Pierre Marquis
Odile Papini
Henri Prade
Editors

A Guided Tour of Artificial Intelligence Research

3 Interfaces and Applications of AI

Springer

A Guided Tour of Artificial Intelligence Research

Pierre Marquis · Odile Papini · Henri Prade
Editors

A Guided Tour of Artificial Intelligence Research

Volume III: Interfaces and Applications of Artificial Intelligence

Springer

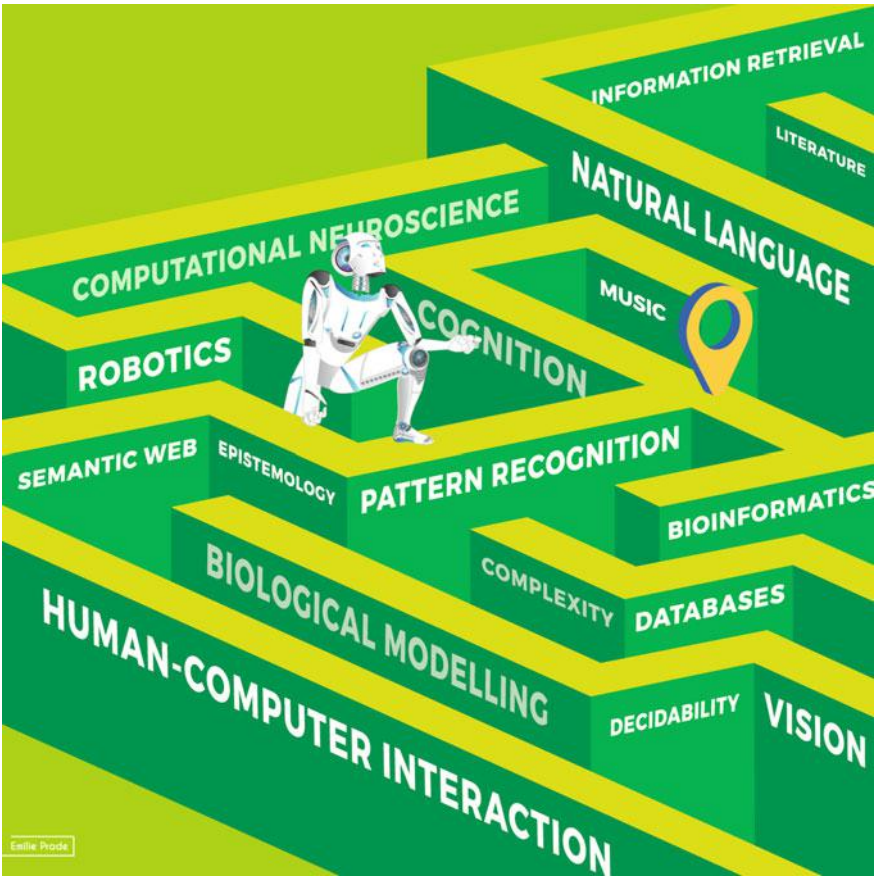
Editors
Pierre Marquis
CRIL-CNRS, Université d'Artois
and Institut Universitaire de France
Lens, France

Odile Papini
Aix Marseille Université, Université de
Toulon, CNRS, LIS
Marseille, France

Henri Prade
IRIT
CNRS and Université Paul Sabatier
Toulouse, France

ISBN 978-3-030-06169-2 ISBN 978-3-030-06170-8 (eBook)
https://doi.org/10.1007/978-3-030-06170-8

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland


INFORMATION RETRIEVAL
LITERATURE
COMPUTATIONAL NEUROSCIENCE
NATURAL LANGUAGE
MUSIC
ROBOTICS
COGNITION
SEMANTIC WEB
EPISTEMOLOGY
PATTERN RECOGNITION
BIOINFORMATICS
COMPLEXITY
DATABASES
BIOLOGICAL MODELLING
DECIDABILITY
VISION
HUMAN-COMPUTER INTERACTION
Emilie Prade

General Presentation of the Guided Tour of Artificial Intelligence Research

Artificial Intelligence (AI) is more than sixty years old. It has a singular position in the vast fields of computer science and engineering. Though AI is nowadays largely acknowledged for various developments and a number of impressive applications, its scientific methods, contributions, and tools remain unknown to a large extent, even in the computer science community. Notwithstanding introductory monographs, there do not exist treatises offering a detailed, up-to-date, yet organized overview of the whole range of AI researches. This is why it was important to review the achievements and take stock of the recent AI works at the international level. This is the main goal of this *A Guided Tour of Artificial Intelligence Research*.

This set of books is a fully revised and substantially expanded version, of a panorama of AI research previously published in French (by Cépaduès, Toulouse, France, in 2014), with a number of entirely new or renewed chapters. For such a huge enterprise, we have largely benefited the support and expertise of the French AI research community, as well as of colleagues from other countries. We heartily thank all the contributors for their commitments and works, without which this quite special venture would not have come to an end.

Each chapter is written by one or several specialist(s) of the area considered. This treatise is organized into three volumes: The first volume gathers twenty-three chapters dealing with the foundations of knowledge representation and reasoning formalization including decision and learning; the second volume offers an algorithm-oriented view of AI, in fourteen chapters; the third volume, in sixteen chapters, proposes overviews of a large number of research fields that are in relation to AI at the methodological or at the applicative levels.

Although each chapter can be read independently from the others, many cross-references between chapters together with a global index facilitate a nonlinear reading of the volumes. In any case, we hope that readers will enjoy browsing the proposed surveys, and that some chapters will tease their curiosity and stimulate their creativity.

July 2018

Pierre Marquis
Odile Papini
Henri Prade

Contents

Preface: Interfaces and Applications of Artificial Intelligence

The project of Artificial Intelligence (AI) to provide machines with advanced capabilities for exploiting data and knowledge puts this area of research at the heart of the sciences of information processing. Maybe due to this state of fact, the contours of AI have somewhat evolved along time and may sometimes raise questions about their exact localization. This is why a whole volume of this Guided Tour of AI Research is dedicated to the interfaces of AI with various scientific, or even artistic, fields, with which strong links exist either at the methodological or at the applicative levels. However, note that this volume does not intend to describe end-user applications of AI in many areas such as logistics, medicine, transport, environment or security, where various AI methods and techniques surveyed in this Guided Tour have been successfully used.

The disciplines considered in this volume and their links with AI are of a different nature. Motivated in general by thematic complementarities, these links have also a historical dimension. Thus, the foreword of this volume reminds us that AI was born for a large part from cybernetics (a point also made in Chapter "Elements for a History of Artificial Intelligence" of Volume I). The links of AI and operations research (OR), whose beginnings precede those of AI by a decade, have been addressed in the afterword of Volume II, considering that it is mainly on the algorithmic side that the two areas meet, where OR has developed methods for specific classes of problems, while AI has rather privileged generic methods.

This third volume has naturally devoted chapters to disciplines that are historically sisters of AI, since born more or less at the same time and since constituted a joint set of research areas together with AI at the beginning: natural language processing, pattern recognition and computer vision, and robotics: The links of these three research areas with AI are, respectively, discussed in Chapters "Artificial Intelligence and Language," "Artificial Intelligence and Pattern Recognition, Vision, Learning" and "Robotics and Artificial Intelligence." Also close and complementary to AI due to their direct links with information, are databases, the semantic Web (which appeared more recently at the interface between databases and AI), information retrieval and human–computer interaction; they are considered in Chapters "Databases and Artificial Intelligence," "Semantic

Web." "Information Retrieval and Artificial Intelligence" and "Cross-Fertilisation Between Human-Computer Interaction and Artificial Intelligence," respectively. All these disciplines are privileged places for past, present or future applications of AI methods (completed with their specific tools). This is also the case for bioinformatics (Chapter "Artificial Intelligence and Bioinformatics") that for a part uses tools, mainly algorithmic, coming from AI. This is completed by two chapters dealing with biological modeling (Chapter "Artificial Intelligence in Biological Modelling") and with computational neurosciences (Chapter "When Artificial Intelligence and Computational Neuroscience Meet"), in relation to AI.

The theoretical and methodological developments of AI have also led to a dialogue with theoretical computer sciences regarding classical topics of this discipline such as computability, decidability, mathematical logic (Chapter "Theoretical Computer Science: Computability, Decidability and Logic") or complexity, and automata in particular (Chapter "Theoretical Computer Science: Computational Complexity"). Moreover, it is probably not exaggerated to say that with time, AI will raise more and more theoretical questions (as this set of volumes already suggests), which should lead to more specific researches in mathematics and in theoretical computer sciences in relation to AI concerns. Besides, AI research and findings have renewed philosophical and epistemological questions (Chapter "Artificial Intelligence: Philosophical and Epistemological Perspectives"), while their cognitive validity raises questions to psychology (Chapter "Artificial Intelligence and High-Level Cognition"). Finally, Chapters "Artificial Intelligence and Literature" and "Music and Artificial Intelligence" deal with the relations of AI with the literature and music, respectively, discussing some of the interactions between science and artistic creation. A brief afterword enumerates some other areas that are also at the interface with AI, but are not covered by this volume, and provides references to them.

Lastly, an epilogue concludes the three volumes by providing an overview of what has been achieved by AI, emphasizing AI as a science, and not just as an innovative technology, and trying to dispel some misunderstandings. It is clear that the chapters in this third volume are largely independent from one another, and that they can be taken in different orders according to the interest of the readers.

Lens, France — Pierre Marquis
Marseille, France — Odile Papini
Toulouse, France — Henri Prade

Foreword: From Cybernetics to Artificial Intelligence

Artificial Intelligence (AI) is still a relatively young research area (just over half a century), but its media coverage has been massive since its inception, sometimes accompanied by polemics. Today, there is hardly any, and the AI research has been broadly diversified, as shown by the three volumes of which we have the last. This variety necessitates a historical perspective: This is what I present in this preface.

As early as 1946, the French publisher Hermann published Norbert Wiener's "Cybernetics or Control and Communication in the Animal and the Machine," which was a great editorial success. And in 1954, an International Congress of Cybernetics (the first one) was held in Namur and I had the chance to participate in it. Artificial Intelligence has only begun to be mentioned as a research area in its own right—or even an autonomous discipline—in the mid-fifties (1955), in the famous "Dartmouth Report," written by John McCarthy, Marvin Minsky, Nathan Rochester and Claude Shannon, the founding fathers of AI. It will be followed by numerous others, as well as by congresses and colloquia on Artificial Intelligence (the baptism of which is a little later). The two domains were obviously neighbors and were often confused, and indeed the interdisciplinary boundaries with other domains such as automation, computer science and cognitive research still remain rather vague today. A vast array of AI topics and subtopics has finally developed. A synthesis like the one presented in this book is therefore welcome.

I had the privilege of being involved in the early developments of AI. First at the Atomic Energy Commission (CEA) where I headed the Analog Calculation Laboratory and then at Euratom where I had set up a "Research Group on Automatic Scientific Information." The subjects we discussed at this time were, mainly:

- Automatic documentation,
- Automatic translation,
- Simulation games,
- Automatic theorem proving.

For obvious economic and political reasons, the subject of machine translation has benefited from the considerable investment but was abruptly reduced when it became apparent that our knowledge of linguistics (especially about semantics) was

still insufficient to lead to acceptable results. In any case, the algorithms that we developed (the objective of which being the development of anti-combinatorial procedures) required computational resources that only a few teams could then have at their disposal. Progress was therefore rather slow, and the impatience of the decision makers was rising. It was reinforced by media excesses in which too many promises were expressed that were not kept. It is interesting to read again the popularization book, written in 1952 and published in 1953 by the NRF in the collection "L'avenir de la science," directed by Jean Rostand. This book was entitled "La Pensée Artificielle" and titled "Introduction à la Cybernétique." The author, Pierre de Latil, was a scientific journalist whom Norbert Wiener's book had strongly impressed. Latil, in his book, emphasizes a mechanism whose research during the Second World War, especially those relating to antiaircraft defense, had shown the importance: the concept of *feedback*. For this concept, which is at the center of Wiener's analyses, Latil suggested a happy neologism in French: rétroaction (p. 52). The concept of feedback was—implicitly or explicitly—the basis of many "automata" projects that were then proposed with Ashby, Gray Walter, McCulloch, etc. The enthusiastic Latil calls these projects revolutionary, despite the modesty of their results and impact. In fact, this impact is essentially on the media and arouses old myths such as the Golem that Wiener will later evoke in "God and Golem, Inc.", MIT Press (1964).

A significant example of the problems that arose in the early 1960s, as well as the directions taken by research, was presented in the talk I gave at the second AFCALTI conference, the French Association of Automatic Computing in 1961. This talk was entitled "Encountering numerical and non-numerical problems in the elaboration of a program dedicated to the resolution of the game of Go-bang" (p. 221 of the report). It elaborated over the theme and problems of the formal representation of "situations," and their evaluation, that of the paths of the trees in which they occur and of the strategies they allow to construct. We were moving from the digital era to the symbolic era.

The "non-digital programming" research line, which began this way, represented an important turning point in the development of "electronic computing machines," previously devoted to the "scientific computation" which dominated space and nuclear engineering. The non-digital theme was widely discussed at the IFIP Congress on Artificial Intelligence held in Munich by Marvin Minsky in 1962. I presented a paper entitled "Research on Artificial Intelligence in EURATOM," but these issues had also been discussed at the two symposia held shortly before by IBM in Blaricum (the Netherlands) with the participation of John McCarthy and many logicians. Some of the papers presented there appeared in the book I edited in 1963 with David Hirschberg at North Holland, which was published in the series "Studies in Logic and the Foundations of Mathematics, Computer Programming and Formal Systems." In particular, one can find in this book an important contribution by John McCarthy: "A Basis for a Mathematical Theory of Computation" (p. 33) which is at the origin of the LISP programming language, as well as the famous article by Chomsky and Schützenberger: "The Algebraic Theory of Context-Free Languages" (p. 118). The book I published in July 1968 at the Presses

Universitaires de France (PUF), in the collection "La Science Vivante," directed by Henri Laugier, under the title "Intelligence Artificielle" gave an update on this research results. The first of its kind, he was featured as such in the *Guinness Book of Records*! Naturally, the themes covered in the nine chapters of this little book can be found in the three volumes of this book: language, games, logic, complexity, control … and even artistic creation.

In 1970, Jean-Claude Quiniou, Jean-Marc Font, Gérard Verroust, Jean-Marc Philippe and Claudine Marenco published a new position book entitled "Les cerveaux non humains" (collection "Le point de la question"). In this book, which is equally misunderstood as that of Latil, we can find, among other things, as an amusing curiosity, the presentation of Boby Lapointe's "bibinary system" (p. 225) and a curiously virulent denunciation of the messianic book of Jacques Bureau "L'ère Logique" (Robert Laffont 1969). Bureau, Engineer, presents the project of a company that would have become "self-adaptive" thanks to the introduction of new technologies, precisely those of control and communication. This utopia resumed an old dream which, after a period of relative oblivion, is reborn today with more vigor than ever in the projects, publications and promotional action of Raymond Kurzweil around these sensational advances to appear (according to Kurzweil) in 2045, the year of "singularity."

Analog and Digital: An Unfinished Debate

Even before the term "Artificial Intelligence" has been imposed, many scientists and educated people worked on projects or models of machines or automata simulating the behavior of animals (foxes, turtles) and even, as we have seen, of human brains. But from the beginning, two quite distinct points of view exist, as reflected in the first published works:

- The engineer's point of view which emphasizes the problems of control and insists on the patterns of feedback: it is the era of "servomechanisms" (the servo/brain homophony in French was quite unfortunate), and Bode's theorem plays an essential role here. From a technical point of view, the use of high gain and high input impedance DC amplifiers is developed. This is the analogical approach.
- The point of view of the logician in search of formalisms and algorithms of symbolic manipulation: predicate calculus, formal coding techniques, decision problems, computational complexity, etc. Propositional calculus and predicate calculus that have developed with the crisis of set theory are now widely published and taught. This is the digital approach.

Although Norbert Wiener was at the beginning a traditional logician, a specialist of set theory and relational algebra, and then a specialist of harmonic analysis and the theory of Brownian motion, he has been involved during the Second World

War, in works on "servomechanisms" and their curly schemes, in particular for air defense. After his stay in Mexico City and his research in collaboration with Arturo Rosenblueth in cardiology and physiology of the nervous system, he had the opportunity to develop a real engineer's point of view where he used the resources of the most advanced results from functional analysis. His 1946 book focused on the phenomena of *feedback* for which he develops a complex mathematical analysis, illustrated by numerous schemas (which Latil will resume and complete): loop patterns and circles.

Among the elementary geometric metaphors, that of the circle is one of the most used, but also one of the most ambiguous: reassuring when it comes to family or friends, disturbing in a reasoning or argumentative process. It then joins other metaphors with a pejorative connotation: labyrinth, complex arborescence, etc. The notion of a loop, above all, presents opposite connotations, which remind one of incompleteness as well as of completion. Circular patterns are numerous in Wiener's essay as in the book of Latil.

But the circle fear has always been intense among philosophers and scholars. It expresses itself strongly in all those who wish to specify the conditions for an effective and honest argumentation. The detection of a circular reasoning is the cause of perplexity, even discomfort, in the reader: an experience of thought often painful … Such a situation is a consequence of the test that logicians, mathematicians and philosophers crossed at the beginning of the century. The "crisis" of set theory, the discovery of paradoxes in formal logic, like the liar paradox, and the precise expression of "limiting statements" made the need to base the scientific disciplines—and especially the most formal ones—on a solid foundation …

A Case Study: The Paradox of Operational Units in Analog Computing

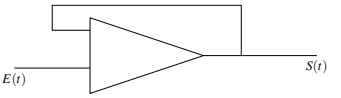
If they looked opposed at a first glance, the points of view of the engineer and of the logician can nevertheless meet: This was indeed the case for Wiener. It so happens that I had the opportunity to sketch their reconciliation at the beginning of my career, at a time when, after working on the foundation of mathematics, with Bachelard, I was interested in the foundations of analogical electronic computation, which was a hot topic those days. The advances in technology and the dramatic increase in the speed of "digital machines," which made it possible to solve differential equations in a manner compatible with "real-time" applications, led to the decline of analog computing. I then discovered, in collaboration with Claude Caillet, that the study of the elementary mechanisms of analogical computation, when carried out to the end, revealed a form of paradox, a profoundly different paradox from those of set theory (which are the basis of the theory of complexity), a paradox whose importance deserves to be emphasized. The path of thought can then continue in the analogical domain in a certain parallelism with the one followed, in

the digital domain. After Post and Turing, John von Neumann, in "The Computer and the Brain" had, by other means, pursued this parallelism.

I therefore think it useful, thus leaving the strict framework of a preface, to present again what I suggest to call "the paradox of operational units in analogical computation."

The units in question here are the electronic assemblies which are the traditional components of an analog machine: adders, inverters (of algebraic sign), integrators, etc. Their main component is a DC amplifier where the feedback plays an essential role, as can be seen in the following diagram (including a loop!):

The objective sought in the design of the analog diagram below is simply to obtain the opposite value of a given value: $S(t) = -E(t)$.



Due to the looping of the output on the input, this input voltage, $E(t)$, and the output voltage $S(t)$ are added algebraically and we have:

$$\varepsilon(t) = E(t) + S(t).$$

This value is multiplied by the gain G of the amplifier. Since this gain is very large (of the order of 10^5), $\varepsilon(t)$ must be very small. So that, the output $S(t)$ which is equal to $G \times \varepsilon(t)$ is finite. This means that $S(t)$ is very close to $-E(t)$, which is the desired result. But $\varepsilon(t)$ is, in fact, the error committed in this computation. The solution therefore exists only because it remains an error: It is precisely the paradox, a "different" paradox! Perhaps, it would not be uninteresting to repeat this analysis by studying the dynamics of the process which leads to equilibrium in the flow of information along the loop and to consider in addition the bandwidth of the amplifier …

Old and New Singularities

In his ambitious book "Darwin Among the Machines" (Helix Books, 1997), whose subtitle is "The Evolution of Global Intelligence," George Dyson evokes Thomas Hobbes and his Leviathan (1651) where automata already played a non-negligible role. The technological developments which followed one another at the end of the First World War provoked speculation that went beyond purely logical questions and tackled all the problems about intelligence. Among the remarkable personalities that arise then, we note in particular that of I. J. Good. This English Mathematician played a decisive role, together with Alan Turing, in deciphering the Enigma code used by the German navy. Quite naturally, like Turing, Good's research activity

then concerned the field of calculators. In 1963, he gave a lecture with the provocative title "Speculations Concerning the First Ultraintelligent Machine." This lecture will be published in "Advances in Computers," vol. 6, 1965. Here is its first sentence:

The survival of man depends on the early construction of an ultraintelligent machine.

Good is thus in a great Anglo-Saxon tradition, which obviously includes Jonathan Swift (1667–1745) with the Academy of Laputa and his machine for producing the literature, Mary Shelley (1797–1851) and the creature of Dr. Frankenstein, as well as Samuel Butler (1835–1901) and the utopia of Erewhon. In 1929, Olaf Stapledon (1886–1950) published "Last Men and First Men" which inspired Fred Hoyle as I. J. Good and, among many anticipatory concepts, presents that of "distributed intelligence."

Since the beginning of the eighties, many utopian speculations associated with the development of Artificial Intelligence have multiplied, with the works, inventions and publications of Raymond Kurzweil (born in 1948). Gifted student, Kurzweil developed models and statistical software. In 1965, he was invited by CBS to perform a piano piece composed on a computer. Then, he developed technologies and software for pattern recognition (characters and sounds) and was Founder of many companies, including the "Kurzweil Foundation" which supports the development of technologies for people with disabilities. In 1990, Kurzweil published "The Age of Intelligent Machines" (MIT Press), and in 1999, "The Age of Spiritual Machines: When Computers Exceed Human Intelligence" (Penguin Books) following the path paved by Good. Starting from numerous extrapolations about the speed and capacity of computers, he predicts the coming of a future when the computer abilities will exceed in all fields that of men, and he even calculates the date when this will be the case: 2045. This is what Kurzweil calls *singularity*. The impact on the media is immense, with several books, including films and international conferences (including "The Singularity is Near: When Humans Transcend Biology," Viking 2005). Now, the focus is on the biological abilities of automata, the possibility of a form of immortality …

Utopia, in any case, seems immortal!

Paul Braffort
(1923–2018)
www.paulbraffort.net

Theoretical Computer Science: Computability, Decidability and Logic

Olivier Bournez, Gilles Dowek, Rémi Gilleron, Serge Grigorieff, Jean-Yves Marion, Simon Perdrix and Sophie Tison

Abstract This chapter deals with a question in the very core of IA: what can be computed by a machine? An agreement has been reached on the answer brought by Alan Turing in 1936. Indeed, all other proposed approaches have led to exactly the same answer. Thus, there is a mathematical model of what can be done by a machine. And this has allowed to prove surprising results which feed the reflection on intelligence and machines.

O. Bournez
LIX, Ecole Polytechnique/CNRS, Palaiseau, France
e-mail: Olivier.Bournez@lix.polytechnique.fr

G. Dowek
INRIA, LSV, ENS-Cachan, Cachan, France
e-mail: gilles.dowek@ens-paris-saclay.fr

R. Gilleron
INRIA Lille Nord Europe, Lille, France
e-mail: remi.gilleron@inria.fr

S. Grigorieff
IRIF, Universite Paris Diderot/CNRS, Paris, France
e-mail: seg@irif.fr

J.-Y. Marion
Universite de Lorraine, LORIA, CNRS, Nancy, France
e-mail: jean-yves.marion@loria.fr

S. Perdrix
INRIA, Universite de Lorraine, LORIA, CNRS, Nancy, France
e-mail: Simon.Perdrix@loria.fr

S. Tison (✉)
University of Lille, Inria, CNRS, CRIStAL, Lille, France
e-mail: Sophie.Tison@univ-lille.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_1

1 Introduction

1.1 Theoretical Computer Science and the Core Themes in IA

> To me there is a special irony when people say machines cannot have minds, because I feel we're only now beginning to see how minds possibly could work—using insights that came directly from attempts to see what complicated machines can do.
>
> "Why People Think Computers Can't" (Minsky 1982)

> Artificial intelligence researchers predict that "thinking machines" will take over our mental work, just as their mechanical predecessors were intended to eliminate physical drudgery. Critics have argued with equal fervor that "thinking machines" is a contradiction in terms. Computers, with their foundations of cold logic, can never be creative or insightful or possess real judgment. Although my own understanding developed through active participation in artificial intelligence research, I have now come to recognize a larger grain of truth in the criticisms than in the enthusiastic predictions.
>
> "Thinking machines: Can there be? Are we" (Winograd 1982)

As witnessed by the above quotes, there is an ongoing debate regarding the founding themes of AI within the pioneers of the subject. That such a debate is inevitable seems clear from the following remark by the neurologist Warren Mc Culloch during the discussion at Hixon symposium, 1948, after John von Neumann's talk on machines (1951):

> I confess that there is nothing I envy Dr. von Neumann more than the fact that machines with which he has to cope are those for which he has, from the beginning, a blueprint of what the machine is supposed to do and how it is supposed to do it. Unfortunately for us in the biological sciences we are presented with an alien, or ennemy's, machine. We do not know exactly what the machine is supposed to do and certainly we have no blueprint of it.

We take no sides in such debates but stress some similarities between the bold ideas questioned by AI and some famous results in Logic and Computability Theory which are presented in this chapter. We shall consider four examples and keep in mind the caution by the French philosopher Jacques Bouveresse about the "*prodiges et vertiges de l'analogie*" ("*wonders and vertigo of analogy*") (1999).

1. In the 17th century, Gottfried Wilhelm Leibniz assigned two goals to science and philosophy (cf. Sect. 6.1):

- An all-embracing precise symbolic language (*characteristica universalis*), built on an "alphabet of thoughts", which would allow a clear vision of the meaning and truth of all assertions.
- A method to handle the assertions of this language (*calculus ratiocinator*) so as to elucidate their meaning and the relations between them.

Restricted to mathematical knowledge, a positive answer to Leibniz' first goal emerged in 1879 with Gottlob Frege's *Begriffsschrift* ("concept-script"), and with the development of mathematics within such a formalized framework done by Alfred Whitehead and Bertrand Russell in the *Principia mathematica*, published in 1910.

That this formalization fully answers Leibniz' first goal was shown half a century later by Kurt Gödel's completeness theorem, 1930: every possible deductive reasoning is captured by the formal logic created by Frege.

Thus, deductive thought can be reduced to a few explicit, simple and precise axioms and deduction rules. This result is at the core of the existing programs for automated theorem proving. One can consider this as showing that an important ability of the human intelligence is mechanizable. Thus, the *calculus ratiocinator* that was hoped for by Leibniz ca 1676 does indeed exist. However, its scope is that of a mathematical language which is far beneath any *lingua characteristica universalis*. Therefore the solution Leibniz was looking for to reduce any dispute, namely "*Calculemus!*", has been shown possible merely inside a limited mathematical scope.

2. In 1936 Alan Turing introduces the so-called Turing machines and proves the existence of a universal Turing machine (cf. Sect. 3.2). The same year Alonzo Church states the celebrated Church–Turing thesis (cf. Sect. 2.7). Considered together, these two results insure that there exist machines which can emulate all other machines whatever they compute. This last result is truly incredible. Let's cite von Neumann (1951):

> We might expect a priori that this is impossible. How can there be an automaton which is as effective as any conceivable automaton, including, for example, one of twice its size and complexity? Turing, nevertheless, proved that this is possible.

Thus, adaptability, flexibility, protean nature (in short, what universality means) are qualities of human intelligence that can be those of a machine. Of course, such an analogy does not bring any answer to the question: can the machines completely match what the human brain can do?

3. In 1938, Stephen Cole Kleene proved an astonishing fixed point theorem (cf. Sect. 3.2). In programming terms, it states that, whatever transformation we consider on programs, there exists a program which has the same input/output behaviour than the associated transformed program. Though it may seem a bit technical, this result has a wide range of applications. For instance, consider the transformation that associates to a (zero input) program P a program $\widehat{P}$ which outputs the program P itself, i.e. the lines of code which constitute P. A fixed point program for this transformation is then a program which outputs its own code! Thus, there exists a program which is able to describe itself explicitly and faithfully. This shows that self-reference, a property sometimes viewed as relevant to consciousness, can also be relevant to a computer program or to a machine. Moreover, there are also machines with both self-reference and universality properties.

A particular version of self-reference is self-reproduction. The evidence of such a phenomenon was discovered by John Von Neumann in 1949. Let us cite von Neumann (1951) again:

> Can one build an aggregate out of such elements in such a manner that if it is put in a reservoir, in which there float all these elements in large numbers, it will then begin to construct other aggregates, each of which will at the end turn out to be another automaton exactly like the original one? This is feasible.

This phenomenon is now popularized by its criminal use: malware and computer viruses witness self-reproduction and also universality. Thus, a feature generally believed to be relevant to the sole living world also appears with programs and machines.

4. Ca 1958, the US logician Haskell Curry notices an analogy between proofs in particular deduction systems of formal logic and the representations of computable functions as terms in the so-called "combinatory logic". This analogy has been clarified by William Alvin Howard in 1969 and shown to be a formal isomorphism between proofs in intuitionistic logic and representations of particular computable functions as terms in typed lambda-calculus. Since then, this isomorphism, now named Curry–Howard isomorphism, has been extended to more powerful logical systems and to proofs in classical logic so as to deal with larger classes of computable functions (cf. Sect. 5). Though the whole class of computable functions cannot be dealt with any variant of this isomorphism, the involved subclasses are quite huge and sufficient in practice.

Curry–Howard isomorphism can be seen as an extension of item 1 supra. The deductive process is not only mechanizable—as a kind of calculus ratiocinator—but it is in fact equivalent to the computational process! Once more, "comparaison n'est pas raison", but nevertheless Curry–Howard isomorphism is an astonishing fact which gives matter for reflection beyond its sole mathematical statement.

1.2 What We Pick and Choose in Theoretical Computer Science

The 2283 and 3176 pages of the Handbooks (van Leeuwen 1990; Abramsky et al. 2001) witness that theoretical computer science is a vast domain. It is simply impossible to cover this domain in a few dozens of pages. Reflecting the authors' taste, this chapter and the next one are devoted to describing a particular but central subject: computation.

The notion of computation is as old as mathematics and continues to flourish:

- Computations over integers: Euclid's algorithm to compute the greatest common divisor of two integers, Sieve of Eratosthenes to enumerate prime numbers, etc.
- Computations over reals: Simpson's rule for numerical approximation of definite integrals, Newton's method for finding successively better approximations to the roots of a real-valued function, etc.
- Geometrical computations: computing areas and volumes of diverses objects, Fortune's sweep line algorithm for generating a Voronoi diagram from a set of points $\{A_1, \ldots, A_n\}$ in a plane (this diagram splits the plane in n polygonal parts: the ith one consists of those points which are closer to A_i than they are to the A_j's, $j \neq i$), etc.

- Sorting items according to some given ordering. There are many ways to sort: insertion, exchange (bubble sort, quicksort), selection (heapsort), merge, distribution, cf. the 388 pages on this topic in Knuth's celebrated book (1973).
- Computations over strings: pattern recognition (Knuth, Morris and Pratt's algorithm 1975), data compression (Ziv and Lempell's algorithm 1978), etc.
- Computations over finite graphs: minimal cover (i.e. a set C of nodes such that every node of the graph is in C or is on an arc with one end in C), shortest path between two nodes (Dijkstra's algorithm 1956), etc.

The quest for a notion of computation model emerged in the first half of the 20th century. A lot of models have been defined, illustrating the non trivial character of the notion of computation and its many facets: sequential computing, parallel computing, quantum computing,…Many of these models are presented below. Each computation model is either a particular mathematical representation of computability or a particular class of theoretical machines which are based on some physical constraints and/or some hypothesis relative to the physical world. For instance, there are sequential machines with no parallelism or a very rudimentary one (e.g. multitape Turing machines) and massively parallel synchronous machines (e.g. cellular automata).

Nowadays some computation models based on machines have physical implementations: computers. This creates a duality somewhat similar to that between syntax and semantics, i.e. between the signifier and the signified.

Though the various computation models are based on different concepts of computation, all the functions that they compute are also computable with Turing machines. Moreover, most models (those strong enough) compute exactly the same functions as do Turing machines: though based on ideas of computation which may be far apart, they nevertheless have the same computation power. This has led to the Church–Kleene–Turing thesis. Such a thesis justifies the name "Computability theory" for the mathematical development of these equivalent models. One of its main objectives is to distinguish what is computable from what is uncomputable.

As a mathematical modelization of the input/output behaviour of algorithms, computability theory is a denotational view on computability. It is completed with a "resource complexity theory" which constitutes a first approach to an intensional view and deals with the following questions: how much time does it take? which space does it need?

It is only around 1980 that the notion of algorithm itself, i.e. the full intensionality of computation, has been given a mathematical formalization. It is remarkable that this duality denotational/intensional is reflected in logic with proof theory: proofs being seen as algorithms via Curry–Howard's isomorphism.

2 Emergence of the Notion of Computability

Several books give a nice presentation of computability. Two classical references for a mathematical approach of computability are Rogers (1967) and Odiffredi (1989). A programming perspective is adopted in Jones (1997), which also treats complexity

theory. Savage (1998) details numerous machine modelizations of computability. Papadimitriou (1994) essentially deals with complexity but is also an introduction to computability.

2.1 Discrete Computation Models Based on Mathematics

Primitive Recursion. First attempts towards a mathematical formalization of computability considered various processes to construct a function from some other ones. In programming terms, such constructions correspond to simple control instructions. The very first such approach led to the so-called "primitive recursive" functions. On the set $\mathbb{N}$ of nonnegative integers, these functions are obtained from a few very simple basic functions (namely, constant 0, successor, projections $\mathbb{N}^k \to \mathbb{N}$) by composition and recursive definitions. In imperative programming, these functions are those which can be obtained using the sole `FOR` loop (excluding any `WHILE` loop). But soon, it was noticed (Ackermann 1928) that this class is not rich enough to completely capture computability.

Herbrand-Gödel Systems of Equations. The first successful formalization was discovered by Jacques Herbrand in 1931. Due his premature death (at age 23) in a mountain accident a few weeks afterwards, he never published it. However, he had described the main lines of his ideas in a letter to Gödel who developed them and published them, crediting Herbrand. This formalization, now called "Herbrand–Gödel Systems of Equations", uses finite systems of functional equations which have the flavor of functional programming.

Kleene Recursive Functions. A popular formalization of computable functions is due to Kleene (1936) (cf. also his celebrated books Kleene 1952, 1967 and Davis 1965). It uses a new construction: minimization. This allows to go from a total function $f : \mathbb{N}^{k+1} \to \mathbb{N}$ to a partial function $g : \mathbb{N}^k \to \mathbb{N}$ such that

$$domain(g) = \{\mathbf{x} \mid \exists y \ f(\mathbf{x}, y) = 0\} \qquad g(\mathbf{x}) = the\ smallest\ y\ such\ that\ f(\mathbf{x}, y) = 0$$

Not surprisingly, Kleene recursive functions are exactly those functions which can be obtained with imperative programs using the two loops `FOR` and `WHILE`. They also coincide with functions obtained via Herbrand–Gödel systems of equations.

2.2 Discrete Computation Models Based on Sequential Machines

Turing Machines. No matter how strong their definitions are, Herbrand–Gödel and Kleene formalizations do not convincibly answer the following basic question: "*Do these formalizations capture all computable functions?*". Gödel himself was not sure

of the answer for the system he formalized on Herbrand's ideas. It is only the very remarkable analysis developed by Turing (1936) (cf. Davis 1965) which brought a compelling positive answer. In his paper, Turing convincingly argues that what can be computed in finitely many steps by a human being, working with paper and pencil according to some "mechanical rules", can also be computed by the device he imagined, now called "Turing Machines". Roughly, this device is a typewriter tape on which a head can read and write characters from a fixed finite alphabet and shift one space to the left or to the right. Writing, moving and halting are precisely ruled by the state of the machine which can vary in a fixed finite set.

One can prove that the functions computable by Turing machines are exactly Kleene recursive functions. Though quite long and tedious, the proof is not a difficult one (cf. Martin Davis classical books, in particular 1958).

Turing Machines with Several Tapes and/or Several Heads. A natural extension of Turing machines consists to allow several tapes and/or several heads. One can also allow multidimensional tapes (with heads able to shift in any direction). It is proved that every function computable with such a machine is also computable with an original Turing machine.

Kolmogorov–Uspensky Machines. In order to test the robustness of Turing device, Kolmogorov and Uspensky (1958) replace the linear tape by an undirected graph tape, each node of which carries a symbol letter. Moreover, at each step there is a distinguished node which plays the role of a head. This graph evolves through time: around the head, some nodes can be removed, new ones can appear, Reading, writing and graph modification are local actions ruled by the state and the labels of the cells in a neighborhood of the head. A constraint is assumed: there is a fixed bound on the degree of the graph (i.e. the maximum number of neighbours of a cell).

Schönhage Machines (Storage Modification Machines). Schönhage (1969, 1980), goes farther: he allows a directed graph with bounded outdegree but arbitrary large finite indegree. His intuition is to view the arcs of the graph as pointers: for some fixed integer k, a node can point to at most k other nodes (hence a bound k on the outdegree of the graph) whereas there is no bound on the number of nodes pointing to a given node (hence no bound on the indegree of the graph).

Though Kolmogorov–Uspensky and Schönhage machines are a priori much more powerful than Turing machines, they compute exactly the same functions.

2.3 Discrete Computation Models Based on Random Access Machines

To get a model of computation much closer to the common notion of computer, Melzak (1961) and Minsky (1961) introduced a machine with infinitely many registers, each one being able to store an arbitrary integer. These registers are indexed by integers. An original feature of the model is indirect addressing: in a single move

the machine can go from a given register with contents an integer n to the register with index n. This feature is the reason for the name "Random Access Machines" (in short RAM) given to this model. Allowing also very elementary operations on the contents of the registers, Elgot and Robinson (1964) (cf. also Cook and Reckhow 1973) showed that this model compute the same functions than Turing machines.

2.4 *Chomsky Type-***0** *Grammars*

Chomsky Type-**0** grammars are another model for discrete computations which manipulate words. Such a grammar consists of a finite alphabet V, a subalphabet T of "terminal symbols", a distinguished nonterminal "start symbol" S and finitely many "production rules" of the form $\alpha \to \beta$ where α, β are words in the alphabet V and α contains at least one nonterminal symbol. Applying the production rule $\alpha \to \beta$ to a word w in alphabet V consists in replacing some factor α in w by the word β. To such a grammar is associated a language which consists of all words in the alphabet T of terminal symbols which are obtained from the fixed one-letter word S by iteratively applying some production rule.

One can show that a language is so obtained if and only if its characteristic function is Turing computable.

2.5 *A Model for Discrete Computation with Massive Parallelism: Cellular Automata*

We now describe another computation model which also does not go beyond computability by Turing machines. However, as concerns length of computations, this model is an order of magnitude more efficient. Also, the model is quite fascinating: we invite the reader to look on the web for videos on "Conway's Game of Life"….

Today technology allows for computers using thousands of CPU's. A challenging question is to get programming languages able to fully use such a massive parallelism. Of course, this is related to the architecture chosen for such a massive parallelism. Two key choices:

– what is the geometrical form of the massively parallel processor array?
– do we require synchronous runs or admit asynchronous ones?

Answering these two questions, John von Neumann set up in 1949 a theoretical framework for massive parallelism: that of cellular automata. He made two fundamental choices: *synchronous runs* and usual *two or three dimensional geometry*. These choices have since been much questioned and other approaches have been developed. None has yet lead to new programming paradigm for massive parallelism.

Von Neumann had in mind a 3-dimensional model emulating some features of brain activity (cf. von Neumann 1951 or Delorme 1999). But, already in dimension

1, cellular automata are a highly complex model.
Formally, a one-dimensional array of cellular automata indexed by the set $\mathbb{Z}$ of positive or negative integers, is a triple (Q, δ, I_0) where Q is the finite set of states, $\delta : Q^3 \to Q$ is the transition function and $I_0 : \mathbb{Z} \to Q$ is the initial distribution of states. Denoting by $I_t(i)$ the state of the cell in position i at time t, the run of the cellular automata array obeys the following rule: $I_{t+1}(i) = \delta(I_t(i-1), I_t(i), I_t(i+1))$, i.e. the next state of a cell depends only on the present states of itself and its two left and right closest neighbours.

This definition extends to arrays of cellular automata indexed by $\mathbb{N}$ (the set of nonnegative integers) or finite arrays indexed by $\{1, \ldots, n\}$ via some adequate modification of the transition function δ so as to deal with the frontier cells (case $i = 0$ for an array indexed by $\mathbb{N}$, cases $i = 1, n$ for a finite array indexed by $\{1, \ldots, n\}$). It also extends to higher dimensional arrays and, more generally, to arrays indexed by the vertices of a bounded degree undirected graph.

Von Neumann's choice of synchronicity appears as a powerful axiom for the theory of cellular automata. Though there are a few interesting theorems with asynchronous cellular automata, their theory does not match the rich theory of the synchronous case. Let us cite some of the results in the synchronous theory.

- In dimension 2, on $\mathbb{Z} \times \mathbb{Z}$, John Von Neumann described in 1949 (cf. 1949) a two-dimensional cellular automaton, on $\mathbb{Z} \times \mathbb{Z}$, which has 29 states and is both Turing-complete (i.e. it can compute every Turing computable function) and self-replicating. This last property makes Von Neumann the father of computer virology. Von Neumann's automaton has been improved to one with 8 states by Codd (1968) (the famous inventor of relational data bases) via an emulation of the structure and behaviour of the brain nervous cells.
- Again on $\mathbb{Z} \times \mathbb{Z}$, John Horton Conway designed in 1970 a fascinating and quite surprising two states cellular automaton: "Life", which has an incredible variety of behaviors.
- *Firing squad synchronisation.* John McCarthy and Marvin Minsky (Minsky 1967) designed a cellular automaton which synchronizes every $\{1, \ldots, n\}$ line: starting from an initial configuration in which cell 1 is in a particular state G (for "general") and cells $2, \ldots, n$ are in a particular quiescent state e (i.e. a state such that $\delta(e, e, e) = e$), at some time all cells enter simultaneously and for the first time in a given state F (for "fire"). Such a synchronisation is also possible for finite connected undirected graphs with bounded degree (Rosenstiehl 1986). There are also *fault-tolerant* such synchronizing automata (Jiang 1992), (cf. also Yunès 2006; Grigorieff 2006 for subsequent improvements)

The pertinence of von Neumann's choice of a simple two dimensional geometry for arrays of cellular automata has been somehow validated by the unfortunate experience of the "Connection Machine" (1980), based on (Hillis 1986) PhD thesis, and to which contributed the Nobel-Prize-winning physicist Richard Feynman (see Hillis 1989). Its $2^{16} = 65536$ processors were connected as a 16-dimensional hypercube. But no one has ever been able to truly exploit the massive parallelism of

such a machine. Adequate programming with such a geometry proved to be far too complex. Since the commercial failure of this machine, implementation of massive parallelism has been frozen for a long time. Nevertheless, presently, there still is a demand for massively parallel programming, coming from the biggest names in the global computer industry. And this demand even appears in New York Times and Wall Street Journal articles…

Going back to von Neumann's ideas and to elementary geometry, new directions are investigated to master massive parallelism (Mazoyer and Yunès 2010).

2.6 A Model Far Apart: Lambda-Calculus

A few months before Turing and independently, Alonzo Church also obtained a mathematical model which he claimed to capture the intuitive notion of effective computation: lambda-calculus, a topic he developed since 1930. His argumentation is presented in Church (1936) (see Davis 1965).

Though we cannot give a detailed exposition of lambda-calculus, let us mention that it is a language consisting of terms built with two operators (cf. also the remark at the end of Sect. 5.4.1):

- *Application*: given terms t, u, the term denoted by $(t\ u)$ is said to be obtained by applying t to u. The intuition is that of applying a function to an input. Now, the conceptual difficulty is that there is no restriction to application in lambda-calculus, every term can play the role of a function and that of an input…For instance, $(t\ t)$ is a term.
- *Abstraction*: given a term t and a variable x (which may occur in t or not) the term denoted by $\lambda x\ t$ is said to be obtained by abstraction. The intuition is that this term represents the function which to any x associates t (which may depend on x or not).

This language of terms is completed with a reduction rule called β-reduction: a term of the form $((\lambda x\ t)\ u)$ (i.e. applying an abstraction term $\lambda x\ t$ to an input u) is reduced to the term $t(u/x)$ obtained by substituting u to every occurrence of x in t (more precisely, the substitution is done on those occurrences of x in t which are not under the scope of some abstraction $\lambda x \ldots$ inside t).

Thus, lambda-calculus is, historically, the first rewriting system. One can prove that it is confluent: if there are sequences of reductions leading from a term t to distinct terms t_1 and t_2 then there is a term u and sequences of reductions leading from t_1 to u and from t_2 to u.

Now, how can lambda-calculus be a computation model? Where are the integers or the words? Church's very original idea is to identify a nonnegative integer n with the term $Church_n$ which represents the functional which iterates n times a function. For instance, the term $Church_3$ representing the integer 3 is $\lambda f\ \lambda x\ (f(f(fx)))$ (the variable names f, x are chosen to help the intuition). In this way, to every term t one

can associate the partial function over nonnegative integers which maps n to p if the term $(t, Church_n)$ can be reduced to $Church_p$ (by confluence, there is at most one such term $Church_p$). Church and Kleene proved that the terms of λ-calculus represent exactly the partial computable functions (cf. Sect. 3.1).

There are typed versions of lambda-calculus where application is restricted as follows: if t, u have respective types $A \to B$ and A for some types A, B then the term $(t\ u)$ is a typed term with type B. This constraint excludes terms such as $(t\ t)$. A given term can have many types. For instance, term $Church_n$ has type $(A \to A) \to (A \to A)$ for every type A. The type constraint insures that the function associated to a term of type $A \to B$ is total. A rather nice property which has a price: not all computable functions can be obtained with typed terms!

In Sect. 5, we show an astonishing property of typed lambda-calculus: a typed term can be regarded as a proof of a formula associated to its type!

To go further, see the following books: Barendregt (1980), Hindley and Seldin (1986), Krivine (1990), Hankin (1994), Barendregt et al. (2013).

2.7 Church Thesis, Diverse Formulations

The fact that all the computation models considered in Sects. 2.1–2.6 lead to the same formal notion of computability has soon been observed. This led to *Church thesis* (also more accurately called *Church–Kleene–Turing thesis*), first expressed by Stephen Kleene, then a student of Church:

> What is effectively computable is computable by some Turing machine.

In this formulation, the first occurrence of the word "computable" refers to an intuitive notion whereas the second one refers to a formal notion, that of Turing machine (Gandy 1980; Copeland 2002; Ord 2006). In their discussions around this thesis, Church, Kleene and Turing did not limit intuitive effective computability to machines or physical computational device but extended it to all possible deductive algorithmic processes, i.e. to all kinds of formal deduction. As Copeland (2002), pertinently argues, this thesis is often confused with another statement, called M thesis in Gandy (1980), and also called *physical version of Church thesis*:

> What is effectively computable by some machine is computable by some Turing machine.

In this M thesis, the notion of machine is intuitive but required to obey all physical laws of our world (Copeland 2002), though resource constraints are not considered. Else it is easy to refute, cf. Ord (2006) and Copeland and Sylvan (1999) for some counterexamples.

Observe that this variant thesis is intimately related to a notion of model for the physical world and the question of its correctness. Indeed, a close variant of the M thesis is

> Every process which admits a mathematical description can be simulated by a Turing machine (Copeland 2002).

Again, if the process is not required to obey all physical laws of our world, this thesis is easy to refute with the same counterexamples than for the physical thesis (Copeland 2002).

These three thesis are distinct: the first one is about the extent of formal systems, the second one is about the physical laws of the surrounding world, whereas the third one is about the mathematical models we have of the physical world (Smith 1999; Copeland 2002; Yao 2003).

Each one of these thesis refers to an intuitive notion hence cannot be formally proved.[1] Nevertheless, one can look for some minimal sets of axioms about formal systems or physical machines which allow to prove these thesis. Such minimal hypothesis on formal systems or physical machines may help consolidate our belief in these thesis (Gandy 1980; Dershowitz and Gurevich 2008; Boker and Dershowitz 2008).

It is also interesting to consider the contrapositive statements obtained from Church thesis and its two variants. They express that every process which computes some function which is not computable by a Turing machine must use some *resource* which either is not algorithmically computable (first thesis) or is not computable by any physical machine (second thesis) or admits no mathematical description (third thesis). In all cases, such a problematic resource can be qualified as *unreasonable*. Independently of any assumption on the truth of these thesis, discussing what makes such a resource unreasonable may be important to understand the surrounding world and the models we have to represent it.

Finally, observe that these thesis express deep facts about our ability to physically or logically describe the surrounding world (Dowek 2007) and, more generally, about the relations between computability, mathematics and physics:

– does nature compute?
– what are the connections between nondeterminism, chaos, unpredictability and randomness (Longo and Paul 2009)?

2.8 *Gandy's Axiomatization of Church Thesis*

In Gandy (1980) Robin Oliver Gandy reduces physical Church thesis to four (quite technical) principles. An interesting fact about this axiomatization is that it allows to get a mathematical proof that what is computable by some physical device obeying these four principles can also be computed by a Turing machine. Otherwise said, Gandy suggests to replace the physical Church thesis by a thesis asserting that any physical deterministic discrete mechanical device must obey these four principles. Though developed for deterministic systems, Gandy's argumentation easily extends to nondeterministic ones.

[1] In fact, in many books about computation and machines, the notion of computation and that of machine are defined as what obeys the associated thesis.

Only discrete digital systems are consider by Gandy who explicitly excludes analogical systems. He assumes that the physical space is the discrete version of the usual three dimensional geometrical space. Dowek (2007) observes that Gandy's four principles are based on two fundamental hypothesis in physics: information transmission has finite speed and information storage has finite density.

Gandy's principles are satisfied by all discrete mathematical models of computation that have been considered up to now, including those which are massively parallel such as Conway's Game of Life (cf. Sect. 2.5). Now, there are theories in physics for which the above two hypothesis of finite speed and finite density either are not valid or take a stronger form (e.g. a limit finite speed in relativistic physics). This leads to question Gandy's axioms in quantum mechanics and relativistic physics. See Nielsen (1997) for a discussion about sources of noncomputability in quantum mechanics and some consequences as concerns Church thesis and our mathematical models in physics. Some principles à la Gandy to capture quantum computing are discussed in Arrighi and Dowek (2008).

Let us mention that Gandy's axiomatization has been simplified and extended by Wilfried Sieg, see Sieg (1994, 1997, 1999, 2008). Also, a totally different axiomatization of the notion of algorithm has been given by Gurevich, cf. Sect. 4.

3 Computability Theory

In Sect. 2 we saw the diversity of computation models and the remarkable fact that they all lead to the same class of computable functions. This fact supports Church–Kleene–Turing thesis which asserts that every one of these models formalizes the intuitive notion of computability.

3.1 Unhalting Computations

Another important fact about computation models is that in each of them there are computations which do not halt. As everybody knows, there is no programming language for the sole computable total functions. It is a commonplace experience for every programmer that some programs do not halt. . .

Whence the notion of *partial computable function* (also called partial recursive function): a function which may not be defined everywhere and for which there is some algorithm which gives its value when defined and rejects or does not halt when undefined.

This denomination "partial computable function", though very common, is not very precise. Indeed, the word "partial" qualifies two very different properties:

- first, it expresses that the function may not be everywhere defined,
- second, it stresses that the computability character of the function is partial. Indeed, the afferent algorithm does not decide if, given some input, the function is defined or not. If defined, the algorithm halts and gives the value of the function, else it may not halt hence we never get the information that the function is undefined.

A much better denomination should be *partial computable partial function* (a denomination suggested by the French logician Daniel Lacombe (cf. Sect. 1.9 in Lacombe 1960) but, unfortunately, it has not become a standard one.

The notion of partial computable function has a counterpart with sets: that of *computably enumerable set* (also called recursively enumerable), that is a set for which there exists a computable enumeration of its elements. Thus, if some element is in the set, we shall know: just wait until it is enumerated. But if an element is not in the set there is a priori no way to know.

This phenomena of partial computability may seem pityful…Yet, it is a fundamental trait which allows for a true mathematical computability theory. Let us stress this point: despite the fact that so many books are entitled "computability theory", *there is no significant theory of computable functions, i.e. of total computable functions, whereas there is a remarkable theory of partial computable functions.*

3.2 Three Wonderful Theorems in Partial Computability

We present three spectacular results. Stated for partial computable functions on the set $\mathbb{N}$ of natural integers, they have obvious counterparts with other families of finitary items: words, finite trees, finite graphs,...

First, a convenient convention: if $f : A \times B \to C$ is a partial function and $a \in A$ then f_a is the partial function $B \to C$ such that, for all $b \in B$, $f_a(b)$ is defined if and only if so is $f(a, b)$ and, if defined, $f_a(b) = f(a, b)$.

Theorem. *There exists a sequence of partial computable functions* $\varphi^{(k)} : \mathbb{N}^{k+1} \to \mathbb{N}$, $k \geq 1$, *satisfying the three following properties.*

1. **Enumeration Theorem** (Turing 1936). *The sequence* $(\varphi_e^{(k)})_{e\in\mathbb{N}}$ *is an enumeration of the family of partial computable functions* $\mathbb{N}^k \to \mathbb{N}$.
2. **Parameter Theorem (or s-m-n Theorem)** (Kleene 1943). *For all* $m, n \geq 1$, *there exists a total computable function* $s_n^m : \mathbb{N}^3 \to \mathbb{N}$ *such that*,

$$\varphi^{(n)}_{s(e,x_1,\ldots,x_m)}(y_1, \ldots, y_n) = \varphi_e^{(m+n)}(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

for all $e, x_1, \ldots, x_m, y_1, \ldots, y_n y$. *In other words, given an index* e *for a binary partial computable function* $f : \mathbb{N}^{m+n} \to \mathbb{N}$, *and a sequence of natural integers* $x_1, \ldots, x_m$, *the function* s_n^m *computes an index for the arity* n *function obtained by freezing to* $x_1, \ldots, x_m$ *the first* m *arguments of* f.
3. **Kleene Fixed Point Theorem (1938)**. *For every partial computable function* $f : \mathbb{N} \to \mathbb{N}$, *there exists* e *such that, for all* x, $\varphi_e^{(1)}(x) = \varphi^{(1)}_{f(e)}(x)$.

Thus, for every (computable) transformation of programs, there is some program such that the transformed program defines the same function as does the original program!

Comments on Universality. Observe that the partial computable function $\varphi^{(1)}$ matches the definition of an *interpreter* of programs: $\varphi^{(1)}(e, x)$ executes program e on input x. For more detailed connections between computability theory and programming see Jones (1997).

In terms of Turing machines, $\varphi^{(1)}$ denotes an universal Turing machine. Now, a natural question is: how complex is such a universal machine? The number of states is an obvious parameter to consider. But it is not an absolute one since a rich alphabet can be used to encode states. This is why the complexity of a Turing machine is measured by the pair *(number of states, number of letters)*. The best known results are[2]

$$(2; 18), (3; 9), (4; 6), (5; 5), (6; 4), (9; 3), (18; 2)$$

cf. Margenstern (2009). What are the best pairs is an open question. It is only known (cf. Kudlek 1996; Pavlotskaya 1973, 1978) that, for all n, the pairs $(1; n)$, $(2; 3)$, $(3; 2)$ and $(n; 1)$ correspond to no universal Turing machine since the halting problem for such machines is decidable.

Finally, let us stress that no enumeration satisfying both the enumeration and the parameter theorems can be injective: together, these two results imply that the set $\{e \mid f = \varphi_e^{(1)}\}$ is infinite for every partial partial computable function. However, a difficult result, due to Friedberh (1958), insures that there exists injective enumerations satisfying the sole enumeration theorem. Failing the parameter theorem, such injective enumerations are not used.

Comments on the Parameter Theorem. This result has a vast number of applications. It is the cornerstone of partial evaluation since the function s freezes one input of program e. It is also used for Futamura–Ershov–Turchin construction of a compiler from an interpreter by specializing it to the program which is to be compiled, cf. again Jones' works (1997).

The parameter theorem states that input and program are somewhat interchangeable hence gives the form of a proved theorem to John von Neumann's original idea about computers: treat similarly input and program.

Comments on Kleene Fixed Point Theorem. Of course, for many transformations f the fixed point program corresponds to the nowhere defined function! For instance, if f modifies the program so that the new output be obtained by incrementing the original output. Nevertheless, this is not always the case and this theorem is, indeed, one of the most powerful tools of computability theory. It has a lot of consequences, in particular to prove undecidabilty results. Smullyan (1993, 1994) thoroughly detailed its relations with logic with his usual brio and humor.

[2] The machine (2; 3) announced in 2007 by Alex Smith, cf. Wolfram prize, considers a notion of universality different from the usual one.

As a pleasant example of application, a simple use of this theorem explicits a program with no input which outputs its own code (such programs are called "quines")!

Kleene fixed point theorem also allowed for a theory of virology which started with Cohen's thesis (1986) and his thesis advisor's paper (Adleman 1988). The exact role of this theorem in virology has been clarified in Marion (2012).

Finally, let us mention that Kleene fixed point theorem is also a basic tool in learning theory (Jain et al. 1999).

3.3 Two "Negative" Results in Partial Computability

Next to these three "positive" theorems, let us look at two "negative" ones.

Undecidability of the Halting Problem. *The domain of the universal partial computable function* $\varphi^{(1)} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *is not computable. Otherwise said, there is no algorithm which, for every* (e, x)*, decides if* $\varphi^{(1)}$ *halts or not on input* (e, x).

Of course, this theorem can be viewed as anti-unemployment: to answer instances of the halting problem, researchers are needed, no way to replace them by some program...

The proof of this result is so simple and nice that we do not resist the temptation to give it. Consider the partial function $f : \mathbb{N} \to \mathbb{N}$ such that $f(x) = 1$ if $\varphi_x^{(1)}(x)$ is not defined and $f(x)$ is undefined if $\varphi_x^{(1)}(x)$ is defined. If the domain of $\varphi^{(1)}$ were computable then f would be partial computable and there would exist e such that f coincides with $\varphi_e^{(1)}$. Let us look at $f(e)$. The definition of f insures that $f(e)$ is defined if and only if $\varphi_e^{(1)}(e)$ is not defined if and only if $f(e)$ is not defined, contradiction! This proves that the domain of $\varphi^{(1)}$ is not computable.

This proof uses a diagonalisation construction: considering $\varphi_x^{(1)}(x)$ means that we identify both arguments of $\varphi^{(1)}$, i.e. we consider a same value for the program and the input. This method was pioneered by Cantor to show that the set of real numbers is uncountable. Another proof of the undecidability of the halting problem can be given using a quantitative argument based on the Richard paradox and the busy beaver function (Rado 1962).

Rice Theorem (1954). *Let* $\mathscr{PC}$ *be the family of partial computable functions* $\mathbb{N} \to \mathbb{N}$. *If* $\mathscr{F} \subset \mathscr{PC}$ *and* $\emptyset \neq \mathscr{F} \neq \mathscr{PC}$ *then the set* $\{e \mid \varphi_e^{(1)} \in \mathscr{F}\}$ *of indexes of partial functions in* $\mathscr{F}$ *is not computable.*

Restated in programming terms, this theorem insures that *there is no nontrivial question about the semantics of programs which can be algorithmically decided.* A result which has fundamental consequences in automatic deduction and program verification and can be seen as a stumbling block for these topics.

Fortunately, this theorem does not forbid the existence of computable subsets X of $\{e \mid \varphi_e^{(1)} \in \mathscr{F}\}$ such that $\mathscr{F} = \{\varphi_e^{(1)} \mid e \in X\}$, i.e. all elements of X are indexes of functions of $\mathscr{F}$ and, though the set X does not contain all indexes of functions in $\mathscr{F}$, it contains at least one index for each function of $\mathscr{F}$. Indeed, finding such sets X is an important research theme in automatic deduction and program verification.

4 Formalization of the Notion of Algorithm

4.1 Denotational versus Operational

What is an Algorithm? Around 1936 the intuitive notion of computability has been given a mathematical formalization. Still, the existence of a mathematical notion of algorithm remained an open question. As stated by Turing (1936):

> What are the possible processes which can be carried out in computing […]?

Indeed, the numerous results supporting Church thesis only show that there are classes of processes, i.e. classes of algorithms, which admit mathematical formalizations and suffice to get all partial computable functions: those classes associated to Turing machines, to RAMs, to programs in the language C,. . .. Now, these classes are clearly distinct. For instance, a Program à la Kleene with `FOR` and `WHILE` loops does not by far act in the same way than any RAM or cellular automaton, it can merely be simulated by such machines so as to get the same input-output function.

Thus, admitting Church thesis, the diverse simulations between the main models of computations show *denotational completeness* of all these models. But the following question remains open: is there a computation model for which the associated class of algorithm contains—up to isomorphism—all those classes associated to the diverse models of computation? In other words, is there an *operationally complete* computation model?

Kolmogorov and Uspensky Machines and Schönhage Machines. Kolmogorov was the first to tackle the problem in Kolmogorov (1953), Kolmogorov and Uspensky (1958). He introduced a powerful extension of the notion of Turing machine, now called Kolmogorov–Uspensky machines (cf. Sect. 2.2). These machines were further extended by Schönhage (cf. Sect. 2.2).

Kolmogorov and Schönhage lacked a mathematical framework to prove or disprove operational completeness for their machines. Such a framework was built some years later by Gurevich, cf. Sect. 4.2 and allowed him to show in Dexter et al. (1997) that Kolmogorov–Uspensky machines and Schönhage machines fail to be operationally complete (cf. Sect. 2.2).

4.2 Getting Operational Completeness

Abstract State Machines. Around 1984, Gurevich (1988, 2000), introduced a radically new notion of machine: "Abstract State Machines" (ASM for short), originally called "Evolving algebras".

Roughly speaking, an ASM is an algebra which evolves over time according to some program. Formally, an ASM consists of one or several domains endowed with functions (relations being identified to Boolean valued functions):

- *Static part.* The domain(s) and some functions—called static— do not vary over time. The static functions are analogous to primitive operations and libraries for a programming language.
- *Dynamic part.* Some functions—called dynamic—may vary over time. They correspond to the environment of a program, by nature dynamic. Of course, the initial values of the dynamic functions are also constituents of the ASM.
- *Program.* Time is discrete and the variation in one unit of time of the dynamic functions obeys a "test and set" program consisting of affectations and conditional tests structured in finite blocks of instructions executed in parallel.

Observe that the static part of an ASM is arbitrary. Indeed, with his model Gurevich stresses an important feature of the notion of algorithm: it is intrinsically oracular! More precisely,

> No algorithm describes from scratch the entire computing process. It necessarily relies on some primitive operations for which it gives no indication on how they are to be computed.

For instance, the program of a Turing machine tells *what* is to be done (move the head right or left, write a particular symbol in the scanned cell, enter a particular new state) but it does not tell *how* these operations are to be done. Though these operations seem extremely elementary, their implementation may be nontrivial. This is the case if you try to emulate this Turing machine by a program in some assembly language. In order to simulate the read/write and change state processes you then need to write a lot of dedicated lines of codes to modify the data encoding the state, the position of the head and the tape contents.

The ASM model is extremely flexible. Choosing carefully the domain(s) and the static and dynamic parts, one can simulate "step by step" every known computation model which runs in sequential time and does a bounded amount of work at each step, i.e. one step of such a computation model is simulated by exactly one step of the ASM. Cf. the rich literature on the subject cited on Gurevich web page. This simulation can also be made an isomorphism, cf. Grigorieff and Valarcher (2010).

Operational Completeness of ASMs. These "step by step" simulation results support *Gurevich Thesis* (also called *Small steps sequential computation thesis*), stated in Gurevich very first paper on this subject (Gurevich 1985):

> Every sequential time computation process which does a bounded amount of work at each step can be simulated step by step by some ASM.

In other words, the thesis asserts that ASMs are operationally complete relative to sequential time small step algorithms.

What about algorithms with cellular automata for which an unbounded amount of work can be done at each step? What about non sequential time algorithms such as the distributed ones? Let us mention that Gurevich extended the ASM model and stated associated thesis to the much more delicate notions of parallel and concurrent algorithms (Blass and Gurevich 2003, 2008).

Operational Completeness of Hyper-KU and Hyper-Schönhage Machines. Dexter et al. (1997) show that ASMs can be simulated "step by step" by generalized

Kolmogorov–Uspensky and by generalized Schönhage machines which therefore constitute operationally complete computation models. These generalized machines are obtained as follows: replace the graph tape by an hypergraph tape with rank 3. In the undirected case of hyper-KU machines, edges become hyperedges: each hyperedge is a set of three nodes. In the directed case of hyper-Schönhage machines, directed arcs become directed hyperarcs: each hyperarc is a list of three nodes.

Operational Completeness of Lambda-Calculus. Ferbus and Grigorieff (2010) show that every ASM can be simulated in lambda calculus in the following way: there is a fixed integer k, depending only on the considered ASM and not on the inputs, such that

- the current state of the ASM is coded by a particular lambda term,
- the lambda term corresponding to the next state is obtained via k successive reductions of the lambda term corresponding to the given state.

Relaxing the "step by step" requirement to "bounded number of steps for one step", we see that lambda calculus is a sequential time computation model which is as rich as any sequential time small-step computation model. Of course, lambda calculus is not a small-step computation model since some reductions modify arbitrarily many subterms in a lambda term. Thus, lambda calculus is *operationally hard* for sequential time small-step algorithms.

Moschovakis' Approach. Another approach of operational completeness has been developed by Moschovakis (2001) based on definition via smallest fixed point. This approach is a more abstract modelization of the notion of algorithm.

4.3 Gurevich's Axiomatization of the Notion of Algorithm

Gurevich (2000) introduces three axioms for the notion of discrete time small-step algorithm:

i. *An algorithm is a state transition system.*
ii. *Its states are structures which all have the same domain(s) and are relative to the same given finite family of function symbols (constant being seen as arity zero functions).*
iii. *A state and its successor state differ only by the interpretations of its function symbols on finitely many points given by a fixed family of terms constructed with the function symbols.*

This axiomatization together with its justification are intimately related to ASM theory. From these axioms, Gurevich formally proves his *small steps sequential computation thesis.*

Getting Church Thesis from Gurevich Thesis. Consider the following fourth axiom

iv. *In initial states all functions are intuitively computable functions.*

Dershowitz and Gurevich (2008) (cf. also Boker and Dershowitz 2008) prove Church thesis for every process which satisfies the four axioms.

The interest of this axiomatization is that it is generic, formal, and based on commonly agreed principles. Also, it provably reduces any doubt about Church thesis to a doubt about one of these elementary principles.

Let us mention that these results have been extended to the notion of parallel algorithm (Blass and Gurevich 2003) and have been related to the notion of quantum algorithm (Grädel and Nowack 2003).

4.4 Operational Completeness and Recursion

We considered in Sect. 2.1 the class of primitive recursive functions obtained with the `FOR` loop and we mentioned that this class is much smaller than the class of computable functions which is obtained using the `WHILE` loop. Going further, Colson (1991) noticed that many usual algorithms to compute a given primitive recursive function use processes which are not relevant to the `FOR` loop hence are not obtained from primitive recursive definitions of that function. Colson then raised the following question:

> To compute a primitive recursive function, are the algorithms issued from the primitive recursive definitions of this function as efficient as those which use any computable process?

Colson gave a negative answer with the strikingly simple primitive recursive function $(x, y) \mapsto \min(x, y)$ which gives the minimum of two nonnegative integers. Indeed, sticking to primitive recursion, a simple definition of the function $\min(x, y)$ goes through definitions of the *if then else*, *zero*, *predecessor* and *subtraction* in $\mathbb{N}$:

$$\begin{aligned} \mathrm{IF}(0, y, z) &= z & \mathrm{IF}(x+1, y, z) &= y \\ Z(0) &= 1 & Z(x+1) &= 0 \\ \mathrm{Pred}(0) &= 0 & \mathrm{Pred}(x+1) &= x \\ \mathrm{Sub}(x, 0) &= x & \mathrm{Sub}(x, y+1) &= \mathrm{Pred}(\mathrm{Sub}(x, y)) \\ \min(x, y) &= \mathrm{IF}(Z(\mathrm{Sub}(x, y)), x, y) \end{aligned}$$

A much simpler definition uses a double recursion (which is not allowed in primitive recursive definitions):

$$\begin{cases} \min(x, 0) = 0 & (*)_1 \\ \min(0, y+1) = 0 & (*)_2 \\ \min(x+1, y+1) = \min(x, y) + 1 & (*)_3 \end{cases}.$$

The algorithm associated to equations (*) clearly runs in time $O(\min(x, y))$, i.e. proportional to $\min(x, y)$. Colson proves that *every algorithm associated to primitive recursive definitions halts either in time $O(x)$ or in time $O(y)$* hence cannot match the algorithm associated to equations $(*)_1 - (*)_3$.

Colson also observes that an extension of primitive recursion to higher order functions (cf. Gödel 1958) allows to get an algorithm running in time $O(\min(x, y))$. Indeed, let C, M denote functionals of types $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ and $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$. To

the following higher order primitive recursive equations is associated an algorithm which faithfully emulates the one associated to equations (*) hence runs in time $O(\min(x, y))$:

$$\begin{array}{rlrl} C(f)(0) & \stackrel{(a)}{=} 0 & M(0) & \stackrel{(c)}{=} \lambda y \cdot 0 \\ C(f)(y+1) & \stackrel{(b)}{=} f(y)+1 & M(x+1) & \stackrel{(d)}{=} C(M(x)) \end{array} \qquad \min(x, y) \stackrel{(e)}{=} M(x)(y)$$

To illustrate this algorithm, we detail the computation of $\min(3, 7)$. Each successive application of equations (d) and (b) mimics the simultaneous decrementation of both arguments given by equation $(*)_3$ above:

$$\begin{aligned} \min(3,7) &\xrightarrow{(e)} \boxed{\mathrm{M}(3)(7)} \xrightarrow{(d)} C(M(2))(7) \xrightarrow{(b)} \boxed{\mathrm{M}(2)(6)}+1 \xrightarrow{(d)} C(M(1))(6)+1 \\ &\xrightarrow{(b)} \boxed{\mathrm{M}(1)(5)}+2 \xrightarrow{(d)} C(M(0))(5)+2 \xrightarrow{(b)} \boxed{\mathrm{M}(0)(4)}+3 \\ &\xrightarrow{(c)} (\lambda y \cdot 0)(4)+3 \longrightarrow 0+3 \longrightarrow 3 \end{aligned}$$

This result has been extended to other families of algorithms, cf. Fredholm (1996); Colson and Fredholm (1998), and to other very simple functions such as the greatest common divisor, cf. Moschovakis (2006) and van den Dries (2003).

5 Deduction and Computation: The Algorithmic Nature of Constructive (i.e. Intuitionistic) Proofs

5.1 Constructive Proofs as a Programming Language

One of the most astonishing results in theoretical computer science and programming is the deep relation that exists between deduction and computation. This relation was first discovered for "constructive proofs" (also called "intuitionistic proofs").
Note. We prefer to use the word "constructive" which stresses an operational feature hence an algorithmic character whereas the word "intuitionistic" is relevant to the philosophical vocabulary.

The Witness Property for Constructive Proofs. This is the simplest illustration of the algorithmic nature of proofs:

> From any constructive proof of a formula of the form $\exists y\, A$, it is possible to computably extract a term u and a constructive proof of the formula $A(u/y)$ obtained from A by substituting the term u to all free occurrences of the variable y.

Similarly, from a constructive proof of a formula asserting the existence of a function, that is a formula of the form $\forall x \exists y\, A$, one can computably extract a program computing such a function:

> Given a term t (the input of the program) and a constructive proof of the formula $\forall x \exists y\ A$, one can computably find a constructive proof of the formula $\exists y\ A(t/x)$ hence also a term u (the output of the program) and a constructive proof of the formula $A(t/x, u/y)$.

Thus, the *language of constructive proofs* is a programming language and the witness extraction mechanism, called *proof reduction*, is the execution mechanism of this programming language.

A Programming Language with Proved Specification. Compared to usual programming languages, the very great benefit with this programming language of constructive proofs is that, in addition to the output u, the extraction process gives a proof of the formula $A(t/x, u/y)$, i.e. a proof that t and u satisfy the given specification A. This is clearly a powerful tool in artificial intelligence for provable specification (Kanovich and Vauzeilles 2007).

A Programming Language in which Every Program Halts. By definition, a program in this language is a proof of some formula $\forall x \exists y\ A$ and, as explained above, for every input t, the program halts and yields an output u satisfying $A(t/x, u/y)$.

Of course, such a property forbids Turing completeness: some computable (total) functions cannot be programmed in this language. Also, even if a computable function can be programmed, the efficiency of some algorithms which compute it may not be matched by any such program.

Nevertheless, this language of constructive proofs (especially in its extended versions with provability in particular theories, cf. Sect. 5.6) expresses very rich classes of computable functions and afferent algorithms. This contrasts with the case of other programming languages in which every program halts, for instance the language of primitive recursion with the sole `FOR` loop, cf. Sect. 2.1.

The reason for this richness is that the language of constructive proofs in a given theory allows to program all functions which are constructively provably total in this theory since any proof of the total character is by itself a program to compute this function in the language of constructive proofs…

For further information on proof languages as programming languages, see the paper by Paulin-Mohring and Werner (1993).

5.2 Formal Logical Systems

A formal logical system consists of the two following items.

- *Morphology.* A vocabulary consisting of constant symbols, function symbols and predicate symbols. With this vocabulary, augmented with an infinite sequence of variables, one constructs terms and formulas.
- *Provability.* A family of axioms and rules to get proofs of some formulas, called theorems of the logical system. Proofs can have diverse forms: lists of formulas, trees of formulas, etc., but it is always required that one can computably check if something is a proof or not.

Usual logical systems have also a particular semantics attached to them: a notion of structure and an afferent notion of satisfaction for formulas. A formula is said to be valid if it is satisfied in every structure. A famous result, show by Kurt Gödel in 1930, insures that, for the so-called "classical" (nonconstructible, i.e. not intuitionistic) logical systems,

> A formula is valid if and only if it is provable.

An analog result (Kripke 1965) also holds for constructive logical systems with respect to an adequate semantics.

For further information on classical and on constructive logical systems, see the books (Cori and Lascar 1983a, b; van Dalen 2008).

Provability in classical logic is still often presented "to the Hilbert" via a list of axioms and the "modus ponens" rule. However, since Gerhard Gentzen's works in 1934, it is known that provability can be presented in very different ways. For instance, sequent calculus or natural deduction have several rules associated to the diverse connectors and quantifiers, and constitute provability systems with far better properties (such as cut elimination).

Also, looking at refutation rather than at provability, the semantic tableaux is an approach much used in automatic deduction.

Linear logic, developed by Girard (1987), is another example. Intuitively, multiplicity counts: $A \wedge A$ is not equivalent to A, i.e. to have two identical resources is not the same as having only one. Technically, this leads to forbid the usual structural contraction and weakening rules in sequent calculus. Also, proofs nets generalize natural deduction in the framework of linear logic. The short book (Girard et al. 1989) is a very clear introduction to this topic.

5.3 Constructive Proofs and Their Denotations

5.3.1 Constructive (i.e. Intuitionistic) Natural Deduction

Constructive (i.e. intuitionistic) natural deduction is a provability system introduced by Gerhard Gentzen (1934) (see also Prawitz 1965). As suggested by its name, this system follows closely the natural way to get proofs: the way mathematicians get their proofs. In particular, to prove a formula of the form $A \Rightarrow B$, a mathematician assumes A and tries to prove B using this hypothesis. This leads to an important turn: instead of defining the family of provable formulas, we define the family of pairs $((A_1, ..., A_n), B)$ where $A_1, ..., A_n$ are formulas considered as hypothesis (called the context) and B is a formula provable from these hypotheses. Such a pair is called a "sequent" and denoted $A_1, ..., A_n \vdash B$.

Axioms and rules of natural deduction are given in Table 1 (where $\bot$ and $\top$ respectively represents "false" and "true").

Table 1 Axioms and rules in constructive (i.e. intuitionistic) natural deduction

$$\frac{}{A_1, ..., A_n \vdash A_i}\ \text{axiom}$$

$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-elim}$	$\dfrac{}{\Gamma \vdash \top}\ \top\text{-intro}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim}_r$ $\quad$ $\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim}_l$	$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro}$
$\dfrac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\text{-elim}$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-intro}_r$ $\quad$ $\dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-intro}_l$
$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash A \Rightarrow B}{\Gamma \vdash B}\ \Rightarrow\text{-elim}$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-intro}$
$\dfrac{\Gamma \vdash \forall x\ A}{\Gamma \vdash A(t/x)}\ \forall\text{-elim}$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash \forall x\ A}\ \forall\text{-intro}$ (if x is not free in Γ)
$\dfrac{\Gamma \vdash \exists x\ A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}\ \exists\text{-elim}$ (if x is not free in Γ, B)	$\dfrac{\Gamma \vdash A(t/x)}{\Gamma \vdash \exists x\ A}\ \exists\text{-intro}$

Note. In Table 1 axioms are represented as rules with no premiss. Also, letters A, B, C denote arbitrary formulas and t denotes an arbitrary term. Thus, each axiom or rule is, in fact, an infinite list of axioms or rules.

Observe that the introduction rule for the connector $\Rightarrow$ follows what we said above on the way mathematicians are proving implications.

A careful examination of the rules in Table 1 shows that each one of them corresponds to a very usual and natural way to argue. As for the axioms, they are rather trivial since they are sequents the conclusion of which also explicitly appears in the context. . .

The reader may wonder why there is no rule for negation. In fact, negation $\neg A$ is identified to the formula $A \Rightarrow \bot$, i.e. the formula which expresses that A implies falsity.

In natural deduction a proof is a tree with nodes labelled by sequents so that (i) leaves are labelled by axiom sequents, (ii) each inner node with N sons (with $N = 1, 2, 3$) is labelled by the conclusion sequent of a rule with N premiss sequents which are the labels of the N sons of the node.

Note. We graphically represent such a tree with its root at the bottom

Provable sequents are those which label some node in some proof.

Observe that each tree satisfying condition (ii) above can be seen as a derived rule the premisses of which are all leaves which are not axioms. Adding such derived rules does not increase the family of provable sequents.

Conventions for the representation of proof trees. The root is at the bottom and the sons of a node lie just above it (as opposed to the usual way in computer science where the root is at the top and sons lie below their father). For clarity, we also superscript each node with a line which runs below the N sons of the node and is marked with the name of the rule involved to go from the N son-nodes to their father-node.

Here is an example of a proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \Rightarrow (q \Rightarrow r), q, p \vdash p \Rightarrow (q \Rightarrow r)}\,\mathrm{ax} \qquad \dfrac{}{p \Rightarrow (q \Rightarrow r), q, p \vdash p}\,\mathrm{ax}}{p \Rightarrow (q \Rightarrow r), q, p \vdash q \Rightarrow r}\Rightarrow\!-\mathrm{e} \qquad \dfrac{}{p \Rightarrow (q \Rightarrow r), q, p \vdash q}\,\mathrm{ax}}{p \Rightarrow (q \Rightarrow r), q, p \vdash r}\Rightarrow\!-\mathrm{e}}{p \Rightarrow (q \Rightarrow r), q \vdash p \Rightarrow r}\Rightarrow\!-\mathrm{i}}{p \Rightarrow (q \Rightarrow r) \vdash q \Rightarrow (p \Rightarrow r)}\Rightarrow\!-\mathrm{i}$$

The usual sequentless framework for provability can easily be recovered:

- a proof of a formula A is a proof of the sequent $\varnothing \vdash A$ which has empty context,
- a proof of a formula A in an axiomatic theory $\mathscr{T}$ is a proof of some sequent $\Gamma \vdash A$ with context Γ included in $\mathscr{T}$.

Remark. As we are in constructive logic, some "classical" properties are not provable. For instance, the following three well-known tautologies of "classical" logic have no constructive proof. The first two ones are formulas $p \vee \neg p$ and $\neg\neg p \Rightarrow p$ written with $\bot$ instead of $\neg$, whereas the last one is Pierce formula (best understood by instantiating q to $\bot$, which is $(\neg p \Rightarrow p) \Rightarrow p$):

$$p \vee (p \Rightarrow \bot), \qquad ((p \Rightarrow \bot) \Rightarrow \bot) \Rightarrow p, \qquad ((p \Rightarrow q) \Rightarrow p) \Rightarrow p.$$

5.3.2 Contexts

Labelling nodes by sequents leads to rather cumbersome proof trees since contexts are lists of formulas which are repeated from node to node. Indeed, the $\Rightarrow$-intro and $\vee$-elim rules are the sole ones in which the contexts vary and these variations are rather modest ones. Clearly, a lighter representation would be welcome.

First, we can restrict to proof trees such that every formula A occurring in the context of some axiom sequent either appears as the conclusion formula of that sequent or is a mobile formula in a $\Rightarrow$-intro or $\vee$-elim rule used in the proof. Otherwise the proof tree can be simplified by removing A from all contexts.

We now describe a lighter representation of proof trees. Forgetting all sequents and keeping the sole names of the rules which are used gives a very light tree. But this tree does not allow to recover the original proof tree: we need to know which

formulas are involved in each application of a rule. These two simple observations lead to the following light representation.

- First, one introduces a name α_i for each formula A_i occurring in some axiom used in the proof. This gives the *named context* $(\alpha_1{:}A_1, ..., \alpha_n{:}A_n)$.
- If an axiom of the form $B_1, \ldots, B_k, \ldots, B_n \vdash B_k$ is used in the proof, it suffices to mention the name of hypothesis B_k.
  For instance, with the named context $(\alpha{:}P \Rightarrow Q, \beta{:}P)$, the proof

$$\dfrac{\dfrac{}{\alpha{:}P \Rightarrow Q,\ \beta{:}P \vdash P \Rightarrow Q}\ \text{axiom} \qquad \dfrac{}{\alpha{:}P \Rightarrow Q,\ \beta{:}P \vdash P}\ \text{axiom}}{\alpha{:}P \Rightarrow Q,\ \beta{:}P \vdash Q}\ \Rightarrow -\text{elim}$$

  can be expressed by the tree corresponding to the term $\Rightarrow -\text{elim}(\alpha, \beta)$.
- For the rules which modify the context (namely, $\Rightarrow$-intro and $\vee-$elim), it is necessary to explicit and name the mobile hypothesis. For instance, with the named context $(\alpha{:}P, \beta{:}P \Rightarrow Q)$, the proof

$$\dfrac{\dfrac{\dfrac{}{\alpha{:}P,\ \beta{:}P \Rightarrow Q \vdash P \Rightarrow Q}\ \text{axiom} \qquad \dfrac{}{\alpha{:}P,\ \beta{:}P \Rightarrow Q \vdash P}\ \text{axiom}}{\alpha{:}P,\ \beta{:}P \Rightarrow Q \vdash Q}\ \Rightarrow -\text{elim}}{\alpha{:}P \vdash (P \Rightarrow Q) \Rightarrow Q}\ \Rightarrow -\text{intro}$$

  cannot be simply rewritten as $\Rightarrow -intro(\Rightarrow -elim(\beta, \alpha))$ since, in order to recover the context $(\alpha{:}P,\ \beta{:}P \Rightarrow Q)$ of the son of the root, we need to tell that the mobile hypothesis is $P \Rightarrow Q$ with name β. Thus, we rewrite the proof tree as the following term: $\Rightarrow -intro(\beta, \Rightarrow -elim(\beta, \alpha))$.

Observe the similarity between names of hypothesis and variables. For instance, in the proof associated to the term $\Rightarrow -intro(\beta, \pi)$, the name β can only name a hypothesis occurring in the subproof associated to the term π and cannot name a hypothesis occurring in the context of the root of the original proof tree. Thus, one can say that the subproof π is in the scope of the variable β and that this variable is bound by the symbol $\Rightarrow$-intro.

5.3.3 Terms Associated to Constructive Proofs

Now, we can attach terms to proofs in natural deduction. For each rule we introduce a function symbol which will be a name for the rule. For instance, the name of the $\Rightarrow$-intro rule is λ. The arity of this function symbol is the number of premisses of the rule augmented with the number of parameters needed to recover the rule from its schema. For each one of its arguments, this function symbol relates the hypothesis which are added to the associated premiss. The formal inductive construction of the term associated to a proof is described in Table 2 (the reader will find in Sect. 5.5 the raison d'être for the chosen symbols).

Table 2 Terms associated to proofs in constructive natural deduction

$$\alpha_i \left\{ \dfrac{}{\Gamma \vdash A_i}\ \text{axiome} \right. \qquad I \left\{ \dfrac{}{\Gamma \vdash \top}\ \top\text{-intro} \right. \qquad \pi \left\{ \dfrac{\vdots}{\Gamma \vdash \bot} \right. \longmapsto \delta_\bot(A,\pi) \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-elim}} \right.$$

$$\pi_1 \left\{ \dfrac{\vdots}{\Gamma \vdash A},\ \pi_2 \left\{ \dfrac{\vdots}{\Gamma \vdash B} \right. \right. \longmapsto \langle \pi_1, \pi_2 \, rangle e \left\{ \dfrac{\dfrac{\vdots}{\Gamma \vdash A} \quad \dfrac{\vdots}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ \wedge\text{-i} \right. \qquad \pi \left\{ \dfrac{\vdots}{\Gamma \vdash A \wedge B} \right. \longmapsto fst(\pi) \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-e}},\ snd(\pi) \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-e}} \right. \right. \right.$$

$$\pi \left\{ \dfrac{\vdots}{\Gamma \vdash A} \right. \longmapsto i(A,B,\pi) \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-i}} \right. \qquad \pi_1 \left\{ \dfrac{\vdots}{\Gamma \vdash A \vee B},\ \pi_2 \left\{ \dfrac{\vdots}{\Gamma, \alpha{:}A \vdash C},\ \pi_3 \left\{ \dfrac{\vdots}{\Gamma, \beta{:}B \vdash C} \right. \right. \right. \longmapsto$$

$$\pi \left\{ \dfrac{\vdots}{\Gamma \vdash B} \right. \longmapsto j(A,B,\pi) \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-i}} \right. \qquad \delta(\pi_1, \alpha{:}A\, \pi_2, \beta{:}B\, \pi_3) \left\{ \dfrac{\dfrac{\vdots}{\Gamma \vdash A \vee B} \quad \dfrac{\vdots}{\Gamma, \alpha{:}A \vdash C} \quad \dfrac{\vdots}{\Gamma, \beta{:}B \vdash C}}{\Gamma \vdash C}\ \vee\text{-e} \right.$$

$$\pi \left\{ \dfrac{\vdots}{\Gamma, \alpha{:}A \vdash B} \right. \longmapsto \lambda\alpha{:}A\, \pi \left\{ \dfrac{\vdots}{\dfrac{\Gamma, \alpha{:}A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-i}} \right. \qquad \pi_1 \left\{ \dfrac{\vdots}{\Gamma \vdash A \Rightarrow B},\ \pi_2 \left\{ \dfrac{\vdots}{\Gamma \vdash A} \right. \right. \longmapsto app(\pi_1, \pi_2) \left\{ \dfrac{\vdots}{\dfrac{\Gamma, A \vdash B} \quad \dfrac{\vdots}{\Gamma \vdash A}}{\Gamma \vdash B}\ \Rightarrow\text{-e} \right.$$

$$\pi \left\{ \dfrac{\vdots}{\Gamma \vdash A} \right. \longmapsto \lambda x\, \pi \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash B}{\Gamma \vdash \forall x\, A}\ \forall\text{-i}} \right. \qquad \pi \left\{ \dfrac{\vdots}{\Gamma \vdash \forall x\, A} \right. \longmapsto app(\pi, t) \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash \forall x\, A}{\Gamma \vdash A(t/x)}\ \forall\text{-e}} \right.$$

$$\pi \left\{ \dfrac{\vdots}{\Gamma \vdash A(t/x)} \right. \longmapsto \langle t, \pi \, rangle e \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash A(t/x)}{\Gamma \vdash \exists x\, A}\ \exists\text{-i}} \right. \qquad \pi_1 \left\{ \dfrac{\vdots}{\Gamma \vdash \exists x\, A},\ \pi_2 \left\{ \dfrac{\vdots}{\Gamma, \alpha{:}A \vdash B} \right. \right. \longmapsto$$

$$\delta_\exists(\pi_1, x\alpha{:}A\, \pi_2) \left\{ \dfrac{\vdots}{\dfrac{\Gamma \vdash \exists x\, A} \quad \dfrac{\vdots}{\Gamma, \alpha{:}A \vdash B}}{\Gamma \vdash B}\ \exists\text{-e} \right.$$

5.4 Reduction of Constructive Proofs

5.4.1 Cuts

A *cut* is a proof ending with an elimination rule whose main premiss is proved using an introduction rule on the same connective or quantifier.

A very simple example is as follows:

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash A} \quad \dfrac{\pi_2}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro}}{\Gamma \vdash A}\ \wedge\text{-elim}$$

Such a proof can obviously be simplified: keep the sole proof π_1 which leads to the same root sequent $\Gamma \vdash A$. Using the term notation, this proof with a cut is written $fst(\langle \pi_1, \pi_2 rangegle)$ and the reduction rule for proofs is

$$\text{fst}(\langle \pi_1, \pi_2 rangegle) \longrightarrow \pi_1$$

We now look at a more complex example of cut:

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma, \alpha : A \vdash B}}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-intro} \quad \dfrac{\pi_2}{\Gamma \vdash A}}{\Gamma \vdash B}\ \Rightarrow\text{-elim}$$

with associated term $((\lambda\alpha{:}A\ \pi_1)\ \pi_2)$. The reduction of this proof is done as follows:

- remove the hypothesis A in all sequents of the proof π_1,
- replace by the proof π_2 each axiom which introduces A in its conclusion.

We see that this reduction is a simple substitution: in the proof π_1 substitute proof π_2 to each occurrence of the variable α associated to the hypothesis A in the context Γ, A. Thus, this reduction rule is the usual β-reduction of λ-calculus (Sect. 2.6):

$$app((\lambda\alpha{:}A\ \pi_1),\ \pi_2) \longrightarrow \pi_1(\pi_2/\alpha)$$

All other rules are constructed in a similar way. Table 3 gives the set of reduction rules for proofs.

Applying these reduction rules, any cut in a proof can be removed. Problem: such a cut elimination can create new cuts!, Fortunately, as insured by the main theorem of this topic, *this process of reduction of proofs halts, giving a cut-free proof.*

Remark. The family of terms associated to proofs in natural deduction endowed with the above reduction rules constitutes an extension of the lambda-calculus sketched in Sect. 2.6. Indeed, lambda-calculus corresponds to terms associated to proofs in

Table 3 Reduction rules for constructive proofs

$$
\begin{aligned}
fst(\langle \pi_1, \pi_2 rangegle) &\longrightarrow \pi_1 \\
snd(\langle \pi_1, \pi_2 rangegle) &\longrightarrow \pi_2 \\
\delta(i(A,B,\pi_1), \alpha{:}A\ \pi_2, \beta{:}B\ \pi_3) &\longrightarrow \pi_2(\pi_1/\alpha) \\
\delta(j(A,B,\pi_1), \alpha{:}A\ \pi_2, \beta{:}B\ \pi_3) &\longrightarrow \pi_3(\pi_1/\beta) \\
app((\lambda\alpha{:}A\ \pi_1), \pi_2) &\longrightarrow \pi_1(\pi_2/\alpha) \\
app((\lambda x\ \pi), t) &\longrightarrow \pi(t/x) \\
\delta_\exists(\langle t, \pi_1 rangegle, x\ \alpha{:}A\ \pi_2) &\longrightarrow \pi_2(t/x, \pi_1/\alpha)
\end{aligned}
$$

which all formulas are built with the sole connector $\Rightarrow$. And the unique afferent rule is β-reduction, cf. the fifth line of Table 3.

5.4.2 Cut Elimination and Witness Extraction

The proof reduction mechanism is exactly the one which allows to extract witnesses from proofs (cf. Sect. 5.1). Indeed, a simple induction on the structure of proofs shows that if a constructive proof of a sequent with empty context is cut-free then this proof ends with an introduction rule. In particular, if π is a proof of a formula of the form $\exists y\ A$ then π ends with some $\exists$-intro rule hence is of the form $\langle u, \pi' rangegle$ where π' is a proof of a formula of the form $A(u/y)$. Thus, the term u is the wanted witness.

5.5 Brouwer–Heyting–Kolmogorov Interpretation

As seen above, due to the witness property, proofs can be seen as a programming language. Also, there is a striking similarity between this language and usual functional programming languages. This similarity does not come from the witness property but from a very different reason: the so-called Brouwer–Heyting–Kolmogorov (for short, BHK) interpretation, that we now detail.

To get a proof of $A \wedge B$ with the $\wedge$-intro rule, one has to get a proof of A and a proof of B. Also, given a proof of $A \wedge B$, one can use it with the $\wedge$-elim rules to get a proof of A or a proof of B. This is much similar to the pair function in set theory. To get a pair constituted with a proof of A and a proof of B one has first to get these two proofs. Also, given a pair constituted with a proof of A and a proof of B, one can use it to get a proof of A or a proof of B. Thus, a proof of $A \wedge B$ is obtained and is used exactly as a pair constituted with a proof of A and a proof of B.

More generally,

- A proof of $\top$ is the element of a singleton set $\{I\}$.
- There is no proof of $\bot$.
- A proof of $A \wedge B$ is a pair constituted with a proof of A and a proof of B.
- A proof of $A \vee B$ is a proof of A or a proof of B.
- A proof of $A \Rightarrow B$ is an algorithm which associates to any proof of A a proof of B.
- A proof of $\forall x\ A$ is an algorithm which associates to any term t a proof of $A(t/x)$.
- A proof of $\exists x\ A$ is a pair constituted with a term t and a proof of $A(t/x)$.

These connections justify the names given to the terms associated to proofs. If π_1 is a proof of A and π_2 is a proof of B, it is natural to denote by $\langle \pi_1, \pi_2 rangegle$ the proof of $A \wedge B$ built with the $\wedge$-intro rule since this proof is the pair constituted with a proof of A and a proof of B.

Also, if π is a proof of B which uses a hypothesis A represented by a free variable α, it is natural to denote by $\lambda\alpha{:}A\ \pi$ the proof of $A \Rightarrow B$ built with the $\Rightarrow$-intro rule since this proof is an algorithm which associates to any proof of A a proof of B.

Moreover, as noticed by Curry, de Bruijn and Howard, it is possible to associate a type to the family of proofs of any given formula. In particular, if $\Phi(A)$ is the type of proofs of the formula A and $\Phi(B)$ is the type of proofs of the formula B then the type of the family of proofs of the formula $A \Rightarrow B$ is the type $\Phi(A) \to \Phi(B)$ of functions from $\Phi(A)$ to $\Phi(B)$, because any proof of $A \Rightarrow B$ is an algorithm mapping any proof of A to a proof of B. Thus,

$$\Phi(A \Rightarrow B) \ = \ \Phi(A) \to \Phi(B)$$

Such striking similarities also hold with the other connectives and quantifiers: Φ *is an isomorphism between formulas and types*, called the *Curry-de Bruijn-Howard isomorphism*. Identifying isomorphic items (as it is usual to do), we see that the type of a proof is the formula that it proves.

5.6 Theories

As seen in Sect. 5.4.2, a key result in the proof of the witness property is that any cut-free constructive proof of a sequent ends with an introduction rule. This property does not extend to all axiomatic theories. For instance, adding the sole axiom $\exists x\ P(x)$, the tree reduced to its root labelled by this axiom constitutes a proof of the formula $\exists x\ P(x)$ which does not end with an introduction rule. Indeed, though the formula $\exists x\ P(x)$ is provable in this theory, there is not term t such that the formula $P(t)$ is provable.

Of course, with no axiom, there are very few provable functions hence the language of constructive proofs is a rather poor programming language. As a consequence, one of the most important questions in the subject of algorithmic interpretation of proofs

is to find axioms which keep valid the key property that every cut-free constructive proof ends with an introduction rule.

5.6.1 More Reduction Rules

A first way to incorporate axioms is to add to the language a constant for each of them. For instance, for each instantiation of the recursive schema, one can add a dedicated constant Rec^A. If π_1 is a proof of $A(0/x)$ and π_2 is a proof of $\forall y\ (A(y/x) \Rightarrow A(S(y)/x))$, then the term $(Rec^A\ \pi_1\ \pi_2)$ denotes a proof of $\forall x\ A$. Also, for every natural integer n, the term $(Rec^A\ \pi_1\ \pi_2\ n)$ will denote a proof of $A(n/x)$. If π_1 and π_2 are cut-free then so is this proof.

Now, the problem is that this proof ends with a $\forall$-elim rule rather than an introduction rule. So, in order to recover the witness property, new reduction rules have to be added, such as the following ones:

$$(Rec^A\ \pi_1\ \pi_2\ 0) \longrightarrow \pi_1$$
$$(Rec^A\ \pi_1\ \pi_2\ S(n)) \longrightarrow (\pi_2\ n\ (Rec^A\ \pi_1\ \pi_2\ n))$$

This approach has been much studied, cf. Gödel (1958), Tait (1967), Martin-Löf (1984), Paulin-Mohring (1993), Werner (1994), ...

5.6.2 Still More Reduction Rules

An alternative solution is to give up axioms and replace them by new nonlogical deduction rules. For instance, one can forget the subset axiom

$$\forall x \forall y\ (x \in \mathscr{P}(y) \Leftrightarrow x \subseteq y)$$

and replace it by the following two rules:

$$\frac{x \subseteq y}{x \in \mathscr{P}(y)}\ \text{fold}, \qquad \frac{x \in \mathscr{P}(y)}{x \subseteq y}\ \text{unfold}$$

It is then necessary to also add new proof reduction rules. In particular, the successive application of the *fold* and *unfold* rules has to be seen as a cut and the proof

$$\cfrac{\cfrac{\cfrac{\pi}{x \subseteq y}\ \text{fold}}{x \in \mathscr{P}(y)}\ \text{unfold}}{x \subseteq y}$$

should be reducible to π.

This approach has also been much studied, cf. Prawitz (1965), Crabbé (1974, 1991), Hallnäs (1983), Ekman (1994), Negri and von Plato (2001), Wack (2005), …

5.6.3 More Formulas

Enriching the language of formulas allows to remove some axioms. A paradigmatic example is the simple theory of types: its axioms can be removed if we allow quantifications over predicates. This leads to an extension of type theory and to higher order functional programming languages. This approach has been developed by Girard (1971), Coquand and Huet (1988), Leivant (1983), Krivine and Parigot (1990), …

5.6.4 Rewriting Rules

Another approach to remove axioms is to identify some terms and/or formulas. For instance, the axiom $\forall x\ (x + 0 = x)$ can be removed if the terms $x + 0$ and x are identified. Also, the recursion axiom can be removed if the formulas $N(y)$ and $\forall c\ ((0 \epsilon c) \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c)$ are identified.

This approach aims to synthetize and to go beyond the other three approaches mentioned above. It has been developed in Dowek et al. (2003), Dowek and Werner (2003), Cousineau and Dowek (2007), . . .

In all these approaches, the reduction process always halts for some theories but not for all theories. When the reduction process always halt, we obtain a programming language in which all programs halt and in which one can program all functions provably total in the involved theory.

5.7 *Other Extensions*

Besides the problem with incorporation of axioms, another very important question is the extension of the algorithmic interpretation of proofs to classical logic, i.e. the logic obtained by adding anyone of the three formulas of the Remark at the end of Sect. 5.3.1.

In this question, the witness property is to be relaxed to Herbrand's theorem:

> From a proof of a formula of the form $\exists y\ A$, one can extract a finite sequence of terms $u_1, \ldots, u_n$ and a proof of the formula
>
> $$A(u_1/y) \vee \cdots \vee A(u_n/y)$$

In this way, *classical proofs appear as non deterministic algorithms*, cf. (Parigot 1992).

Finally, besides natural deduction, other logical calculus have been considered; in particular, cf. (Curien and Herbelin 2000; Urban 2001), the sequent calculus (care!

the denomination "sequent calculus" is somewhat confusing: both natural deduction and sequent calculus use the same notion of sequent, their difference lies in their deduction rules).

5.8 *Cut Elimination and Consistency Results*

Years before Curry–Howard isomorphism was explicited, a remarkable relation between deduction and computability was shown in 1931: Gödel's first incompleteness theorem (1931) (see Davis 1965).

Let us say that a theory in inconsistent (or incoherent) if it proves both a formula and its negation. It is easy to show that an inconsistent theory proves every formula, which makes it trivial. Gödel's first incompleteness theorem states that there is a small set $\mathscr{P}$ of very simple true arithmetical properties such that if a consistent theory with a computable list of axioms is powerful enough to prove all properties in $\mathscr{P}$ (possibly via some coding of integers and the $+$, $\times$ operations) then this theory is necessarily incomplete: there are statements which can be neither proved nor disproved (i.e. the negation of the statement can not be proved). The assumption about the computable list of axioms causes no problem: it necessarily holds for every explicit mathematical theory $\mathscr{T}$. Thus, the whole family of true arithmetical statements is a strict subset of the family of formulas provable in $\mathscr{T}$!

In fact, in 1931, five years before Turing's paper (1936), Gödel's first incompleteness theorem was stated for theories with a primitive recursive list of axioms. But its proof goes through when the list of axioms is computable.

Robinson (1950) proved that $\mathscr{P}$ can be taken to be the following very simple set of seven axioms:

$$\begin{array}{c} Succ\, x \neq 0 \\ x = 0 \ \vee\ \exists y\ x = Succ\, y \\ Succ\, x = Succ\, y \Rightarrow x = y \end{array} \qquad \begin{array}{c} x + 0 = x \\ x + Succ\, y = Succ(x + y) \end{array} \qquad \begin{array}{c} x \times 0 = 0 \\ x \times Succ\, y = (x \times y) + x \end{array}$$

A remarkable instance of such a neither provable nor disprovable statement is a very natural formula expressing the coherence of the theory, this is Gödel's second incompleteness theorem.

Before Gödel's incompleteness theorems, David Hilbert invited the logical community to prove the coherence of mathematical theories by "finitistic" means hence by means formalizable in these theories. This is known as *Hilbert's program.* Of course, Gödel's theorems forced to a drastic revision of Hilbert's program. Nevertheless, this program should not be abandoned: other methods have to be invented that can overcome this stumbling block (hence cannot be formalizable in the theory itself) and still keep "convincing". The first to succeed was Gerhard Gentzen (who worked around Hilbert): he proved the consistency of Peano arithmetic using a transfinite induction which cannot be formalized in this theory (1936).

Later, Schütte (1951) rewrote Gentzen's argument as a proof of cut elimination in a formalization of arithmetic where (following the strategy described in Sect. 5.6.2) the recursion axioms have been replaced with a rule which has infinitely many premisses, the so-called ω-rule:

$$\frac{A(0),\ A(1),\ A(2),\ldots}{\forall x\ A(x)}\ \omega$$

This cut elimination property easily implies the consistency of the theory. Indeed, assume that theory is inconsistent. Then there is a proof of $\perp$ hence also a cut-free proof of $\perp$. Such a cut-free proof necessarily ends with an introduction rule which has to be an introduction of $\perp$ (since the proved formula is $\perp$). But there is no such rule, contradiction!

6 Decidable versus Undecidable

6.1 Automatic Deduction and Decidability of Logical Theories

> Non disputemus, sed calculemus.
> The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: "let us calculate, without further ado, to see who is right".
>
> Gottfried Wilhelm Leibniz 1687

As shown by the above citation, the idea of automatic deduction (with a considerable ambition) goes back to Leibniz (1646–1716), philosopher, mathematician, a truly multifaceted genius. To achieve such a goal, he proposed to find a universal language to express all possible ideas: a " lingua characteristica universalis". Now, the overall ambition has been drastically reduced. Rather than looking for a universal language, a large variety of languages are considered in relation with the diverse mathematical structures. For each of these languages, the first natural question is about the existence of an algorithm to decide if a formula is a theorem or not.

We list below a few examples of the numerous results obtained since the last century.

6.1.1 Entscheidungsproblem (The Decision Problem)

Given a logical language, Gödel completeness theorem insures that a closed formula is true in all structures for this language if and only if it is provable in this logic. As discussed in detail in chapter "Automated Deduction" of Volume 2, an interesting question, called the decision problem, is to know if the set of theorems of this logic is computable, (we also say that the logic is decidable) or not.

Care! Classically, the question is given another form: is the set of *satisfiable formulas* computable or not? (Satisfiable means that the formula is true in some structure, i.e. that its negation is not provable).

When the logic is undecidable, the decision problem can be refined to subfamilies of prenex formulas with given quantifer prefix. The answer then depends on both the logical language and the quantifer prefix. After about 80 years of research, the question is now completely solved, the answer being rather technical.

Below, we stick to the question relative to the set of provable formulas.

For languages with equality, the decidable logical languages or prefix classes of logical languages are as follows:

- The logical languages which contains only unary relation symbols, constants and at most one unary function symbol—hence no function or relation symbol (different from equality) with arity at least 2—(Rabin 1969).
- Formulas with universal quantifier prefix $\forall \ldots \forall$ for any logical language (Gurevich 1976).
- Formulas with quantifier prefix of the form $\forall \ldots \forall \exists \ldots \exists$ for purely relational languages (constants are allowed but there is no function symbol) (Ramsey 1930).
- Formulas with quantifier prefix of the form $\forall \ldots \forall \exists \forall \ldots \forall$ for languages containing only relations and at most one unary function (Shelah 1977).

For a complete analysis, see the book by Börger et al. (1997).

6.1.2 Arithmetic of Natural Integers

Arithmetic with Addition and Multiplication. Using Gödel's incompleteness theorem, it is straightforward to see that the family of true sentences of the structure $\langle \mathbb{N}; =, +, \times rangegle$ is not computable. This was noticed by Church in 1935.

Undecidability of Diophantine Arithmetics. As concerns the prefix classes, the best known undecidability result is that of the "diophantine problem", cf. Yuri Matijasevitch, Julia Robinson and Martin Davis:

> There exist two polynomials P, Q in $m+1$ variables (and with coefficients in $\mathbb{N}$) such that the set of natural integers a satisfying the formula
>
> $$\exists x_1 \ldots \exists x_m \; P(a, x_1, \ldots, x_m) = Q(a, x_1, \ldots, x_m)$$
>
> is not computable.

One can even get such polynomials with $m = 9$ (Matijasevich 1977, cf. Jones 1982). It is conjectured that $m = 3$ is possible.

Purely Additive Arithmetic (Multiplication is Removed). Presburger showed in 1929 that the family of true sentences of this arithmetics is decidable. Though it may seem quite a poor theory, additive arithmetics allows to modelize a lot of problems in computer science. Thus, Presburger's result is of outmost importance in computer science since it gives algorithms for all theseü problems.

Monadic Second-Order Theory of Natural Integers with the Successor Function. This is the theory of the structure $\langle \mathbb{N}; =, \in, Succrangegle \rangle$ (Care: no addition nor multiplication) with quantifications over integers and over sets of integers. The family of true sentences of this arithmetics is also decidable (Büchi 1960). The proof is obtained by finite automata techniques, cf. Sect. 3.1 in chapter "Theoretical Computer Science: Computational Complexity". This result is also of outmost importance in computer science.

6.1.3 Arithmetic of Rational Numbers

The family of true sentences of this arithmetics is undecidable because it is possible to define the set $\mathbb{Z}$ of integers in the structure $\langle \mathbb{Q}; =, +, \times rangegle \rangle$, hence also the set $\mathbb{N}$ of natural integers (since, by Lagrange's theorem, every natural integer is the sum of four squares). This is a difficult result proved by Robinson (1949) and it is the source of a lot of undecidability results.

The definition of $\mathbb{Z}$ in $\mathbb{Q}$ found by Julia Robinson in 1948 is a formula with quantifier prefix $\forall^2\exists^7\forall^6$ (where $\forall^2$ means $\forall\forall$). About 70 years later, Koenigsmann (2016) obtained a purely universal definition, i.e. one with quantifier prefix $\forall \ldots \forall$. It is believed (but still an open question which is related to famous conjectures in number theory) that $\mathbb{Z}$ is not Diophantine over $\mathbb{Q}$, i.e. is not definable by a formula with quantifier prefix $\exists \ldots \exists$.

6.1.4 Words

Words and Concatenation. Consider the family Σ^* of words on an alphabet Σ with at least two letters. Endow Σ^* with the concatenation operation (which maps the pair of words $(a_1 \ldots a_k, b_1 \ldots b_\ell)$ to the word $a_1 \ldots a_k b_1 \ldots b_\ell$). The set of formulas true in this structure is undecidable (Quine 1946) since it allows to easily encode the structure $\langle \mathbb{N}; =, +, \times rangegle \rangle$.

Contrasting with Matijasevitch's result on integers, cf. Sect. 6.1.2, Makanin 1977, proved the decidability of the family of true formulas of the form

$$\exists x_1 \ldots \exists x_k \; C(x_1, \ldots, x_k) = D(x_1, \ldots, x_k)$$

where C, D are words built on alphabet Σ augmented with variables (which represent words and not letters).

Words and Suffix Adjunction of Letters. Replacing concatenation by unary operations which add a letter as a suffix (i.e. $Succ_a(a_1 \ldots a_n) = a_1 \ldots a_n a$), one can consider the monadic second-order structure $\langle \Sigma^*, \mathscr{P}(\Sigma^*); =, \in, (Succ_a)_{a\in\Sigma} rangegle$ where $\in$ is the membership relation ("monadic second-order" means that we can quantify over words and over sets of words). It turns out that the set of true formulas in this structure is computable (Rabin 1969). This is a major result of theoretical computer science. Its proof is quite difficult.

6.1.5 Real and Complex Algebras

If arithmetic in $\mathbb{N}$ and $\mathbb{Q}$ is undecidable, this is no more the case with reals or complex numbers: the set of formulas true in the structure $\langle \mathbb{R}; =, +, \times rangegle$ (resp. $\langle \mathbb{C}; =, +, \times rangegle$) is computable (Tarski 1931). This is another key result in the subject. Care: this result is about the sole algebraic structure of reals (or complex numbers) hence all usual items of real analysis (sequences, limits,…) are out of scope.

An important related question remains open: what if we add the exponential function to the real algebra? Is the theory of $\langle \mathbb{R}; =, +, \times, x \mapsto e^x rangegle$ decidable or not? It is known (Macintyre and Wilkie 1995) that this question is related to a difficult open problem in number theory, Schanuel's conjecture (1960).

6.1.6 Elementary Geometry

Using representation of the Euclidean plane and space with Cartesian coordinates, elementary geometry can be reduced to questions in the real algebra hence it is decidable (Tarski 1931). Thus, all questions about conics, quadrics, nine-point circle, harmonic points,…, which were so much in fashion years ago in mathematics, can be solved using an algorithm for real algebra!

6.2 A Few Other Problems

Dominoes. Given a finite family of squares, all of the same size but with sides colored in diverse ways, *is it possible to pave the discrete Euclidean plane* $\mathbb{Z} \times \mathbb{Z}$ *with such squares so that ant two adjacent squares have identical adjacent sides?* This problem is undecidable: the set of finite such families for which it is possible is not computable (Berger 1966).

The same question can be raised in the hyperbolic plane. Care, there is no square nor rectangle in the hyperbolic plane! However, for every $s \geq 5$, there are regular polygons with right angles and s sides. For each $s \geq 5$, the answer is the same: the problem is undecidable (Margenstern 2007, 2008).

Petri Nets. As discussed in chapter "Artificial Intelligence in Biological Modelling" of this volume, a very efficient model for concurrency is that of Petri nets (Petri 1962). A simple form of this model consists of the following items:

- two fixed disjoint finite families S, T: "places" and "transitions",
- a fixed family of "arcs" $F \subseteq (S \times T) \cup (T \times S)$ between places and transitions,
- a dynamic distribution $M : S \to \mathbb{N}$ of "tokens" in places.

The inputs (resp. outputs) of a transition $t \in T$ are all the places s such that $(s, t) \in F$ (resp. $(t, s) \in F$ i.e. there is an arc from s to t (resp. from t to s).

A distribution M admits a successor if there exists some transition $t \in T$ such that $M(s) > 0$ for every input s of t. Fixing such a transition t, the successor distribution M' of M is obtained by removing one token to each input of t and adding one token to each output of t.

A run of a Petri net is then a sequence of successive distributions. Observe that this is a nondeterministic run since there may be several successors to a given distribution. Also, a distribution may have no successor, blocking the run.

The interesting question is that of reachability: is it possible to go from a given distribution to another given distribution via some run? A deep and difficult result is that this problem is decidable (Mayr 1984).

Varia. See chapter "Reasoning with Ontologies" of Volume 1 and chapter "Databases and Artificial Intelligence" of this volume.

6.3 Deciding…with High Probability To Be Correct

In many practical cases, uncertainty is unavoidable and arguing with probabilities is a natural approach, see chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1. In the context of Turing machines, uncertainty comes when the state is not precise enough to determine—together with the scanned symbol—what is to be done in a unique way. This leads to the notion of nondeterministic Turing machine: at some steps there may be several possible transitions (taken from a fixed finite set). Assume some probability distribution on these different possible transitions. One can then consider the set of inputs for which the probability to be accepted is $\geq \delta$ for some fixed $\delta > 0$. Simulating in parallel all possible computations, it is easy to see that this set is computably enumerable hence it is also accepted by some deterministic Turing machine, cf. Leeuw et al. (1956). Thus, as concerns acceptance of languages, probabilities do not bring anything new.

7 Computability on Reals

We review different models to compute with real numbers. Their motivations are far apart from each other.

7.1 Computable Analysis

Introduced by Turing (1936), computability with reals was developed by the Polish school (Banach and Mazur 1937; Grzegorczyk 1957; Mostowski 1957) and by Lacombe (1955), cf. the book (Pour-El and Richards 1989).

The natural idea to get a notion of computable real is to consider those reals which have computable representations. As is well-known, there are many ways to represent a real a. It turns out that the effectivization of all usual representations lead to the same notion. For instance, for all integer $k \geq 2$, the following conditions are equivalent

- there exists a computable sequence $(q_n)_{n\in\mathbb{N}}$ of rational numbers which converges to a and the identity function is a Cauchy modulus, i.e. $|q_m - q_n| < 2^{-k}$ for all $m, n \geq k$.
- the development of the real a in base k is a computable sequence of digits,
- the left Dedekind cut $\{q \in \mathbb{Q} \mid q < a\}$ of a is a computable set,

Let us stress that the identity function as Cauch modulus can be replaced by any unbounded monotone computable function.

Alas, the above equivalences badly fail with sequences of reals (Mostowski 1957). The "right" notion, which is also the larger one, turns out to be that associated to the Cauchy representation, i.e. a sequence $(a_i)_{i\in\mathbb{N}}$ of reals is computable if there exists a doubly indexed computable sequence $(q_{i,n})_{i,n\in\mathbb{N}}$ of rational numbers such that, for each i, the simply indexed sequence $(q_{i,n})_{n\in\mathbb{N}}$ (obtained by freezing i) converges to a_i and admits the identity as a Cauchy modulus.

As for the notion of computable function $f : [a, b] \to \mathbb{R}$ where a, b are computable reals, two conditions are required:

- f is "sequentially computable", i.e. maps computable sequences of reals to computable sequences of reals,
- f is effectively uniformly continuous, i.e. it admits a computable modulus of uniform continuity $\nu : \mathbb{N} \to \mathbb{N}$ such that, for all $x, y \in [a, b]$ and $n \in \mathbb{N}$, if $|x - y| < \nu(n)$ then $|f(x) - f(y)| < 2^{-n}$.

In case f has domain an interval I with computable endpoints, the effective uniformly continuity has to be true in $[a, b]$, in a uniform way, for every $a, b \in \mathbb{Q}$ such that $[a, b] \subseteq I$. In other words, the modulus of uniform continuity is now a computable function $\nu : \mathbb{Q}^2 \times \mathbb{N} \to \mathbb{N}$.

The facts that the diverse representations of reals do not lead to the same notions of computable sequence of reals and computable function over reals and that the Cauchy representation is the right one are witnessed by the following striking result: *If x is computable then so is $3x$ and we can compute approximations of $3x$ from approximations of x but there is no algorithm which, for every real x, computes the base 10 decimals of $3x$ from those of x.*

Since computable functions over reals have to be continuous but the characteristic function of the singleton set $\{0\}$ is discontinuous as a function $\mathbb{R} \to \mathbb{R}$, we have
Rice's Theorem (1954). *One cannot decide if a real is 0 or not.*

7.2 Blum, Shub and Smale Machines

The Blum et al. (1989, 1998) has been introduced to get some insight on the algebraic complexity of problems over reals. In that model some fixed simple functions over reals are given and considered to be performed in unit time. This allows to measure the complexity of a problem in terms of the number of operations to be done.

Since the test $x = 0$ is among the possible operations being performed in unit time, the Blum, Shub and Smale is totally incompatible with computable analysis…

7.3 Shannon's General Purpose Analog Computer

The *General Purpose Analog Computer* (GPAC) was introduced by Shannon (1941), as a model of analog computers, in particular of Vannevar Bush's differential analyzer (1931) and other analog electronics.

In its present refined version, cf. Graça (2007), the basic units of a GPAC are adders and multipliers (to perform addition and multiplication) constant units (i.e. constant functions with value any real a) and integrators which are operators $(u, v) \mapsto w$ on functions such that, for some fixed t_0,

$$w(t) = \int_{t_0}^{t} u(x)v'(x)dx\ .$$

These units can be freely connected in a circuit and *feedback connexions* are allowed for integrators. To such a circuit is associated a differential equation which, given explicit initial conditions for the integrators, has a unique solution, said to be GPAC-generated.

For instance, the GPAC of Fig. 1, with integrator initial condition $\theta(0) = 1$, generates the function satisfying $\theta(t) = 1 + \int_0^x \theta(x)\,dx$, i.e. the exponential function e^t. As for the one with integrator initial conditions $\varphi(0) = 1$, $\psi(0) = 0$, it generates the functions satisfying $\varphi(t) = 1 - \int_0^x \psi(x)\,dx$ and $\psi(t) = \int_0^x \varphi(x)\,dx$, i.e. the functions $\varphi(t) = \cos(t)$ and $\psi(t) = \sin(t)$.

It turns out (Shannon 1941, see Graça 2007) that the GPAC-generated functions are exactly the differentially algebraic functions, i.e. the solutions of differential

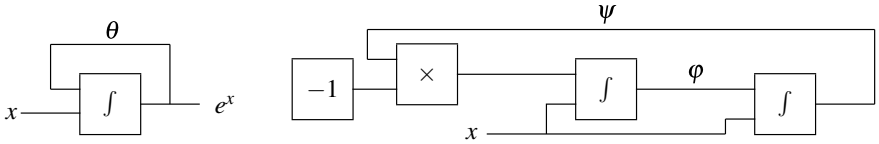


Fig. 1 Two GPACs. With integrator initial conditions $\theta(0) = 1$, $\varphi(0) = 1$ and $\psi(0) = 0$, they generate the function $\theta(x) = e^x$ and the functions $\varphi(x) = \cos(x)$ et $\psi(x) = \sin(x)$

equations of the form $p(x, y, y', y'', \ldots, y^{(k)}) = 0$ where p is a polynomial. They are also the components of solutions $\mathbf{y} = (y_1, \ldots, y_k)$ of differential systems $\mathbf{y}' = \mathbf{p}(t, \mathbf{y})$, $\mathbf{y}(0) = \mathbf{a}$ where $\mathbf{p}$ is a k-tuple of polynomials and $\mathbf{a}$ is a k-tuple of reals (Graça and Costa 2003).

Not all functions of computable analysis are GPAC-generated. For instance, Riemann's zeta function $\zeta(x) = \sum_{n\geq 1} \frac{1}{n^x}$ and the function $\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$ (the analytic extension of the usual factorial function on $\mathbb{N}$) are not GPAC-generated.

Nevertheless, a spectacular result by Bournez et al. (2007), insures that analog computability matches computable analysis.

Theorem. *The functions of computable analysis are exactly those which are GPAC-computable.*

where GPAC-computability is defined as follows (Graça 2004):

- A GPAC is computable if the real numbers for the constant units and the lower bounds of integrators are all computable reals.
- A function $f : \mathbb{R} \to \mathbb{R}$ is GPAC-computable if there exist two functions g, ε which are generated by some computable GPACs and such that, for all x, we have $\forall t \; |f(x) - g(t, x)| \leq \varepsilon(t, x)$ and $\lim_{t \to +\infty} \varepsilon(t, x) = 0$.

7.4 Computation Models with Continuous Time

The most popular such model is the neural network model (Siegelmann and Sontag 1995) which loosely simulates neurons (in human and animal beings). The reader will find in Bournez and Campagnolo (2008) a survey of the different models for continuous time computation and their properties as concerns computability and complexity.

7.5 Unifying These Models?

It is clearly impossible to unify all the computation models dealing with reals. We already noticed that all functions in computable analysis are continuous, which is not the case in Blum, Shub and Smale model, cf. Sect. 7.2. Let us also mention that in some of the continuous time models it is possible to encode a noncomputable sequence of digits into a single real and to perform exact arithmetic on reals and hence get super-Turing computation! See Siegelmann and Fishman (1998) for the case of neural networks.

Though recent results show the equivalence between some of these models, see Bournez et al. (2006, 2007), Graça and Costa (2003), it remains unclear whether one of these models could lead to some thesis in the vein of Church–Kleene–Turing thesis, see Bournez and Campagnolo (2008) for a discussion on this question.

7.6 Computability Beyond Church–Kleene–Turing Thesis?

Following Ord (2006), we briefly present some of the diverse ways to get systems or machines to go beyond Turing computability.

Oracle Computability. The first approach was proposed by Turing himself in his PhD dissertation (1939): use Turing machines which can question a fixed oracle considered as a blackbox. This notion is heavily used in computability and complexity theory in order to get "relativized results" and compare (possibly noncomputabe) sets and functions in a much broader sense than that of computable or polynomial time reduction (cf Sect. 2.4.1 in chapter "Theoretical Computer Science: Computational Complexity").

There are many kinds of machines in the vein of oracle Turing machines:

- Coupled Turing machines, introduced by Copeland (1997), are Turing machines connected to an input canal,
- Networks of machines with possibly noncomputable synchronization time, cf. Copeland and Sylvan (1999),
- Machines coupled to physical processes such as *scatter-machines*, cf. Beggs et al. (2008),
- Machines using biaised coins as oracles, cf. Ord and Kieu (2009).

Speeding-up Machines. A different approach considers "speeding-up machines", i.e. machines which perform a first step in time 1, the second one in time $1/2$, …and the nth step in time 2^{-n}. This is, indeed, an old idea going back to Blake (1926), Russell (1935), Weyl and Kirschmer (1927). This idea is also at work with machines running in ordinal time (Hamkins 2002). It has been argued that some physical systems could implement this idea: quantum mechanics (Calude and Pavlov 2002) or black holes (Hogarth 1994).

Other Machines. Various extra features have been added to Turing machines: unbounded nondeterminism (Spaan et al. 1989), infinite inputs (cf. Weihrauch 2000 for type 2 Turing machines), infinite set of states (Ord 2002), noisy Turing machines (Asarin and Collins 2005), etc…

8 Conclusion

The mathematical formalization of computability, this intuitive notion going back to antiquity, is one of the major mathematical achievement in the 20th century. It is the basic tool in computer science and, historically, its starting point.

What can we expect in the near future?

Programming. Pointing an unexpected relation between computations and proofs, Curry–Howard isomorphism (cf. Sect. 5) has created a bridge between two subjects a priori far apart, and stressed how fundamental they are. Fruitful research is currently

being done around this isomorphism and much is to be expected as concerns proof assistants and programming languages with provably correct programs.

Computer Virology. Kleene fixed point theorem (cf. Sect. 3.2) is the theoretical basis of computer virology. This topic is to be an unending sequence of actions and counteractions between malicious hackers and those who try to stop them.

Algorithms. As seen in Sect. 4, formalization of the operational notion of algorithm is somewhat advanced but far to be fully achieved. More is to be done to capture the different facets of computation.

Numerical Analysis. There is an obvious common point between computability and numerical analysis: both are interested in computations. Of course, computation has to be extended to real numbers: this is the subject of computable analysis (cf. Sect. 7), which dates back to the very beginning of computability since it was explicitly treated in Turing (1936).

Though computability is essentially discrete whereas numerical analysis deals with the continuous world, both subjects should deeply interact. Indeed, two processes are often involved in the mathematical study of some discrete phenomena: first, they are often modeled with continuous mathematics (because they give us a better intuitive grasp), second, for computing explicit values, one proceeds to a subsequent discretization. Though the second process is by no way the inverse of the first one, this is somewhat puzzling. As explained by Kolmogorov (1983, p. 30):

> Quite probably, with the development of the modern computing technique, it will be clear that in very many cases it is reasonable to conduct the study of real phenomena avoiding the intermdiate stage of stylizing them in the spirit of mathematics of the infinite and the continuous and passing directly to discrete models.

Probabilities and Statistics. As unexpected as it can be, there is a deep connection between computability and the theory of probabilities and statistics. Indeed, though it is the central intuitive notion, the concept of random item has no formal definition in probability theory. Random items are seen globally and never individually. This question has been considered by Richard von Mises and Andrei N. Kolmogorov. It was one of the motivations that lead to Kolmogorov complexity (cf. Sect. 5.2 in chapter "Theoretical Computer Science: Computational Complexity"). It turns out that, using computability, the notion of random item can get a formal mathematical definition. In fact, it can be done in two equivalent ways, one mixing computability, measure theory and topology (due to Martin-Löf 1966), the other one based on Kolmogorov complexity.

Mathematics. Obviously, the distinction *computable/noncomputable* is important in all mathematical domains (cf. Sect. 6.1) and should become a basic question for any working mathematician.

References

Abramsky S, Gabbay DM, Maibaum TS (eds) (2001) Handbook of logic in computer science. Elsevier, New York

Ackermann W (1928) Zum Hilbertschen Aufbau der rellen Zahlen. Math Ann 99:118–133

Adleman LM (1988) An abstract theory of computer viruses. Advances in cryptology (CRYPTO'88), vol 403. Lecture Notes in Computer Science. Springer, New York, pp 354–374

Arrighi P, Dowek G (2008) A quantum extension of Gandy's theorem. Colloque "La thèse de Church: hier, aujourd'hui, demain

Asarin E, Collins P (2005) Noisy turing machines. In: Proceeding of ICALP'05, vol 3580. Lecture Notes in Computer Science. Springer, Berlin, pp 1031–1042

Banach S, Mazur S (1937) Sur les fonctions calculables. Ann. Soc. Pol. de Math. 16:223

Barendregt H (1980) The lambda-calculus: its syntax and semantics. North-Holland, Amsterdam

Barendregt HP, Dekkers W, Statman R (2013) Lambda calculus with types. Perspectives in logic. Cambridge University Press, Cambridge

Beggs EJ, Costa JF, Loff B, Tucker JV (2008) Computational complexity with experiments as oracles. Proc R Soc Lond A 464(2098):2777–2801

Berger R (1966) The undecidability of the domino problem. Mem Am Math Soc 66:1–72

Blake R (1926) The paradox of temporal process. J Philos 23(24):645–654

Blass A, Gurevich Y (2003) Abstract state machines capture parallel algorithms. ACM Trans Comput Log 4(4):578–651

Blass A, Gurevich Y (2008) Abstract state machines capture parallel algorithms: correction and extension. ACM Trans Comput Log 9(3):1–32

Blum L, Cucker F, Shub M, Smale S (1998) Complexity and real computation. Springer, Berlin The citation 'Blurn et al. (1988)' has been changed 'Blurn et al. (1988)'. Please check and confirm the changes

Blum L, Shub M, Smale S (1989) On a theory of computation and complexity over the real numbers; NP completeness, recursive functions and universal machines. Bull Am Math Soc 21(1):1–46

Boker U, Dershowitz N (2008) The Church-Turing thesis over arbitrary domains. In: Avron A, Dershowitz N, Rabinovich A (eds) Pillars of computer science: essays dedicated to Boris (Boaz) Trakhtenbrot on the occasion of his 85th birthday, vol 4800. Springer, Berlin, pp 199–229

Börger E, Grädel E, Gurevich Y (1997) The classical decision problem. Springer, Berlin

Bournez O, Campagnolo ML (2008) A survey on continuous time computations. In: Cooper S, Löwe B, Sorbi A (eds) New computational paradigms. Changing conceptions of what is computable. Springer, Berlin, pp 383–423

Bournez O, Campagnolo ML, Graça DS, Hainry E (2006) The general purpose analog computer and computable analysis are two equivalent paradigms of analog computation. In Cai J, Cooper SB, Li A (eds) Proceedings of Theory and applications of models of computation, third international conference, TAMC 2006, Beijing, China, 15–20 May 2006. Lecture Notes in Computer Science, vol 3959. Springer, pp 631–643

Bournez O, Campagnolo ML, Graça DS, Hainry E (2007) Polynomial differential equations compute all real computable functions on computable compact intervals. J Complex 23(3):317–335

Bouveresse J (1999) Prodiges et vertiges de l'analogie: de l'abus des belles-lettres dans la pensée. Raisons d'Agir

Bush V (1931) The differential analyzer. a new machine for solving differential equations. J Frankl Inst 212:447–488

Büchi J R (1960) Weak second-order arithmetic and finite automata. Z Math Logik und Grundlagen der Math 6:66–92

Calude C, Pavlov B (2002) Coins, quantum measurements, and Turing's barrier. Quantum Inf Process 1(1–2):107–127

Church A (1936) An unsolvable problem of elementary number theory. Am J Math 58:345–363

Codd EF (1968) Cellular automata. Academic Press, Cambridge

Cohen F (1986) Computer viruses. PhD thesis, University of Southern California

Colson L (1991) About primitive recursive algorithms. Theor Comput Sci 83(1):57–69
Colson L, Fredholm D (1998) System T, call-by-value and the minimum problem. Theor Comput Sci 206(1–2):301–315
Cook SA, Reckhow RA (1973) Time bounded random access machines. J Comput Syst Sci 7(4):354–375
Copeland BJ (1997) The broad conception of computation. Am Behav Sci 40:690–716
Copeland BJ (Fall 2002) The Church-Turing thesis. In: Zalta EN (ed) The stanford encyclopedia of philosophy. Stanford University. Available online at: http://plato.stanford.edu/entries/church-turing/
Copeland BJ, Sylvan R (1999) Beyond the universal Turing machine. Australas J Philos 77:46–66
Coquand T, Huet G (1988) The calculus of constructions. Inf Comput 76:95–120
Cori R, Lascar D (1983a) Logique mathématique: Tome 1: Calcul propositionnel, algèbres de Boole, calcul des prédicats. Dunod, Paris
Cori R, Lascar D (1983b) Logique mathématique: Tome 2: fonctions récursives, théorème de Gödel, théorie des ensembles. Dunod, Paris
Cousineau D, Dowek G (2007) Embedding pure type systems in the lambda-pi-calculus modulo. In: Ronchi Della SR (ed) Typed lambda calculi and applications, vol 4583. Springer, Berlin, pp 102–117
Crabbé M (1974) Non-normalisation de ZF. Manuscript
Crabbé M (1991) Stratification and cut-elimination. J Symb Log 56(1):213–226
Curien P-L, Herbelin, H (2000) The duality of computation. SIGPLAN Notices 35(9):233–243. International conference on functional programming, ICFP'00, Montreal
Davis M (1958) Computability and unsolvability. Dover publications
Davis M (1965) The undecidable: basic papers on undecidable propositions. Unsolvable problems and computable functions. Raven Press. Reprinted by Dover Publications, Incorporated in 2004
Delorme M (1999) An introduction to cellular automata. In: Delorme M, Mazoyer J (eds) Cellular automata: a parallel model. Kluwer, pp 3–51
Dershowitz N, Gurevich Y (2008) A natural axiomatization of computability and proof of church's thesis. Bull Symb Log 14(3):299–350
Dexter S, Boyle P, Gurevich Y (1997) Gurevich local evolving algebras and Schönage storage modification machines. J Univers Comput Sci 3(4):279–303
Dowek G (2007) Les métamorphoses du calcul : une étonnante histoire de mathématiques. Le Pommier
Dowek G, Hardin T, Kirchner C (2003) Theorem proving modulo. J Autom Reason 31:32–72
Dowek G, Werner B (2003) Proof normalization modulo. J Symb Log 68(4):1289–1316
Ekman J (1994) Normal proofs in set theory. PhD thesis, Chalmers university of technology and University of Göteborg
Elgot C, Robinson A (1964) Random-access stored-program machines, an approach to programming languages. J Assoc Comput Mach 11(4):365–399
Ferbus M, Grigorieff S (2010) ASM and operational algorithmic completeness of lambda calculus. In: Andreas Blass ND, Reisig W (eds) Fields of logic and computation, essays dedicated to Yuri Gurevich on the occasion of his 70th birthday. LNCS, vol 6300. American Mathematical Society, pp 301–327
Fredholm D (1996) Computing minimum with primitive recursion over lists. Theor Comput Sci 163(1–2):269–276
Friedberh RM (1958) Three theorems on recursive enumeration. i. decomposition. ii. maximal set. iii. enumeration without duplication. J Symb Log 23:309–316
Gandy RO (1980) Church's thesis and principles for mechanisms. In: Barwise J, Keisler HJ, Kunen K (eds) The Kleene symposium. North-Holland, pp 123–148
Gentzen G (1934) Untersuchungen über das logische schließen. Mathematische Zeitschrift 39:405–431
Gentzen G (1936) Die widerspruchfreiheit der reinen zahlentheorie. Mathematische Annalen 112:493–565

Girard J-Y (1971) Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In: 2d scandinavian logic symposium. North Holland, pp 63–92
Girard J-Y (1987) Linear logic. Theor Comput Sci 50:1–102
Girard J-Y, Lafont Y, Taylor P (1989) Proofs and types. Cambridge tracts in theoretical computer science, vol 7. Cambridge University Press, Cambridge
Gödel K (1931) Über formal unentscheidbare sätze der principia mathematica und verwandter systeme I. Monatschefte fur Mathematik und Physik 38:173–198
Gödel K (1958) Ü eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. Dialectica 12:280–287
Graça DS (2004) Some recent developments on shannon's general purpose analog computer. Math Log Q 50(4–5):473–485
Graça DS (2007) Computability with polynomial differential equations. PhD thesis, Instituto Superior Técnico
Graça DS, Costa JF (2003) Analog computers and recursive functions over the reals. J Complex 19(5):644–664
Grädel E, Nowack A (2003) Quantum computing and abstract state machines. Lecture Notes in Computer Science, pp 309–323
Grigorieff S (2006) Synchronization of a bounded degree graph of cellular automata with non uniform delays in time $D \log_m D$. Theor Comput Sci 356(1–2):170–185
Grigorieff S, Valarcher P (2010) Evolving MultiAlgebras unify all usual sequential computation models. In: Marion J-Y, Schwentick T (eds) STACS 2010, pp 417–428
Grzegorczyk A (1957) On the definitions of computable real continuous functions. Fundam Math 44:61–71
Gurevich Y (1976) The decision problem for standard classes. J Symb Logic 41:460–464
Gurevich Y (1985) A new thesis, abstract 85T–68-203. Not Am Math Soc 6:317
Gurevich Y (1988) Logic and the challenge of computer science. In: Boerger E (ed) Current trends in theoretical computer science. Computer Science Press, pp 1–50
Gurevich Y (2000) Sequential abstract state machines capture sequential algorithms. ACM Trans Comput Log 1(1):77–111
Hallnäs L (1983) On normalization of proofs in set theory. PhD thesis, University of Stockholm
Hamkins JD (2002) Infinite time Turing machines. Minds Mach 12(4):521–539
Hankin C (1994) Lambda calculi: a guide for computer scientists, vol 3. Graduate texts in computer science. Oxford
Hillis WD (1986) The connection machine. MIT Press (Partiellement disponible sur books.google.fr)
Hillis WD (1989) Richard Feynman and the connection machine. Phys Today 42(2):78–83
Hindley JR, Seldin JP (1986) Introduction to combinators and λ-calculus. Cambridge University Press, Cambridge
Hogarth M (1994) Non-Turing computers and non-Turing computability. Philos Sci Assoc 1:126–138
Jain S, Osherson D, Royer JS, Sharma A (1999) Systems that learn. MIT press, Cambridge
Jiang T (1992) The synchronization of non-uniform networks of finite automata. Inf Comput 97(2):234–261
Jones JP (1982) Universal diophantine equation. J Symb Log 47:549–571
Jones ND (1997) Computability and complexity, from a programming perspective. MIT press, Cambridge
Kanovich MI, Vauzeilles J (2007) Strong planning under uncertainty in domains with numerous but identical elements (a generic approach). Theor Comput Sci 379(1–2):84–119
Kleene SC (1936) General recursive functions of natural numbers. Mathematische Annalen 112:727–742
Kleene S C (1943) Recursive predicates and quantifiers. Trans Am Math Soc 53:41–73
Kleene SC (1952) Introduction to metamathematics. D. Van Nostrand Co, New York

Kleene SC (1967) Mathematical logic. Wiley, New Jersey
Knuth DE (1973) The art of computer programming: sorting and searching, vol 3. Addison-Wesley, Boston
Koenigsmann J (2016) Defining $\mathbb{Z}$ in $\mathbb{Q}$. Ann Math 183(1):73–93
Kolmogorov AN (1953) On the notion of algorithm. Uspekhi Mat Nauk 8:175–176 In russian
Kolmogorov AN (1983) Combinatorial foundations of information theory and the calculus of probabilities. Russ Math Surv 38(4):29–40
Kolmogorov AN, Uspensky, V (1958) On the definition of an algorithm. Uspekhi Mat Naut 13(4). English translation in AMS translation vol 21 (1963), 217–245
Kripke S (1965) Semantical analysis of intuitionistic logic. In: Crossley J, Dummett MAE (eds) Formal systems and recursive functions, pp 92–130
Krivine JL (1990) Lambda-calcul. Masson
Krivine J-L, Parigot M (1990) Programming with proofs. J Inf Process Cybern EIK 26(3):149–167
Kudlek M (1996) Small deterministic Turing machines. Theor Comput Sci 168:241–255
Lacombe D (1955) Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. Comptes Rendus de l'Académie des Sciences Paris 241:151–153
Lacombe D (1960) La théorie des fonctions récursives et ses applications. Bulletin de la Société Mathématique de France 88:393–468
Leeuw Kd, Moore EF, Shannon CE, Shapiro N (1956) Computability by probabilistic machines. In: Shannon C, MacCarthy J (eds) Automata studies. Princeton University Press, Princeton, pp 183–212
Leivant D (1983) Reasoning about functional programs and complexity classes associated with type disciplines. In: Foundations of computer science, FOCS'83, vol 88. IEEE Computer Society Press, pp 460–469
Longo G, Paul T (2009) Le monde et le calcul. Réflexions sur calculabilité, mathématiques et physique. In: Logique et Interaction : Géométrie de la cognition, volume Actes du colloque et école thématique du CNRS Logique, Sciences, Philosophie, Cerisy. Hermann
Macintyre AJ, Wilkie A (1995) On the decidability of the real exponential field. In: Odifreddi PG (ed) Kreisel 70th birthday volume. CLSI
Makanin G S (1977) The problem of solvability of equations in the free semigroup. Math USSR Sb 32:129–198
Margenstern M (2007) The domino problem of the hyperbolic plane is undecidable. Bull EATCS 93:220–237
Margenstern M (2008) The domino problem of the hyperbolic plane is undecidable. Theor Comput Sci 407:28–84
Margenstern M (2009) Surprising areas in the quest for small universal devices. Electron Notes Theor Comput Sci 225:201–220
Marion J-Y (2012) From Turing machines to computer viruses. Philos Trans R Soc A 370:3319–3339
Martin-Löf P (1966) The definition of random sequences. Inf Cont 9:602–619
Martin-Löf P (1984) Intuitionistic type theory. Bibliopolis, Naples
Matijasevich Y (1977) Some purely mathematical results inspired by mathematical logic. In: Proceedings of fifth international congress on logic, methodology and philosophy of science, London, Ontario, 1975. Reidel, Dordrecht, pp 121–127
Mayr EW (1984) An algorithm for the general Petri net reachability problem. SIAM J Comput 13(3):441–460. Preliminary version in Proceedings of the thirteenth annual ACM symposium on Theory of computing, pp 238–246, 1981
Mazoyer J, Yunès J-B (2010) Computations on cellular automata. In: Rozenberg G, Bäck TH, Kok JN (eds) Handbook of natural computing. Springer, to appear
Melzak Z (1961) An informal arthmetical approach to computability and computation. Can Math Bull 4(3):279–293
Minsky M (1961) Recursive unsolvability of post's problem of 'tag', and other topics in theory of turing machines. Annals of Mathematics 74:437–455

Minsky ML (1967) Computation: finite and infinite machines. Prentice-Hall, Upper Saddle River
Minsky ML (1982) Why people think computers can't. AI Mag 3(4):3–15. Reprinted in Technology Review, Nov/Dec 1983, and in The computer culture, (Donnelly, Ed.) Associated University Presses, Cranbury NJ, 1985
Moschovakis Y (2001) What is an algorithm? In: Engquist B, Schmid W (eds) Mathematics unlimited - 2001 and beyond. Springer, Berlin, pp 919–936
Moschovakis Y (2006) On primitive recursive algorithms and the greatest common divisor function. Theor Comput Sci 301(1–3):1–30
Mostowski A (1957) On computable sequences. Fundam Math 44(1):37–51
Negri S, von Plato J (2001) Structural proof theory. Cambridge University Press, Cambridge
Nielsen MA (1997) Computable functions, quantum measurements, and quantum dynamics. Phys Rev Lett 79(15):2915–2918
Odiffredi P (1989) Classical recursion theory. North-Holland
Ord T (2002) Hypercomputation: computing more than the turing machine. Arxiv preprint math/0209332
Ord T (2006) The many forms of hypercomputation. Appl Math Comput 178(1):143–153
Ord T, Kieu TD (2009) Using biased coins as oracles. Int J Unconv Comput 5:253–265
Papadimitriou CH (1994) Computational complexity. Addison-Wesley, Boston
Parigot M (1992) $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In: Voronkov A (ed) Logic programming and automated reasoning, vol 624. Springer, Berlin, pp 190–201
Paulin-Mohring C (1993) Inductive definitions in the system Coq - rules and properties. In: Bezem M, de Groote JF (eds) Typed lambda calculi and applications, vol 664. Springer, Berlin, pp 328–345
Paulin-Mohring C, Werner B (1993) Synthesis of ml programs in the system Coq. J Symb Comput 15(5/6):607–640
Pavlotskaya L (1973) Solvability of the halting problem for certain classes of turing machines. Math Notes Acad Sci USSR 13:537–541
Pavlotskaya L (1978) Sufficient conditions for the halting problem decidability of turing machines. Avtomaty i Mashiny, pp 91–118 (in Russian)
Petri C A (1962) Kommunikation mit automaten. Dissertation, Schriften des IIM (Institut für Instrumentelle Mathematik an der Universität Bonn), Bonn
Pour-El MB, Richards JI (1989) Computability in analysis and physics. Springer, Berlin
Prawitz D (1965) Natural deduction, a proof-theoretical study. Almqvist & Wiksell, Stockholm
Quine W V (1946) Concatenation as a basis for arithmetic. J Symb Logic 4:105–114
Rabin M O (1969) Decidability of second-order theories and automata on infinite trees. Trans Am Math Soc 141:1–35
Rado T (1962) On a simple source for non-computable functions. Bell Syst Tech J 41:877–884
Ramsey F (1930) On a problem of formal logic. In: Proceedings of the London Mathematical Society, vol 30, pp 264–286
Robinson J (1949) Definability and decision problems in arithmetic. J Symb Log 14:98–114
Robinson RM (1950) An essentially undecidable axiom system. In: Proceedings of the international congress of mathematics, pp 729–730
Rogers HJ (1967) Theory of recursive functions and effective computability. McGraw Hill, New York
Rosenstiehl P (1986) Existence d'automates finis capables de s'accorder bien qu'arbitrairement connectés et nombreux. Int Comput Centre 5:245–261
Russell B (1935) The limits of empiricism. In: Proceedings of the aristotelian society, vol 36. Blackwell Publishing, The Aristotelian Society, pp 131–150
Savage J (1998) Models of computation: exploring the power of computing. Addison Wesley, Boston
Schönhage A (1969) Universelle Turing Speicherung. In: Dörr J, Hotz G (eds) Automatentheorie und formale Sprachen, Bericht einer Oberwolfachtagung. BI Mannheim (Oktober 1969), vol 3, pp 369–383

Schönhage A (1980) Storage modification machines. SIAM J Comput 9(3):490–508
Schütte K (1951) Beweisetheoretische erfassung der unend lichen induktion in der zahlentheorie. Math Ann 122:369–389
Shannon CE (1941) Mathematical theory of the differential analyser. J Math Phys MIT 20:337–354
Shelah S (1977) Decidability of a portion of the predicate calculus. Isr J Math 28:32–44
Sieg W (1994) Mechanical procedures and mathematical experience. Mathematics and mind. pp 71–117
Sieg W (1997) Step by recursive step: Church's analysis of effective calculability. Bull Symb Log 3(2):154–180
Sieg W (1999) Hilbert's programs: 1917–1922. Bull Symb Log 5(1):1–44
Sieg W (2008) Church without dogma-axioms for computability. In: Cooper S, Löwe B, Sorbi A (eds) New computational paradigms. Changing conceptions of what is computable. Springer, New York
Siegelmann HT, Fishman S (1998) Analog computation with dynamical systems. Physica D 120:214–235
Siegelmann HT, Sontag ED (1995) On the computational power of neural nets. J Comput Syst Sci 50(1):132–150
Smith WD (1999) History of "Church's theses" and a manifesto on converting physics into a rigorous algorithmic discipline. Technical report, NEC Research Institute. Avalaible on http://www.math.temple.edu/~wds/homepage/works.html
Smullyan RM (1993) Recursion theory for metamathematics. Oxford University Press, Oxford
Smullyan RM (1994) Diagonalization and self-reference. Oxford University Press, Oxford
Spaan E, Torenvliet L, van Emde Boas P (1989) Nondeterminism fairness and a fundamental analogy. Bull EATCS 37:186–193
Tait WW (1967) Intentional interpretations of functionals of finite type I. J Symb Log 32:198–212
Tarski A (1931) Sur les ensembles définissables de nombres réels I. Fundam Math 17:210–239
Turing AM (1936) On computable numbers, with an application to the entscheidungsproblem. Proc Lond Math Soc 42(2):230–265
Turing AM (1939) Systems of logic based on ordinals. Proc Lond Math Soc 45:161–228
Urban C (2001) Strong normalisation for a Gentzen-like cut-elimination procedure. Typed lambda calculi and applications, vol 2044. Springer, Berlin, pp 415–429
van Dalen D (2008) Logic and structure. Springer, Berlin
van den Dries L (2003) Generating the greatest common divisor, and limitations of primitive recursive algorithms. Found Comput Math 3(3):297–324
van Leeuwen J (ed) (1990) Handbook of theoretical computer science, vol A: Algorithms and Complexity, B: Formal Models and Semantics. The MIT Press and Elsevier
von Neumann J (1951) A general and logical theory of automata. In: Jeffries L (ed) Cerebral mechanisms in behavior–the hixon symposium. Wiley, pp 1–31. Reprinted in Papers of von Neumann J (1987) Computing and computer theory. Aspray W, Burks A (eds) MIT Press
von Neumann J (1966) Theory of self-reproducing automata. University of Illinois Press. (Edité et complété par Arthur W. Burks à partir d'un cours donné par von Neumann en 1949)
Wack B (2005) Typage et déduction dans le calcul de réécriture. PhD thesis, Université Henri Poincaré, Nancy 1
Weihrauch K (2000) Computable analysis: an introduction. Springer, Berlin
Werner B (1994) Une théorie des constructions inductives. PhD thesis, Université Paris 7
Weyl H, Kirschmer G (1927) Philosophie der Mathematik und Naturwissenschaft. Oldenbourg Wissenschaftsverlag

Winograd T (1982) Thinking machines: can there be? are we? In: Sheehan J, Sosna M (eds) The boundaries of humanity: humans, animals, machines. University of California Press, Berkeley, pp 198–223. Reprinted in The Foundations of Artificial Intelligence, D. Partridge and Y. Wilks, Cambridge Univerity Press, 1991, pp 167–189

Yao AC-C (2003) Classical physics and the Church-Turing thesis. J ACM 50(1):100–105

Yunès J-B (2006) Fault tolerant solutions to the firing squad synchronization problem in linear cellular automata. J Cell Autom 1(3):253–268

Theoretical Computer Science: Computational Complexity

Olivier Bournez, Gilles Dowek, Rémi Gilleron, Serge Grigorieff, Jean-Yves Marion, Simon Perdrix and Sophie Tison

Abstract How much time, space and/or hardware resource does require an algorithm? Such questions lead to surprising results: conceptual simplicity does not always go along with efficiency. A lot of quite natural questions remain open, e.g., the famous $P = NP$ problem raised in 1970. The so elementary model of finite automata, adequately tailored to diverse data structures, proves to be a flexible and powerful tool in the subject whereas quantum computing opens astonishing perspectives. An elegant tool for proofs of lower bounds for time/space complexity is a totally different notion of complexity: Kolmogorov complexity which measures the information contents.

O. Bournez
LIX, Ecole Polytechnique/CNRS, Palaiseau, France
e-mail: Olivier.Bournez@lix.polytechnique.fr

G. Dowek
INRIA, LSV, ENS-Cachan, Cachan, France
e-mail: gilles.dowek@ens-paris-saclay.fr

R. Gilleron
INRIA Lille Nord Europe, Lille, France
e-mail: remi.gilleron@inria.fr

S. Grigorieff
IRIF, Universite Paris Diderot/CNRS, Paris, France
e-mail: seg@irif.fr

J.-Y. Marion
Universite de Lorraine, CNRS, LORIA, Nancy, France
e-mail: jean-yves.marion@loria.fr

S. Perdrix
CNRS, INRIA, Universite de Lorraine LORIA, Nancy, France
e-mail: Simon.Perdrix@loria.fr

S. Tison (✉)
University of Lille, Inria, CNRS, CRIStAL, Lille, France
e-mail: Sophie.Tison@univ-lille.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_2

1 Introduction

1.1 Complexity of Human Beings, Complexity of Machines

Some figures, given in 1948 at the Hixxon Symposium (von Neumann 1951).

> With any reasonable definition of what constitutes an element, the natural organisms are very high complex aggregations of those elements. The number of cells in the human body is somewhere of the general order of 10^{15} or 10^{16}. The number of neurons in the central nervous system is somewhere of the general order of 10^{10}. We have absolutely no past experience with systems of this degree of complexity. All artificial automata made by man have numbers of parts which by any comparable schematic count are of the order of 10^3 to 10^6.

As for 2016, these figures are as follows…

- *Number of cells in the human body.* Past estimates have been revised to 3.0×10^{13} human cells in the 70 kg "reference man".
- *Complexity of a human brain.* A human brain contains about 85×10^9 neurons, of which about 20% are in the cerebral cortex and 80% in the cerebellum. Each neuron has on average 7, 000 synaptic connections to other neurons. The brain of a three-year-old child has about 10^{15} synapses. This number declines with age, stabilizing by adulthood. Estimates vary for an adult, ranging from 10^{14} to 5×10^{14} synapses.
- *Number of transistors on an integrated circuit (IC).* As of 2016, the largest transistor count in a commercially available single-chip processor is over 7.2×10^9—the Intel Broadwell-EP Xeon. In other types of ICs, such as field-programmable gate arrays, Intel's (previously Altera) Stratix 10 has the largest transistor count, containing over 30×10^9 transistors (cf. Wikipedia). In 2018 AMD commercialized a single-chip processor with 19.2 billion transistors.
- *Petascale computing* ($peta = 10^{15}$). As of 2016, the world's most powerful machine is the Chinese supercomputer "93 petaflop Sunway TaihuLight". It can perform up to 93×10^{15} calculations per second. It has more than 10.5×10^6 processing cores and 40, 960 nodes and it runs on a Linux-based operating system. In 2018 IBM and Nvidia have built a 200 petaflop machine built.

These figures witness an impressive technological growth since 1948: in less than 70 years the largest number of transistors in a circuit has been multiplied by about 270 millions! A phenomenon precised in Moore's law:

> The number of transistors in a dense integrated circuit doubles approximately every two years.

Stated by Gordon E. Moore in 1965, this law has proved accurate since 1958 upto 2015. Since then this technological explosion has started to decline: in 2019 the two years delay has become a three years delay.

As a result, the scale complexity of machines now exceeds that of some constituents of the human being. This gives new elements for the discussion on the

seminal themes of AI, cf. Sect. 1.1 in Chapter "Theoretical Computer Science: Computability, Decidability and Logic".

Also, the efficiency issue becomes crucial when dealing with such huge masses of data (hundreds of petabytes) and petaflop computing machines. As we shall see, simplicity and efficiency do not really go together. Efficient algorithms to structure, access, update and query such gigantic sets of data may be quite sophisticated and their conception requires much care.

An approach to feasibility—hence to usefulness—of algorithms is to estimate the order of magnitude of the time and space resources they use. This is the subject of algorithmic complexity in computer science.

1.2 What We Pick and Choose

In this second chapter we continue our survey of theoretical computer science. Among a large variety of topics, we picked a few ones.

Sections 1 and 2 survey computational complexity. In the previous chapter, we considered a set or a function to be computable if some Turing machine can compute it. Now, the computation time can be totally irrealistic with respect to physical limitations, even with the most permissive one (such as the age of the universe…). Complexity theory takes care of the notion of resource used to compute a function or a set, especially the time and space resources. Thus, the mathematical formalization of computability is somehow refined to a more practical notion, that of complexity of computation.

Section 3 deals with automata and their many applications in computer science. Besides its intrinsic elegance (a question of taste, of course), automata theory has met great success because most problems about them can be decided using quite weak resources. This property is the reason that makes automata so useful in practice.

Section 4 briefly discusses the very promising subject of quantum computing.

Section 5 looks at a different notion of complexity: Kolmogorov complexity which measures the information contents. It turns out to be a very convenient tool for complexity lower bounds.

The interested reader will find complementary information on these topics in the following much praised references: Papadimitriou (1994) is a classic, Jones (1997) stresses on programming theory, Savage (1998) details different computation models. Some other references (Sipser 2006; Goldreich 2008; Arora and Barak 2009).

2 A Finer Look at Computability: Complexity Theory

2.1 Physical Resources Limitations

Time for Gods, Time for Human Beings.
Ulysses renounces Calypso island, this miniature paradise and renounces eternity to return to his old identity, that of a man who ages and dies. Greek mythology teaches us that there at least two kinds of time: a stilled eternal time for gods and a linear finite time for human beings.

Computability theory deals with the sole question of the existence of an algorithm to solve a given problem with no reference to time. On the opposite, complexity theory deals with the computation time. Somehow, entering complexity theory means leaving Olympus and going back to a linear perception of time which then becomes a basic parameter in computations.

How Many Units of Time Since the Big Bang: 10^{41}.
The longest duration A which has some physical interpretation is the age of the universe. In the Big Bang cosmological model, this is about fifteen billions years, i.e. $A = 15 \times 10^9 \times 365.25 \times 24 \times 3600 \approx 4,734 \times 10^{17}$ s.

As for the shortest duration τ which makes sense, physicists consider that it is the time for a photon to travel (at speed of light $c \approx 300,000$ km/s) a distance equal to the diameter of a proton $d = 10^{-15}$ m. Thus, $\tau = \dfrac{d}{c} = \dfrac{10^{-15}}{3 \times 10^8} \approx 3,333 \times 10^{-24}$ s.

As a consequence, the number $N = \dfrac{A}{\tau} = \dfrac{4,734 \times 10^{17}}{3,333 \times 10^{-24}} \approx 10^{41}$ represents the highest number of units of time since the Big Bang.

Observe that $10^{41} = 2^{136.199} < 2^{2^{7.098}}$.

Amount of Matter in the Universe: $\leq 10^{80}$ **elementary particles**.
Observe that $10^{80} = 2^{265.75} < 2^{2^{8.054}}$.

Complexity Theory as "Feasible" **Computability**.
The above figures show that algorithms which require exponential time or space are irreallistic. This stresses how fundamental it is to know the order of magnitude of time and space complexity and to look for algorithms running in time and space bounded by functions much lower than the exponential function.

In particular, the complexity of learning (see Chapter "Statistical Computational Learning" of Volume 1), optimisation, planification (see Chapter "Meta-Heuristics and Artificial Intelligence" of Volume 2) and reasoning (see Chapter "Reasoning about Action and Change" of Volume 1) problems is a crucial question of IA in order to develop feasible (exact or approximate) algorithms.

2.2 Complexity of a Few Particular Problems

Many problems admit algorithms running in time surprisingly much lower than a priori expected. We mention a few ones.

2.2.1 The O, Ω and Θ Notations

Knuth (1976) popularized variants of classical mathematical notations which prove quite convenient in computer science. Let f, g be functions $\mathbb{N} \to [0, +\infty[$.

- $f = O(g)$ (or $f \in O(g)$) means that there exists $c > 0$ such that $f(n) \leq c\, g(n)$ for all $n \in \mathbb{N}$.
- $f = \Omega(g)$ (or $f \in \Omega(g)$) is equivalent to $g = O(f)$, i.e. there exists $c > 0$ such that $f(n) \geq c\, g(n)$ for all $n \in \mathbb{N}$.
- $f = \Theta(g)$ (or $f \in \Theta(g)$) is equivalent to the conjunction of conditions $f = O(g)$ and $g = O(f)$, i.e. there exists $c, d > 0$ such that $d\, g(n) \leq f(n) \leq c\, g(n)$ for all $n \in \mathbb{N}$.

2.2.2 Example 1: Multiplication of Natural Integers

There are many very different algorithms to multiply two integers a and b. Suppose $d \geq 2$ and the base d expansions of a and b have respectively n and m digits. The "school teacher algorithm"—the one everybody knows—computes the product $a \times b$ via nm multiplications of digits and $0(nm)$ additions of digits:

1. To produce the trapezoidal set of digits consisting of m lines of n or $n + 1$ digits, one performs exactly nm multiplications of digits and at most $(n - 1)m$ additions of carries (since there is no carry for the first digit of each line).
2. To sum the two first lines and then sum the result with the third line and so on up to the m-th line, one performs at most $(n - 1)(m - 1)$ additions of digits and at most $(n - 1)(m - 1)$ additions of carries.

In particular, considering the case $m = n$, multiplying two numbers with n digits requires $O(n^2)$ elementary operations on digits. Such an algorithm is said to have complexity $O(n^2)$.

Surprisingly as it may seem, there are algorithms for multiplication with much lower complexity. Of course, such algorithms are conceptually more complex: somehow, one exchanges time with brainpower...

Karatsuba and Ofman (1962) get complexity $O(n^{\log_2(3)})$ (observe that $n^2 = n^{\log_2(4)}$ whereas $n^{\log_2(3)} = n^{1,5849\cdots}$). Their algorithm uses the so-called "divide and conquer" strategy: iteratively divide each one of the integers a, b. To simplify, assume $d = 2$ and let $p = \lceil n/2 \rceil$ and $q = \lfloor n/2 \rfloor$ (so that $p + q = n$). For some α, γ we have $a = 2^p \alpha + \beta$ and $b = 2^p \gamma + \delta$ with $0 \leq \beta, \delta < 2^p$. Observe that $\alpha, \gamma < 2^q$ since $2^p \alpha \leq a < 2^n = 2^p\, 2^q$ and $2^p \gamma \leq b < 2^n = 2^p\, 2^q$. Thus,

$$a \times b = (2^p\alpha + \beta) \times (2^p\gamma + \delta) \;=\; \alpha\gamma\; 2^{2p} + (\alpha\delta + \beta\gamma)\; 2^p + \beta\delta \qquad (1)$$

$$= \alpha\gamma\; 2^{2p} + [(\alpha + \beta)(\gamma + \delta) - \alpha\gamma - \beta\delta]\; 2^p + \beta\delta\;. \qquad (2)$$

Line (1) shows that a multiplication of two numbers with n digits can be reduced to four multiplications of numbers with $\lceil n/2 \rceil$ digits. Iterating this process leads to a useless algorithm running in time $O(n^2)$…Now, observe that $\alpha + \beta$, $\gamma + \delta$ have at most $p + 1$ digits since $\alpha + \beta, \gamma + \delta \leq (2^p - 1) + (2^q - 1) < 2^{p+1}$. Thus, using line (2), we reduce to three multiplications of numbers with at most $\lceil n/2 \rceil + 1$ digits and, iterating this process, one gets Karatsuba's algorithm running in time $O(n^{\log_2(3)})$.

The algorithm in Schönhage and Strassen (1971) is much more sophisticated: based on the discrete Fourier transform, it runs in time $O(n \log(n) \log(\log(n)))$.

The best known algorithm (Fürer 2009) runs in time $O(n \log(n) \log^*(n))$ where $\log^* : \mathbb{R} \to \mathbb{N}$ is the function which, applied on input x, gives the least number k of times one has to iterate the log function on input x in order to be at most 0. In other words, $\log^*(x) = 0$ if $x \leq 0$ and $\log^*(x) = 1 + \log^*(\log(x))$ if $x > 0$. Thus,

when $x \in$	$]-\infty, 0]$	$]0, 1]$	$]1, 2]$	$]2, 4]$	$]4, 16]$	$]16, 256]$	$]256, 2^{256}]$	$]2^{256}, 2^{2^{256}}]$
$\log^*(x) =$	0	1	2	3	4	5	6	7

Though this function $\log^*$ goes to infinity when n grows, it does so with extreme slowness. In particular, in view of the limits of time seen Sect. 2.1, the $\log^*(n)$ factor will never exceed the constant 6 in practice.

Let us mention that Karatsuba's algorithm is implemented in every symbolic computation software. This is not the case for Schonhage-Strassen and Fürer algorithms because, up to now, the constant hidden in the $O(\ldots)$ notation is too huge and spoils their performance.

2.2.3 Example 2: Evaluation of a Polynomial

The complexity of a problem is very sensitive to the exact conditions in which a solution is to be used. We illustrate this phenomenon with the problem of evaluating a degree 4 polynomial $P(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$. The usual algorithm (Horner 1819) rewrites the polynomial as follows:

$$P(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + xa_4)))$$

and evaluates it via 4 multiplications and 4 additions. It has been proved that this complexity is optimal as concerns the number of additions (Ostrowski 1954) and also that of multiplications (Pan 1966). Nevertheless, if one has to evaluate this polynomial for a large number N of values of x (for instance to get the graph of the polynomial), this is no more the best algorithm. To perform the N evaluations, Hörner's algorithm requires $4N$ additions and $4N$ multiplications. Now, some preconditioning allows to perform the N evaluations with $5N$ additions and $3N$ multiplications. Since addi-

tion is much easier than multiplication, the exchange of N multiplications with N additions is welcome. Let us detail this preconditioning

$$P(x) = a_4[(x+\alpha)(x+\beta)]^2 + \gamma(x+\alpha)(x+\beta) + (x+\delta) \quad (3)$$
$$= (x+\delta) + (x+\alpha)(x+\beta)[\gamma + a_4(x+\alpha)(x+\beta)] \quad (4)$$

where $\alpha, \beta, \gamma, \delta$ are such that

$$\begin{aligned} 2a_4(\alpha+\beta) &= a_3 \\ a_4((\alpha+\beta)^2 + 2\alpha\beta) + \gamma &= a_2 \\ 2a_4\alpha\beta(\alpha+\beta) + \gamma(\alpha+\beta) + 1 &= a_1 \\ a_4\alpha^2\beta^2 + \alpha\beta\gamma + \delta &= a_0 \end{aligned}$$

The price of this preconditioning is independent of N and negligible for large values of N. Indeed, a few arithmetic operations give $\alpha, \beta, \gamma, \delta$: the first equation yields $\alpha + \beta$, then the next two ones are a linear system giving $\alpha\beta$ and γ. Having their sum and product, we get α, β as the roots of the equation $z^2 - (\alpha+\beta)z + \alpha\beta = 0$. Finally, the last equation simply gives δ.

Such a beneficial preconditioning can also be done with the general degre n polynomial (cf. Knuth 1981, p. 471–475). This even leads to an evaluation process using at most $\lfloor n/2 \rfloor + 2$ multiplications (observe that $\lfloor n/2 \rfloor + 2 < n$ for $n \geq 5$).

2.2.4 Exemple 3: The Fast Fourier Transform (FFT)

The discrete Fourier transform of a sequence $(x_0, \ldots, x_{n-1})$ of real or complex numbers is the sequence $(y_0, \ldots, y_{n-1})$ such that $y_k = \sum_{\ell=0}^{\ell=n-1} x_\ell \, e^{(-2i\pi/n)k\ell}$. In other words, letting $w = e^{-2i\pi/n}$,

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & w & w^2 & \ldots & w^{n-1} \\ 1 & w^2 & w^4 & \ldots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \ldots & w^{(n-1)^2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

This notion has a lot of important technological applications, for instance in coding and decoding of pictures. Thus, an efficient way to compute the discrete Fourier transform is of utmost importance. The obvious algorithm runs in quadratic time $O(n^2)$. The FFT reduces this time to $O(n \log n)$ and constitutes one of the most important algorithm in computer science. Indeed, due to the very slow growth of the log function, $O(n \log n)$ time is "almost" linear time in practice. Though the general public completely ignores this fact, the FFT constitutes a true technological revolution.

2.2.5 Example 4: Shortest Path in a Graph

To find a shortest path between two nodes in a directed graph with n nodes and a arcs, time $O(a + n \log n)$ is sufficient (Dijkstra 1959).

2.3 Complexity Theory

The "Turing Award Lectures" by Rabin (1987) and Cook (1983) are remarkable papers on the genesis of complexity theory that we strongly recommend.

2.3.1 Is it Possible to Define the Complexity of a Given Problem? …Sometimes

The natural approach to get the complexity of a problem is to establish upper and lower complexity bounds for the problem. In practice, the methods to get upper bounds are quite different from those to get lower bounds. Also, it is usual to get upper and lower bounds "up to a linear factor" in order to avoid taking care of some insignificant details of the algorithm. We can then use the notations $O(f)$ and $\Omega(g)$ (cf. Sect. 2.2.1) for an upper bound f and a lower bound g up to a linear factor.

If these bounds f, g are linearly related then we can use the Θ notation and say that the complexity of the problem is $\Theta(f)$ or, equivalently, $\Theta(g)$. One also says that the order of magnitude of the complexity is f or, equivalently, g.

A Simple Example: Sorting. Fix some integer $n \geq 1$. Given an enumeration $\sigma = (x_1, \ldots, x_n)$ of a linearly ordered set X, we want to sort these n elements as an increasing sequence. The number of comparisons of two elements occurring in the run of a sorting algorithm gives a significant order of magnitude of its running time.

Assume we successively compare some pairs x_i, x_j and let $\mathscr{E}_\parallel$ be the family of enumerations of X which cannot be distinguished from σ by the k first of these comparisons. Observe that $\mathscr{E}_\parallel$ is the whole family of $n! = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$ enumerations of X. Also, the $(k+1)$-th comparison discriminates at most half of the enumerations in $\mathscr{E}_\parallel$. Thus, $\mathscr{E}$ contains at most $n!/2^k$ enumerations. In order to characterize the given enumeration σ we need that σ be the unique enumeration in $\mathscr{E}_\parallel$. This requires that $n!/2^k \leq 1$ hence that the number k of comparisons be at least $\log_2(n!)$. Since $\log_2(n!) \geq \log_2((n/2)^{n/2}) = (n/2)\,(\log_2(n) - 1)$, we see that any sorting algorithm requires an $\Omega(n \log n)$ number of comparisons.

There are many sorting algorithms which use $O(n \log n)$ comparisons, for instance the "merge sort". Thus the complexity of sorting (viewed as the number of comparisons which are done) is $\Theta(n \log n)$.

2.3.2 But…Not Always: Blum's Speed-Up Theorem

A naive idea is that, for every problem, there is a best algorithm, or a best one up to a linear factor. Alas, things are far more complicated.

Blum Speed-Up Theorem (1967). *Let $\alpha : \mathbb{N} \to \mathbb{N}$ be a computable function which is monotone nondecreasing and unbounded. Then there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that, if a function $C : \mathbb{N} \to \mathbb{N}$ is an upper bound of the time (or space) complexity of some algorithm computing f then so is the function $\alpha \circ C$.*

Clearly, this theorem is all the more interesting when α when it has very slow growth, i.e. is much smaller than the identity function: $\alpha(p) \ll p$. For instance, letting α be the log function, Blum speed-up theorem insures that there is a function f for which if C is an upper bound of the time (or space) complexity of some algorithm computing f then so is $\log(C)$ hence also $\log(\log(C))$, $\log(\log(\log(C)))$…There are explicit examples of problems for which the complexity admits such a logarithmic speed-up, for instance the set of true formulas in the monadic second-order theory of the structure $\langle \mathbb{N}; =, \in, Succrangegle \rangle$ or the structure $\langle \Sigma^*, \mathscr{P}(\Sigma^*); =, \in, (Succ_a)_{a\in\Sigma} rangegle \rangle$ (cf. Chapter "Theoretical Computer Science: Computability, Decidability and Logic", Sects. 6.1.2 and 6.1.4).

2.3.3 Complexity and Computation Models

The complexity of a problem can be very sensitive to the particular computation model we consider. For instance, palindrome recognition (words which are read the same way from left to right or from right to left) requires quadratic time (i.e. $O(n^2)$ where n is the length of the word) when done on a usual one tape, one head Turing machine (a result which admits an elegant proof (Paul 1979) using Kolmogorov complexity, cf. Sect. 5). Now, on Turing machines with two heads, linear time complexity is easy to obtain: move head 2 to the end of the word and then read the word left to right with head 1 and right to left with head 2. Linear time can also be similarly obtained with two tapes, each one with one head.

Fortunately, reasonable sequential models (cf. Chapter "Theoretical Computer Science: Computability, Decidability and Logic", Sects. 2.2 and 2.3) are robust in the following sense: any such model can simulate any other one at polynomial time cost (most often quadratic time cost). Some authors (Slot and ven Emde Boas 1984, the monography (Arora and Barak 2009)) express this robustness as a *strong variant of Church thesis*:

> Every computation model which can be physically implemented can be simulated by a Turing machine at polynomial cost.

However, such versions of Church thesis are much more controversial than the versions seen in Chapter "Theoretical Computer Science: Computability, Decidability and Logic" Sect. 2.7. In particular, when comparing analogical machines with digital ones, the proof of this strong Church thesis presented in Vergis et al. (1986) requires simplifications which have been questioned.

2.3.4 Time Complexity and Space Complexity

For sequential algorithms, the most popular computational resources are time and space, cf. Cobham (1965) and Cook (1983): time is the number of elementary steps in the run of the algorithm and space is the maximum needed memory to store all data at any computation step. Now, *elementary step* and *unit of space* are far to be absolute notions. Indeed, they are chosen in dependence with the context of computation. Let's look at some examples.

For evaluation of a polynomial, cf. Sect. 2.2.3, addition and multiplication of numbers are seen as elementary steps. As a consequence, storing an integer only needs a unit of space…

For multiplication of integers, cf. Sect. 2.2.2, we considered that adding two integers is an elementary step. This has been questioned: for RAM's (cf. Chapter "Theoretical Computer Science: Computability, Decidability and Logic", Sect. 2.3), Cook and Reckhow (1973) suggested that the cost for time and space be logarithmic in the length of the representation.

In contrast with the above example, for addition of integers in a given basis, cf. Sect. 2.2.2, an elementary step will be an addition of two digits.

For Turing machines, an elementary step is simply considered to be a transition. Now, a transition is itself a complex operation: first, read the scanned cell and then, using the transition function, determine the next state and store it, determine the symbol which is to replace the scanned one and replace it, finally, determine the move of the head and move it. We could go farther in the analysis and consider that reading the scanned cell is also a complex operation involving some kind of pattern matching.

In general, for a given algorithm, the complexity heavily depends on the chosen model of computation: Turing machines or RAM's for non parallel algorithms, PRAM's, Boolean circuits or cellular automata for parallel algorithms. Besides time and space, one can also consider another resource: the number of processors and their topology, cf. (van Emde Boas 1990).

In conclusion, it seems fair to say that the notions of elementary step and unit of space are arbitrary choices…which look reasonable in a particular context.

This fact is well illustrated in Gurevich's context of Abstract State Machines where an elementary step is then a transition step leading from one state to the next one by applying the program. Indeed, running the program involves computing the values of a bounded number of terms using the current values of the environment and the primitive operations which can be arbitrarily complex. Thus, "elementary" is relative to primitive operations considered as oracles with unit cost.

Worst-Case Complexity versus Average-Case Complexity. Worst-case complexity gives an upper bound which may be attained for very peculiar cases, either quite rare or never met in practice. This is why average-case complexity is in many cases much more significant (Levin 1986). The sole problem with average-case complexity is far more difficult to obtain than the worst-case one.

Parameterized Complexity. Fixing a particular parameter in the input may sometimes drastically lower the complexity. This is the subject of parameterized complexity (Cook and Reckhow 1973).

Axiomatic and Complexity Theory. Let us mention that Manuel Blum (1967), see also (Seiferas 1990), defined an axiomatic system to develop a complexity theory. In particular, this theory applies in the framework of every computation model satisfying the axioms. Since this is the case of all usual computation models, this theory has a very large scope. For instance, Blum's speed-up theorem in Sect. 2.3.2 is proved in this theory.

2.4 Complexity Classes

The variety of models of computation leads a vast family of complexity classes, i.e. classes of problems obtained by bounding some resources in a given computation model. Deciding which classes are equal has proved to be quite complex. In particular, there is a lot of unsolved very natural questions of the following forms: compare complexity classes associated to different computation models, compare complexity classes associated to different resources on the same computation model.

2.4.1 Reduction and Completeness

An important concept is that of completeness. A problem A in a complexity class $\mathscr{C}$ is $\mathscr{C}$-complete if it belongs to $\mathscr{C}$ and every problem B in $\mathscr{C}$ can be "simply" reduced to A. A very popular notion of reduction is as follows: assuming A, B are sets of words in some fixed alphabet Σ, then B reduces to A if there is a function $f : \Sigma^* \to \Sigma^*$ which is computable in polynomial time and such that $B = f^{-1}(A)$, i.e. a word $x \in \Sigma^*$ is in B if and only if the word $f(x)$ is an A. Another popular reduction is obtained by replacing polynomial time computability (for f) by logarithmic space. This is a finer notion of reduction since logarithmic space complexity implies polynomial time complexity.

2.4.2 The Polynomial Time Complexity Class *PTIME*

This is the complexity class consisting of all problems which are computable in polynomial time, i.e. for which can be solved by some algorithm running in polynomial time on some Turing machine. Polynomial time is relative to the size of the input where size means length for a word, logarithm for an integer (i.e. length of the binary representation), number of nodes and arcs for a graph, …

This class is robust: the same class is obtained if Turing machines are replaced by RAM's or Kolmogorov machines or Schönhage machines (cf. Sect. 2.2 in Chapter

"Theoretical Computer Science: Computability, Decidability and Logic"). Also, it is closed under polynomial time reductions.

There are many complete problems in this class: deciding if an Horn clause is satisfiable, computing the value of a monotone Boolean circuit, deciding if a context-free language is empty, etc.

PTIME is often considered as the class of " tractable" problems. This is a reasonable claim as long as the degree of the polynomial for the time complexity is quite low. However, an algorithm working in time n^{100} will match one working in time $2^{0.0000001 \times n}$ only for quite large values of n…

Bounding the degree, one consider subclasses $DTIME(n^\alpha)$ of *PTIME*, where α is any positive real. Clearly, $PTIME = \bigcup_{\alpha>0} DTIME(n^\alpha)$ and a hierarchy theorem holds true: $DTIME(n^\alpha) \subset DTIME(n^\beta)$ whenever $\alpha < \beta$.

2.4.3 The Non-Deterministic Polynomial Time Complexity Class *NP*

This is the non-deterministic version of *PTIME*. It is defined with a variant notion of Turing machines: non-deterministic Turing machines. For such machines, at each step there may be several possible transitions and the machine "chooses" one of them randomly. Thus, for a given input there may be a lot of distinct runs. These runs can be seen as the branches of the tree of successive possible transitions.

An input is accepted by a non-deterministic machine if there is at least one run which accepts it (though there can be a lot of different runs which reject).

Such a non-deterministic Turing machine runs in polynomial time if all runs (either accepting or rejecting) halt within some fixed polynomial time bound.

A problem is in *NP* if it is the set of accepted inputs of some non-deterministic Turing machine which runs in polynomial time.

A priori this sounds like fantasy computation…However, a lot of usual algorithms have non-deterministic features and face some choices (for instance, to get a covering tree for an input undirected graph, one starts by non-deterministically picking some vertex in the graph, then one chooses some edge, etc). Either these choices are done via some random function which works as an "oracle" (usual programming languages offer a random function based on current time) or the algorithm is completed by some effective listing of all possible "oracles" for the needed choices.

Last and not least, it also turns out that non-determinism is a very fruitful mathematical notion.

This complexity class *NP* was independently introduced by Cook (1971) and Levin (1973) and they both proved the existence of *NP*-complete problems. Richard Karp papers then the classical book by Garey and Johnson (1979) give an impressive list of *NP*-complete problems relative to diverse domains: logic (3-SAT), graph theory (vertex cover, clique), optimization (knapsack problem). See Chapter "Reasoning with Propositional Logic—From SAT Solvers to Knowledge Compilation" of Volume 2 for information on SAT Solvers.

As in the deterministic case, one can bound the degree and consider the class $NTIME(n^\alpha)$, for $\alpha > 0$, of problems which can solved by a non-deterministic

Turing machine in time n^α. We also have $NP = \bigcup_{\alpha>0} NTIME(n^\alpha)$ and a hierarchy theorem holds true: $NTIME(n^\alpha) \subset NTIME(n^\beta)$ whenever $\alpha < \beta$.

2.4.4 The *PTIME* $\stackrel{?}{=}$ *NP* Problem

This famous problem (also known as $P = NP$) is not a purely theoretical problem: its solution could have practical consequences in daily life (Cook 2003). A negative answer, i.e. $PTIME \neq NP$, would show the soundness of the hypothesis on which some cryptographic methods are based. A positive answer, i.e. $PTIME = NP$, if it had a reasonable proof, would allow tractable proofs in some logical systems hence would bring remarkable applications in automatic deduction.

A priori, it seems plausible that non determinism could seriously lower the time complexity of a problem but, up to now, this is "terra incognita". The sole known result in this vein is the following inclusion: $DTIME(n) \subseteq \Sigma_4 - ATIME(n/log^*(n))$ where $\Sigma_4 - ATIME$ refers to time complexity with 4-alternating Turing machines, [1] a notion which leads to a priori much larger classes than $NTIME$ (but this is also an open problem), and $\log^*$ is the very slow growing function seen in Sect. 2.2.2 supra.

The $PTIME \stackrel{?}{=} NP$ problem goes back to 1970 and is still an open question…The strict inclusion $DTIME(n) \subset NTIME(n)$ (Paul et al. 1983, cf. also the book Balcázar et al. 1990) is the sole analog result which has been solved up to now. However, there is no known explicit example of a problem in $NTIME(n)$ but not in $DTIME(n)$. In fact, the proof of this strict inclusion is based on the strict inclusion in $\Sigma_4 - ATIME(n/log^*(n))$ and a general hierarchy theorem for deterministic time but this proof is a proof by contradiction hence it gives no explicit example.

The above considerations give some idea why the $PTIME \stackrel{?}{=} NP$ problem is so difficult…Another puzzling result related to this problem is Levin's optimal algorithm for *NP* problems (see Gurevich (1988) for detailed explanations).

2.4.5 The Polynomial Space Complexity Class *PSPACE*

This complexity class consists of all problems which can be solved by a Turing machine such that, for every input, the number of visited cells during the run is bounded by a fixed polynomial in the length of the input.

[1] Alternating Turing machines are the non-deterministic ones with two special features: (1) there are two kinds of states: the existential ones and the universal ones; this gives two kinds of transitions according to the state from which the transition is done, (2) an input is accepted if there is a subtree of the tree of possible transitions (cf. supra) such that each non terminal existential transition has exactly one son in the subtree and all the sons of a universal transition in the subtree are also in the subtree. Such a machine is Σ_4 if the root transition is existential and on each branch there are at most four blocks of successive transitions of the same (existential or universal) kind.

There are many *PSPACE* complete problems: the set QBF of true quantified Boolean formulas (where the quantification is over the set $\{true, false\}$ of truth values), a generalization of the Go game, parsing context-free grammars,…

There are a few nice equality relations between some space and time complexity classes (Chandra et al. 1981):

$$PSPACE = NP\text{space} = Parallel\ PTIME = APTIME$$

where *NP*space is the non-deterministic version of *PSPACE*, and *Parallel PTIME* is polynomial time on parallel RAM machines and *APTIME* is polynomial time on alternating machines (cf. Footnote supra). Thus, the space analog of the $PTIME \stackrel{?}{=} NP$ problem has a positive solution. Indeed, this equality comes from a nice inclusion proved by Savitch (1970) between deterministic and non-deterministic space complexity classes: $NSPACE(n^\alpha) \subseteq DSPACE(n^{2\alpha})$ for any $\alpha > 0$.

2.4.6 The Complexity Zoo (Aaronson et al. 2010)

"*When open questions generate plethora of definitions…*". The classification of complexity classes has proved to be a kind of entomologist's work. Due to pending open questions, more and more classes have been introduced: more than five hundreds classes up to now…

Inside the class *PTIME*, besides the $DTIME(n^\alpha)$'s seen supra, $\alpha > 0$, the classes $NLOGSPACE$ and NC^i, AC^i are worth mentioning (Karp and Ramachandran 1990).

$NLOGSPACE$ is the class associated to non-deterministic Turing machines which visit $O(\log n)$ cells on inputs of length n. Examples of complete problem for $NLOGSPACE$: accessibility in directed graphs, 2-SAT.

NC^i ("Nick's class" in reference to Nick Pippenger) and AC^i are classes associated to parallel computation with uniform sequences of Boolean circuits having a polynomial number of gates and depth $O((\log(n)^i)$ (the depth is the length of the longest directed path from an input node to the output node, it is also the running time): in NC^i the OR and AND gates have two inputs, whereas in AC^i they can have an arbitrary number of inputs (i.e. unlimited fan-in) (thus, in one step an arbitrary long conjunction or disjunction of Boolean values can be computed). In particular, AC_0 is the class associated to constant-depth unlimited fan-in Boolean circuits. A interesting complete problem in NC^1 is the evaluation of an instantiated Boolean formula (Buss 1987).

Since a polynomial fan-in OR (resp. AND) can be simulated using a tree of binary OR's (resp. AND's) of depth $O(\log n)$, we have $NC^i \subseteq AC^i \subseteq NC^{i+1}$.

Known inclusion relations between these classes ($\subset$ means strict inclusion):

$$AC^0 \subset NC^1 \subseteq LOGSPACE \subseteq NLOGSPACE \subset PTIME \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$

The strict inclusion, $AC^0 \subset NC^1$ is an important result by Furst, Saxe and Sipser (1984). Another known strict inclusion is $PTIME \subset EXPTIME$. Whether some other inclusions are strict is an open problem.

2.4.7 The Quasi-Linear Time Complexity Class $DTIME(n(\log n)^{O(1)})$

Though it does not occur for combinatorial problems, this class contains the Fast Fourier Transform (cf. Sect. 2.2.4) and the usual arithmetical functions on integers (cf. Sects. 2.2.2 and 2.2.3). It is also of outmost importance in recursive analysis since Richard Brent (1967) proved that all usual analytical functions (exponential, sine, cosine, Γ,…) are computable in time $n(\log n)^{O(1)}$ (this uses the fact that multiplication of integers can be done in time $n(\log n)^{O(1)}$, cf. Sect. 2.2.2).

2.5 Characterization of Complexity Classes

Complexity classes are associated to particular computation models and resource bounds. It turns out that many of them can also be defined with no reference to any computation model or any resource bound. This is the subject of "Implicit Computational Complexity".

The class *PTIME* has so be characterized in many ways: via bounded primitive recursion (Cobham 1965), via fixed-point logic (Immerman 1986), via second-order logic with restricted comprehension axiom (Leivant 1991), via ramified recursion (Bellantoni and Cook 1992; Leivant 1994), via a restriction to Lambda-Calculus (Leivant and Marion 1993), via linear types (Girard 1998), via first-order logic with a restriction on universal quantification (Marion 2001).

Parsons (1970, 1971, 1972) proved that the class of all primitive recursive functions can be seen as that of functions provably total in first order arithmetic with induction axioms restricted to existential formulas (i.e. for any primitive recursive function f, the relation $f(x) = y$ is definable by a formula $\phi(x, y)$ such that the formula $\forall x\ \exists! y\ (\phi(x, y)$ (where $\exists!$ means "there exists exactly one") is provable in this constrained arithmetic.

A famous result due to Fagin (1974) characterizes *NP* as those relations definable by existential second-order formulas.

2.6 Complexity and Data Representation: Avižienis Parallel Addition

The way data are represented may have a strong impact on the complexity of problems. Who would use the unary representation of integers to multiply integers? Or

Roman numerals, a variant using addition and subtraction of digits I, V, X, L, C, D, M (for 1, 5, 10, 50, 100, 500, 1000)? Nowadays, for most people the right way to represent integers is to use the positional representation, i.e. Hindu-Arabic numerals using ten digits $0, 1, \ldots, 9$ or the base b variant using digits $0, 1, \ldots, b-1$, where $b \geq 2$ is any fixed integer. However, surprisingly as it may seem, a more sophisticated representation is used in computers. Replacing the digits $0, 1, \ldots, b-1$ by $-a, -a+1, \ldots, -1, 0, 1, \ldots, a-1, a$ in the positional base b representation, one can represent every integer if $2a+1 \geq b$. If $2a+1 > b$ then there are several representations: for instance, $24 = 4^2 + 2 \times 4 = 2 \times 4^2 + (-2) \times 4$. Why use such systems? As is well-known, with the positional representation, an addition involves carries which may propagate. This simple fact forces to add the successive digits sequentially, hence in linear time in the length of the numbers to be added. Nevertheless, Algirdas Avižienis (1961), an American Lithuanian researcher, showed that one can avoid carry propagation and get a parallel addition in constant time if $b \geq 3$ and $b+2 \leq 2a+1 < 2b$. This is implemented in computers with $b = 4$ and $a = 2$.

3 Finite Automata

Finite automata or finite-state machines are an abstract computational model having a central role in theoretical computer science. The study of finite automata has its roots in the theory of computation, in programming languages, in complexity theory, in logic and in formal language theory. Its origin can be arbitrarily fixed in 1954 with the Kleene's theorem stating the equivalence between regular expressions and finite automata. Indeed, the theorem was the first of many results showing the equivalence between finite automata and other formalisms based on automata, on grammars, on logic and on algebra. Since 1954, finite automata have been thoroughly studied. Finite automata are now a standard notion for every computer scientist. But, it should be noted that finite automata and their extensions are still used and studied in many domains, such as language processing, verification, database theory and bio-informatics, with many recent exciting results.

The purpose of this section is to present a "tour d'horizon" of finite automata theory showing some of the most important results among them equivalence results between many formalisms, introducing most useful extensions of finite automata, and giving examples of applications in different domains. Among the different approaches, we will focus on finite automata as a computational model with an algorithmic perspective. Obviously, many choices have been done by the authors and, while assuming our choices, we apologize for the many interesting works not presented here. Important bibliographic sources that we have used are: books by Hopcroft and Ullman, among them (Hopcroft and Ullman 1979); Chapters "Heuristically Ordered Search in State Graphs"–"Reasoning with Propositional Logic—From SAT Solvers to Knowledge Compilation" from Volume 2 in van Leeuwen and editor (1990) by Perrin, Berstel et Boasson, Salomaa, Thomas et Courcelle; many chapters in Rozenberg and Salomaa

(1997); Sakarovitch's book (2009); an online book on tree automata (Comon et al. 2007); a book on weighted automata (Droste et al. 2009).

3.1 Finite String Automata

Finite string automata (or finite word automata) are much more restrictive than Turing machines. Indeed, they process finite strings over a finite alphabet from left to right, there is no auxiliary memory and they are acceptors outputting Yes or No. The abstract model of finite automata is quite simple: a finite number of states, a finite number of rules where each rule gives a new state for a given state and a given letter, initial and final conditions defined by particular states. More formally, given a finite alphabet of letters A, a *finite string automaton* $\mathscr{M}$ is defined to be a quadruplet (Q, I, F, Δ) where Q is a finite set of states, I is a finite subset of Q of initial states, F is a finite subset of Q of final states, and Δ is a finite set of transition rules which is a subset of $Q \times A \times Q$. One can also view a finite string automaton $\mathscr{M}$ as a directed graph (Q, Δ) where Q is the set of vertices and Δ is the set of edges together with two special subsets of vertices, one for the initial vertices and one for the final vertices. It is common to draw finite string automata as directed graphs using drawing conventions for the initial vertices and the final vertices. It should be noted that the above definition corresponds to non deterministic automata: there can be several initial states, and given a state q and a letter a there can be several rules of the form (q, a, q') in Δ. A finite automaton is *deterministic* if there is (at most) one initial state and for every pair (q, a) in $Q \times A$, there is at most one rule (q, a, q') in Δ. Also, to avoid dead-end computations, a finite automaton is said to be *complete* if there is at least one initial state and for every pair (q, a) in $Q \times A$, there is at least one rule (q, a, q') in Δ.

It remains to define how a finite automaton accepts or not a finite string u over the alphabet A. A *run* of $\mathscr{M}$ over a string $u = a_1 \dots a_n$ in A^* is a string $q_0 q_1 \dots q_n$ in Q^* such that q_0 is an initial state in I, and, for every $i < n$, (q_i, a_{i+1}, q_{i+1}) is a rule in Δ. The reader can easily note that, for a complete finite automaton (respectively deterministic finite automaton), there exists at least one run (respectively at most one run) for every string u. A string u is said to be *accepted or recognized* by $\mathscr{M}$ if there exists a run of $\mathscr{M}$ over u such that the last state of the sequence is a final state in F. Such a run is called a *successful run*. The set of strings (also called language) accepted by $\mathscr{M}$ is denoted by $L(\mathscr{M})$. A language L (a subset of A^*) is said to be *recognizable* if there is a finite automaton $\mathscr{M}$ such that $L = L(\mathscr{M})$.

Below is an example of a complete deterministic finite automaton with initial state i, final states p, q and set of rules (a rule (q, a, q') in Δ is written $q \xrightarrow{a} q'$):

$$\begin{array}{cccc} i \xrightarrow{a} p & p \xrightarrow{a} p & q \xrightarrow{a} r & r \xrightarrow{a} r \\ i \xrightarrow{b} i & p \xrightarrow{b} q & q \xrightarrow{b} i & r \xrightarrow{b} r \end{array}$$

It is easy to verify that it defines the language of finite strings that do not contain aba as a substring.

We now give a list of base results on finite automata. Proofs can be easily found in many textbooks on language theory. For every finite string automaton, there exists a complete finite automaton recognizing the same language. For every finite string automaton with n states, there exists a complete deterministic finite automaton recognizing the same language with at most 2^n states. The above definition of finite automata implies that, along a run, a letter is processed when a rule is applied. A first extension is to consider that an automaton can change the current state without processing any letter. This can be modeled by adding in the abstract model definition so-called ε-rules of the form (q, ϵ, q') where ϵ denotes the empty string, and by adapting the computational model accordingly. Again, for every finite string automaton with ε-rules, there exists a finite automaton without ε-rules recognizing the same language. Thus, all above variants of finite string automata are equivalent meaning that every model corresponds to the class of recognizable languages.

Pumping lemmas express important properties of recognizable languages and are useful to prove that certain languages are not recognizable. The basic version states that, for every recognizable language L, there exists an integer k such that, for every string u in L of length greater than k, u can be factorized as $u = lvr$ and, for every $n \geq 0$, $lv^n r$ is in L. It can be used to prove that the language $\{a^n b^n \mid n \geq 0\}$ and parenthesis languages (also called Dyck languages) are not recognizable.

Closure and decidability properties of the recognizable languages are noteworhty, the class of recognizable languages being closed under various and numerous operations. The class of recognizable languages is closed under union, intersection and complement. The class is also closed under string substitution—each letter is substituted by a regular language, and thus by homomorphism, i.e. string substitution such that each letter is replaced by a single string, and it is closed by inverse string homomorphism. It is decidable whether a recognizable language is empty (resp. finite). The complexity of the emptiness problem is discussed below and depends on the type of automaton given as input for defining the recognizable language.

As said in the introduction, recognizable languages can also be defined in an algebraic way. For this, the residual (or Brzozowski derivative) $u^{-1}L$ of a language L for a string u is the set of strings v such that uv is in L. This allows to define the binary relation $\equiv_L$ over strings by $u \equiv_L v$ if $u^{-1}L = v^{-1}L$. It is an equivalence relation compatible with right concatenation over A^*, i.e. it is a right congruence relation for the algebraic structure A^* with concatenation. The *Myhill–Nerode Theorem* states that a language L is recognizable if and only if the number of congruence classes of the relation $\equiv_L$ is finite. A consequence is the existence of a minimal deterministic automaton for every recognizable language L where minimal is defined with respect to the number of states. Several algorithmic constructions of the minimal automaton have been proposed. A general approach following the Myhill–Nerode Theorem is to merge undistinguishable states, i.e. states recognizing the same language: e.g Hopcroft's algorithm successively refines partitions of the set of states of a deterministic automaton, starting from the initial partition $(F, Q - F)$. Brozowski's approach is quite different: starting from a possibly non-deterministic automaton,

the algorithm reverses the edges and determinizes the automaton, obtaining a deterministic automaton for the mirror language, and then repeating this reversal and determinization.

We now present other formalisms which have been shown equivalent to finite automata from the expressiveness point of view. We will conclude by some complexity results for decision problems for recognizable languages which depend on the chosen representation of the input recognizable language.

Regular Grammars. Finite automata are an abstract computational model defining acceptors for strings and allowing to define recognizable languages. Another perspective is to define a generative process for strings. This perspective has been taken for defining regular grammars in Natural Language Processing. Indeed, regular grammars are defined to be a set of production rules (or rewrite rules) in order to generate strings. Formally, given a finite alphabet A, a *grammar* $\mathscr{G}$ is defined to be a triplet (N, P, S) where N is a finite set of special symbols called non terminals (or variables), P is a finite set of production rules of the form $l \to r$ where l and r are strings over $A \cup N$, and S is the start symbol (or axiom) in N. The generative process is starting from S and iteratively replacing a substring l of the current string by the substring r for a rule $l \to r$ in P. The process is repeated until the string belongs to A^*, i.e. until no non terminal occurs in the string. Without restrictions on the rules, grammars correspond to the Type-0 grammars in the Chomsky hierarchy and correspond to the class of recursively enumerable languages that can be recognized by a Turing machine (see Sect. 2.4 in Chapter "Theoretical Computer Science: Computability, Decidability and Logic"). Regular grammars correspond to the Type-3 grammars in the Chomsky hierarchy where rules satisfy the following very restrictive conditions: every rule $l \to r$ in P is such that l is a non terminal and r is a letter in A, possibly followed by a non terminal. It is easy to show, in a constructive way, that the class of languages generated by regular grammars is equal to the class of recognizable languages.

Regular Expressions. Another perspective is to consider descriptions of languages with so-called regular expressions introduced by Kleene (1956). Regular expressions are defined with set operations over string languages: concatenation defined by $X.Y = XY = \{w \mid w = uv, u \in X, v \in Y\}$, union denoted by $+$ in infixed notation, and the star operation in postfixed notation defined by $X^* = \{w \mid w = u_1 \dots u_n, n \geq 0, u_1, \dots, u_n \in X\}$, i.e. the infinite union of the X^n. A regular expression defines a language and a language is said to be a *rational language* if it can be defined by a regular expression.

The Kleene's Theorem Kleene (1956) states the equivalence between rational languages and recognizable languages. Due to many applications in many different fields (programming languages, databases, document description languages, …), algorithms for transforming regular expressions in finite automata and vice versa have been thoroughly studied. We just sketch the base ideas in this section.

First, let us consider the *construction of a finite string automaton given a regular expression*. A proof-theoretic construction, known as Thompson's construction, is to define an automaton construction for each of the set operations concatenation,

union and star. This is quite easy using finite string automata with ε-transitions. But, the number of ε-transitions can be high and their elimination is costly. A more efficient construction proposed by Glushkov (1961) produces a non deterministic automaton whose states correspond to the positions in the expression, but whose number of transitions can be quadratically larger than the Thompson automata. This construction has many applications and it has been improved in many ways for particular classes of regular expressions, see McNaughton and Yamada (1960), Berry and Sethi (1986), Brüggeman-Klein (1993, 1998).

Second, let us consider the inverse *construction of a regular expression from a finite string automaton*. Let us consider set variables X_i, one such variable X_i for each state q_i of the input automaton. Each variable X_i is designed to capture the set of strings reaching q_i when starting from an initial state. One can write a set of equations of the form $X_i = \sum_j X_j a_{ij}$ or $X_i = \sum_j X_j a_{ij} + \varepsilon$ when q_i is an initial state, whenever there is a rule (q_j, a_{ij}, q_j). Such a system can be solved using replacements and the fact that the unique solution of an equation $X = XY + Z$, when $\varepsilon \notin Y$, is ZY^*. Ehrenfeucht and Zeiger (1976) have shown that there exist finite automata with n states for which the smallest equivalent regular expression has a size exponential in n.

The *star height of a regular expression* is, roughly speaking, the nesting depth of the star set operator in the expression. The *star height of a regular (or recognizable) language* is the minimum star height of a regular expression defining the language. The question whether the star height can be bounded has been answered negatively by Eggan (1963), even for binary alphabets as proved by Dejean et Schützenberger (1966). The second question is to compute the star height of an input regular language given by a finite automaton. The problem was open for a long period, a non elementary algorithm was proposed by Hashiguchi (1988), a much more efficient one has been proposed by Kirsten in 2005, the tool STAMINA[2] proposes an implementation.

Monadic Second Order Logic (MSO). A string of length n can be modeled by a first order structure where the domain is the set of positions $\{0, \ldots, n-1\}$, with the successor function and the natural linear order on positions, and, for every letter a in the alphabet A, the unary predicates P_a collecting the positions where the corresponding letter a occurs. Let us consider the first order language with first order variables x, y, ... ranging over positions in word models, and built from atomic formulas of the form $x + 1 = y$ (the position following x is y), $x < y$ (the position x is before y) and $P_a(x)$ (letter a occurs at position x) for every letter a in A. Let us also consider second order variables, denoted by X, Y, ...for finite sets of positions, and the corresponding predicates. Adding the usual connectives and quantifiers over first order variables and second variables and interpreting over finite strings defines the so-called monadic second order logic over finite strings. Such a logic allows to define properties of strings and so string languages as sets of strings satisfying a formula. A language is said to be *MSO-definable* if it is definable by an MSO-formula without free variables. Büchi (1960b) and Elgot (1965) have proven the equivalence between MSO-definable languages and recognizable languages. Indeed,

[2] http://stamina.labri.fr/

first, from a finite automaton, it is easy to build an MSO-formula encoding the automaton and its behavior. For the other direction, the proof uses closure properties of recognizable languages, under union, complement and projection. It is worth noticing that each alternation of quantifiers implies to compute the complement language and, consequently, an exponential procedure due to the determinization process. This is unavoidable as satisfiability of MSO logic over finite strings is non elementary.

Thus, MSO-logic over finite strings is decidable as one can compute, given as input an MSO-formula without free variables, a corresponding automaton and then decide emptiness of the recognized language. As a consequence the theory of integers with successor and quantification over sets is decidable (see Sect. 6.1.2 in Chapter "Theoretical Computer Science: Computability, Decidability and Logic"). Note that the result for finite strings can be viewed as a particular case of the same result over infinite strings proved afterwards by Büchi (1960a) and McNaughton (1966).

Alternating Automata. We have shown different equivalent formalisms based on machines, generative models, descriptive models and logics. We come back to the automaton point of view with alternating automata introduced in Chandra et al. (1981) and Brzozowski and Leiss (1980). The idea was to introduce a class of automata for which complementation was easy. Indeed, remember that complementation for non deterministic automata is exponential because a determinization procedure is required. Let us consider two rules (q, a, q_1) and (q, a, q_2) of a non deterministic automaton, starting from state q, for a string $u = av$ to be accepted, it is sufficient to have a successful run starting from q_1 or a successful run starting from q_2 when reading v. Moreover, for u to be discarded, all runs must be computed. The base idea of alternating automata is to handle in their definition these two cases: a disjunctive case for which one run satisfies a condition; and a conjunctive case for which all runs must satisfy a condition. Equivalent definitions have been proposed based on existential states and universal states or based on logic formulas. For every such formalism, the closure by complementation is easy and it is proved that alternating automata are equivalent to finite automata. Obviously, complexity issues remain because it has been proved in Chandra et al. (1981) that there exist alternating finite automata with k states for which a deterministic complete equivalent automaton has at least 2^{2^k} states.

Some Complexity Results. We present some complexity results depending on the input representation of the recognizable language. Note that we have given above some complexity bounds for the conversion between formalisms. For instance, deciding whether a string satisfies an MSO-formula is linear with respect to the size of the string and the size of an automaton for the formula but the construction of such an automaton is non elementary.

- Decide whether a string belongs to a language is *DLOGSPACE*-complete for a deterministic automaton, *NLOGSPACE*-complete for a finite automaton, P-complete for an alternating automaton.

- Decide the equivalence of automata, i.e. equality between the two recognized languages is *NLOGSPACE*-complete for deterministic automata, *PSPACE*-complete for finite automata and for alternating automata.

In 2005, a new technique "antichain algorithm" has been designed, keeping the determinization step implicit for solving problems such as universality, inclusion or equivalence of automata (Wulf et al. 2006). More recently, Bonchi and Pous (2015) designed an optimized algorithm for proving language equivalence of nondeterministic finite automata, by introducing a very elegant technique: bisimulation up to congruence; their approach outperforms experimentally previous ones.

3.2 Beyond Finite String Automata

3.2.1 Weighted Automata and Rational Series

Finite string automata process input strings and compute Boolean outputs. A natural extension is to produce probabilistic outputs. This can be done by so-called *probabilistic automata* introduced by Rabin (1963), Paz (1971). The base idea is to add non negative real weights for initial states, final states and rules. The weight of a run is computed by multiplying the weight of the initial state, the weights of the rules used along the run and the weight of the final state. The weight of a string is the sum of weights over all successful runs over the input string. Introducing normalization conditions on weights and reachability conditions for states, it is proved that a probabilistic automaton defines a probability distribution over the set A^* of finite strings over an alphabet A. Moreover, the set of strings with non zero probability, sometimes called the support of the probability distribution, is a recognizable language. It should be noted that deterministic probabilistic automata are strictly less expressive than (non deterministic) probabilistic automata. Probabilistic automata have the same expressive power than Hidden Markov Models as shown by Denis et al. (2005).

The computation model for probabilistic automata considers sums of products of weights. Therefore weighted automata can be defined in a similar way with weights in a semiring $(K, +, \times, 0, 1)$. Then, a weighted automaton defines a function from the set A^* of finite strings into the semiring K. Such a function is defined to be a *recognizable series* and it has been shown that recognizable series are equivalent to rational series defined in an algebraic way (Berstel and Reutenauer 1982; Sakarovitch 2009; Droste et al. 2009).

3.2.2 Pushdown Automata and Context-Free Languages

Up to now, we have considered regular grammars which are Type-3 grammars in the Chomsky hierarchy. But a very important class of grammars is the Type-2 grammars,

also called context-free grammars, because they define context-free languages widely used for programming languages. A *context-free grammar* $\mathcal{G}$ is a grammar $\mathcal{G} = (N, P, S)$ such that, for every rule $l \to r$ in P, l is a non terminal in N. Along the generating process, every non terminal can be rewritten independently of its context hence the name "context-free". A *context-free language* is a language that can be generated by a context-free grammar. Every regular language is a context-free language and there are context-free languages, a prototypic example is parenthesis languages, which are not regular. Such languages show that automata for context-free languages must be able to check the opening and closing of parentheses. This is done by using a stack which can be "pushed down" leading to so-called *pusdown automata*. There is a huge number of scientific papers on automata and context-free languages. Base results can be found from (van Leeuwen 1990; Rozenberg and Salomaa 1997), but some important results are posterior. For instance, the equivalence problem for deterministic puhdown automata remained open for a long period and was solved by Sénizergues (2002). A very interesting subclass of pushdown automata has been defined in Alur and Madhusudan (2004, 2009): *visibly pushdown automata* (and their restriction to nested words, *nested word automata* where the input symbol determines when the pushdown automaton can push or pop. The corresponding class of visibly pushdown languages generalizes the class of recognizable languages while keeping many of its decision and closure properties and has many applications, e.g. for modelling unranked ordered tree languages and for processing XML documents.

3.2.3 Transducers: Automata with Output

Another extension to finite string automata is to produce an output string while reading the input string. Such an extension gave rise to Moore machines, to Mealy machines and, more generally, to string transducers. A *finite string transducer* is a finite string automaton for which an output string u over an alphabet B is associated with every rule (q, a, q'). Such a rule of a finite tree transducer is denoted by $(q, a/u, q')$ or by $q \xrightarrow{a/u} q'$. Finite string transducers process strings as finite string automata but at each application of a rule $(q, a/u, q')$ the string u is concatenated to the current output. Let us consider the deterministic string transducer with initial state i, set of final states Q, and the rules

$$\begin{array}{lll} i \xrightarrow{a/x} i & p \xrightarrow{a/x} q & q \xrightarrow{a/x} q \\ i \xrightarrow{b/y} p & p \xrightarrow{b/y} q & q \xrightarrow{b/xy} q \\ i \xrightarrow{c/yxy} q & p \xrightarrow{c/yy} q & q \xrightarrow{c/yy} q \end{array}$$

The transducer is taken from Schützenberger (cf. Eilenberg 1974, p. 305). It defines a bijection from $(a + b + c)^*$ into $(x + y)^*$ based on the following partitions:

$$(a+b+c)^* = a^* + a^*b + a^*(c + b(a+b+c))(a+b+c)^*$$
$$(x+y)^* = x^* + x^*y + x^*(yxy + y(x+y+yy))(x+xy+yy)^*$$

A finite string transducer defines a *rational relation* over $A^* \times B^*$. Finite transducers have been thoroughly studied in language theory. We refer the reader to the textbook by Sakarovitch (2009). We only give some milestone results showing differences between automata and transducers. Finite string transducers can be deterministic or not, may contain or not ε-rules. The equivalence of non deterministic string transducers without ε-rules was proved undecidable in (Griffiths 1968). The equivalence has been proved for several subclasses, e.g. finite-valued transducers (Culik and Karhumüki 1986). It is decidable whether a finite transducer is functional (Schützenberger 1975); every functional transducer is equivalent to an unambiguous one (Eilenberg 1974) and it is decidable whether it can be realized by a deterministic one (Berstel and Boasson 1979; Choffrut 1978). Equivalence of transducers remains decidable when the domain is restricted to a context-free language as a consequence of results in Plandowski's thesis (Plandowski 1995).

3.3 Automata Over Discrete Structures

3.3.1 Bottom-Up Tree Automata

Trees are ubiquitous in computer science with different definitions in graph theory, in machine learning or in computational biology. In this section, we consider finite ranked trees. A finite ranked tree has a root, every node has a label and a finite ordered number of sons fixed according to the label. More formally, we consider a finite ranked alphabet Σ in which each symbol has a fixed arity defining its number of children. The set of trees can be inductively defined by: every symbol of arity 0 (also called constant) is a tree; if a symbol f has arity n, and $t_1, \ldots, t_n$ are trees, then $f(t_1, \ldots, t_n)$ is a tree. A tree can be seen as a directed graph with a special node called the root, with one ingoing edge for every node, and with a number of outgoing edges defined by the arity of the symbol. By convention, a tree is drawn with the root at the top and leaves (symbols of arity 0) at the bottom.

The model of finite string automata can be extended to trees but, contrarily to the string case, defining the model and its computational process from the root to leaves (in a top-down mode) or from leaves to the root (in a bottom-up mode) implies different properties of tree automata. Let us consider the bottom-up mode. A *bottom-up finite tree automaton* $\mathscr{M}$ is a triplet $\mathscr{M} = (Q, F, \Delta)$ where Q is a finite set of states, F is a subset of Q of final states, and Δ is a finite set of rules of the form $f(q_1, \ldots, q_n) \to q$ where f is a symbol of arity $n \geq 0$. Let us note that initial states are not considered because rules for symbols of arity 0 have the form $a \to q$ and can be considered as initial rules. Such an automaton is said to be *deterministic* if there are no two rules with the same left-hand side. A *run* of an automaton $\mathscr{M}$ over

a tree t is a tree r with the same domain than t and for which the labeling in Q is compatible with rules of $\mathscr{M}$ over t. A tree t is said to be *accepted or recognized* by $\mathscr{M}$ if there exists a run r of $\mathscr{M}$ over t such that the root state of r is a final state in F. Such a run is called a *successful run*. The tree language (set of trees) accepted by $\mathscr{M}$ is denoted by $L(\mathscr{M})$. A tree language is said to be *recognizable* if there is a finite automaton $\mathscr{M}$ such that $L = L(\mathscr{M})$.

Many results can be extended from strings to trees: determinization of bottom-up tree automata, closure under Boolean operations, pumping lemmas, decidability of the emptiness problem. Regular tree grammars can be defined which define regular tree languages equivalent to recognizable tree languages. Regular expressions can be defined—in a more tedious way—and an analog of the Kleene's Theorem can be proven. Also, a Myhill-Nerode's Theorem can be proven leading to the notion of minimal deterministic bottom-up automaton. The MSO logic can be extended to finite trees, MSO-definable languages are the recognizable languages, and MSO logic over finite trees is decidable (Thatcher and Wright 1968). Note that this result, using an adequate notion of automata, has been extended to infinite trees in the fundamental *Rabin's Tree Theorem* (Rabin 1969). We now emphasize the differences between the case of strings and the case of trees.

Top-Down Tree Automata. They are defined by reversing arrows in the rules of a bottom-up automaton and by exchanging final states for initial ones. Thus a run can be defined in a similar way but trees are now processed in a top-down mode. Top-down tree automata also define recognizable tree languages. The difference comes from determinism. Indeed, a top-down tree automaton is said to be deterministic if, for every state q and every symbol f of arity n, there is at most one rule $q \to f(q_1, \ldots, q_n)$. And, it should be noted that deterministic top-down tree automata define a proper subclass of recognizable tree languages. For instance, the reader can easily show that the language $\{f(a, b); f(b, a)\}$ is recognizable but can not be recognized by a deterministic top-down automaton.

Tree Walking Automata. The base idea is to visit nodes following vertices of a binary tree considered as a graph. Roughly speaking, a tree walking automaton can check whether the current position is the root, a leaf, a right child or a left child. Rules allow to annotate the current position with a state and to remain in the current position or to go to the father node or to go to the right child if exists or to go to the left child if exists. A successful run on a tree t starts from an initial state at the root of t, applies rules and ends with a final state at the root of t. It is easy to show that tree languages accepted by walking tree automata are recognizable tree languages. The converse is false and has been a long-standing problem. Indeed, it has been shown that recognizable tree languages strictly include languages recognized by tree walking automata that also strictly include languages recognized by deterministic tree walking automata (Bojańczyk and Colcombet 2006, 2008).

Closure by Tree Homomorphism. Whereas, inverse tree homomorphisms preserve recognizability, recognizable tree languages are not closed by tree homomorphism. For this, let us consider the recognizable tree language $\{g(f^n(a)) \mid n \geq 0\}$ and the tree homomorphism defined by the rule $g(x) \to h(x, x)$, the image language

$\{h(f^n(a), f^n(a)) \mid n \geq 0\}$ is not a recognizable tree language. When the duplication of variables is forbidden, i.e. when considering linear homomorphisms only, the closure property is true. Deciding, given an homomorphism and a recognizable language, whether the corresponding image is recognizable—a long-standing open problem—is EXPTIME-complete (Creus et al. 2016). Let us note, that generalization of tree automata, so called tree automata with constraints have been defined and capture homomorphic images of recognizable tree languages.

First-Order Logic. As said before, MSO logic can be adapted for finite trees and MSO-definable tree languages correspond to recognizable tree languages. But the difference between the string case and the tree case appears when restricting MSO logic to first order (FO) logic. Indeed, in the string case, FO-definable tree languages correspond to star-free languages (McNaughton and Papert 1971), i.e. languages defined by regular expressions over concatenation, complementation and union, and they correspond also to aperiodic languages (Schützenberger 1965). The situation is quite more complex for trees. A decidable characterization of FO-definable tree languages is non trivial and has been obtained by Benedikt and Ségoufin (2009).

3.3.2 Graph Automata

Even if several notions of graph automata have been defined, generalizing tree automata to process graphs while keeping good closure and decision properties and a nice correspondence with logic, remains an open question. By using tree-decomposition, tree automata can be used for representing set of graphs of bounded tree-width and Courcelle's theorem states that every property definable in the monadic second-order logic of graphs can be decided in linear time on graphs of bounded treewidth. The interested reader can refer to the introduction by Courcelle to these difficult questions in Chapter "Reasoning with Propositional Logic—From SAT Solvers to Knowledge Compilation" of Volume 2 of (van Leeuwen and editor 1990) and in textbooks by Courcelle and Engelfriet (Courcelle 2010; Courcelle and Engelfriet 2012).

3.4 Automata and Applications

Automata have been used in many different domains. Here follow examples of these numerous applications.

Verification and Model-Checking. As already mentioned, there is a strong connection between logic and automata and verification is a major application of automata; e.g. tree automata have been designed a long time ago in the context of circuit verification. Roughly speaking, the automata-theoretic approach reduces the satisfiability and model-checking problems to standard decision problems about automata. Automata can be used both for modeling or abstracting the behaviours of a system, and for specifying the property to avoid: then model-checking reduces to checking

emptiness of the intersection of the two corresponding automata. E.g. properties to be verified are often described in temporal logics, such as linear temporal logic (LTL) and an equivalent Büchi automaton can be computed from a LTL formula. Extended models of automata have been designed to express more complex behaviours, such as timed automata -automata with real-valued variables, for modeling and verification of real time systems. The reader is referred to the abundant bibliography on verification and model-checking, e.g. (Baier and Katoen 2008).

Pattern Matching. The question is to define algorithms for searching substrings in strings. The Knuth, Morris and Pratt's algorithm (1977) was a first linear algorithm for string matching. It was based on finite string automata techniques. Due to the practical importance of this problem, for instance consider the `grep` and `egrep` Unix commands, many algorithms, some of them based on automata techniques, have beeen proposed for string matching and regular expression matching. An entry point is the chapter (Crochemore and Hancart 1997) by Crochemore and Hancart in Rozenberg and Salomaa 1997. It should be noted that the reserach domain is still very active with many questions for approximate string matching of small strings in very large databases of very long strings in bioinformatics because of the development of sequencing technologies.

Codes in Information Theory. Theory of codes has its origin in the theory of information by Shannon in the 1950's. One of its branch is the theory of error-correcting codes related to algebra and finite automata theory. Indeed, codes can be defined as formal languages (see Berstel and Perrin 1985). We refer the reader to the revised textbook by Reutenauer et al. (2009) for a complete overview of the subject.

Parsing. Parsing, in computer science, is testing whether a string belongs to a formal language defined by a formal grammar. Moreover, if the string belongs to the language, a parser produces a proof in the form of a data structure (a parse tree or a dependency graph). For context-free languages, the output is a parse tree which represents a derivation of the string according to the given context-free grammar. Many automata-based algorithms have been proposed using pushdown automata and they are used to define interpreters and compilers for programming languages.

Natural Language Processing. Automata-based algorithms have also been proposed for parsing texts using various types of grammars adapted to natural language. It should also be noted that string transducers and tree transducers have been defined and studied for the translation problem. Also, probabilistic context-free grammars have been defined. It was proved that it is decidable whether a probabilistic context-free grammar defines a probability distribution (Etessami and Yannakakis 2009). Probabilistic parsing allows to compute the most probable parse tree or the k-best parse trees.

Information Extraction. Extracting structured information from raw texts or from semi-structured documents (for instance HTML documents) can be seen as a problem of transforming texts or trees. Therefore, methods based on string transducers or on tree transducers have been proposed.

Semi-Structured Databases. Semi-structured documents, a prototypic example is XML documents, have a tree structure as defined above but a node may have an unbounded number of children. Finite tree automata and finite tree transducers have been adapted accordingly and finite string automata—see visibly pushdown automata mentioned before, and finite string transducers have been defined for their linear representations with markup languages. The interested reader can refer to Chapter "Artificial Intelligence in Biological Modelling" in Comon et al. (2007). Automata techniques are also used to process XPath queries. E.g., designing automata-based algorithms for querying streams of semi-structured documents is an area of active research.

Graph Databases. Graph databases have gained renewed interest in the last years due to the adoption of data formats like RDF and their applications in areas such as the Semantic Web. Typical queries select pairs of nodes connected by a path whose sequence of edge labels satisfies a given regular expression: this correspond to Regular Path Queries (RPQs). Extensions have been designed: e.g. conjunctive regular path query (CRPQs) are obtained by adding conjunctions and existential quantification over variables whereas two-way Regular Path Queries (2RPQ) extend RPQs by allowing inverse relations. Regular expressions with memory (REMs), based on register automata, have been defined for querying on graphs with data. Regular expressions can also be used for defining regular constraints on graph databases.

Description Logics. Automata-based approaches can be used for deciding satisfiability of Description Logics. The approach relies on the tree model property and builds a tree-automaton accepting e.g. the tree models of a concept w.r.t. a TBox. Satisfiability is then reduced to an emptiness test (see e.g. Baader et al. (2008) for a complete introduction).

Rewriting Systems and Automatic Deduction. Rewriting systems are introduced in Chapter "Semantic Web" by Dershowitz and Jouannaud in van Leeuwen and editor (1990). Important properties are termination and confluence. Automata have been used to solve decidability problems for rewriting systems. E.g. the theory of tree rewrite systems without variables has been proven decidable by means of automata techniques. For subclasses of rewriting systems, automata techniques have also been developed to build finite representations of the sets of descendants (resp. ancestors) of a recognizable language, allowing to decide "extended word problem". The interested reader is refered to Chapter "Databases and Artificial Intelligence" in Comon et al. (2007).

Compiler Design. The first step, lexical analysis, uses regular expressions to tokenize the input and can be designed by using finite string automata. The second step is parsing or syntactical analysis of a context-free language. Parsing, in computer science, is testing whether a string belongs to a formal language defined by a formal grammar. Moreover, if the string belongs to the language, a parser produces a proof in the form of a data structure (a parse tree or a dependency graph). For context-free languages, the output is a parse tree which represents a derivation of the string according to the given context-free grammar. Many automata-based algorithms have

been proposed using pushdown automata. Also tree automata can be used in compiler design for code selection or code optimization.

Grammatical Inference. For HMMs and probabilistic grammars, a first problem is, given the model (base automaton or rules) is to learn weights which can be done using Viterbi-like algorithms. Grammatical inference considers the more general problem to learn the structure for automata and both the structure and the weights for weighted automata. For the non probabilistic case, regular languages are not learnable from positive examples, i.e. when only examples in the language are given. With positive and negative examples, many learning algorithms construct the deterministic minimal automaton for the language. For the probabilistic case, the problem is to infer the probabilistic model from examples drawn according to the target distribution (defined by an unknown target probabilistic automaton or grammar). The above methods have been extended to the probabilistic case but it should be noted that more recent results use matrix (or tensor) factorization methods based on the formulation of automata computations with linear algebra. The reader is referred to de la Higuera (2010).

Control and Game Theory. Automata have been widely used in control theory. Also strategies in games and probabilistic games have been modeled by automata and probabilistic automata. Many domains such as decision theory, operations research and artificial intelligence use Markov decision processes as a mathematical tool. And Markov decision processes are related to probabilistic automata and more generally weighted automata. Note also that, as said above, hidden Markov models are also related to probabilistic automata. Thus, with these applications in mind, many recent works study decision problems and their complexity for weighted automata.

4 Quantum Computing

A Quantum Turing Machine. Though Richard Feynman (1960, 1984) was the first to guess the potential impact of quantum computers, the first model of quantum computation is the quantum Turing machine (QTM) introduced by Deutsch (1985). This machine is an extension of a probabilistic machine where every transition is done with a given amplitude (which is a complex number). QTM's use quantum features such as superposition of states, interference and entanglement. Bernstein and Vazirani (1993) proved the existence of a universal QTM, i.e. a QTM which can approximately simulate every QTM in an efficient way. Later, Deutsch (1989) also introduced quantum circuits. But, as proved by Yao (1993), they are polynomially equivalent to quantum Turing machines.

Among the fundamental properties of QTM, deciding whether the machine is halted or not, is a conceptually hard problem: measuring the configuration of a QTM can potentially dramatically alter its configuration. Solutions based on hating-qubits (Ozawa 2002) and classically controlled QTM (Perdrix and Jorrand 2006) have been introduced to solve this problem and also to account for more recent models of quantum computation based on measurements (Nielsen 2003; Perdrix 2005) and entangled

resources (Raussendorf and Briegel 2001; Danos et al. 2010). Measurement-based models are not only very promising in terms of physical implementations, they can also be used to reduce the quantum depth of a quantum computation up to a logarithmic factor compare to the quantum circuit model (Browne et al. 2011).

Algorithms and Quantum Complexity. Quantum computability has an important impact in algorithmic complexity. This has been proved by Deutsch (1985), Deutsch and Jozsa (1992). They considered the following problem. Let X be the set of functions $f : \{0, 1\}^n \to \{0, 1\}$ which are either constant or equilibrated in the sense that the sets $f^{-1}(0)$ and $f^{-1}(1)$ have the same number of elements. The problem is to decide whether a function in X is constant or equilibrated. Every classical deterministic algorithm for this problem has to question f at $2^{n-1} + 1$ elements (i.e. half of X plus one element) but there exists a quantum algorithm which questions f only once. Though this algorithm has no practical application, it is the first proof that quantum computation can beat classical computation.

The first very useful example has been found by Grover (1996). To find an element in a database with size n, every classical deterministic algorithm needs $\Omega(n)$ queries whereas Grover's quantum algorithm needs only $O(\sqrt{n})$ queries.

Several quantum algorithms provides a polynomial speed up, for instance for solving graph problems (Dürr et al. 2006; Magniez et al. 2005). General techniques, like quantum walks, have been introduced to design and study quantum algorithms (Ambainis 2007; Buhrman and Špalek 2006).

The main quantum complexity class is BQP (Bernstein and Vazirani 1993). Its definition is similar to that of the probabilistic class BPP. The classe BQP (resp. BPP) consists of all decision problems which can be solved in polynomial time by some quantum (resp. probabilistic) Turing machine with an error probability less than $1/3$.

It is known, that $PTIME \subseteq BPP \subseteq BQP \subseteq PSPACE$, cf. (Bernstein and Vazirani 1993). Since it is not known whether the inclusion of $PTIME$ in $PSPACE$ is strict or not, the same is true for BPP and BQP. Nevertheless, there are several quantum algorithms which seem to witness such a strict inclusion, e.g., Simon (1994) and Shor (1994). Shor's algorithm allow factorization and the computation of the discrete logarithm in polynomial time whereas no known classical algorithm for these problems runs in polynomial time. The Harrow-Hassidim-Lloyd (HHL) algorithm (Harrow et al. 2009) is a quantum algorithm solving linear equations. Intuitively, the algorithm can sample from the solution of some linear systems, providing an exponential speed-up over its classical counterpart.

Another important open question is to compare the classes NP and BQP. Up to now, there is no known polynomial time quantum algorithm solving an NP-complete problem.

5 Algorithmic Information Theory

How to measure the amount of information in a text? This is the question underlying this section. It turns out that this notion of complexity, though not a resource complexity, is a powerful tool to prove lower bounds for resource complexity. Reference books on this subject are Downey et al. (2010); Vereshchagin et al. (2013); Nies (2009), for short presentations see (Zenil 2011; Ferbus-Zanda and Grigorieff 2004, 2011, 2014).

5.1 Shannon Entropy

In ASCII code (American Standard Code for Information Interchange) every letter is coded by 8 bits hence a text of length n is represented by a sequence of $8n$ bits. Now, in a given text, letters are not uniformly distributed. Thus, coding letters with high frequency by fewer bits than those with low frequency, one can code the length n text by a sequence of less than $8n$ bits. How short can be such a code?

This is the question solved by Claude Shannon in his celebrated paper (1948) which opened a new subject, quantitative information theory, with tremendous applications (telephone, fax, etc.).

Shannon introduced a notion of *entropy*: if the frequencies of the different letters in the text are $f_1, \ldots, f_n$ then the associated entropy H is the real number $H = -(f_1 \log f_1 + \ldots + f_n \log f_n)$. Observe that $0 \leq H < 1$. Two of Shannon's main results insure that

1. every binary encoding of the text contains at least nH bits,
2. there exists some binary encoding of the text containing at most $n(H + 1)$ bits.

5.2 Kolmogorov Complexity

Around 1964, the Russian mathematician Andrei N. Kolmogorov (Kolmogorov 1965) completely revisited Shannon's theory. Rather than looking at the sole letter by letter encodings of a text T, he considers the text T globally and looks at the shortest length of a program (in any programming language or any machine) which does write down the text T. Care: we are not looking at the time complexity of the run of the program but solely at the length of the program considered as a word. This shortest length is called the *Kolmogorov complexity* of the text T.

A priori, it seems that Kolmogorov complexity is much dependent on the particular chosen programming language or computation model.

Of course, richer is the alphabet of programs, shorter are these programs: grouping letters by pairs, a program of length $2n$ in an alphabet Σ with k letters can be seen as having length n in the alphabet $\Sigma \times \Sigma$ with k^2 letters. To remove such a trivial

dependency, we consider that all programs are written in a binary alphabet and consider Kolmogorov complexity relative to binary programs.

This normalization to binary programs being done, how much does the Kolmogorov complexity depend on the chosen programming language or computation model? Kolmogorov proves a striking result: this dependency is merely $O(1)$, given two programming languages or computation models, there exists d such that for any text T, the number d bounds the difference between the Kolmogorov complexities of T relative to these two programming languages or computation models. As Kolmogorov nicely says:

> […] various "reasonable" possibilities of choice [of programming language or computation model] appearing here will lead to estimates of [Kolmogorov] complexities which differ by hundreds rather than by tens of thousands of bits. Therefore such expressions such as the "[Kolmogorov] complexity" of the text of the novel "War and Peace" must be viewed as being practically uniquely determined.

The Kolmogorov complexity of length n texts varies in a wide range.

Texts with Low Kolmogorov Complexity. If all letters of T are the same letter a then, using a loop, a simple program to write T is $\ll$*write n times the letter a*$\gg$. This program has length $O(1) + \log n$ since we need $\log n$ bits to write down the bound n of the loop. Thus, the Kolmogorov complexity of T is at most $O(1) + \log n$ where the $O(1)$ term does not depend on the particular text T and only reflects how to express the loop in the programming language. Same thing if the text T consists of the first n decimals of the real number π since there are plenty of algorithms to enumerate these digits.

Observe that if f is a computable function and $n = f(p)$ then, replacing n by $f(p)$ and adding a fixed subprogram to compute f, we get a program with length $O(1) + \log p$ which outputs T. In case $f(x) = 2^x$, this shows that the Kolmogorov complexity of T is at most $O(1) + \log(\log(n))$ when its length n is of the form 2^p for some p.

Texts with High Kolmogorov Complexity. An obvious upper bound of the Kolmogorov complexity of a length n text T written in an alphabet A with k letters is $O(1) + n \log k$:

- encoding letters of A by binary words with length $\lceil \log k \rceil$, we get a binary encoding U of T with length $n \lceil \log k \rceil$,
- the program $\ll$*write down the text in alphabet A which is encoded by U*$\gg$, outputs the text T and has length $O(1) + n \log k$ (since U has to be explicitly written).

This upper bound is optimal. Indeed, there are $2^p - 1$ binary words with length strictly less than p hence at most $2^p - 1$ programs with length strictly less than p. Looking at the outputs of these programs, we see that there are at most $2^p - 1$ words with Kolmogorov complexity strictly less than p. Thus, to get all texts of length n in an alphabet with k letters we must have $2^p - 1 \geq k^n$ hence $2^p > k^n$ and $p > n \log k$.

5.3 A Formal Notion of Random Infinite Sequence of Bits

Though the intuitive notion of random infinite sequence of bits underlies the whole subject of probability theory, it is simply ignored in its treatment based on measure theory (as axiomatized by Kolmogorov in 1933, English translation (1956)). Indeed, probability theory does not look at individual sequences but at sets of sequences hence gives no clue on this notion of random sequence. Kolmogorov was not satisfied by this situation and his development of Kolmogorov complexity was also aimed at answering this question.

Using tools from recursive analysis (a mix of mathematical analysis and computability theory), this notion of random sequence was first formalized in 1965 by the Swedish logician Per Martin-Löf, then a student of Kolmogorov. Around 1973, Levin and Chaitin, independently found a different approach, heavily based on Kolmogorov complexity, leading to the same notion of random sequence.

5.4 Practical Applications of Kolmogorov Complexity

As can be expected, though Kolmogorov complexity is a perfectly defined theoretical notion, it is not a computable function from words to integers…This can be seen as a stumbling block for any application but it is not the case, cf. Li and Vitanyi's book (2008). In particular, Kolmogorov complexity is a powerful tool to prove lower bounds for time and space complexity.

There are also fruitful applications in learning theory (Denis and Gilleron 2001; Chater and Vitanyi 2007), a central theme in IA, as witnessed by the 13 chapters in which this theme occurs: mainly Chapters "Statistical Computational Learning" and "Reinforcement Learning" in Volume 1, Chapter "Designing Algorithms for Machine Learning and Data Mining" in Volume 2 and Chapter "Artificial Intelligence and Pattern Recognition, Vision, Learning" in this volume, but also Chapters "Representations of Uncertainty in Artificial Intelligence: Beyond Probability and Possibility" and "Compact Representation of Preferences" in Volume 1, Chapters "Belief Graphical Models for Uncertainty Representation and Reasoning"–"Planning in Artificial Intelligence", "Constrained Clustering: Current and New Trends" in Volume 2 and Chapters "Artificial Intelligence in Biological Modelling", "When Artificial Intelligence and Computational Neuroscience Meet" and "Robotics and Artificial Intelligence" in this volume.

Also, around 2000, Paul Vitanyi, using computable approximations of Kolmogorov complexity based on usual compression utilities (gzip, …), created a remarkably efficient implementable method for document and pattern classification, (Cilibrasi and Vitanyi 2005; Cilibrasi et al. 2004; Li et al. 2004).

6 Conclusion

As a very natural refinement of computability theory, algorithmic complexity measures the resources used in a computation. As witnessed by this chapter, this is a delicate and difficult subject.

What are main issues in the subject?

Pending Open Questions. As seen in Sect. 2.4.4, very natural, easy to state questions are open since 1970…The plethora of complexity classes introduced in the literature witnesses how much difficult is the subject. Let us mention that the Clay Institute put the $PTIME \stackrel{?}{=} NP$ problem among the seven most important mathematical problems (for each of which it offers a one million $ award).

More and More Sophistication. As seen in Sect. 2.2, some efficient algorithms are really sophisticated and tricky and some may have overwhelming impact, such as the Fast Fourier Transform. Obviously, there is still much more to expect.

Also, as shown in Sect. 2.2.3, efficiency may be much dependent of the context: optimality is quite a fragile notion.

Good Old Automata Theory is Well and Alive. Going back to 1956, this theory proved to be incredibly flexible, cf. Sect. 3: avatars tailored for quite unexpected structures (infinite words, trees, infinite trees, transfinite ordinals, linear orderings…) are continuously appearing, allowing to decide more and more problems.

Quantum Computation. This is now one of the most important topic, with huge expectations (cf. Sect. 4)

Formalization of Algorithms and the Development of Complexity Theory. As for now, there is still much to expect in the development of complexity theory. In particular, Gurevich's formalization of the notion of algorithm (cf. Sects. 4.2 and 4.3 in Chapter "Theoretical Computer Science: Computability, Decidability and Logic") is a promising new approach for a quantitative analysis of resources used during a computation.

References

Aaronson S, Kuperberg G, Granade C (1992–2010). Complexity zoo. http://qwiki.stanford.edu/wiki/Complexity_Zoo

Alur R, Madhusudan P (2004) Visibly pushdown languages. In: Proceedings of the 36th annual ACM symposium on theory of computing, Chicago, IL, USA, June 13–16, 2004, pp 202–211

Alur R, Madhusudan P (2009) Adding nesting structure to words. J ACM 56(3)

Ambainis A (2007) Quantum walk algorithm for element distinctness. SIAM J Comput 37(1):210–239

Arora S, Barak B (2009) Computational complexity: a modern approach. Cambridge University Press

Avizienis A (1961) Signed-digit representations for fast parallel arithmetic. IRE Trans Electron Comput (now IEEE Trans Electron Comput) 10(3):389–400

Baader F, Horrocks I, Sattler U (2008) Description logics. In van Harmelen F, Lifschitz V, Porter B (eds). Handbook of knowledge representation, Chap. 3. Elsevier, pp 135–180
Baier C, Katoen J-P (2008) Principles of model checking (representation and mind series). The MIT Press
Balcázar JL, Diaz J, Gabarró J (1990) Structural complexity. EATCS monographs on theoretical computer science
Bellantoni S, Cook S (1992) A new recursion-theoretic characterization of the poly-time functions. Comput Complex 2:97–110
Benedikt M, Segoufin L (2009) Regular tree languages definable in FO and in FOmod. ACM Trans Comput Log 11:4:1–4:32
Bernstein E, Vazirani U (1993) Quantum complexity theory. In: Proceedings of the 25th annual ACM symposium on the theory of computation. ACM press, New York, pp 11–20
Berry G, Sethi R (1986) From regular expressions to deterministic automata. Theor Comput Sci 48:117–126
Berstel J, Boasson L (1979) Transductions and context-free languages. Teubner (ed)
Berstel J, Perrin D (eds) (1985) Theory of codes. Academic Press
Berstel J, Perrin D, Reutenauer C (2009) Codes and automata. Cambridge University Press, Encyclopedia of Mathematics and its Applications
Berstel J, Reutenauer C (1982) Recognizable formal power series on trees. Theor Comput Sci 18:115–148
Blum M (1967) A machine-independent theory of the complexity of recursive functions. J ACM 14(2):322–336
Bojańczyk M, Colcombet T (2006) Tree-walking automata cannot be determinized. Theor Comput Sci 350:164–173
Bojańczyk M, Colcombet T (2008) Tree-walking automata do not recognize all regular languages. SIAM J Comput 38:658–701
Bonchi F, Pous D (2015) Hacking nondeterminism with induction and coinduction. Commun ACM 58(2):87–95
Brent R (1976) Fast multiple-precision evaluation of elementary functions. J Assoc Comput Mach 23:242–251
Browne DE, Kashefi E, Perdrix S (2011) Computational depth complexity of measurement-based quantum computation. In: Theory of quantum computation, communication, and cryptography (TQC'10), vol 6519. LNCS, pp pp 35–46
Brüggemann-Klein A (1993) Regular expressions to finite automata. Theor Comput Sci 120(2):197–213
Brüggemann-Klein A, Wood D (1998) One-unambiguous regular languages. Inf Comput 142:182–206
Brzozowski JA, Leiss E (1980) On equations for regular languages, finite automata, and sequential networks. Theor Comput Sci 10:19–35
Büchi JR (1960a) On a decision method in a restricted second order arithmetic. In: Tarski A, Henkin L (eds) Proceedings of international congress on logic, methodology and philosophy of science. Stanford University Press, pp 1–11
Büchi JR (1960b) Weak second-order arithmetic and finite automata. Z Math Log Grundl Math 6:66–92
Buhrman H, Špalek R (2006) Quantum verification of matrix products. In: Proceedings of the seventeenth annual ACM-SIAM symposium on discrete algorithm, SODA'06. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. pp 880–889
Buss SR (1987) The boolean formula value problem is in ALOGTIME. In: ACM (ed) Proceedings of the 19th annual ACM symposium on theory of computing. ACM Press, New York, pp 123–131
Chandra AK, Kozen DC, Stockmeyer LJ (1981) Alternation. J Assoc Comput Mach 28:114–133
Chater N, Vitanyi P (2007) 'Ideal learning' of natural language: positive results about learning from positive evidence. J Math Psychol 51(3):135–163
Choffrut C (1978) Sur les traductions reconnaissables. ITA 12(3)

Cilibrasi R, Vitanyi P (2005) Clustering by compression. IEEE Trans Inf Theory 51(4):1523–1545
Cilibrasi R, Vitanyi P, de Wolf R (2004) Algorithmic clustering of music based on string compression. Comput Music J 28(4):49–67
Cobham A (1965) The intrinsic computational difficulty of functions. In: Bar-Hillel Y (ed) Proceedings of the international conference on logic, methodology, and philosophy of science, pages. North-Holland, Amsterdam, pp 24–30
Comon H, Dauchet M, Gilleron R, Löding C, Jacquemard F, Lugiez D, Tison S, Tommasi M (2007) Tree automata techniques and applications. http://tata.gforge.inria.fr/
Cook SA (1971) The complexity of theorem proving procedures. In: Proceedings third annual ACM symposium on theory of computing, pp 151–158
Cook SA (1983) An overview of computational complexity. Commun ACM 26(6):400–408
Cook SA (2003) The importance of the P versus NP question. JACM 50(1):27–29
Cook SA, Reckhow RA (1973) Time bounded random access machines. J Comput Syst Sci 7(4):354–375
Courcelle B (2010) Graph algebras and monadic second-order logic. Cambridge University Press
Courcelle B, Engelfriet J (2012) Graph structure and monadic second-order logic—a language-theoretic approach, vol 138 of Encyclopedia of mathematics and its applications. Cambridge University Press
Creus C, Gascón A, Godoy G, Ramos L (2016) The HOM problem is exptime-complete. SIAM J Comput 45(4):1230–1260
Crochemore M, Hancart C (1997) Automata for matching patterns. Handbook of formal languages, vol 2. Linear modeling: background and application. Springer, pp 399–462
Culik K, Karhumäki J (1986) The equivalence of finite valued transducers (on HDT0L languages) is decidable. Theor Comput Sci 47(3):71–84
Danos V, Kashefi E, Panangaden P, Perdrix S (2010) Semantic techniques in quantum computation. In: Extended measurement calculus. Cambridge University Press, pp 235–310
de la Higuera C (2010) Grammatical inference: learning automata and grammars. Cambridge University Press, New York, NY, USA
Dejean F, Schützenberger MP (1966) On a question from Eggan. Inf Control 9:23–25
Denis F, Dupont P, Esposito Y (2005) Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. Pattern Recognit 38:1349–1371
Denis F, Gilleron R (2001) Pac learning under helpful distributions. ITA 35(2):129–148
Deutsch D (1985) Quantum theory, the church-turing principle and the universal quantum computer. Proc R Soc Lond Ser A A400:97–117
Deutsch D (1989) Quantum computational networks. Proc R Soc Lond Ser A 425:73
Deutsch D, Jozsa R (1992) Rapid solution of problems by quantum computation. Proc R Soc Lond Ser A 439:553–558
Dijkstra E (1959) A note on two problems in connexion with graphs. Numer Math 1:269–271
Downey RG, Hirschfeldt DR (2010) Algorithmic randomness and complexity. Springer
Droste M, Kuich W, Vogler H (eds) (2009) Handbook of weighted automata. Springer
Dürr C, Heiligman M, Høyer P, Mhalla M (2006) Quantum query complexity of some graph problems. SIAM J Comput 35(6):1310–1328
Eggan LC (1963) Transition graphs and the star-height of regular events. Mich Math J 10:385–397
Ehrenfeucht A, Zeiger HP (1976) Complexity measures for regular expressions. J Comput Syst Sci 12:134–146
Eilenberg S (1974) Automata, languages and machines, vol A. Academic Press
Elgot CC (1965) Decision problems of finite automata design and related arithmetics. Trans Am Math Soc 98:21–52
Etessami K, Yannakakis M (2009) Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. J ACM 56(1):1–65

Fagin R (1974) Generalized first-order spectra and polynomial-time recognizable sets. In Karp R (ed) complexity of computation. SIAM-AMS Proceedings, vol 7. American Mathematical Society, pp 43–73
Ferbus-Zanda M, Grigorieff S (2004) Is randomness native to computer science? In Current trends in computer science, vol 2. World Scientific, pp 141–179
Ferbus-Zanda M, Grigorieff S (2011) Is randomness native to computer science? Ten years later. In Zenil H (ed) Randomness through computation. Some answers, more questions. World Scientific, pp 243–263
Ferbus-Zanda M, Grigorieff S (2014) Kolmogorov complexity in perspective, part 1: information theory and randomness. In Dubucs J, Bourdeau M (eds)Constructivity and computability in historical and philosophical perspective, vol 34 of Logic, epistemology, and the unity of science. Springer, pp 57–94
Feynman RP (1960) There's plenty of room at the bottom: an invitation to open up a new field of physics. Eng Sci (California Institute of Technology) 23(5):22–36
Feynman RP (1984) Quantum-mechanical computers. J Opt Soc Am B 1:464
Fürer M (2009) Faster integer multiplication. SIAM J Comput 39(3):979–1005. Earlier version in proceedings of the 39th annual ACM ACM STOC 2007 conference, pp 57-66
Fürsr ML, Saxe JB, Sipser M (1984) Parity, circuits, and the polynomial-time hierarchy. Math Syst Theory 17(1):13–27
Garey MR, Johnson DS (1979) Computers and intractability. In: Freeman WH (ed) A guide to the theory of NP-completeness
Girard J-Y (1998) Light linear logic. Inf Comput 143(2):175–204
Glushkov VM (1961) The abstract theory of automata. Russ Math Surv 16:1–53
Goldreich O (2008) Computational complexity: a conceptual perspective. Cambridge University Press
Griffiths TV (1968) The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines. J ACM 15(3):409–413
Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th annual ACM symposium on theory of computing. ACM Press, New York, pp 212–219
Gurevich Y (1988) Kolmogorov machines and related issues. Bull EATCS 33:71–82
Harrow AW, Hassidim A, Lloyd S (2009) Quantum algorithm for linear systems of equations. Phys Rev Lett 103(15):150502
Hashiguchi K (1988) Algorithms for determining relative star height and star height. Inf Comput 78:124–169
Hopcroft JE, Ullman JD (1979) Introduction to automata theory, languages, and computation. Addison-Wesley
Horner WG (1819) A new method of solving numerical equations of all orders, by continuous approximation. Philos Trans R Soc Lond 2:308–335
Immerman N (1986) Relational queries computable in polynomial time. Inf Control 68(1–3):86–104
Jones ND (1997) Computability and complexity, from a programming perspective. MIT press
Karatsuba AA, Ofman Y (1962) Multiplication of many-digital numbers by automatic computers. Dokl Akad Nauk SSSR 145:293–294
Karp RM, Ramachandran V (1990) Parallel algorithms for shared memory machines. In: Leeuwen JV (ed), Handbook of theoretical computer science A: algorithms and complexity. Elsevier Science Publishers and The MIT Press, pp 870–941
Kleene SC (1956) Representation of events in nerve nets and finite automata. Princeton University Press, pp 3–42
Knuth D (1976) Big omicron and big omega and big theta. SIGACT News, Apr–June, pp 18–24
Knuth DE (1981) The art of computer programming, vol 2: seminumerical algorithms, 2nd edn. Addison-Wesley
Knuth DE, Morris J, Pratt V (1977) Fast pattern matching in strings. SIAM J Comput 6(2):323–350
Kolmogorov AN (1956) Foundations of the theory of probability. Chelsea Publishing Company

Kolmogorov AN (1965) Three approaches to the quantitative definition of information. Probl Inform Transm 1(1):1–7
Leivant D (1991) A foundational delineation of computational feasiblity. In: Proceedings of the sixth IEEE symposium on logic in computer science (LICS'91). IEEE, Computer Society Press
Leivant D (1994) Predicative recurrence and computational complexity I: word recurrence and poly-time. In: Clote P, Remmel J (ed), Feasible mathematics II. Birkhäuser, pp 320–343
Leivant D, Marion J-Y (1993) Lambda calculus characterizations of poly-time. Fundam Inform 19(1,2):167,184
Levin L (1973) Universal search problems. Probl Inf Transm 9(3):265–266
Levin L (1986) Average case complete problems. SIAM J Comput 15(1):285–286
Li M, Chen X, Li X, Ma B, Vitanyi P (2004) The similarity metric. IEEE Trans Inf Theory 50(12):3250–3264
Li M, Vitanyi P (2008) An introduction to Kolmogorov complexity and its applications, 3rd edn. Springer
Magniez F, Santha M, Szegedy M (2005) Quantum algorithms for the triangle problem. In: Proceedings of the sixteenth annual ACM-SIAM symposium on discrete algorithms, SODA'05. Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, pp 1109–1117
Marion J-Y (2001) Actual arithmetic and feasibility. In: CSL, vol 2142 of Lecture notes in computer science. Springer, pp 115–129
McNaughton R (1966) Testing and generating infinite sequences by a finite automaton. Inf Control 9:521–530
McNaughton R, Papert S (1971) Counter-free automata. The MIT Press
McNaughton R, Yamada H (1960) Regular expressions and state graphs for automata. Trans Electron Comput 9:39–47
Nielsen MA (2003) Universal quantum computation using only projective measurement, quantum memory, and preparation of the 0 state. Phys Rev A 308(2–3):96–100
Nies A (2009) Computability and randomness, vol 51 of Oxford logic guides. Oxford Science Publications
Ostrowski AM (1954) On two problems in abstract algebra connected with Horner's rule. Studies in Mathematics and Mechanics presented to Richard von Mises. Academic Press, pp 40–48
Ozawa M (2002) Halting of quantum turing machines. In: Unconventional models of computation, vol 2509 of LNCS, pp 58–65
Pan VY (1966) Methods of computing the values of polynomials. Russ Math Surv 21(1):105–137
Papadimitriou CH (1994) Computational complexity. Addison-Wesley
Parsons C (1970) On a number theoretic choice schema and its relation to induction. In: Myhill J, Kino A, Vesley R (ed) Intuitionism and proof theory. Studies in logic and the foundations of mathematics. North-Holland, pp 459–473
Parsons C (1971) Proof-theoretic analysis of restricted induction schemata. J Symb Log 36:361
Parsons C (1972) On n-quantifier induction. J Symb Log 37:466–482
Paul W (1979) Kolmogorov complexity and lower bounds. In: Budach L (ed) Second international conference on fundamentals of computation theory. Akademie Berlin, pp 325–334
Paul W, Pippenger NJ, Szemeredi E, Trotter WT (1983) On determinism versus nondeterminism and related problems. In: Proceedings of IEEE FOCS'83. IEEE computer society, pp 429–438
Paz A (1971) Introduction to probabilistic automata. Academic Press
Perdrix S (2005) State transfer instead of teleportation in measurement-based quantum computation quantum computation. Int J Quantum Inf 3(1):219–223
Perdrix S, Jorrand P (2006) Classically-controlled quantum computation. Math Struct Comp Sci 16:601–620
Plandowski W (1995) The complexity of the morphism equivalence problem for context-free languages. PhD thesis, Department of Informatics, Mathematics, and Mechanics, Warsaw University
Rabin MO (1963) Probabilistic automata. Inf Control 6(3):230–245
Rabin MO (1969) Decidability of second-order theories and automata on infinite trees. Trans Am Math Soc 141:1–35

Rabin MO (1987) ACM turing award lecture, 1977: complexity of computations. ACM Press/Addison-Wesley Publishing Co, pp 625–633
Raussendorf R, Briegel HJ (2001) A one-way quantum computer. Phys Rev Lett 86:5188–5191
Rozenberg G, Salomaa A (ed) (1997) Handbooks of formal languages: vol 1. word, language, grammar; vol 2. Linear modeling; vol 3. Beyond words. Springer
Sakarovitch J (2009) Elements of automata theory. Cambridge University Press
Savage J (1998) Models of computation. Exploring the power of computing. Addison Wesley
Savitch WJ (1970) Relationship between nondeterministic and deterministic tape classes. JCSS 4:177–192
Schönhage A, Strassen V (1971) Schnelle multiplikation großer zahlen. Computing 7:281–292
Schützenberger MP (1965) On finite monoids having only trivial subgroups. Inf Control 8:190–194
Schützenberger MP (1975) Sur les relations rationnelles. In: Automata theory and formal languages. 2nd GI conference, Kaiserslautern, May 20–23, 1975, pp 209–213
Seiferas JI (1990) Machine-independent complexity. In: Leeuwen JV (ed) Handbook of theoretical computer science A: algorithms and complexity. Elsevier Science Publishers and The MIT Press, pp 163–186
Sénizergues G (2002) L(A)=L(B)? a simplified decidability proof. Theor Comput Sci 281:555–608
Shannon CE (1948) The mathematical theory of communication. Bell Syst Tech J 27:379–423
Shor P (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Goldwasser S (ed) Proceedings of the 35nd annual symposium on foundations of computer science. IEEE Computer Society Press, pp 124–134
Simon DR (1994) On the power of quantum computation. In: Proceedings of the 35th annual symposium on foundations of computer science. IEEE Computer Society Press, pp 116–123
Sipser M (2006) Introduction to the theory of computation, 2nd edn. Thomson Course Technology
Slot CF, van Emde Boas P (1984) On tape versus core an application of space efficient perfect hash functions to the invariance of space. In: Proceedings of the sixteenth annual ACM symposium on theory of computing, Washington, DC, April 30–May 2, 1984. ACM Press, pp 391–400
Thatcher JW, Wright JB (1968) Generalized finite automata with an application to a decision problem of second-order logic. Math Syst Theory 2:57–82
van Emde Boas P (1990) Machine models and simulations, chapter 1. The MIT Press and Elsevier, pp 1–66
van Leeuwen J (ed) (1990) Handbook of theoretical computer science, volume A: algorithms and complexity, B: formal models and semantics. The MIT Press and Elsevier
Vereshchagin NK, Uspensky VA, Shen A (2013) Kolmogorov complexity and algorithmic randomness (in Russian). MCNMO, (English version to be published by AMS)
Vergis A, Steiglitz K, Dickinson B (1986) The complexity of analog computation. Math Comput Simul 28(2):91–113
von Neumann J (1951) A general and logical theory of automata. In: Jeffries L (ed) Cerebral mechanisms in behavior—the Hixon symposium. Wiley, pp 1–31. Reprinted in Aspray W, Burks A (ed), Papers of John von Neumann on computing and computer theory, 1987. MIT Press
Wulf MD, Doyen L, Henzinger TA, Raskin J (2006) Antichains: a new algorithm for checking universality of finite automata. In: Proceedings of 18th international conference on computer aided verification, CAV. Seattle, WA, USA, Aug 17–20, pp 17–30
Yao AC-C (1993) Quantum circuit complexity. In: Proceedings of 34th IEEE symposium on foundation of computer science. IEEE Computer Society Press
Zenil H (ed) (2011) Randomness through computation. World Scientific, some answers, more questions

Databases and Artificial Intelligence

Nicole Bidoit, Patrick Bosc, Laurence Cholvy, Olivier Pivert and Marie-Christine Rousset

Abstract This chapter presents some noteworthy works which show the links between Databases and Artificial Intelligence. More precisely, after an introduction, Sect. 2 presents the seminal work on "logic and databases" which opened a wide research field at the intersection of databases and artificial intelligence. The main results concern the use of logic for database modeling. Then, in Sect. 3, we present different problems raised by integrity constraints and the way logic contributed to formalizing and solving them. In Sect. 4, we sum up some works related to queries with preferences. Section 5 finally focuses on the problematic of database integration.

1 Introduction

Research in databases and artificial intelligence have been maintaining close relations for more than thirty years. "Logic and databases" was the first scientific field at the intersection of databases and artificial intelligence (Gallaire and Minker 1987; Gallaire et al. 1981; Reiter 1983; Gallaire et al. 1983, 1984). Its aim was to formalize

N. Bidoit (✉)
LRI, Université Paris Sud, Paris, France
e-mail: Nicole.Bidoit@lri.fr

P. Bosc · O. Pivert
IRISA, Rennes, France
e-mail: Patrick.Bosc@irisa.fr

O. Pivert
e-mail: Olivier.Pivert@irisa.fr

L. Cholvy
ONERA, Toulouse, France
e-mail: Laurence.Cholvy@onera.fr

M.-C. Rousset
University Grenoble Alpes & Institut Universitaire de France, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France
e-mail: Marie-Christine.Rousset@imag.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_3

in logic some of the problems raised by databases. This approach has first met some difficulties in a community which did not clearly distinguish basic concepts used in databases from technological considerations. But its interest has gradually been truly appreciated. This research first focused on relational databases, then considered more complex information like incomplete information, deduction rules, dynamic integrity constraints, fuzzy information, legal information etc. This research also addressed new functionalities of databases like for instance, querying distributed databases, cooperative answers generation, preference-based queries answering or studying confidentiality of information.

Logic is one of the most useful formalisms in this area: first order logic, possibilistic logic (Dubois and Prade 2004), temporal logic, (de Amo and Bidoit 1993, 1995), epistemic logic (Reiter 1988; Demolombe and Jones 1996), deontic logic (Cuppens and Demolombe 1996; Carmo et al. 1997), situation calculus (Reiter 1993), description logic (Baader et al. 2003). But some other formalisms are also used, like for instance, fuzzy sets (Zadeh 1965) or CP-nets (Brafman and Domshlak 2004).

An exhaustive description of all the contributions at the intersection of databases and the artificial intelligence goes beyond the scope of this chapter. We will only address some of them. Section 2 sums up the seminal work of the "Logic and database" area which opened a wide research field at the intersection of databases and artificial intelligence. Section 3 deals with dynamic integrity constraints. Section 4 considers preference-based queries. Finally, Sect. 5 addresses the problem of database integration.

2 Modeling Relational Databases with Logic

2.1 Seminal Work

Reiter (1983) has been one of the first to promote the use of logic in the databases. His work aimed at using first order logic to model relational databases and describe their functionalities: complex information modeling, expressing queries and query evaluation, database updating... The use of logic has been motivated by the fact that this formal tool allows one to express sentences (formulas) and to reason based on these sentences. Reiter and his colleagues have shown that these two aspects exist in databases: one need to express information (data, constraints) and reason with them (queries must be answered, constraints must be checked...) Reiter has shown that modeling databases with logic can be done according to two different approaches: according to the model theory approach, a database instance is an interpretation of a particular first order language; according to the proof theory approach, a database instance is a set of first order formulas. In the following, we define a relational database with respect to the model theory approach.

Definition 1 A relational database is a triplet (L, I, IC) so that:

- L is a first order language corresponding to the database schema. It is defined as follows:
  - Any attribute value of the database is represented by a constant symbol of L. To simplify, the same symbol is used.
  - Any attribute domain T of the database is represented by an unary predicate symbol T, called type.
  - Any n-ary relation schema R of the database is modeled by a n-ary predicate symbol R.
  - The binary predicate for equality $=$ is introduced.
- $I = (D_I, i)$ is an interpretation of the language L corresponding to a state or an instance of the database. Its domain D_I and its interpretation function i are defined as follows:
  - D_I is isomorphic to the set of constant symbols of L. It is thus isomorphic to the set of attribute values of the database.
  - $i(=) = \{(a, a) : a \in D_I\}$. I.e., the predicate = is interpreted by the diagonal of D_I^2.
  - Any type T is interpreted by the subset of D_I which contains the constants associated with the values of the attribute domain T.
  - Any n-ary predicate R which represents a n-ary relation schema is interpreted by a set of elements of D_I^n corresponding to the tuples of the instance of the relation R in the database state.
- IC is a set of formulas of L called *integrity constraints*. They are defined by:
  - Any constraint on the states of the database (primary key, functional or inclusion dependency, ?) is represented by a formula in IC.
  - The formula $\forall x \quad T(x) \leftrightarrow (x = a_i^1) \vee ... \vee (x = a_i^n)$ belongs to IC, for any attribute domain $T = \{a^1 ... a^n\}$.
  - The formula $\forall x_1 ... \forall x_n \ R(x_1, ..., x_n) \rightarrow T_1(x_1) \wedge ... \wedge T_n(x_n)$ belongs to IC for any n-ary relation schema R whose attribute domains are $T_1, ..., T_n$.

One will notice that, because of the simplification on the choice of the constants and their interpretation, the interpretation I is indeed, an Herbrand interpretation.

Definition 2 The database (R, I, IC) is *consistent* iff $\models_I IC$. I.e., the interpretation I satisfies IC or equivalently, I is a model of IC.

In these works, the only integrity constraints which can be modeled are those that can be expressed in first order logic. In Sect. 3, we will come back to the notion of integrity constraint. We will see that there are some other kinds of integrity constraints, called dynamic integrity constraints, whose expression needs the use of temporal logic.

As for database querying, logic has proved to be useful for query simplification, query equivalence etc. These results were provided for queries expressed in relational algebra which is one of the most popular language in databases. These results are based on the fact that any algebraic query can be reformulated as a first order formula as it is shown in the following:

Let DB be a relational database, Q be a query expressed in relational algebra and $answer(Q, DB)$ be the answer of Q when evaluated over DB. Let (R, I, IC) be the logical representation of DB. Then, there is a formula of L associated with Q, denoted $t(Q, x_1, ...x_n)$ and whose free variables are $x_1...x_n$, such that: answer $(Q, DB) = \{< d_1...d_n > \in D_I^n \ : \ \models_I Q(d_1...d_n)\}$.[1]

For instance, consider two binary relations $Employee(e : Person; d : Department)$ and $Phone(e : Person; n : num)$. The first one relates employees to the departments they belong to, and the second one associates employees to their telephone numbers. Consider the algebraic query Q: $\prod_n \ \sigma_{d=CS} \ (Employee(e, d) \bowtie Phone(e, n))$. It aims at retrieving the telephone numbers of the employees who belong to the computer-science department. Its translation in logic is: $t(Q, x) = \exists y(Employee(y, CS) \wedge Phone(y, x))$.

But, if any algebraic query can be reformulated as a logical formula, the reverse is not true. More precisely, it has been shown that some logical formulas do not correspond to any algebraic query. This is the case of the disjunction f $Employee(x, computer) \vee Employee(Sally, y)$ which aims to find the pairs of individuals (e, d) so that e is an employee of the computer science department and then d can be anything or conversely, d is the department $Sally$ belongs to and e can be anything. Expressing such a formula in relational algebra is impossible. Note that the "answer" $\{< e, d > \ : \ \models_I f\}$ may be an infinite set of pairs. Thus, the language of first order logic is, in some sense, more powerful than the relational algebra for expressing database queries. In the next section, we will see that it is even too powerful for expressing queries since it allows one to express queries which have no meaning in the context of information and databases modeling.

Let us come back to the consequences of the previous property. Since a relational database can be expressed in logic and any algebraic query can be expressed as a logical formula, some of the problems raised in the database context can be studied and solved in logic. For instance, showing that two algebraic queries Q and Q' are equivalent (i.e., they provide identical answers in any coherent database state) comes down to showing that $IC \models t(Q, x_1...x_n) \leftrightarrow t(Q', x_1...x_n)$ i.e., showing that $t(Q, x_1...x_n) \leftrightarrow t(Q', x_1...x_n)$ is a logical consequence of IC. In the same way, showing that the answer of an algebraic query Q is always empty comes down to showing that the set of formulas $IC \cup t(Q, x_1...x_n)$ is inconsistent. This has been used in the domain of *cooperative answering.*

[1] Remember that by convention, we take the same symbol to represent a constant and the individual which interprets it.

2.2 Domain-Independent Formulas

The previous section emphasized the fact that the language of first order logic can be used in the context of databases to model information, queries and integrity constraints. However, some logical formulas do not have a clear meaning and thus must be discarded. For instance, the formula $Employee(x, computer) \vee Employee(Sally, y)$ already discussed above, or the formula $\forall x \exists y\, Phone(x, y)$ are problematic, even if they are well-formed formulas. Indeed, the last formula means that the property of having a telephone number is universal and thus has no meaning since every individual satisfies it. In a database which manages employee identifiers, department identifiers, etc.... expressing such a formula as an integrity constraint is considered as a conceptual error. It would imply that any object, even a telephone number, has got a telephone number, which is a nonsense. Indeed, what is meant is "any employee has got a telephone number" which is written $\forall x \exists y (Employe(x) \rightarrow Phone(x, y))$. Now, the property of having a telephone number is restricted to employees.

Another example of a frequent error consists in modeling the query "who does not belong to the CS department ?" by the formula $\neg Department(x, CS)$. In a database which manages employee identifiers, department identifiers, etc.... the answer will necessarily contain all the telephone numbers, department identifiers etc. which obviously do not belong to the CS department. In fact, what is meant by this query is "who are the employees not belonging to the CS department ?" and must be modeled by $Employee(x) \wedge \neg Department(x, CS)$.

The only formulas modeling queries for database processing are the *domain-independent formulas* (Kuhns 1967). The formulas which have been pointed out above are not domain-independent. The valuation of domain-independent formulas remains the same when one changes the interpretation domain without modifying the interpretation of predicates. Domain-independent formulas are defined by:

Definition 3 *(Domain-independent formulas)* The formula $F(x_1, ..., x_n)$ is domain-independent iff for any pair of interpretations $I =< D_I, i >$ and $I^* =< D_I \cup \{*\}, i >$ where I^* differs from I by one domain element $*$, we have:

$$\{< d_1, ..., d_n > \in D_I^n :\models_I F(d_1, ..., d_n)\} = \{< d_1, ..., d_n > \in D_{I^*}^n :\models_{I^*} F(d_1, ..., d_n)\}.$$

Although domain-independent formulas characterize logic formulas meaningful as database queries, the class of domain-independent formulas turns out not to be decidable. Thus, there is no algorithm which proves that any formula, modeling an integrity constraint or a query, is domain-independent. Studies have been carried out in order to find decidable subsets of domain-independent formulas. Among them, one finds the class of evaluable formulas (Demolombe 1992), the class of range restricted formulas (Nicolas 1982) or the class of *Safe formulas* (Ullman 1980).

Let us mention here a different approach to solve the same issue and according to which formulas expressing semantic integrity constraints or queries are not restricted.

This approach rather modifies the semantic of the language so that the valuation domain is restricted to *active domains* i.e, the set of individuals which have an occurrence in the interpretation of one predicate or in the formula expressing the query or integrity constraint. For instance, consider two predicates R (binary), S (unary) and the interpretation $I =< D_I, i >$ shown below, supposing that D_I=$\{a_1, a_2, ..., b_1, ...\}$ is infinite:

R	
a_1	b_1
a_2	a_2

S
a_3
a_2

The active domain *adom(I)* of I is the finite set $\{a_1, a_2, a_3, b_1\}$. The first order formula $\neg S(x)$ is not a domain-independent formula as shown previously but the number of valuations $\nu(x) \in$ *adom(I)* such that $\models_\nu \neg S(x)$ is finite. It is $\{a_1, b_1\}$ which is the answer to the query $\neg S(x)$ over I according to the active domain semantics.

Among the strongest results in the theory of query languages, recalled in (Abiteboul et al. 1995), are those showing the equivalence between the four following languages:

- first order logic restricted to domain-independent formulas
- first order logic restricted to Range-restricted formulas
- first order logic whose semantic is restricted to active domain
- relational algebra.

These equivalences strengthen each solution provided to the initial problem and allows the use of any of them without loosing generality. For instance, using the "active domain" approach in database is quite common for simplicity reasons.

Finally, let us notice that even if these results are quite old, they remain of interest inn the context of information modeling and its validation. This issue arises in database and in artificial intelligence and can be captured by: how can we be sure that the formula intending to model a given piece of information, really represents it ? Identifying that the formula written to express some property is domain-dependent proves an conceptual error although, writing a domain-independent formula does not eliminate any modeling error.

3 Integrity Constraints

The relational model like most database models[2] is quite poor from a semantic point of view. It allows one to specify tables (relations) whose cells contain elementary values. The number of columns of the table and the values allowed in each column are part of the table specification. However, table description through the relational model, is unable to exclude specific value combination, neither does it enables the inverse that is to enforce conditioned value occurrence. In general, the relational

[2]The relational model has been chosen in the introduction but models such as non normalized, complex value data and semi-structured models are concerned as well.

model does not allow to capture complex properties nor general laws that data should verify in order to conform to the real world applications.

The relational model, like other data models, is enriched with mechanism allowing to complement the data structure specification of tables with properties related to the application domain. These properties which are metadata are called integrity constraints. Integrity constraints acquisition and management (maintenance) are fundamental in several respects: (1) as mentioned above, the key objective is to ensure data reliability that is their compliance with the application domain, (2) like typing in programming languages, integrity constraints have a powerful leverage effect for query and update optimization at the logical and physical level; constraints serve to model data and to efficiently manage data up to avoiding the evaluation of a query; for instance, based on the declared integrity constraints, one may statically identify that a query answer is empty.

Application evolution, from relational database to XML data systems, comes with the increased need to develop techniques ensuring data reliability and highly efficient management.

This section does not aim to address integrity constraint system features exhaustively (Abiteboul et al. 1995; Bidoit and Collet 2001), and even less to cover commercial systems. Our goal is to review some of the problems related to integrity constraints illustrating the link between database and artificial intelligence. The first part focuses on elementary notions and more specifically on first order logic formalization of integrity constraints. The second part is dedicated to dynamic integrity constraints and temporal logic.

3.1 Integrity Constraints and First Order Logic

We postpone for now the discussion on constraint types and focus on static integrity constraints. A static integrity constraint is a property, no matter how complex, which can be checked by a simple test on the database current state. For instance, the property stating that an employee is assigned to only one department, is a static constraint.

Classically, a constraint is specified by a closed first order formula. Why? Besides the relative simplicity that first order logic provides for expressing properties, most problems related to integrity constraints are directly translated in logical terms allowing one to reuse existing formal results and tools as well as to develop new ones. Here follows a broad overview of the most known and common problems (see (Abiteboul et al. 1995; Bidoit and Collet 2001) for an extensive presentation and bibliography).

Entailment. Integrity constraints are metadata. It is fundamental, for instance, in order to validate the database schema, to be able to answer the following question: given a set of integrity constraints $\mathscr{C}$, is there any other constraint which are enforced by $\mathscr{C}$? and what are these constraints? This decision problem is well-known as the entailment problem in first order logic. The entailment, denoted $\mathscr{C} \models c$, checks

whether a formula c is true as soon as the set of formulas $\mathscr{C}$ satisfied. From a purely syntactic point of view, the problem comes to exhibit an inference system (axiomatization) used, when appropriate, to build a proof of c from the formulas in $\mathscr{C}$. Algorithmic and complexity issues of integrity constraint entailment have been investigated for specific classes of constraints called dependencies. The best known axiomatization is that of Armstrong for functional dependencies (Armstrong 1974). The frontier between logic and databases is drawn by the entailment complexity. Considering sub-classes of constraints such as acyclic, unary or tuple generating dependencies has been motivated by their good complexity properties as well as their relevance from the application point of view.

Coherence. Once constraints dedicated to a specific application domain have been specified, it is unavoidable to check consistency and to answer the following question: do data exist that satisfy these constraints? This problem is strongly related to satisfiability of a set of formulas which is known as undecidable. However satisfiability and consistency slightly differ: a set of formulas is satisfiable as soon as one model exists, even if this model is empty while a set of formulas is coherent if a non empty model exists for this set.

Semantic Optimization. Query optimization is a critical issue and traditionally its investigation combines two approaches. On the one hand, physical optimization makes use of the physical database schema (access paths like indexes) to generate efficient query execution code: integrity constraints like keys and foreign keys entail database index creation which foster query compilation. On the other hand, semantic query optimization takes place at an earlier stage by metadata based rewriting.[3] In extreme case, semantic optimization replaces query evaluation and produces the query answer avoiding data access. Example: the query extracting people having two partners while a constraint tells that every body has at most one partner.

Technics such as chase (Maier et al. 1979) for semantic optimization are among the most elegant ones. Formalizing both queries and constraints in first order logic allows one to use partial subsumption to "simplify" queries. Description logics have greatly contributed to semantic query optimization (Chakravarthy et al. 1990).

Description logics have extensively been used and contributed to semantic optimization (Hacid and Rigotti 1995; Bergamaschi et al. 1997; Calvanese et al. 1998; Beneventano et al. 2003) for their ability to provide a unique framework to express schemas, integrity constraints and queries.

Although it is impossible here to review all issues related to integrity constraints and leading to cross fertilization between artificial intelligence and databases, we ought to have a short discussion about integrity constraint maintenance methods.

Integrity constraint maintenance. Integrity constraints allow one to control the database evolution and thus checking database consistency arise essentially upon updates. But, when exactly? Choosing when constraint checking is activated leads to different classes of methods. The post update methods control and, if necessary,

[3]Functional dependencies help in a significant way the optimization of data sorting which arises when evaluating SQL group by, order by and distinct command (Simmen et al. 1996).

handle integrity violation through cancellation, repair or adaptation, after update execution: the efficiency of this optimistic and naive strategy relies on filtering the relevant constraints that are checked (relevant w.r.t. the updates) and also on developing incremental check. The pre-update methods are related to static analysis and takes on the challenge to predict, before executing the updates, the correctness of the result w.r.t. integrity constraints. These methods cannot be general. A dynamic variant of such strategy has been motivated by programming technics and introducing pre-condition enforcing valid update processing. Transaction schemas and active rules systems offer alternative solutions, often partial ones to integrity maintenance.

3.2 Dynamic Constraints: First Order and Temporal Logics

Whatever the type (static, dynamic, transaction), integrity constraints participate to database evolution control: changing data relies on these constraints in order to validate the changes and maintain data integrity/quality. To be checked, a transaction constraint needs to access both the database state before the update and that after. The constraint stating that salaries can only increase is an example of a transaction constraint. A dynamic constraint requires, in general, the whole state history of the database, that is the sequence of states from the creation of the database to the current state. The constraint stating that an employee cannot be reassigned to a department where she has been working in the past, is an example of a dynamic integrity constraint.

Dealing with dynamic constraints requires first to capture the notion of database history. We choose an abstract, simple model leaving aside a number of interesting problems such as concrete time measures, durations, calendar, problem induced by time granularity changes, multi-temporality (validity versus transaction), efficient storage of database history, etc. Dealing with abstract temporal or historical database is generally based on two equivalent simple temporal data representations.

On the one hand, the implicit approach considers a temporal database $\mathscr{I}$ over a schema (language) $\mathscr{R}$ as a sequence of static states $I_1, \ldots, I_n$ that is of interpretation of the language $\mathscr{R}$ as defined in 2. Each state I_{i+1} of the sequence has been obtained from an update over the previous state I_i. On the other hand, the explicit representation of a temporal database relies on data time stamping with time stamps being stored in the database as regular data. Time is assumed discrete and linear and the domain of the time stamp attribute is $\mathbb{N}$. Translating an implicit temporal database $\mathscr{I}$ into a time stamped instance uses an extension $\mathscr{R}^{est}$ of the schema $\mathscr{R}$ simply obtained by adding an attribute T to each relation schema R, leading to a schema R^{est}. Formally, the instance of R^{est}, denoted $I^{est}(R^{est})$, is given by $I^{est}(R^{est}) = \bigcup_{i=1}^{n}(I_i(R) \times \{i\})$.

In the implicit case, the query languages used to express dynamic or temporal integrity constraints are built from the linear temporal logic TL (Prior 1957; Emerson 1990; Chomicki and Toman 1998). Formulas of TL over a language $\mathscr{R}$ extend first order formulas with the following rules: if φ_1 and φ_2 are formulas then $\varphi_1 until \varphi_2$ et $\varphi_1 since \varphi_2$ are TL formulas.

A database history $\mathscr{I}$ satisfies a TL formula $\varphi(\mathbf{x})$ at time point $i \in [1, n]$, given a valuation ν of the free variables $\varphi(\mathbf{x})$, denoted $[\mathscr{I}, i, \nu] \models$, if the following holds:

- $[\mathscr{I}, i, \nu] \models \varphi_1(\mathbf{x_1})\ until\ \varphi_2(\mathbf{x_2})$ iff there exists $j > i$ such that $[\mathscr{I}, j, \nu] \models \varphi_2(\mathbf{x_2})$ and for each k such that $i < k < j$, $[\mathscr{I}, k, \nu] \models \varphi_1(\mathbf{x_1})$.
- $[\mathscr{I}, i, \nu] \models \varphi_1(\mathbf{x_1})\ since\ \varphi_2(\mathbf{x_2})$ iff there exists $j < i$ such that $[\mathscr{I}, j, \nu] \models \varphi_2(\mathbf{x_2})$ and for each k such that $i > k > j$, $[\mathscr{I}, k, \nu] \models \varphi_1(\mathbf{x_1})$.

Based on the temporal operators *until* and *since*, other operators may be derived such as *next*, *prev*, ...

In the explicit case, queries and constraints are expressed through first order logic, with the restrictions explained in Sect. 2, and by distinguishing two types of variables, data variables and temporal ones. The language obtained is thus a first order two-sorted logic, denoted TS–FO.

For instance, expressing that an employee cannot be reassigned in a department where she has been working in the past, is expressed by:

- using TL : $\forall e, d\ G(Employee(e, d) \rightarrow \neg(True\ Since\ Employee(e, d)))$ where G is the temporal modality "always".
- using TS–FO : $\forall t, \forall e, d\ (Employee(e, d, t) \rightarrow \neg(\exists t'\ (t' < t \wedge Employee(e, d, t))$ where t and t' are temporal variables whereas e and d are data variables.

The comparative study of the temporal query languages TL and TS–FO is probably one of the topics that led to rather unexpected results. The choice of explicit versus implicit representations of time has no impact at the level of data representation, however it has an impact on the language expressivity. As opposed to the results established by Gabbay (1980) and Kamp (1968) in the propositional case, comparing TL and TS–FO expressivity showed that:

1. the restriction of TL to the future *until*, *next* modalities is strictly less expressive than TL (Abiteboul et al. 1999);
2. TL is strictly less expressive than TS–FO (Abiteboul et al. 1999; Bidoit et al. 2004; Toman 2003).

This result has been proved using communication complexity on the hand, and independently using Ehrenfeucht-Fraïssé games for the order invariant fragments of TL and TS–FO. For instance, the very simple property stating that there exists two distinct states for which employee assignments to departments are exactly the same, is invariant w.r.t. the time order; it is straightforward to express this property in TS–FO: $\exists t_1, t_2\ (\forall e, d\ (Employee(e, d) \leftrightarrow Employee(e, d)))$. However, this property cannot be expressed in TL.

These results have motivated a number of investigations aiming at extending TL to build an implicit temporal language as powerful as TS–FO : Wolper (1983) introduces an extension of TL based on regular expression; Toman (2003) proves that there is no temporal modality able to reach this goal; (Abiteboul et al. 1999; Herr 1997) propose temporal iterators and fixed-point operators (Vardi 1988; Bidoit and Amo 1999) studies adding the operator "now" and (Abiteboul et al. 1999; Bidoit and Objois 2009) provide a hierarchy of these languages w.r.t. to expressivity.

As for static constraints, we conclude this subsection by providing a few pointers to methods dedicated to dynamic constraint maintenance. Two kinds of methods have been investigated. The first ones are based on the hypothesis that the database history is fully stored and used for constraint checking leading to technics similar to those developed for static constraints. The second methods try to avoid the storage of the whole database evolution and instead enrich the current database state with data relevant to the constraint checking mechanism (Chomicki 1995; Chomicki and Toman 1995): each update entails auxiliary relation updates. The main issue here is to use as least auxiliary relations as possible. For a given set of constraints, the number of auxiliary relations is required to be fixed and their content should only depend on the database. The contribution of such methods resides in decreasing secondary memory consumption and also improving execution time. However these methods suffer from the fact that storage and time optimization are pre-determined by and for a given set of integrity constraints, excluding the ability afterwards to deal with (check and evaluate) other constraints or queries at all. Bidoit and Amo (1998) proposes to treat temporal constraint checking using refinement technics borrowed from program specification: given a set of temporal constraints viewed as an abstract specification, a set of parameterized transactions together with composition rules, viewed as a concrete specification, is generated. This method, which is not general, however allows one to deal with a large class of temporal constraints.

3.3 Concluding Remarks

To conclude, it is important to highlight that integrity constraint definition and maintenance is a research topic which is still active and will remain active for a long time because integrity constraints provide a way to fill the gap between semantically poor data models and real world applications, highly demanding w.r.t. to semantic issues. For instance, although not developed in this section, the semi-structured data model and the web data exchange model XML require the definition and verification of integrity constraints for improving the quality of data management, the accuracy of reasoning and for optimization purposes. Many research works (Davidson et al. 2007; Arenas 2009) have addressed these problems for the XML format: keys, reference and functional dependencies are classical constraints that are useful for XML applications; path constraints are "new" constraints linked to the XML data format (Buneman et al. 2001; Buneman et al. 2003; Fan and Siméon 2003) In this context too, logic and more precisely modal logics (Kripke 1963) have been investigated as they offer a unique and simple formalization of graph properties as well as powerful reasoning mechanisms for these structures: labelled graphs (or trees) are commonly used to represent XML data (Calvanese et al. 1999; Alechina et al. 2003; Demri 2003). Specifying schemas and constraints, more specifically reference constraints has been investigated in (Bidoit and Colazzo 2007; Bidoit and de Amo 1998).

4 Database Preferences Queries

4.1 Introduction

The last two decades have witnessed a growing interest in the expression of preferences in database queries. The motivations for extending database queries with preferences are manifold. First, it appeared desirable to provide users with more expressive query languages, capable of faithfully reflecting the user intentions. Secondly, introducing preferences into queries provides a basis for rank-ordering the answers, which is particularly helpful when the result of a query is large. Finally, when a classical query produces an empty result, a relaxed (thus less restrictive) version has more chance to be satisfied by some of the elements of the database.

The approaches that aim to integrate preferences inside database queries may be classified into two categories (Hadjali et al. 2011) according to whether they are of a quantitative or a qualitative nature (see chapter "Compact Representation of Preferences" of Volume 1). In the first family of approaches, preferences are expressed in a quantitative way by means of a monotonous *scoring function* (the global score is positively correlated to partial scores, and each of these is computed by a function of one or several attribute values). As the scoring function associates a numerical degree with each tuple, tuple t_1 is preferred to tuple t_2 if the score of t_1 is greater than the score of t_2. On the other hand, in qualitative approaches, preferences are defined by means of *binary preference relations*. These two families of approaches are presented hereafter through some of their most typical representatives.

4.2 Quantitative Approaches

4.2.1 Explicit Scores Attached to Entities

The approach proposed by Agrawal and Wimmers (2000) enables a user to express his/her preference for an entity, either by associating it with a score between 0 and 1, or by expressing a veto (using the symbol $\natural$) or an indifference statement (default case) related to this entity. An entity is represented by a tuple in which the value of a field either belongs to the domain of the corresponding attribute or is equal to * (symbol that stands for any domain value other than those specified in the query). In order to illustrate these notions, let us consider a relation *car* of schema (#*i*, *make*, *model*, *type*, *color*, *price*, ...) describing different vehicles. A user expressing the preferences {(⟨Renault, Clio, red⟩, 0.4), (⟨Renault, Clio, *⟩, $\natural$), (⟨Opel, Corsa, green⟩, $\natural$), (⟨Ford, Fiesta, white⟩, 0.8)} means that he/she has a strong preference for white Ford Fiestas, a much lower preference for red Renault Clios, and that he/she absolutely rejects green Opel Corsas as well as any Renault Clio that is not red. The approach also includes a generic operator that makes it possible to combine preferences from several users.

The approach proposed by Koutrika and Ioannidis (2004) follows the same general philosophy but extends (Agrawal and Wimmers 2000) by considering a more general format for user preference profiles. It also makes it possible to express negative preferences ("I do not like SUVs") and preferences about the absence of values ("I prefer cars without ESP").

4.2.2 Fuzzy-Set-Based Approach

As classical sets can be used for defining Boolean predicates, fuzzy sets (Zadeh 1965)—which aim to describe classes of objects whose boundaries are vague—can be associated with gradual predicates (see chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1).

Generally speaking, atomic fuzzy predicates correspond to adjectives of the natural language such as *recent*, *big*, *fast*, etc. A fuzzy predicate P can be modeled by a function μ_P (usually of a triangular or trapezoidal shape) of one or several domains in the unit interval $[0,\ 1]$. The degree $\mu_P(x)$ represents the extent to which element x satisfies the gradual predicate P (or, equivalently, the extent to which x belongs to the fuzzy set whose membership function is μ_P). An atomic fuzzy predicate may also compare two attribute values by means of a gradual comparison operator such as "approximately equal" or "much greater than".

It is possible to alter the semantics of a fuzzy predicate by means of a *modifier*, which is generally associated with an adverb of the natural language. For instance, the modified predicate *very expensive* is more restrictive than *expensive*, and *rather high* is less demanding than *high*. The semantics of the modified predicate $mod\ P$ (where mod is a fuzzy modifier) can be defined compositionally, and several approaches have been proposed to do so, among which $\mu_{mod\,P}(x) = \mu_P(x)^n$.

Atomic and modified predicates can take place in compound conditions which go far beyond those that can be expressed in a classical querying framework. Conjunction (resp. disjunction) is interpreted by means of a triangular norm (resp. conorm) $\top$ (resp. $\bot$), for instance the minimum or the product (resp. the maximum or the probabilistic sum). As for negation, it is modeled by: $\forall x,\ \mu_{\neg P}(x) = 1 - \mu_P(x)$.

Operators of weighted conjunction and disjunction can also be used to assign different weights to the predicates of a query.

The operations of relational algebra can be extended in a rather straightforward manner to fuzzy relations (i.e., to relations resulting from fuzzy queries, where tuples are assigned a membership degree) by considering fuzzy relations as fuzzy sets on the one hand, and by giving a gradual meaning to the operations whenever it appears appropriate. It is worth emphasizing that the fuzzy-set-based approach to preference queries provides a *compositional* framework, contrary to most of the other approaches (either quantitative or qualitative). The definitions of the extended relational operators can be found in Bosc et al. (1999). As an illustration, we give hereafter the definition of the fuzzy selection, where r denotes a (fuzzy or classical) relation and φ is a fuzzy predicate.

$$\mu_{\sigma_\varphi(r)}(x) = \top(\mu_r(x),\ \mu_\varphi(x))$$

where $\top$ denotes a triangular norm (for instance the minimum).

The language SQLf described in Bosc and Pivert (1995), Pivert and Bosc (2012) extends the SQL norm so as to authorize the expression of fuzzy queries.

The fuzzy-set-based approach has also been applied to the querying of multimedia databases in Fagin (1998).

4.2.3 Top-*k* Queries

In the top-k approach (Chaudhuri and Gravano 1999), the user specifies ideal values for certain attributes as well as the number k of answers (the best ones) that he/she wants to obtain. The distance between an attribute value and the ideal value is computed by means of a simple difference, after a normalization step which maps every domain to the unit interval $[0,\ 1]$. The global distance is computed by aggregating the elementary distances using a function which can be the minimum, the sum, or the Euclidean distance. The global score obtained by a tuple is the complement to 1 of its global distance to the ideal object specified in the query. The computation steps are as follows:

1. from the threshold k, the chosen aggregation function, and statistics about the content of the relation considered, a threshold α that will be applied to the global score is derived;
2. a Boolean query calculating the set of elements whose score is at least equal to α—or a superset of it—is built;
3. this query is evaluated and the global score attached to every answer is calculated;
4. if at least k tuples having a score at least equal to α have been obtained, the k best are returned to the user; otherwise, the procedure is executed again (starting from Step 2) using a lower value of α.

4.3 Qualitative Approaches

4.3.1 Pareto-Order-Based Approaches

In the last decade, many algorithms have been proposed for efficiently computing the non-dominated answers (in the sense of Pareto order) to a given preference query. Seen as points in a multidimensional space, these answers constitute a so-called *skyline*. A pioneering work in this domain is that by Börzsönyi et al. (2001). First let us recall the principle of Pareto-order-based preference queries.

Let $\{G_1, G_2, ..., G_n\}$ be a set of atomic partial preferences. We denote by $t \succ_{G_i} t'$ (resp. $t \succeq_{G_i} t'$) the statement "tuple t satisfies preference G_i better than (resp. at least as well as) tuple t'". In the sense of Pareto order, a tuple t dominates another tuple

t' if and only if $\forall i \in [1,\ n],\ t \succeq_{G_i} t'$ and $\exists k \in [1,\ n],\ t \succ_{G_k} t'$. In other words, t dominates t' if it is at least as good as t' w.r.t. every preference, and it is strictly better than t' w.r.t. at least one preference.

Clearly, the approach based on Pareto order does not require any commensurability assumption between the satisfaction levels associated with the different elementary preferences, contrary to the fuzzy-set-based approach for instance. As a consequence, some points of the skyline (i.e., some elements of the result) may perform very poorly w.r.t. some atomic conditions (whereas they can be excellent w.r.t. some others), and the skyline approach only provides a strict partial order whereas the fuzzy approach yields a complete preorder. Kießling (2002), Kießling and Köstler (2002) laid the foundations of a preference query model based on Pareto order for relational databases. A preference algebra including an operator called *winnow* has also been proposed by Chomicki (2003) so as to integrate formulas expressing user preferences inside a relational framework (and SQL). In a similar spirit, Torlone et Ciaccia (2002) have introduced an operator named *Best* that aims to return the non-dominated tuples of a relation.

In such an approach, when preferences concern multiple attributes, the risk of obtaining many incomparable tuples tends to get high. Several techniques have been proposed for defining an ordering between two tuples that are incomparable in the sense of Pareto order, by exploiting for instance: (i) the number of tuples that each of the considered ones dominate (notion of k-representativity introduced by Lin et al. (2007)), or (ii) an order between the attributes concerned by the preferences, see e.g. the notions of k-dominance defined by Chan et al. (2006a), and k-frequency proposed by the same authors (Chan et al. 2006b).

4.3.2 CP-nets

The use of the structure called CP-net (Conditional Preference Network) for modeling database preference queries has first been suggested by Brafman and Domshlak (2004)—but this preference approach was initially developed in Artificial Intelligence (Boutilier et al. 2004) (cf. chapter "Compact Representation of Preferences" of Volume 1). A CP-net is a graphical representation of statements expressing conditional preferences of type *ceteris paribus*. The underlying idea is that the preferences of the user generally express that, in a given context, a partially described state of affairs is strictly preferred to another partially described state of affairs, the two states being mutually exclusive, according to the *ceteris paribus* semantics, i.e., all other things being considered equal in the descriptions of the two states. Using a CP-net, a user can describe how his/her preferences on the values of a given variable depend on the values of other variables. For instance, a user may formulate the following statements:

s_1: I prefer SUVs to sedans;
s_2: as for SUVs, I prefer the make Ford to Chrysler;
s_3: as for sedans, I prefer the make Chrysler to Ford;
s_4: concerning Ford cars, I prefer the color black to white.

In the CP-net approach applied to database querying (Brafman and Domshlak 2004), a preference is represented by a binary relation over a relation schema (where the attributes are assumed to be binary). Let R be a relation schema; a preference query Q over R consists of a set $Q = \{s_1, ..., s_m\}$ of statements (usually between sub-tuples of R, according to the *ceteris paribus* semantics).

From Q, one may infer a set of preference relations $\{>_{CP}(1), \ldots, >_{CP}(m)\}$, from which one may derive a global preference relation $>_{CP}(Q)$ that defines a strict partial order on the tuples of R.

It is worth emphasizing that the *ceteris paribus* semantics is opposed to the so-called *totalitarian* semantics which is implicitly favored by the database community (including those who advocate an approach based on Pareto order). The totalitarian semantics means that when evaluating the preference clause of a query, one does not take into account the values of the attributes that do not appear in this clause. Obviously, with the *ceteris paribus* semantics, the number of incomparable tuples is in general much higher than with the totalitarian one.

4.3.3 Domain Linearization

The approach proposed in Georgiadis et al. (2008) considers preferences defined as preorders on relational attributes and their respective domains. Let us consider again a relation *car* of schema (#*i*, *make*, *model*, *type*, *color*, *price*, ...) describing vehicles. An example of preference query in the sense of (Georgiadis et al. 2008) is made of the following statements:

(1) I prefer Volkswagen to both Opel and Ford (P_1);
(2) I prefer the colors black and grey to white (P_2);
(3) I prefer the type sedan to coupe, and coupe to SUV (P_3);
(4) the make is as important as the type, whereas the combination make-type is more important than the color (P_4).

Such statements define binary preference relations: (1), (2) and (3) on attribute domains, (4) on the set of attributes. These relations are supposed to be reflexive and transitive, i.e., to be preorders. The authors propose a technique for linearizing the domains associated with these partial preorders (let us recall that a domain, in the sense of domain theory, is a partially ordered set). This way, one can build a sequence of blocks (i.e., an ordered partition) of the result of the query. In such a sequence, each block contains tuples that are incomparable in the sense of the user preferences. The first block contains the elements that are the most preferred, and in every other block, for every element, there exists an element that is more preferred in the preceding block.

The algorithms proposed in Georgiadis et al. (2008) compute the sequence of blocks that constitute the result of a preference query without building the order induced on the tuples themselves. The idea is to exploit the semantics of a preference expression for linearizing the Cartesian product of all the attribute values that appear in this expression. Concretely, one moves from a set of statements expressing partial

preferences to a lattice of queries, then to a lattice of answers, and finally to a sequence of blocks that constitutes the result.

With respect to the approaches based on Pareto order, the originality of this technique lies in the use of partial (as opposed to strict) preorders for modeling independent positive preferences. This makes it possible to distinguish between the notion of "equally preferred tuples" on the one hand and "incomparable tuples" on the other hand.

4.3.4 Possibilistic-Logic-Based Approach

In Hadjali et al. (2011) the authors present a preference query model based on possibilistic logic (Dubois and Prade 2004), (see chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1), where the queries involve symbolic weights expressed on a linearly ordered scale.

For handling these weights, it is not necessary to give them a precise value, which leaves the user the freedom not to specify any default order on the priorities between the preferences (contrary to CP-nets where such an order is induced by the structure of the preference graph). However, the user may specify a partial order between the preferences.

In the case of binary preferences, the possibilistic encoding of the conditional preference "in context c, a is preferred to b" is a pair of possibilistic formulas: $\{(\neg c \vee a \vee b, 1), (\neg c \vee a,\ 1-\alpha)\}$. Hence, if c is true, one must have a or b (which are the only possible choices), and in context c, it is somewhat imperative that a be true. This corresponds to a constraint of the form $N(\neg c \vee a) \geq 1-\alpha$ where N measures the necessity of the event given as an argument; this expression is itself equivalent to $\Pi(\neg a|c) \leq \alpha$ where Π is the possibility measure dual to N.

This constraint expresses that the possibility *not to have a* is upper bounded by α, i.e., $\neg a$ is all the more impossible as α is small. To move from the scale of necessity degrees to a scale of satisfaction (or possibility) degrees, the authors use a scale reversal operator denoted by $1-(.)$. The priority level $1-(\alpha)$ associated with a preference is thus transformed into a satisfaction degree α when this preference is violated. Even if the values of the weights are unknown, a partial order between the different choices, founded on the operator *leximin* (Dubois et al. 1997), can be induced.

A parallel may be established between this approach and that based on fuzzy set theory where atomic conditions in a query may be assigned a weight reflecting their importance. These two approaches are in fact complementary and may be interfaced, which makes it possible to handle gradual (rather than binary) preferences on numerical attributes.

4.4 Concluding Remarks

It is well known that scoring functions cannot model all preferences that are strict partial orders (Fishburn 1999), not even some that may appear in a natural way in database applications (Chomicki 2003). For instance, scoring functions cannot capture skyline queries (see Hadjali et al. 2011). However, the skyline approach, and more generally dominance-based approaches, have some notable drawbacks: they produce in general a large number of incomparable tuples, they suffer from dominance rigidity (there is no distinction between tuples that are dominated by far and those that are near to dominant tuples), and they focus on the "best" answers only whereas quantitative approaches yield a layered set of items. Let us also mention that qualitative approaches are rather limited when it comes to combining preferences while the fuzzy-set-based approach makes it possible to express a great variety of trade-offs between criteria due to the large range of connectives coming from fuzzy logic.

The aspects related to the implementation of these models, in particular query optimization, could not be dealt with here, due to space limitation, but they are of course crucial in a database context, where the volume of data to manage is in general very large. Some elements about this issue may be found e.g. in Pivert and Bosc (2012).

5 Database Integration

5.1 Motivations

The goal of data integration is to provide a uniform access to a set of autonomous and possibly heterogeneous data sources in a particular application domain. This is typically what we need when, for instance, querying the *deep web* that is composed of a plethora of databases accessible through Web forms. We would like to be able with a single query to find relevant data no matter which database provides it.

The goal of a mediator (Wiederhold 2002) on top of existing data sources is to give users the illusion that they interrogate a centralized and homogeneous database management system by providing a query interface based on a single global schema (also called mediated schema). In contrast to a standard database management system, a mediator does not contain any data, which remain stored in the different data sources according to a format and a schema specific to each data source, but contains abstract descriptions of those data in the form of views. The views describe the content of each data source in function of the mediated schema. Formally, a view is a query (i.e., a logical formula) defined over the relations of the mediated schema and identified by a name. For answering to user queries that are expressed using the relations of the mediated schema, the extensions of the relations in the queries are not available: only the extensions of views are known by the mediator. The problem

of answering queries asked to a mediator is thus formally equivalent to the problem of computing the answers from views extensions. This problem is harder than the problem of standard evaluation of a query for which we have the complete information on the extensions of the relations appearing in the query. The difficulty comes from the fact that the instances of the relations in the query must be inferred from the instances (or extensions) of the views and from the definitions of these views. Even in simple cases, one cannot infer all the instances of the query's relations, as it can be illustrated in the following example.

Example 1 Let us consider a mediated schema that contains a single binary relation $Reservation$ relying a person to the persons for whom s/he has made a reservation. Consider the query *Q(x,y) : Reservation(x, y)* asking all pairs of persons (x, y) such that the person x has made a reservation for the person y. Suppose that only three very specific databases are available for answering such a query :

- DB1, that can only provide persons that have made a reservation for themselves and for somebody else. The content of this database can be described by the view $V1$ defined by $V1(x) : Reservation(x, x) \wedge \exists y(y \neq x \wedge Reservation(x, y))$.
- DB2, that can only provide persons that have made reservations. The content of this database can be described by the view $V2$ defined by $V2(x) : \exists y Reservation(x, y)$.
- DB3, that can only provide persons for whom reservations have been made. The content of this database can be described by the view $V3$ defined by $V3(x) : \exists y Reservation(y, x)$.

Suppose that the extensions of these views are: $V1(a)$, $V2(a)$, $V2(b)$, $V3(c)$. They enable the entailment of the incomplete extension of the relation $Reservation$: $Reservation(a, a), Reservation(a, ?), Reservation(b, ?), Reservation(?, c)$. The only precise answer that we can infer with certainty for the query Q is $< a, a >$. The other precise answers, such as $< a, c >$ for example, are possible but not certain.

5.2 Query Answering By Rewriting

The problem is to compute *all* the precise answers that are certain. An answer is precise if it is totally instantiated. An answer to a query is certain if it is part of the result of the evaluation of the query against all the extensions of the relations in the query that are compatible with the views extensions and definitions.

In the setting of mediator-based integration of distant data sources, the problem of query evaluation, that is already more complicated than the standard problem of query evaluation on top of a database as we have just explained it, is made even more complex by the fact that the data in the views extensions are not easily available. The cost of the transfer of these data into the mediator is prohibitive since they are distributed and stored in distant data sources. In addition, these data are very often evolving and volatile. This make impossible to base the computation of certain

answers on reasoning on views extensions. The only resources available within the mediator are the views definitions. The computation of the answers can only be done by *rewriting* the query in terms of views. This consists in reformulating the input query into a union of queries built on the names of the views, called query rewritings in function of the views. Each of these rewritings, being a query using names of views only, can then be evaluated in a standard manner against the extensions of the views involved in the rewritings. More precisely, the rewritings represent the query plans enabling the extraction from the different data sources of the elements of answers that are relevant for computing the certain answers of the input query. Their concrete execution requires however software interfaces (called *wrappers*) between the mediator and the data sources.

Finding rewritings that are equivalent (modulo views definitions) to the input query is not always possible. In general, we merely compute (maximal) rewritings *subsumed* by the input query. A rewriting is subsumed by the input query if, by replacing in the body of the rewriting each view by its definition, we obtain a logical formula that logically implies the body of the input query. Because of this logical implication, a rewriting subsumed by the input query provides a query plan whose execution returns answers that are guaranteed to be relevant to the input query.

Given a query and a set of views, the problem of rewriting queries using views consist in determining if it is possible to compute the set of all rewritings that are maximally subsumed by the query.

Example 2 Consider a mediated schema allowing one to define queries on employees of a company using the following relations: *Employee(e:Person, d:Department)*, $Phone(e : Person, p : PhoneNumber)$, $Office(e : Person, b : RoomNumber)$. Let us suppose that the data is stored in two distinct databases DB1 and DB2 whose content is specified in function of the relations of the mediated schema using the following two views:

- $V1(e, b, d) : Office(e, b) \wedge Employee(e, d)$
- $V2(e, p) : Phone(e, p) \wedge Employee(e, \text{“}toy\text{”})$.

DB1 provides information on employees, their office number and their department. DB2 provides phone numbers of the employees of the toy department.

Let us consider the query: $Q(p, b) : Phone(\text{“}sally\text{”}, p) \wedge Office(\text{“}sally\text{”}, b)$ asking the phone and office numbers of Sally. The only rewriting that can be obtained for this query using the two views $V1$ and $V2$ is: $Q_v(p, b) : V2(\text{“}sally\text{”}, p) \wedge V1(\text{“}sally\text{”}, b, d)$.

It is worthwhile to notice that the execution of the query plan corresponding to this rewriting does not guarantee to return answers, for several reasons. First, if Sally is not a member of the toy department, the execution of the query plan will not bring any result. This is due to the incompleteness of the available data for the relations in the mediated schema, that is declared in the view definitions: the only way to obtain phone numbers is to use $V2$, but its definition specifies that $V2$ can only provide phone numbers for employees of the toy department. Another cause for incompleteness is related to the fact that, in absence of additional information,

we do not know if the databases whose content is specified by views definitions are complete with respect to these definitions.

A view extension is complete if we can assume that it contains all the answers to the query defined by the view. For instance, stating the completeness of the $V2$ extension in the above example means that we have the guarantee that the database DB2 whose content is modeled by $V2$ definition contains effectively *all* the phone numbers of *all* the employees of the toy department. This completeness assumption is often too strong in the setting of information integration where it is reasonable to assume the soundness of views extensions but not their completeness. Stating that the $V2$ extension is sound (without being necessarily complete) means that DB2 contains phone numbers of employees of the toy department only, but not necessarily for all of them.

5.3 Decidability and Complexity

A lot of work (Beeri et al. 1997; Levy 2001; Abiteboul and Duschka 1998; Calvanese et al. 2000a, b; Goasdoué 2001) has been done on the decidability and the complexity of the problems of query rewriting using views and of answering queries using views, in function of the languages used for expressing respectively the queries, the views and the rewritings, and depending on the assumptions made on the views extensions. In particular, (Abiteboul and Duschka 1998; Calvanese et al. 2000a) shows the influence of the completeness assumption of the views extensions on the complexity of the problem of answering queries using views. It has been shown in Abiteboul and Duschka (1998) that under the soundness assumption on the views extensions, answering Datalog queries from extensions of views defined as conjunctive queries is polynomial (in data complexity), whereas this problem is co-NP-complete if the views extensions are assumed to be complete. If the views and the queries are expressed in Datalog, then in both cases (soundness and completeness of views extensions), the problem of answering queries using views is undecidable. These kinds of results have been extended in Calvanese et al. (2000a) to languages of queries and views belonging to the description logics family (Baader et al. 2003).

The problem of rewriting queries using views has been studied in (Beeri et al. 1997; Goasdoué 2001) when the languages for queries, views and rewritings belong to the CARIN (Levy and Rousset 1998) family that combines Datalog with description logics (see chapter "Reasoning with Ontologies" of Volume 1).

It has been shown in Calvanese et al. (2000b) that evaluating the rewriting of a query does not guarantee to find *all* the answers that can be obtained by evaluating the query on top of the views extensions, even if the rewriting is equivalent to the query modulo the views definitions. This shows an additional cause for the possible incompleteness of the answers, which is the limit of the expressive power of the language for specifying the rewritings. It is possible that a rewriting, defined in a language more expressive than the rewriting language imposed for modeling the

allowed query plans, leads to more answers than any rewriting in the considered rewriting language.

Goasdoué (2001) provides a sufficient condition that guarantees to obtain by rewritings all the answers that it is possible to obtain by evaluating the query from views extensions. If the query has a finite number of maximal rewritings defined as conjunctive queries with inequalities, then the result of the evaluation of the query against the views extensions is exactly the union of the answers obtained by executing the query plans corresponding to the maximal rewritings. As a consequence of this condition, a mediator will be able to compute all the answers in time that is polynomial in the size of the data (even if it is exponential in the size of the queries and of the views definitions). This result has been applied to design and implement the PICSEL mediator (Goasdoué et al. 2000; Rousset et al. 2002) in collaboration with France Telecom R& D.

More recently, description logics have evolved towards the design of tractable fragments such as the DL-Lite family (Calvanese et al. 2007) with good computational properties for querying data through ontologies.

Ontologies are at the core of the Semantic Web (Berners-Lee et al. 2001). They provide a conceptual view of data and services available through the Web in order to facilitate their handling. Answering conjunctives queries over ontologies is central for implementing the Semantic Web. The DL-Lite family (Calvanese et al. 2007) has been specially designed to guarantee a polynomial data complexity for the problem of answering conjunctive queries over data constrained by lightweight ontologies. Reformulating the query in function of the constraints and axioms declared in the ontology is necessary for guaranteeing the completeness of the answers. The important point is that this reformulation step (just like rewriting the query using views) is a reasoning problem independent of the data.

A major result of (Calvanese et al. 2007) is that DL-Lite is one of the maximal subset of first-order logic for which the problem of answering queries on top of massive data in presence of logical constraints on the schema is *tractable*.

DL-Lite is a subset of the ontology web language OWL[4] recommended by the W3C and more precisely of the recent standard OWL2.[5] DL-Lite extends RDFS[6] with the possibility to declare disjoint classes and to express functionality constraints on relations. RDFS is the W3C standard to describe metadata on resources in Linked Data and the Semantic Web.

The results obtained for DL-Lite have been generalized to *decentralized* query rewriting using views in Abdallah et al. (2009). For scalability as well as for robustness and data privacy, it is indeed relevant to study a fully decentralized model of the Semantic Web seen as a huge peer-to-peer data and ontology management system.

[4] http://www.w3.org/2004/OWL/.

[5] http://www.w3.org/TR/owl2-overview/.

[6] http://www.w3.org/TR/rdf-schema/.

6 Conclusion

This chapter first presented the seminal work on "logic and databases" which opened a wide research field at the intersection of databases and artificial intelligence. Then it showed some links between the two areas by focusing on integrity constraints satisfaction, preference-based queries and database integration.

This chapter does not intend to present a complete overview of relations between databases and artificial intelligence. In particular, some recent extensions of databases require using artificial intelligence techniques. For instance, querying databases which stores uncertain data requires using techniques from uncertainty management (see chapters "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" and "Representations of Uncertainty in Artificial Intelligence: Beyond Probability and Possibility" of Volume 1); querying databases which stores inconsistent data requires using inconsistency-tolerant techniques (see chapter "Argumentation and Inconsistency-Tolerant Reasoning" of Volume 1) or information fusion techniques (see chapter "Belief Revision, Belief Merging and Information Fusion" of Volume 1).

References

Abdallah N, Goasdoué F, Rousset MC (2009) DL- LITE$_{\mathcal{R}}$ in the light of propositional logic for decentralized data management. In: International joint conference on artificial intelligence (IJCAI)

Abiteboul S, Duschka OM (1998) Complexity of answering queries using materialized views. In: ACM (ed) PODS '98. Proceedings of the seventeenth ACM SIG-SIGMOD-SIGART symposium on principles of database systems, ACM Press, New York, NY 10036, USA

Abiteboul S, Herr L, van den Bussche J (1999) Temporal connectives versus explicit timestamps to query temporal databases. J Comput Syst Sci 58(1):54–68

Abiteboul S, Hull R, Vianu V (1995) Foundations of databases. Addison-Wesley

Agrawal R, Wimmers E (2000) A framework for expressing and combining preferences. Proc SIGMOD 2000:297–306

Alechina N, Demri S, de Rijke M (2003) A modal perspective on path constraints. J Log Comput 13(6):939–956

Arenas M (2009) Xml integrity constraints. In: Encyclopedia of database systems. Springer, pp 3592–3597

Armstrong W (1974) Dependency structures of data base relationships. In: Proceedings of IFIP congress, North Holland, Amsterdam, pp 580–583

Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) (2003) The description logic handbook: theory, implementation, and applications. Cambridge University Press

Beeri C, Levy A, Rousset MC (1997) Rewriting queries using views in description logics, editor = ACM. In: PODS '97, Proceedings of the sixteenth ACM SIG-SIGMOD-SIGART symposium on principles of database systems, May 12–14, 1997. ACM Press, Tucson, Arizona, New York, NY 10036, USA

Beneventano D, Bergamaschi S, Sartori C (2003) Description logics for semantic query optimization in object-oriented database systems. ACM Trans Database Syst 28:1–50

Bergamaschi S, Sartori C, Beneventano D, Vincini M (1997) Odb-tools: a description logics based tool for schema validation and semantic query optimization in object oriented databases. In: AI*IA, pp 435–438

Berners-Lee T, Hendler J, OLassila (2001) The semantic web. Scientific American, p 279
Bidoit N, de Amo S (1998) A first step towards implementing dynamic algebraic dependences. Theor Comput Sci 190(2):115–149
Bidoit N, Amo SD (1998) A first step towards implementing dynamic algebraic dependences. TCS 190(2):115–149
Bidoit N, Colazzo D (2007) Testing xml constraint satisfiability. Electr Notes Theor Comput Sci 174(6):45–61
Bidoit N, Objois M (2009) Fixpoint and while temporal query languages. J Log Comput 19(2):369–404
Bidoit N, de Amo S, Segoufin L (2004) Order independent temporal properties. J Log Comput 14(2):277–298
Bidoit N, Amo SD (1999) Implicit temporal query languages: towards completeness. In: Proceedings of the 19th conference on foundations of software technology and theoretical computer science, pp 245–257
Bidoit N, Collet C (2001) Contraintes d'intégrité et règles actives. In: Bases de Données et Internet (Modèles, Langages, et systèmes). Hermès, pp 47–74
Bőrzsőnyi S, Kossmann D, Stocker K (2001) The skyline operator. In: Proceedings of the 17th IEEE international conference on data engineering, pp 421–430
Bosc P, Pivert O (1995) SQLf: a relational database language for fuzzy querying. IEEE Trans Fuzzy Syst 3(1):1–17
Bosc P, Buckles B, Petry F, Pivert O (1999) Fuzzy sets in approximate reasoning and information systems—the handbook of fuzzy sets series. Chap Fuzzy databases. Kluwer Academic Publishers, pp 403–468
Boutilier C, Brafman R, Domshlak C, Hoos H, Poole D (2004) CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. J Artif Intell Res (JAIR) 21:135–191
Brafman R, Domshlak C (2004) Database preference queries revisited TR2004-1934. Computing and information science, Tech Rep. Cornell University
Buneman P, Fan W, Weinstein S (2003) Interaction between path and type constraints. ACM Trans Comput Log 4(4):530–577
Buneman P, Davidson SB, Fan W, Hara CS, Tan WC (2001) Reasoning about keys for xml. In: DBPL, pp 133–148
Calvanese D, Giacomo GD, Lenzerini M (1999) Representing and reasoning on xml documents: a description logic approach. J Log Comput 9(3):295–318
Calvanese D, Giacomo GD, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: the dl-lite family. J Autom Reason (JAR) 39(3):385–429
Calvanese D, De Giacomo G, Lenzerini M (2000a) Answering queries using views in description logics. In: Proceedings of AAAI 2000
Calvanese D, De Giacomo G, Lenzerini M, Vardi M (2000b) Answering regular path queries using views. In: Proceedings of ICDE 2000
Calvanese D, Lenzereni M, Nardi D (1998) Description logics for conceptual data modeling. In: Logics for databases and information systems. Kluwer
Carmo J, Demolombe R, Jones A (1997) Toward a uniform logical representation of different kinds of integrity constraints. In: Proceedings of ECSQARU-FAPR'97, LNAI 1244. Springer, pp 614–620
Chakravarthy U, Grant J, Minker J (1990) Logic-based approach to semantic query optimization. TODS 15(2):162–207
Chan C, Jagadish H, Tan K, Tung A, Zhang Z (2006) Finding k-dominant skylines in high dimensional space. Proc of SIGMOD 2006:503–514
Chan C, Jagadish H, Tan K, Tung A, Zhang Z (2006b) On high dimensional skylines. In: Proceedings of EDBT 2006, LNCS 3896, pp 478–495

Chaudhuri S, Gravano L (1999) Evaluating top-k selection queries. In: Proceedings of the 25th VLDB conference, pp 399–410
Chomicki J (1995) Efficient checking of temporal integrity constraints using bounded history encoding. ACM Trans Database Syst 20(2):149–186
Chomicki J (2003) Preference formulas in relational queries. ACM Trans Database Syst 28:1–40
Chomicki J, Toman D (1995) Implementing temporal integrity constraints using an active dbms. IEEE Trans Knowl Data Eng 7(4):566–582
Chomicki J, Toman D (1998) Temporal logic in information systems. In: Chap 3: Logic for databases and information systems, Kluwer Academic Publisher, pp 31–70
Cuppens F, Demolombe R (1996) A deontic logic for reasoning about confidentiality. In: Proceedings of 3rd international workshop on deontic logic in computer science (DEON'96)
Davidson SB, Fan W, Hara CS (2007) Propagating xml constraints to relations. J Comput Syst Sci 73(3):316–361
de Amo S, Bidoit N (1993) Contraintes dynamiques d'inclusion et schémas transactionnels. In: Neuvièmes Journées Bases de Données Avancées
de Amo S, Bidoit N (1995) A first step towards implementing dynamic algebraic dependencies. In: 893 L (ed) Proceedings of 5th ICDT
Demolombe R (1992) Syntactical characterization of a subset of domain independent formulas. J ACM 39
Demolombe R, Jones A (1996) Integrity constraints revisited. J Interes Group Pure Appl Log 4(3)
Demri S (2003) Modal logics for semistructured data. Invited talk at "Third workshop on methods for modalities (M4M-3)"
Dubois D, Prade H (2004) Possibilistic logic: a retrospective and prospective view. Fuzzy Sets Syst 144(1):3–23
Dubois D, Fargier H, Prade H (1997) Beyond min aggregation in multicriteria decision: (ordered) weighted min, discri-min, leximin. In: Yager R, Kacprzyk J (eds) The ordered weighted averaging operators—theory and applications. Kluwer Academic Publisher, pp 181–192
Emerson EA (1990) Temporal and modal logic. In: van Leeuwen J (ed) In: Handbook of theoretical computer science volume B: formal models and semantics. Elsevier
Fagin R (1998) Fuzzy queries in multimedia database systems. Proc of PODS 1998:1–10
Fan W, Siméon J (2003) Integrity constraints for xml. J Comput Syst Sci 66(1):254–291
Fishburn P (1999) Preference structures and their numerical representations. Theor Comput Sci 217(2):359–383
Gabbay DM, Pnueli A, Shelah S, Stavi J (1980) On the temporal basis of fairness. In: POPL, pp 163–173
Gallaire H, Minker J (1978) Logic and databases. Plenum
Gallaire H, Minker J, Nicolas J (1981) Advances in database theory, vol 1. Plenum
Gallaire H, Minker J, Nicolas J (1983) Advances in database theory, vol 21. Plenum
Gallaire H, Minker J, Nicolas JM (1984) Logic and databases: a deductive approach. ACM Surv 16(2)
Georgiadis P, Kapantaidakis I, Christophides V, Nguer E, Spyratos N (2008) Efficient rewriting algorithms for preference queries. Proc of ICDE 2008:1101–1110
Goasdoué F (2001) Réécriture de requêtes en termes de vues dans carin et intégration d'informations. PhD thesis, Université Paris Sud XI - Orsay
Goasdoué F, Lattes V, Rousset MC (2000) The use of carin language and algorithms for information integration: the picsel system. Int J Coop Inf Syst 9:383–401
Hacid MS, Rigotti C (1995) Combining resolution and classification for semantic query optimization. In: DOOD, pp 447–466
Hadjali A, Kaci S, Prade H (2011) Database preference queries—a possibilistic logic approach with symbolic priorities. Ann Math Artif Intell 63(3–4):357–383
Herr L (1997) Langages de requête pour les bases de données temporelles. PhD thesis, Université Paris-Sud 11

Kamp HW (1968) Tense logic and the theory of linear order. PhD thesis, University of California, Los Angeles
Kießling W (2002) Foundations of preferences in database systems. In: Proceedings of the 2002 VLDB conference, pp 311–322
Kießling W, Köstler G (2002) Preference SQL—design, implementation, experiences. In: Proceedings of the 2002 VLDB conference, pp 990–1001
Koutrika G, Ioannidis YE (2004) Personalization of queries based on user preferences. In: Bosi G, Brafman RI, Chomicki J, Kießling W (eds) In: Preferences, vol 04271 of Dagstuhl Seminar Proceedings. IBFI, Schloss Dagstuhl, Germany
Kripke S (1963) Semantical considerations on modal logic. Acta Philos Fenn 16:83–94
Kuhns J (1967) Answering questions by computers—a logical study. Rand Memo RM 5428 PR, Rand Corporation, Santa Monica, California
Levy A (2001) Answering queries using views: a survey. vLDB J
Levy A, Rousset MC (1998) Combining horn rules and description logics in carin. Artif Intell 101
Lin X, Yuan Y, Zhang Q, Zhang Y (2007) Selecting stars: the k most representative skyline operator. Proc of the ICDE 2007:86–95
Maier D, Mendelzon AO, Sagiv Y (1979) Testing implications of data dependencies. ACM Trans Database Syst 4(4):455–469
Nicolas JM (1982) Logic for improving integrity checking in relational databases. Acta Inform 18(3)
Pivert O, Bosc P (2012) Fuzzy preference queries to relational databases. Imperial College Press, London, UK
Prior A (1957) Time and modality. In: John Locke lectures for 1955–56. Oxford University Press
Reiter R (1983) Towards a logical reconstruction of relational database theory. In: On conceptual modelling: perspectives from artificial intelligence, databases and programming languages. Springer
Reiter R (1988) What should a database know. In: Proceedings of PODS
Reiter R (1993) Proving properties of states in the situation calculus. Artif Intell 64(2)
Rousset MC, Bidault A, Froidevaux C, Gagliardi H, Goasdoué F, Reynaud C, Safar B (2002) Construction de médiateurs pour intégrer des sources d'informations multiples et hétérogène: le projet picsel. Inf Interact Intell 2:9–58
Simmen D, Shekita E, Malkemus T (1996) Fundamental techniques for order optimization. In: Jagadish H, Mumick I (eds) Proceedings of the 1996 ACM SIGMOD international conference on management of data. Montreal, Quebec, Canada, June 4–6, 1996, ACM Press, pp 57–67
Toman D (2003) On incompleteness of multi-dimensional first-order temporal logics. In: Proceedings of the 10th international symposium on temporal representation and reasoning, pp 99–106
Torlone R, Ciaccia P (2002) Finding the best when it's a matter of preference. In: Proceedings of the 10th Italian national conference on advanced data base systems (SEBD 2002), pp 347–360
Ullman JD (1980) Principles of database systems. Computer Science Press
Vardi M (1988) A temporal fixpoint calculus. In: Proceedings of the 5th ACM symposium on principles of programming languages, pp 250–259
Wiederhold G (2002) Mediators in the architecture of future information systems. IEEE Comput
Wolper P (1983) Temporal logic can be more expressive. Inf Control 56(1–2):72–99
Zadeh L (1965) Fuzzy sets. Inf Control 8:338–353

Artificial Intelligence and Language

Nicholas Asher and Pierre Zweigenbaum

Abstract This chapter provides an overview of the role of artificial intelligence in natural language processing. We follow the chronology of the development of natural language processing systems (Sect. 2). This review is necessarily partial and subjective: rather than providing a general introduction to natural language processing, it focuses on logical and discursive aspects (Sect. 3) and on the contributions of machine learning (Sect. 4).

1 Introduction

Artificial intelligence (AI) aims to understand and to model human intelligence by computational means. Among the most striking human features, and a sign of their intelligence, is their ability to use a language system to communicate: this language system is much more complex than any communication system used by other known animal species. Since the beginning of computer science, researchers have investigated the computational processing of all levels of human language: this is the domain of natural language processing (NLP, also called *computational linguistics* or *language engineering* depending on the facet one wishes to emphasize). The main interactions and contributions of AI to linguistics have taken place in the domain of semantics and pragmatics, i.e., in the modeling of the contents of a sentence, a text or a dialogue. The application of machine learning to virtually every part of natural language processing in the last decades is another locus of this interaction.

From a formal linguistics point of view, the contents of a sentence is captured, at least in part, by a formalism that provides the truth conditions of the sentence, i.e., the conditions that should obtain for the sentence to be true. Formal linguists

This chapter extends an earlier version written in French with Laurence Danlos.

N. Asher (✉)
IRIT, CNRS, Toulouse, France
e-mail: Nicholas.Asher@irit.fr

P. Zweigenbaum
LIMSI, CNRS, Université Paris-Saclay, Orsay, France
e-mail: pz@limsi.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_4

generally use formal language representations such as first-order logic or higher-order logics. Given the translation of a natural language sentence into a formula of some logical representation language, these conditions are represented by a set of models which make this formula true. To test the accuracy of this translation and the adequacy of the semantic analysis, these linguists use the logical consequences of the formal language. A semantic analysis receives empirical support depending on the accuracy of the consequences of the semantic analysis. For instance, an analysis of the semantics of adverbial phrases such as *at midnight* or *with enthusiasm* that predicts that (1b–d) are logical consequences of the analysis of (1a) is *a priori* better verified than an analysis of (1a) which does not logically entail (1b–d).

(1) a. John buttered his toast with enthusiasm at midnight.
b. John buttered his toast with enthusiasm.
c. John buttered his toast.
d. John buttered something.

An important property for semantic analysis is compositionality, of which several versions exist. Roughly speaking, compositionality tells us that the contents of a complex expression must be a function of the contents of its component expressions, of its syntactic structure, and, since the advent of dynamic semantics, that will be discussed in more detail below, of its discursive context and even of its enonciative context. For many philosophers and linguists, the division between semantics and pragmatics lies in the use of discursive and enonciative context. But for many others, and at least for some of us, this division is somewhat arbitrary, for much research can be cited in which the discursive context even interacts with word meaning (Kamp 1973; Asher 2011). In the following we shall not dwell on the division between semantics and pragmatics. We shall instead examine in more detail the claim that giving the meaning of a sentence or a word in isolation is not appropriate to capture the contents of this sentence or this word in a particular context: the unit of analysis for the conceptual analysis of contents must be at least a discourse.

Linguistics is an old science. Linguistic studies date back to the fifth century before the Common Era with the syntactic work of Indian grammarian Panini and the discussions on word meaning in Platon's *Cratulus*. Since the Chomskian and Montagovian revolutions of the nineteen fifties and sixties, we have well-formalized models in syntax and semantics. The main issue is that very few of these models are operational for more than a toy subset of a natural language. And as is shown by research originating in AI that we shall sketch below, there is indeed a yawning gap between these theories and an effective computation of the meaning of a text or discourse.

A challenge these theories face is the need for large human efforts to compile the lexical, semantic and conceptual resources they require. The advent of very large text corpora has opened an avenue for data-driven methods in which another sub-discipline of AI, machine learning, contributes algorithms that automatically acquire the knowledge needed to perform NLP tasks. Text annotation efforts are still required to support supervised machine learning; less supervised methods that reduce or dispense with human annotation are therefore actively investigated. The design of good features to represent text for machine learning algorithms is another

area where human intervention is needed, although automatically learning word and text representations for NLP is a hot topic at the writing of this chapter.

2 The First Efforts

The first efforts of AI researchers on language consisted in modeling the information conveyed in text and language-mediated human interaction. For reasons that belong to the main involved researchers, this research was not grounded in existing linguistic descriptions or theories (for instance, Chomsky's theories on syntactic structure (Chomsky 1964, 1965) and Montague's grammar (Thomason 1974) for semantics). This lack of interest by AI pioneers for sentence syntax and semantics may be attributed to a recognition, at least implicit, of the contextual dependency of sentence meaning, and a desire to model the information conveyed by a whole text, discourse or dialogue directly. Conversely, researchers such as Marvin Minsky and Roger Schank took advantage of the development of high-level programming languages such as LISP to write'semantic grammars' that modeled very simple texts (Minsky 1969, 1994; Schank and Abelson 1977). These grammars consisted in a structure whose slots must be filled with a representation of clauses or sentences, using relations driven by verbs and their arguments. These structures were called *scripts* or *frames* and encoded typical knowledge on an object or event, such as *going to the restaurant*. Scripts were triggered by the occurrence of a word in the text or in its title, and included predicates and arguments. These arguments could be variables that were to be linked in the text to the arguments of the verb predicates. By exploiting scripts, a program could answer simple questions such as *Who ate in the restaurant?* even though this piece of information was not explicit in the text, by instantiating the script participants with nouns found in the text. Once the script was instantiated, the question could be matched to the script elements and the program could match the subject of verb *to eat* in the question to the agent of predicate EAT in the script.

(2) Alice went to the restanrant. Alice ordered the day's special.

The weaknesses of these methods were obvious from the start. First, the semantic representations were fragile. For nearly each new text a new script must be added to the library. In addition, the inferential systems underlying scripts were weak and the information that could be obtained from a script was roughly what had been introduced in the script initially. Finally, the interface between language and scripts was naive and could only handle very simple texts written by the script authors themselves. Around that time, Joseph Weizenbaum had used an even simpler approach, pattern matching, to develop the interactive system ELIZA (Weizenbaum 1966) that simulated a psychotherapist. The system used a library of English sentence templates. It could recognize words such as *father*, *mother* or larger patterns in an utterance written by a human (the'patient'). ELIZA then randomly selected one of the sentence templates in its library that were associated to this input pattern to generate an

answer. With an appropriate choice of sentences in its library, ELIZA led some of its users to believe that it was a machine with thought and language capabilities. ELIZA had passed the Turing test! The philosopher John Searle's article *Chinese Room Experiment* brought a compelling argument against this type of approach (Searle 1980). Searle's argument is simple: he builds a 'thought experiment' in which an English-speaking person in a room executes the instructions of a program to answer questions in Chinese. The person receives a question in Chinese, which is for her a string of characters or digits and sends back, according to her instructions, another string of characters that represents an answer in Chinese (see the discussion in chapter "Artificial Intelligence: Philosophical and Epistemological Perspectives" of this volume). Searle's intuition, shared by many, is that the person in the room has no idea of the contents of the Chinese expressions she handles.

The script and frames approach thus fell short of meeting a main objective of its inventors—creating a machine that would understand human language. There are several reasons why scripts, in their original conception, have largely been abandoned. Schank and Minsky aimed at processing the whole complexity of the language system without studying nor addressing the complexity of the interfaces between syntax and semantics, between semantics and pragmatics, or between sentence and discourse.

An analysis of the syntax-semantics interface, as sketched by linguist and philosopher Richard Montague in the 1960s, provided a means to generate, starting from lexical information, subtle semantic representations written in a logical language with a well-defined semantics and a powerful notion of logical consequence. To compute meaning, Montague grammars used a language containing the simply-typed lambda-calculus and intensional, higher order logic whose notion of consequence was that of higher-order logic or the somewhat weaker notion of consequence defined by the set of generalized Henkin models for this language. Montague provided expressions with intensional semantics, a much more refined semantics than an extensional one where assignments of extensions to terms and of truth values to sentences had proved to be insufficient to capture the truth conditions of sentences and semantic relations like entailment between sentences. Unfortunately, early AI researchers on NLP did not know, or rejected, the Montagovian research program. What early AI researchers on NLP did come up with, the scripts approach, was *ad hoc* in comparison to a formal semantic system like Montague Grammar and lacked a notion of logical consequence and a reasoning system. In addition, it lacked robustness, in today's parlance.

Although the scripts approach did not lead to results that met its inventors' goals, it gave rise to other approaches that are still in use in information extraction, notably in the *Message Understanding Conferences* (MUC5 1993), that rely on patterns to extract information on targeted topics. The scripts approach also demonstrated the importance of conceptual knowledge for text understanding, and provided a method to elicit important objects and events in a text. This method underlies contemporary NLP tools such as *named entity recognition* (NER) systems. However, there is now a consensus that pattern-recognition-based methods can only provide shallow semantic information.

Ideas from scripts have also been adopted and persist in lexical semantics, in the description and modeling of word meaning. The FrameNet project at Berkeley

(Johnson and Fillmore 2000), initially designed as a computational lexical resource for English and now adapted or being adapted to several other languages including French, Chinese, Brazilian Portuguese, German, Spanish, Japanese, Swedish, and Korean (https://framenet.icsi.berkeley.edu), associates content words (i.e., verbs, adjectives and common nouns) to frames or scripts. This lexical project is less ambitious than Schank's approach; it does not claim to build a system that captures word meaning. But it too inherits the difficulties of scripts as long as they are not provided with clear semantics nor with a system that exploits these semantics, though see (Löbner 2014) for efforts in this direction.

The need to exploit deeper content representations had been clearly identified. A research stream in AI on language thus turned towards the study and deeper modeling of semantics and pragmatics through methods based on logic.

3 Logic, AI, and NLP

The development of automatic provers for first-order logic, together with the plausible or non-monotonic reasoning systems of John McCarthy and Ray Reiter, two figures of artificial intelligence, have also had a strong influence on natural language processing. This is particularly true of discourse, where the work of Jerry Hobbs and Grosz and Sidner must be cited (Grosz 1979; Grosz and Sidner 1986).

3.1 Logic for Syntax and Semantics

In the 1980s, a number of researchers, equipped with logic methods and with rewriting and constraint resolution methods, have studied the syntax-semantics interface to address some of the weaknesses of the scripts approach. Computer scientist Ron Kaplan teamed with linguist Joan Bresnan to build a new syntactic formalism, Lexical Functional Grammar (LFG), that combined computational methods with linguistic descriptions (Bresnan 1982; Kaplan and Bresnan 1983). Also to be mentioned are other similar formalisms such as Head Driven Phrase Structure Grammar (HPSG) (Pollard 1984; Pollard and Sag 1987) and Tree Adjoining Grammars (TAG, (Joshi et al. 1975)), based upon advanced lexicons but using relatively simple combination rules such as feature structure unification. Logic also enabled researchers such as Ed Stabler and Mark Johnson to encode Chomsky's *Government and Binding* grammars (Chomsky 1981) using first-order logic formulations (Johnson 1989). This work produced syntactic parsers able to build a forest of parse trees for complex sentences, displaying a fine-grained linguistic analysis.

In parallel, in-depth studies continued to explore Lambek grammars (Lambek 1958) and categorial grammars (Oehrle et al. 1988). A categorial grammar consists of two parts: a lexicon that assigns each word a type or category, and a set of inference rules that compute the type of a word sequence. Although this grammar is a phrase

structure grammar like Chomsky's grammars, its rules are closer to, or even identical to, logical axioms. In fact, this compositional grammar has a close link to simply-typed lambda calculus. Its advantage is that axioms are set once and for all and encode the compositional structure of a language, so that in the end the grammar is simply determined by the lexicon.

On the semantics side, progress and cross-fertilization continued too but advances on more 'intelligent' machines were slow in coming and less successful. The aim, which is still present in today's formal linguistics, was to build from a syntactic representation a logic formula (in first-order or higher-order logic) that would represent the contents of a sentence or a text in terms of evaluation or truth conditions. In other words, meaning was defined by the models of this logic formula. Given the completeness of first-order logic, the hope was to perform automatic reasoning on text content for at least a large fraction of language.

To go from syntax to a logic formula that represents the meaning of a sentence, one needs (1) to build lexical entries, (2) to combine these lexical entries, and (3) to give a specific scope to any operator introduced in the resulting translation. The second step had been extensively studied by Montague and other semanticists of this school, using Church's typed lambda-calculus. For instance, given the syntactic tree for sentence (3), we obtain the information that *loves Bob* is a verb phrase with (present) tense, verb *love*, and direct object *Bob*.

(3) Alice loves Bob.

By translating *Bob* into a logic constant b and giving the verb a representation in higher-order logic with the λ operator of (Church 1940), $\lambda x \lambda T \lambda y T(\text{loves}(y, x))$, where T is an operator of type $(e \to t) \to (e \to t)$ that will be specified by the tense of the verb, reduction rules in λ-calculus predicted formula (4) for the verb phrase

(4) λy Present(loves(y, b))

that could then combine with the translation of the subject to yield a logic formula giving the truth conditions for the full sentence. Montague's school showed how to generalize this procedure to many complex constructions of natural languages, including verb tenses, quantifiers, modalities, ellipses, coordination, and relative clauses.

However, the first step that consists in providing words with appropriate representations raised and still raises many issues if one wants to endow a machine with the ability to perform logical reasoning on lexical information. We shall return to these issues later. For now, let us discuss the last step of the processing chain that we just sketched. This third step consists in providing a scope to quantifiers and operators that are obtained in the translation of a text into logic. This is where formal linguistics encountered difficulties with the standard first-order logic formalisms. Let us take a trivial example.

(5) A man entered the bar. He was smoking a cigar.

Pronoun *he* in the second sentence clearly depends on the quantifier introduced by determiner *a*. However, in the methods of classical logic, the scope of the quantifier ultimately stops with the end of the first sentence. There was no elegant solution neither in classical logic nor in Montague's grammar. This is why linguists reinvented a semantics for texts, 'dynamic semantics', that was very similar to the semantics of programs. In the semantics of programs, every command performs either a transition of the state of the machine or a test on the current state. A possible action consists in giving a random value to a variable intuitively bound to an existential quantifier; it is very important for the semantics of programs that the value assigned to the variable be propagated to the following states of the machine, unless it is replaced. This is also what happens in Example (5): the value of the variable bound by the existential quantifier is 'propagated' to the starting state of the processing of the second sentence, and that value can then be reused in its interpretation. In building various dynamic semantics methods, linguists had discovered or reinvented the resources of theoretical computer science that are used for the semantics of programming languages (Kamp and Reyle 1993; Groenendijk and Stokhof 1991; Harel 1984). Since then interesting interactions have taken place between abstract semantics methods for programming languages (the continuation method) and natural language semantics (de Groote 2006; Barker and Shan 2006).

This transition to dynamic semantics for texts made it possible to handle in much more detail linguistic problems related to *anaphora*, in which the interpretation of a word or phrase depends on the preceding linguistic context. In the mid 1980s, researchers, mostly European, including linguists and computer scientists with an interest in dynamic semantics, among others in Hans Kamp's group, started to study formal semantics implementations. Some researchers began to use the PROLOG programming language to build syntax-semantics interfaces for dynamic semantics and for LFG (Frey et al. 1983). These implementations continue nowadays with more and more powerful syntactic parsers and increasingly sophisticated algorithms for the syntax-semantics interface.

Dynamic semantics has been very successful over the last thirty years. However, some weaknesses were uncovered in the processing of anaphoric expressions by dynamic semantics. This semantics does not differentiate multiple antecedents that are judged possible. Research in linguistics showed as early as the end of the 1980s that better predictions on the interpretation of various elements that depend on the discourse context for their content were possible if discourse context was construed not as a sequence of values assigned to variables, but instead as a relational structure that contains discourse constituents linked by discourse or rhetoric relations (Fox 1987; Asher 1993; Lascarides and Asher 1993). Later, research in psychology and psycholinguistics confirmed experimentally that anaphora resolution depends upon the discourse structure of text (Kehler et al. 2008).

3.2 Discourse Structure

In AI discourse structure, which is composed of content elements linked by relations that express their discourse roles in the text, had already been a topic of study in the late 1970s and in the 1980s (Schank and Abelson 1977; Grosz 1979; Hobbs 1985; Polanyi 1958; Grosz and Sidner 1986). These authors had suggested several models for discourse structure. Schank and Abelson on the one hand, and Grosz and Sidner on the other hand, have developed the idea, which has its modern origin in philosopher Paul (Grice 1975), of an analysis of a text or of a dialogue based upon intentions and plans. These researchers hypothesized that the discourse structure of a text corresponded to the structure of a recursive plan in the mind of the speaker. For Grosz and Sidner, these plans were built with two operators: one related two segments of a discourse if the second segment provided details or specified the plan of the first segment; the other was a sequence operator that ordered actions in a plan, in a similar way to the operator ';' of dynamic logic. Mann and (Thompson 1986; 1987)'s approach to discourse structure continued in this direction, enriching the number and the descriptive capacity of intentional relations that link text segments. The cross-fertilization of research on plans and discourse was productive (Grosz and Kraus 1993; Lochbaum 1998). These authors benefited from planning languages and logics in their work, and from advances in models in modal logic of knowledge, belief, and intentions, provided by numerous AI researchers, including Joe Halpern, Ron Fagin and Moshe Vardi (Halpern et al. 1995) through conferences such as *Theoretical Aspects of Reasoning about Knowledge*.

The intentional approach to discourse structure nevertheless had a weakness that required strong hypotheses for it not to make this approach implausible. The problem was that the speaker's intentions needed to be inferred from what had been uttered, and that these inferences are often uncertain or even false. To solve this problem, hypotheses had to be made about speakers and listeners. These hypotheses, also coming from Grice, assumed that the participants in a conversation shared the knowledge that speakers are strongly cooperative with their listeners and say what they believe to be true. With these hypotheses and axioms that link language features to particular intentions (for instance if the speaker asked a question, this meant that they wanted to obtain an informative and truthful answer), researchers in this domain had succeeded in analyzing manually pedagogical dialogues that they had built and to infer their discourse structure.

Coherence relations (also termed *rhetorical* relations) have also been useful not only to develop text semantics but also for computational applications such as text summarization (Marcu 1997). Better performing discourse parsers make it possible to build advanced systems that take advantage of discourse structure for information or opinion mining.

3.3 Lexical Semantics

Let us recall that a suitable processing of semantics must go through an encoding of word contents, hence through considerations of *lexical semantics*. In the logical framework given by Montague's grammar, each lexical element is represented by a λ-term. For instance the word *cat* is represented by the term $\lambda x\ \mathrm{cat}(x)$. By assigning each word an appropriate type and relying upon the syntactic structure of a sentence, semanticists were able to build the logical structure of a sentence. The problem is that $\lambda x\ \mathrm{cat}(x)$ does not provide any element to make inferences purely based on lexical information. For instance, to infer (6b) from (6a), we need auxiliary information that does not come directly from the lexical entries created in Montague's semantic tradition.

(6) a. Boris is a cat.
b. Boris is an animal.

The solution to this problem in the Montagovian approach was to build *meaning postulates* that supported these inferences, an approach pushed by (Dowty 1979). But linguists had not really tried to codify these postulates systematically. In AI two paths were explored to bring techniques to solve this problem.

The first one consists in trying to axiomatize in a more or less complete manner in first-order logic the information associated with lexemes; an example is the CYC knowledge base developed by Douglas Lenat, which continues probably to be the largest effort in this direction (Lenat 1995). This approach is still current in the discipline of formal ontology and in conferences such as FOIS and ISMIS, as well as in general AI conferences, and was most successful with the axiomatization of temporal and spatio-temporal information.

A second path consisted in using tools such as feature structures and unification, developed for syntactic processing in LFG and HPSG, with concept lattices associated to words. Inheritance systems such as those studied by (Horty, Thomason and Touretsky 1987) were used as inference engines. The key idea of this approach is to simplify the encoding of lexical knowledge by using a language that is much less expressive than first-order logic. This approach is most visible in linguistics in the work of (James Pustejovsky 1995) and his theory of the generative lexicon. In AI these techniques are used in the domains of terminology and ontology: we refer the reader to chapter "Knowledge Engineering" of Volume 1, that describes work pertaining to the creation and use of ontologies, and to chapter "Reasoning with Ontologies" of Volume 1.

However, these approaches were not readily accepted in formal semantics. They were created without much care for their logical foundations, and were therefore not really in a position to produce sophisticated lexical knowledge in the form needed by logic-based approaches. Weaknesses in the formalization of these approaches and difficulties in integrating feature structures in λ-calculus, which remains the main device for meaning composition in formal semantics, prevented this path from having much impact on formal semantics until recently.

These two approaches are not exclusive though. A number of studies have used knowledge representation languages (see chapter "Reasoning with Ontologies" of Volume 1) that have a strong logical foundation, such as Conceptual Graphs (Sowa 1984) or Description Logics (Brachman and Schmolze 1985), to represent concepts and word senses. Their combination with syntactic models based on feature structures and predicate-argument structure such as LFG provided them with the benefits of both approaches.

Nevertheless, as in all other cases, the implementation of these methods on real texts and at scale met with several difficulties. On the one hand, the rigidity of λ-calculus must be reconciled with the flexibility of natural language, of which metonymy is an example. For instance, (Bouaud et al. 1996) study Expression (7):

(7) la dilatation d'une sténose
the dilation of a stenosis

If the action *to dilate* is defined as acting on a *physical object* (for instance an artery), and given that a stenosis is a pathological *state* (the narrowing of an artery) [1], Expression (7) does not enable the construction of a representation that satisfies the stated constraints because types *object* and *state* are incompatible. This exemplifies a common phenomenon: the fact that a word that refers to a state is used to refer to the object that undergoes this state, a specific form of metonymy. Thus, in some sense, such predications involve meaning shifts, and mechanisms to account for these, such as type coercion, are then necessary, (Pustejovsky 1995), and require complicating the composition methods of formal semantics using the λ-calculus (Asher 2011). This has led to work at the intersection of theoretical computer science and semantics, notably in type theory (Luo 2010, 2012; Cooper 2011; Asher and Luo 2012).

On the other hand, an exhaustive description of the lexical representations needed for a given language requires either a colossal human effort, such as was endeavored by CYC, or automatic knowledge acquisition methods from large text corpora or encyclopedia, which the existence of the Web and of Wikipedia (http://www.wikipedia.org/) makes possible today.

3.4 Pragmatics and Non-Monotonic Logic

We have already mentioned the classical linguistic distinction between semantics and pragmatics, between literal contents ('what was said') and implied content ('what was suggested'). Some linguists think this distinction is very clear; others think it is not. But most linguists take seriously the idea that the contents of a word, sentence or discourse includes a core that gives rise to classical deductive inference and a 'periphery' that supports'soft' inference. Within this'peripheral' content lies scalar implicature. Let us consider for instance the dialogue in (8):

(8) a. A: Did all linguists take part in the demonstration?

[1] This defines *selectional restrictions*.

b. B: A few linguists took part in it.

The answer to question (8a) does not say that not all linguists took part in the demonstration, but implies it. This implication is called a *scalar implicature*. Let us note additionally that it is defeasible (i.e., undoable). This means that B can continue with:

(8c) B: Indeed, all of them were there.

without making her contribution in (8b) inconsistent, which would not be the case if the implicature were a non-defeasible consequence.

Defeasible inferences are ubiquitous in language and mastering them constitutes an important part of linguistic competence. These inferences are indeed at the core of implicature analysis. But they are present too in the deduction of a discourse structure for a text. The inferences that conclude that a rhetorical relation obtains between two successive discourse constituents are very often defeasible inferences. They also play an important role in preferential attachment decisions for prepositional phrases in a syntactic parse tree: for instance, in

(9) The murderer lunged towards John. John hit the murderer with a hammer.

the prepositional phrase *with a hammer* can attach either to the noun phrase *the murderer* to specify the denotation of the noun phrase, or to the verb phrase *hit the murderer* as an adverb of manner, which would be the preferred attachment in this discourse context. Defeasible inferences make up an important part of the anaphoric link deduction mechanism, as shown by (Megumi Kamayama 1995). She gives as an example:

(10) John hit Arnold Schwarzenegger. He felt pain.

The task here is to determine the referent of *He*. Given the general information available at this time about Arnold Schwarzenegger, the preferred but defeasible solution for the pronoun was *John*. Finally, we can cite lexical disambiguation as another domain in which defeasible inference is used. Let us take for instance the English verb *enjoy*. In

(11) John enjoyed the meal.

the verb requires an event as direct object: John appreciated the dishes served in the meal, or the conversation during the meal, the background of the meal, etc. The preferred interpretation of this sentence is that John appreciated the dishes of the meal. However this interpretation remains defeasible: in another context John might have liked the conversation or the ambiance though he might have found the dishes unsatisfactory.

It is AI that provided linguists and philosophers with precise formalizations of defeasible reasoning by developing non-monotonic logic. The circumscription method of (John McCarthy 1980) and the default logic of (Ray Reiter 1980), developed in the late 1970s, offered well-defined formal frameworks to study defeasible inference for non-monotonic logic, many papers, and a series of international conferences on this topic (see chapter "Knowledge Representation: Modalities, Conditionals and Nonmonotonic Reasoning" of Volume 1). In part, McCarthy's and Reiter's

formalizations answered a challenge launched by one of the advocates of script/frame models, Marvin Minsky, who had claimed that logic could not adequately formalize a sentence such as

(12) Birds fly.

(think about the case of ostriches). Since then the linguistic study of sentences that contain *generic* noun phrases such as (12) confirmed that McCarthy's and Reiter's efforts provided linguistics with precious tools. These methods and their offspring have been applied to a number of linguistic phenomena. To cite but a few: presupposition (Mercer 1987), rhetorical relation inference (Lascarides and Asher 1993; Hobbs et al. 1993; Asher and Lascarides 2003), modalities (Veltman 1996), the analysis of verb tense (Asher 1992), scalar implicature (van Rooij and Schultz 2004), and generic noun phrases (Asher and Morreau 1995; Pelletier and Asher 1997).

These logic based approaches contributed powerful mechanisms to handle a number of linguistic phenomena. However they still proved too fragile to process arbitrary texts and provide a true empirical validation of the underlying theories. Uncertainty processing with non-monotonic reasoning required for example a considerable axiomatization effort to succeed in modeling simple phenomena. Here too, there is hope that the development of machine learning techniques (see Sect. 4 below) and the advent of large-coverage, powerful syntactic parsers will make these methods really usable.

3.5 Reasoning and Dialogue

The formalization of reasoning by AI researchers has an important role in dialogue too. Several kinds of reasoning are relevant for dialogue analysis. The first kind is a reasoning on *coordination* between the speaker and the listener about the meaning of the message or dialogue turn. The pioneering research of philosopher (Lewis 1969) introduced the concept of signaling game in which an actor knows the state of the world and emits a signal whereas another actor must 'interpret' the signal, i.e. associate it with a possible state of the world. The equilibria of these games determines the information content of the signal. Since Lewis's work, economists have developed the concept further and contributed many results and important refinements (Crawford and Sobel 1982; Farrell 1993; Rabin 1990). Linguists and philosophers have started to use the formalism of games to analyze the content of linguistic expressions as well as pragmatic inferences (van Rooij 2004). However, little attention had been given to the development of a computational model of reasoning in terms of game theory until the early 2000s (Bonzon et al. 2006; Asher et al. 2017; Asher and Paul 2016).

A second kind of reasoning that is relevant for dialogue analysis is that activated by the computation by the speaker and of her dialogue turn, anticipating the replies of the listener. The speaker often chooses to say what she says based upon her computation of the effect of the message on the listener. In this purpose, the speaker

must model the beliefs and preferences of the listener and the way she will react. This is a reasoning about decision, but inasmuch as the listener must do the same thing, the reasoning becomes that of a game with a notion of equilibrium. A delicate issue consists in determining the granularity of this reasoning. It is clear that speakers adjust their message while learning things about their audience. Hence this reasoning must be efficient. Besides, the listeners' beliefs and preferences are revealed only partially, through what they say and do. Therefore this reasoning system must be qualitative. With the advent of compact, qualitative systems for the representation of preferences (for instance, see Boutilier 1999; Domshlak 2002 or also chapter "Music and Artificial Intelligence" of this book), AI researchers have had much interest in reasoning in the context of game theory, especially in Boolean games, in which actions are represented by propositional formulas (Bonzon 2007). This will in turn have important consequences for formal and computational linguistics, since this provides the means to build models of this reasoning that are both qualitative and reasonably efficient. This chapter of linguistics remains to be written but the means are there to make a big leap toward the ultimate goal of AI in natural language processing: a near-complete automatic processing of the language behavior of a human agent.

4 Machine Learning in NLP

4.1 The Rise of Large Text Corpora

With the advent of fast computers with a large enough memory to store huge corpora of real examples, linguists in various fields faced the following question: can the theories developed on hand-built examples cope with real, large-sized data?

Text data compilation efforts performed for English since the 1970s increased along decades—the British National Corpus (Burnard 1995), the Penn Tree Bank (Marcus et al. 1993), the Message Understanding Conferences data (MUC5 1993), the Penn Discourse TreeBank (Prasad et al. 2008), the English Gigaword (Graff and Cieri 2003), the Google Books (Davies 2011), to name but a few—and other compilations followed suit in other languages. The very large text corpora have reached a size that exceeds that of the totality of language data a human being is exposed to: Brysbaert et al. 2016's upper limit estimates for a sixty-year old human being exposed to a few billion words, whereas the Google Books corpus of American English contains 155 billion words. These corpora indeed differ in their word distributions, but their sheer size opens new perspectives for the study of languages and of their acquisition.

In parallel with the wider availability of language data, researchers started to organize shared frameworks to evaluate NLP components (e.g. MUC5 1993; Voorhees and Harman 2005). Shared task definition, shared input data and expected output (*gold standard*), shared evaluation criteria and procedures, all this contributed to a

better comparison of approaches and assessment of the state of the art. Many NLP shared task series are currently active, let us cite among others TAC (Text Analysis Conference, https://tac.nist.gov/) and SemEval (Semantic Evaluation [e.g. Bethard et al. 2017a]) on top of the more information retrieval oriented TREC (https://trec.nist.gov/).

Implementations of linguistic theories that use purely symbolic rules grounded in expert knowledge (a priori knowledge, introspection or the result of corpus studies by humans) have proved fragile when faced with this challenge. Morphological and syntactic parsers, which were the most mature, adapted better, but for semantics, discourse and pragmatics, only limited fragments of language or very specific corpora (with short texts or domain-specific texts) provided partial validations for some theories. Human-authored lexical and semantic resources were lacking to build systems with enough coverage and robustness.

4.2 Casting NLP Tasks as Prediction Problems

Machine learning methods have invaded the domain of NLP, starting in the mid-1990s (Klavans and Resnik 1996), as logic-based symbolic approaches required a too large investment in axiomatizing the lexical, semantic and conceptual knowledge needed for a large coverage of existing corpora. With the advent of large corpora, machine learning methods, especially numerical methods (see Chapter "Statistical Computational Learning" of Volume 1), have become more and more attractive for NLP (see Manning and Schütze 1999). Their last incarnation at the writing of this chapter is the 2010s' wave of neural network methods for NLP (see Sect. 4.4 below). The lag in building such large corpora in languages other than English has caused an equivalent lag in the invasion of these methods for these other languages.

Machine learning has brought an evolution of methods from algorithms based on human knowledge (e.g., grammar rules for a parser), in general with binary decision rules (a sentence is or is not grammatical), to algorithms based on knowledge learned from annotated data (e.g., sentences annotated with parse trees by humans), in general with decision rules that aim at finding the most likely solution (computing the likelihood of a set of possible parse trees, then selecting the one(s) with the highest likelihood). Note, however, that echoes the developments in nonmonotonic logic that compute the most plausible interpretation discussed briefly in Sect. 3.4. Current machine learning approaches frame an NLP task as a prediction task (or a series of smaller prediction tasks) that can be formulated as a supervised machine learning problem. Then instead of asking humans to provide expert knowledge to the system (for instance grammar rules), human informants are asked to annotate a large enough sample of input text with the desired output: for instance, parse trees (see Marcus et al. 1993) if the goal is to train a syntactic parser.

For simple classification problems (for instance part-of-speech tagging, which determines the correct part-of-speech of a word in the context in which it is used), machine learning methods have produced very satisfying results (as early as Schmid

1994; Brill 1995). And for more complex problems for which sufficient well-annotated data were available, they succeeded better than analytic and symbolic methods. For instance, syntactic parsers based upon machine learning methods outperform the results of parsers based upon human-authored rule bases (Collins 1997). They now underlie a large part of the computational syntax-semantic interfaces. Let us also cite, in semantics, word sense disambiguation (Kilgarriff and Palmer 2000) or semantic role labeling (Carreras and Màrquez 2005) based upon supervised learning. Machine translation is another example of a task where machine learning methods have attracted most of current investigations. Traditional rule-based systems were replaced or augmented with statistical phrase-based systems (see for instance Allauzen and Yvon 2012), which are now in turn supplanted by neural methods, which now underlie commercial systems such as Google's or Systran's (Deng et al. 2017).

For complex problems with less well-annotated data (a common situation in text semantics), machine learning methods have not yet produced the expected results. To understand the issues, let us recall the distinction between supervised machine learning, which relies upon complete annotations, and unsupervised or semi-supervised learning, which can use unannotated data (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2). For linguistic problems, supervised learning has been used most often.

However, for text semantics and discourse pragmatics, supervised learning is faced with fundamental problems. A major issue, for instance, in corpora that are annotated with discourse structure or temporal structure (TimeML, e.g., the THYME corpus (Styler et al. 2014)), is that these structures are complex and much less well understood than syntactic structures. This complexity is such that to obtain an inter-annotator agreement that reaches a usual level in computational linguistics, annotation guidelines must simplify or strongly regulate human annotators' intuitions. Complexity also leads to consistency issues: for instance, temporal structure relation annotations are often globally inconsistent with temporal relation axioms, e.g. those of (Allen 1983). Finally, the complexity of the annotation task makes human annotation long and costly; researchers are then faced with a scarcity of data, which moreover are often noisy. Despite these issues, annotated corpora have been prepared and distributed, for instance for temporal structure detection (Styler et al. 2014). In one such task (Bethard et al. 2017b), the best-performing methods (Tourille et al. 2017) determined the relation of an event to the document time with an F1-score above the inter-annotator agreement (though still below the agreement between annotator and adjudicator), but fell short of human annotators by 30pt F1-score in the detection of 'narrative containers' (Bethard et al. 2017b), a relation that is much more difficult to determine reliably. By training and testing their systems on the two subdomains of the THYME corpus, researchers were also able to study the dependence of supervised learning systems to their training domain and to test domain-adaptation methods: a loss of 15–20pt F1-score was observed compared to in-domain testing (Bethard et al. 2017b).

For these reasons, much current research investigates less supervised or unsupervised machine learning methods. Among the former is bootstrapping. To reduce the

need for annotated data, bootstrapping approaches were explored in which only a few seed examples are provided (Yarowsky 1995; Brin 1999; Agichtein and Gravano 2000; Ravichandran and Hovy 2002). For instance, to extract book–author relations, (Brin 1999) looked for instances of five examples of (author, title) pairs and abstracted patterns from their occurrences in 24 million web pages. These patterns were used to find more (author, title) pairs, which led to more patterns, and so on.

Distant supervision (a method to create *weakly labeled data*) leverages the existence of an external knowledge base to mitigate the absence or scarcity of annotations (Mintz et al. 2009). For example, given instances of relations stored in the Freebase knowledge base (Bollacker et al. 2008) between pairs of entities, one can detect these entities in a large corpus and collect the sentences that contain these pairs of entities. Assuming that these sentences are likely to express their Freebase relation, one can then use them as noisy positive examples to train a classifier to recognize this relation (Mintz et al. 2009).

Transfer learning acquires knowledge while learning to solve a source task (e.g., predicting the next word in a sentence) and applies it to another task (e.g., measuring semantic similarity) (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2). In NLP this is of particular interest when no human annotation is needed for the source task: for instance, training to predict the next word in a sentence only needs a word-segmented corpus. This is sometimes construed as unsupervised learning, but in truth a supervised classifier is trained, albeit no human annotation was needed. For instance, the word2vec algorithm (Mikolov et al. 2013) (see below, Sect. 4.4) trains a system to predict nearby words given a source word, and as a by-product of the resulting trained classifier creates a vector-space representation of each input word such that words with similar distributions of neighbors obtain vectors that are close to each other.

Another current challenge is to find how to mix machine learning approaches with a priori human knowledge in an optimal way. The possibilities of including symbolic constraints in machine learning methods are limited to constraints that can be expressed in logic fragments of very low expressivity (generally propositional logic). Instead, the easiest way to do so is to add a priori knowledge (e.g., lexical knowledge, known patterns, etc.) as features that annotate the training and test examples of supervised learning. Another direction consists in using a priori knowledge to generate training examples for a classifier. For instance, (Goikoetxea et al. 2015) generate pseudo-sentences along random walks that follow the links among the synonym sets ('synsets') of the WordNet lexical database (Miller et al. 1990; Fellbaum 1998), sampling a word from each traversed synset. They apply standard methods to the resulting corpus to create word embeddings (see below, Sect. 4.4) that rival those obtained from a much larger natural corpus. We return to this topic in Sect. 4.4 in the context of neural network representation learning.

4.3 Unsupervised Learning for NLP

Unsupervised learning explores the distribution of a set of objects, for instance words or texts, to uncover hidden relations (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2). The simplest form of unsupervised machine learning used in language processing is clustering, which aims to group together objects that are more similar to one another than to those in other groups. This has been used extensively to help elicit texts with similar topics or genres (Biber 1989); to acquire word senses from their use in monolingual or multilingual corpora (Diab 2000; Apidianaki 2008); to rediscover the tree of language families based upon errors made by second-language learners of English (Nagata and Whittaker 2013)—to name but a few applications.

The basic method to cluster texts consists in representing them as vectors of words, where each cell contains a measure of association between a word and a text (e.g., occurrence count, tf.idf) and defining a similarity on the resulting vector space (e.g., cosine). Observing that different words may have similar meanings, (Deerwester et al. 1990)'s Latent Semantic Indexing (LSI) model further applies singular-value decomposition to the word–text matrix to represent a document with a smaller number of dimensions that are not words anymore, but linear combinations thereof. Both words and texts can be represented according to these dimensions, and words with similar meanings are expected to be mapped to nearby positions in this reduced vector space.

Pursuing in this direction of exploiting the word–text matrix and reducing its dimension, topic models (e.g., Blei et al. 2003; Steyvers and Griffiths 2006) consider that texts "are a mixture of topics, where a topic is a probability distribution over words" (Steyvers and Griffiths 2006). Among these models, probabilistic latent semantic indexing (pLSI) (Hofmann 1999) assumes that topics are drawn from a document-specific distribution over topics. In contrast, latent Dirichlet allocation (LDA) (Blei et al. 2003) defines a proper generative framework in which the topic distribution is itself governed by a parameter sampled from a smooth distribution. Topics identified this way can be exploited directly for various purposes, such as studying the evolution of themes over time (Hall et al. 2008). LDA has also been used as a tool in a number of NLP tasks, for instance unsupervised relation discovery (Yao et al. 2012). Topic models were extended to parallel text collections in two languages (Zhao and Xing 2007), inferring word alignments. (Mimno et al. 2009) generalized them to a larger number of languages and to non-parallel, comparable texts, such as Wikipedia articles: each tuple of comparable texts in the considered languages is assumed to share the same tuple-specific distribution over topics. (Vulić et al. 2011) used this model to find word translations in comparable Wikipedia articles.

Other examples of unsupervised learning include the segmentation of texts into words (Brent 1999) and of words into morphemes (Creutz and Lagus 2002). (Brown et al. 1992) clustered words according to their context of occurrence, building distributional classes, by maximizing the likelihood of the training text for the corresponding bi-class language model.

4.4 Learning Semantic Representations

The availability of very large corpora enabled researchers to implement distributional semantics (Harris 1954; Habert and Zweigenbaum 2002), according to which the meaning of a word is determined by its contexts of use. One can represent all the usages of a word in a corpus with a vector that encodes the number of times this word has a predicate-argument relation with any other word in the corpus or some longer-distance or more complex dependency—or, to simplify processing, the number of times this word occurs within a maximal distance from any other word in the corpus. Instead of simple co-occurrence counts, these vectors can be populated with association strengths computed using mutual information or other scores that reveal how much these co-occurrences depart from a baseline distribution. Words that can be used in similar contexts are expected to have similarities in meaning, hence the proximity of word vectors (typically measured by the cosine of the angle they make) has proved to be a useful test for the semantic similarity of words at least with respect to certain tasks like word class elicitation and thesaurus induction (Hirschman et al. 1975; Schütze 1992; Grefenstette 1994; Habert et al. 1996).

By construction, these vectors are high-dimensional (they have the size of the vocabulary) and sparse; the application of dimension reduction algorithms such as Singular Value Decomposition leads to denser, lower-dimensional spaces (Schütze 1992). Besides singular value decomposition, researchers in distributional semantics have also used non negative matrix factorization (Lee and Seung 2000); unlike singular value decomposition, non negative matrix factorization yields positive values at every point in the reduced space, and this makes a probablistic interpretation of these values possible and facilitates the interpretation of what the reduced dimensions mean (Asher et al. 2016). Word vectors with similar properties can also be built through methods that seem quite different (Mnih and Kavukcuoglu 2013; Mikolov et al. 2013), to which we return below. Clustering word vectors with respect to cosine similarity (or an associated distance) leads to the unsupervised acquisition of word classes, which is of interest to lexical semanticists. The method of (Brown et al. 1992), cited above, directly produces word classes from a corpus without an intermediate vector representation. Unsupervised word classes have been used as additional features in supervised classification, for instance for chunking and named entity recognition (Turian et al. 2010).

Adding to this line of research, the renewal of neural network classifiers in the 2010s (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2) contributed to an evolution from a representation of semantics as logical formulas or structured objects to algebraic representations. Neural network classifiers learn a feature representation of their input words that is adapted to their task (Collobert et al. 2011). These feature representations are dense, low-dimensional vectors that have come to be called *word embeddings*. Efficient algorithms such as word2vec (Mikolov et al. 2013) have been proposed to build them from large corpora. However, research has shown that these methods may not be so different from the above methods of dimension reduction (Levy and Goldberg 2014b).

Another research direction, which we already touched upon briefly in Sect. 4.2, is how to make these corpus-based representations and a priori knowledge collaborate. Very large efforts have been invested in the creation of lexical knowledge bases such as WordNet (Fellbaum 1998), and have helped in many NLP tasks: could they be a source of continuous representations too? (Faruqui and Dyer 2015) showed that the answer is positive: they built vectors of 172,418 features based on information found in 8 lexical knowledge bases or corpora including WordNet, FrameNet, etc. (Goikoetxea et al. 2015) transformed the synonymy and hyponymy relations of WordNet into a pseudo-corpus that is then processed by standard methods such as word2vec to generate word vectors. These alternative methods rival corpus-based word vectors. They can also be combined to them: corpus-based word vectors can be 'retrofitted' to pre-existing knowledge bases by transforming the corpus-based vector space a posteriori (Faruqui et al. 2015) or can be made closer to them at construction time by biasing their learning objective (Yu and Dredze 2014).

Let us cite some of the advantages of these representations. They are continuous rather than binary, i.e., they support the notion of a degree of semantic similarity or relatedness. They are acquired with little effort from unannotated text (Mikolov et al. 2013), although starting from a higher level of analysis such as syntactic relations (Levy and Goldberg 2014a) is more consistent with linguistic theory. They are low-dimensional, making further computations lighter. For instance, (Grégoire and Langlais 2017) could exhaustively evaluate the semantic similarity of the cross-product of 370,000 $\times$ 270,000 sentences represented as 300-dimensional vectors, i.e., about 100 billion cosine computations, a feat that would not have been feasible with more complex representations and comparison operations. They can be optimized for various tasks, for instance for the detection of specific types of entities or relations (Lample et al. 2016), entity linking (Ferré et al. 2017), semantic role labeling (FitzGerald et al. 2015), question-answering (Sharp et al. 2016) or machine translation (Sutskever et al. 2014). Optimizing them at the same time for multiple tasks, through multi-task learning, can lead to better results for the individual tasks (Collobert and Weston 2008). Variants have been designed to take into account infra-word information: character n-grams fare better than words for POS tagging (Ling et al. 2015) and parsing (Bojanowski et al. 2017) and can be added to word representations to improve named entity recognition (Lample et al. 2016). In languages with a non-alphabetic writing system such as Chinese (Chen et al. 2015) or Japanese (Misawa et al. 2017), character-level embeddings have also been shown to be relevant, as well as infra-character embeddings based upon radicals (Sun et al. 2014; Li et al. 2015). Conversely, methods have been designed to represent larger text units directly (Le and Mikolov 2014).

A shared representation can be created for multiple languages (Klementiev et al. 2012; Faruqui and Dyer 2014; Chandar et al. 2014; Gouws et al. 2015; Zhang et al. 2017; Conneau et al. 2018; Ruder et al. 2017) and lead to NLP systems that can analyze text in several languages using the same model (Johannsen et al. 2015) or that can perform cross-language text comparisons in a shared space (Grégoire and Langlais 2017). Finally, word embeddings provide a natural input to the neural network classifiers that are increasingly used nowadays.

Among their drawbacks, they require large-size training data, and an external training task such as the prediction of the next word in a sequence of words (what a language model does). By default, polysemous words are represented by the centroid of their sense vectors unless a word sense disambiguation step is applied to create sense vectors (Iacobacci et al. 2015; Li and Jurafsky 2015), although directions have been explored to elicit word senses automatically and represent them separately (Huang et al. 2012). Explicability is another issue. In common neural architectures, the predicted result, e.g. the predicted category in text categorization, depends on the interaction of a very large number of parameters. This makes it difficult to pinpoint which part of the text actually supported the decision. Attention models, which include a more explicit focus on specific text spans, may help in this matter by revealing those text spans that most supported the decision.

Another problem is interpretability. Linguistics, perhaps in contrast to AI, is an empirical science, and so a good theory should not only be predictive but also explanatory: it should help us understand why and how the phenomena are as they are. Moving away from symbolic methods to statistical ones has made explanations more difficult, a theme we already touched upon in our brief discussion of dimension reduction. While this will be no surprise to philosophers who have studied the foundations of physical theories like quantum mechanics, neural network architectures seem to introduce more complications in the effort to understand what they do, because of the presence of the hidden layers in these architectures. In addition, little is really understood about the complexity results of neural architectures though see Eldan and Shamir 2016 for a very interesting discussion of one case.

A final problem with these representations is to figure out how to compose meanings that are represented via the methods of linear algebra. While vectorial representations of individual words have found a great deal of success, at the present time researchers have not found methods of composition that match the power of the lambda calculus and the logical representations that it yields. For instance, despite intensive efforts, it is still not possible to infer from *A woman petted a dog* that there was a dog, using methods of algebraic composition like vector addition, vector multiplication or simple generalizations thereof. This suggests that some hybrid form of meaning representation using both symbolic and stochastic methods is perhaps what we need (Asher et al. 2016).

In another direction, the construction of large corpora where texts are associated to images or videos (e.g. Young et al. 2014) enables researchers to correlate information conveyed through these multiple modalities. This benefits the resolution of some of the above-mentioned tasks, such as word sense disambiguation (Barnard et al. 2003), as well as that of scene analysis tasks (Wang et al. 2016) (see chapter "Artificial Intelligence and Pattern Recognition, Vision, Learning" of this volume). The joint analysis of text and images or videos also contributes to extend distributional semantics to features that are obtained from both language processing and image perception (Bruni et al. 2011). This can be seen as a way to ground language in the world, an important complementary area that attracts increasing interest.

5 Conclusion

In this chapter we have sketched a view of the history of the role of artificial intelligence in natural language processing. We focused on logical, semantic and discursive aspects (Sect. 3) and on the contributions of machine learning (Sect. 4), since it is in these two domains of natural language processing that we see the most immediate interactions between linguistics and artificial intelligence. Natural language processing provided artificial intelligence with new and exciting problems and applications during almost the whole history of these two young disciplines. We believe that this interaction, which benefits both disciplines, will continue to expand.

References

Agichtein E, Gravano L (2000) Snowball: extracting relations from large plain-text collections. In: Proceedings of the fifth ACM Conference on Digital Libraries, ACM, New York, NY, USA, DL '00, pp 85–94. https://doi.org/10.1145/336597.336644

Allauzen A, Yvon F (2012) Statistical methods for machine translation. In: Gaussier E, Yvon F (eds) Textual Information Access, ISTE/Wiley, Paris, chap 7, pp 223–304

Allen JF (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843. https://doi.org/10.1145/182.358434

Apidianaki M (2008) Translation-oriented word sense induction based on parallel corpora. In: Calzolari N, Khalid Choukri BM, Mariani J, Odijk J, Piperidis S, Tapias D (eds) Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco

Asher N (1992) A default, truth conditional semantics for the progressive. Linguist Philos 15:463–508

Asher N (1993) Reference to abstract objects in discourse. Kluwer Academic Publishers, Netherlands

Asher N (2011) Lexical meaning in context: a web of words. Cambridge University Press, Cambridge

Asher N, Lascarides A (2003) Logics of conversation. Cambridge University Press, Cambridge

Asher N, Luo Z (2012) Formalization of coecions in lexical semantics. In: Chemla E (ed) Proceedings of Sinn und Bedeutung, Paris

Asher N, Morreau M (1995) What some generic sentences mean. In: Carlson G, Pelletier F (eds) The generic book. University of Chicago Press, Chicago, pp 300–339

Asher N, Paul S (2016) Evaluating conversational success: weighted message exchange games. In: Hunter J, Simons M, Stone M (eds) 20th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL), New Jersey, USA

Asher N, van de Cruys T, Bride A, Abrusán M (2016) Integrating type theory and distributional semantics: a case study on adjective-noun compositions. Comput Linguist 42(4):703–725

Asher N, Paul S, Venant A (2017) Message exchange games in strategic conversations. J Philos Log 46(4):355–404. https://doi.org/10.1007/s10992-016-9402-1

Barker C, Shan K (2006) Types as graphs: continuations in type logical grammar. J Log Lang Inf 15(4):331–370

Barnard K, Johnson M, Forsyth D (2003) Word sense disambiguation with pictures. In: Regina Barzilay ER, Siskind JM (eds) Proceedings of the HLT-NAACL 2003 Workshop on Learning Word Meaning from Non-Linguistic Data, pp 1–5. http://www.aclweb.org/anthology/W03-0601.pdf

Bethard S, Carpuat M, Apidianaki M, Mohammad SM, Cer D, Jurgens D (eds) (2017a) Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics. http://www.aclweb.org/anthology/S17-2000

Bethard S, Savova G, Palmer M, Pustejovsky J (2017b) SemEval-2017 Task 12: Clinical TempEval. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics, pp 565–572. https://doi.org/10.18653/v1/S17-2093

Biber D (1989) A typology of English texts. Linguistics 27:3–43

Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet Allocation. J Mach Learn Res 3:993–1022

Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146. http://aclweb.org/anthology/Q17-1010

Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '08, pp 1247–1250. https://doi.org/10.1145/1376616.1376746

Bonzon E (2007) Modélisation des interactions entre agents rationnels: les jeux booléens. PhD thesis, Université Paul Sabatier, Toulouse

Bonzon E, Lagasquie-Schiex MC, Lang J, Zanuttini B (2006) Boolean games revisited. In: Brewka G, Coradeschi S, Perini A, Traverso P (eds) ECAI 2006. IOS Press, pp 265–270

Bouaud J, Bachimont B, Zweigenbaum P (1996) Processing metonymy: a domain-model heuristic graph traversal approach. In: Tsujii JI (ed) Proceedings of the 16th COLING, Copenhagen, Denmark, pp 137–142

Boutilier C (1999) Sequential optimality and coordination and multiagent systems. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Stokholm, pp 527–534

Brachman RJ, Schmolze J (1985) An overview of the KL-ONE knowledge representation system. Cogn Sci 9:171–216

Brent MR (1999) An efficient, probabilistically sound algorithm for segmentation and word discovery. Mach Learn 34(1):71–105

Bresnan J (ed) (1982) The mental representation of grammatical relations. MIT Press, Cambridge

Brill E (1995) Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Comput Linguist 21(4):543–565

Brin S (1999) Extracting patterns and relations from the world wide web. In: Atzeni P, Mendelzon A, Mecca G (eds) The The World Wide Web and databases. Springer, Berlin, Heidelberg, pp 172–183

Brown PF, Pietra VJD, de Souza PV, Lai JC, Mercer RL (1992) Class-based n-gram models of natural language. Comput Linguist 18(4):467–79

Bruni E, Tran GB, Baroni M (2011) Distributional semantics from text and images. In: Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics, Association for Computational Linguistics, Edinburgh, UK, pp 22–32. http://www.aclweb.org/anthology/W11-2503

Brysbaert M, Stevens M, Mandera P, Keuleers E (2016) How many words do we know? practical estimates of vocabulary size dependent on word definition, the degree of language input and the participant's age. Front Psychol 7

Burnard L (1995) Users Reference Guide for the British National Corpus. British National Corpus Consortium, Oxford University Computing Services, Oxford, UK, version 1.0 edn, 524p. http://homepages.abdn.ac.uk/k.vdeemter/pages/teaching/NLP/practicals/bnc-doc.pdf

Carreras X, Màrquez L (2005) Introduction to the CoNLL-2005 shared task: semantic role labeling. In: Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), Association for Computational Linguistics, Ann Arbor, Michigan, pp 152–164. http://www.aclweb.org/anthology/W/W05/W05-0620

Chandar APS, Lauly S, Larochelle H, Khapra MM, Ravindran B, Raykar VC, Saha A (2014) An autoencoder approach to learning bilingual word representations. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) Advances in Neural Information Processing

Systems 27: Annual Conference on Neural Information Processing Systems. Montreal, Quebec, Canada, pp 1853–1861
Chen X, Xu L, Liu Z, Sun M, Luan H (2015) Joint learning of character and word embeddings. In: Proceedings of IJCAI, pp 1236–1242
Chomsky N (1964) Syntactic structures. Mouton and Co., The Hague
Chomsky N (1965) Aspects of the theory of syntax. MIT Press, Cambridge, MA
Chomsky N (1981) Lectures on government and binding. Foris, Dordrecht
Church A (1940) A formulation of the simple theory of types. J Symb Log 5:56–68
Collins M (1997) Three generative, lexicalized models for statistical parsing. In: Proceedings of the Conference on 35th Annual Meeting and 8th Conference of the European Chapter, Association for Computational Linguistics, pp 16–23
Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, ACM, New York, NY, USA, ICML '08, pp 160–167. https://doi.org/10.1145/1390156.1390177
Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186
Conneau A, Lample G, Ranzato M, Denoyer L, Jgou H (2018) Word translation without parallel data. In: ICLR
Cooper R (2011) Copredication, quantification and frames. In: Logical Aspects of Computational Linguistics (LACL'2011) LNAI, vol 6736
Crawford V, Sobel J (1982) Strategic information transmission. Econometrica 50(6):1431–1451
Creutz M, Lagus K (2002) Unsupervised discovery of morphemes. In: Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning, Association for Computational Linguistics, pp 21–30. https://doi.org/10.3115/1118647.1118650, http://www.aclweb.org/anthology/W02-0603
Davies M (2011) Google books (American English) Corpus (155 billion words, 1810–2009). Available online at http://googlebooks.byu.edu/
Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by Latent Semantic Analysis. J Am Soc Inf Sci 41(6):391–407
Deng Y, Kim J, Klein G, Kobus C, Segal N, Servan C, Wang B, Zhang D, Crego J, Senellart J (2017) SYSTRAN purely neural MT engines for WMT2017. In: Proceedings of the Second Conference on Machine Translation, Association for Computational Linguistics, Copenhagen, Denmark, pp 265–270. http://www.aclweb.org/anthology/W17-4722
Diab M (2000) An unsupervised method for multilingual word sense tagging using parallel corpora. In: ACL-2000 Workshop on Word Senses and Multi-Linguality, Association for Computational Linguistics, Hong Kong, China, pp 1–9. https://doi.org/10.3115/1117724.1117725. http://aclweb.org/anthology/W00-0801
Domshlak C (2002) Modeling and reasoning about preferences with CP nets. PhD thesis, Ben Gurion University
Dowty DR (1979) Word meaning and Montague grammar: the semantics of verbs and times in generative semantics and Montague's PTQ, vol 7. Studies in linguistics and philosophy. Kluwer, Dordrecht
Eldan R, Shamir O (2016) The power of depth for feedforward neural networks. In: Conference on Learning Theory, pp 907–940
Farrell J (1993) Meaning and credibility in cheap talk games. Games Econ Behav 5:514–531
Faruqui M, Dyer C (2014) Improving vector space word representations using multilingual correlation. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Gothenburg, Sweden, pp 462–471. http://www.aclweb.org/anthology/E14-1049
Faruqui M, Dyer C (2015) Non-distributional word vector representations. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International

Joint Conference on Natural Language Processing (Volume 2: short papers), Association for Computational Linguistics, pp 464–469. https://doi.org/10.3115/v1/P15-2076

Faruqui M, Dodge J, Jauhar SK, Dyer C, Hovy E, Smith NA (2015) Retrofitting word Vectors to Semantic Lexicons. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Denver, Colorado, pp 1606–1615. http://www.aclweb.org/anthology/N15-1184

Fellbaum C (ed) (1998) WordNet: an electronic database. MIT Press, Cambridge, MA

Ferré A, Zweigenbaum P, Nédellec C (2017) Representation of complex terms in a vector space structured by an ontology for a normalization task. In: BioNLP 2017, Association for Computational Linguistics, Vancouver, Canada, pp 99–106. http://www.aclweb.org/anthology/W17-2312.pdf

FitzGerald N, Täckström O, Ganchev K, Das D (2015) Semantic role labeling with neural network factors. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp 960–970. http://aclweb.org/anthology/D15-1112

Fox B (1987) Discourse structure and anaphora. Cambridge University Press, Cambridge

Frey W, Reyle U, Rohrer C (1983) Automatic construction of a knowledge base by analyzing texts in natural language. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp 727–730

Goikoetxea J, Soroa A, Agirre E, (2015) Random walks and neural network language models on knowledge bases. In: Human Language technologies: the (2015) conference of the North American Chapter of the Association for Computational Linguistics. Denver, CO, pp 1434–1439

Gouws S, Bengio Y, Corrado G (2015) BilBOWA: fast bilingual distributed representations without word alignments. In: Bach F, Blei D (eds) Proceedings of the 32nd International Conference on Machine Learning, Lille, France, JMLR Workshop and Conference Proceedings, vol 37

Graff D, Cieri C (2003) English Gigaword, LDC2003T05. Linguistic Data Consortium, Philadelphia, web download

Grefenstette G (1994) Explorations in Automatic Thesaurus Discovery. Natural Language Processing and Machine Translation, Kluwer Academic Publishers, London

Grégoire F, Langlais P (2017) BUCC 2017 Shared Task: a first attempt toward a deep learning framework for identifying parallel sentences in comparable corpora. In: Proceedings of the 10th Workshop on Building and Using Comparable Corpora, Association for Computational Linguistics, Vancouver, Canada, pp 46–50. http://aclweb.org/anthology/W17-2509

Grice HP (1975) Logic and conversation. In: Cole P, Morgan JL (eds) Syntax and semantics volume 3: speech acts. Academic Press, Cambridge, pp 41–58

Groenendijk J, Stokhof M (1991) Dynamic predicate logic. Linguist Philos 14:39–100

de Groote P (2006) Towards a Montagovian account of dynamcs. In: SALT 16, CLC Publications, pp 148–155

Grosz B (1979) Utterance and objective: issues in natural language communication. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1979), pp 1067–1076

Grosz B, Sidner C (1986) Attention, intentions and the structure of discourse. Comput Linguist 12:175–204

Grosz BJ, Kraus S (1993) Collaborative plans for group activities. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, Los Altos, California, pp 367–373

Habert B, Zweigenbaum P (2002) Contextual acquisition of information categories: what has been done and what can be done automatically? In: Nevin BE, Johnson SM (eds) The Legacy of Zellig Harris: Language and Information into the 21st Century, vol 2. Mathematics and Computability of Language. John Benjamins, Amsterdam, pp 203–231

Habert B, Naulleau E, Nazarenko A (1996) Symbolic word clustering for medium-size corpora. In: Proceedings of 16th International Conference on Computational Linguistics. Denmark, Copenhagen, pp 490–495

Hall D, Jurafsky D, Manning CD (2008) Studying the history of ideas using topic models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics

Halpern J, Fagin R, Moses Y, Vardi M (1995) Reasoning About Knowledge. MIT Press, Cambridge, MA

Harel D (1984) Dynamic logic. In: Gabbay D, Guenther F (eds) Handbook of Philosophical Logic, Volume II Extensions of Classical Logic, vol 2. D. Reidel Publishing Co., Dordrecht, pp 497–604

Harris ZS (1954) Distributional structure. Word 10(2–3):146–162

Hirschman L, Grishman R, Sager N (1975) Grammatically-based automatic word class formation. Inf Process Manag 11:39–57

Hobbs JR (1985) On the coherence and structure of discourse. Tech Rep csli-85-37, Center for the Study of Language and Information, Stanford University

Hobbs JR, Stickel M, Appelt D, Martin P (1993) Interpretation as abduction. Artif Intell 63(1–2):69–142

Hofmann T (1999) Probabilistic Latent Semantic Indexing. In: Proceedings of the Twenty-Second Annual International SIGIR Conference

Huang E, Socher R, Manning C, Ng A (2012) Improving word representations via global context and multiple word prototypes. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: long papers), Association for Computational Linguistics, Jeju Island, Korea, pp 873–882. http://www.aclweb.org/anthology/P12-1092

Iacobacci I, Pilehvar MT, Navigli R (2015) SensEmbed: learning sense embeddings for word and relational similarity. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: long papers), Association for Computational Linguistics, Beijing, China, pp 95–105. http://www.aclweb.org/anthology/P15-1010

Johannsen A, Martínez Alonso H, Søgaard A (2015) Any-language frame-semantic parsing. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp 2062–2066. http://aclweb.org/anthology/D15-1245

Johnson C, Fillmore CJ (2000) The FrameNet tagset for frame-semantic and syntactic coding of predicate-argument structure. In: Proceedings ANLP-NAACL, Seattle, WA

Johnson M (1989) Parsing as deduction: the use of knowledge of language. J Psycholinguist Res 18(1):105–128

Joshi AK, Levy LS, Takahashi M (1975) Tree adjunct grammars. J Comput Syst Sci 10(1):136–163

Kameyama M (1995) Indefeasible semantics and defeasible pragmatics. Quantifiers, deduction and context, vol 57. CSLI lecture notes, pp 476–482

Kamp H (1973) Free choice permission. Proc Aristot Soc 74:57–74

Kamp H, Reyle U (1993) From Discourse to the Lexicon: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Kluwer Academic Publishers

Kaplan R, Bresnan J (1983) Lexical-functional grammar: a formal system for grammatical representation. In: Bresnan J (ed) The Mental Representation of Grammatical Relations. MIT Press, Cambridge MA

Kehler A, Kertz L, Rohde H, Elman J (2008) Coherence and coreference revisited. J Semant (Special Issue on Processing Meaning) 25(1):1–44

Kilgarriff A, Palmer M (2000) Special issue on senseval: evaluating word sense disambiguation programs. Comput Humanities 34(1–2):1–13

Klavans JL, Resnik P (eds) (1996) The Balancing Act. MIT Press, Cambridge, MA

Klementiev A, Titov I, Bhattarai B (2012) Inducing crosslingual distributed representations of words. In: Proceedings of COLING 2012, The COLING 2012 Organizing Committee, Mumbai, India, pp 1459–1474. http://www.aclweb.org/anthology/C12-1089

Lambek J (1958) The mathematics of sentence structure. Am Math Mon 65:154–170

Lample G, Ballesteros M, Subramanian S, Kawakami K, Dyer C (2016) Neural architectures for named entity recognition. In: Proceedings of NAACL-HLT, Association for Computational Linguistics, San Diego, CA, pp 260–270

Lascarides A, Asher N (1993) Temporal interpretation, discourse relations and commonsense entailment. Linguist Philos 16(5):437–493

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14), pp 1188–1196

Lee DD, Seung HS (2000) Algorithms for non-negative matrix factorization. Advances in Neural Information Processing Systems, vol 13. pp 556–562

Lenat D (1995) Cyc: a large-scale investment in knowledge infrastructure. Commun ACM 38(11):33–38

Levy O, Goldberg Y (2014a) Dependency-based word embeddings. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (volume 2: short papers), Association for Computational Linguistics, Baltimore, Maryland, pp 302–308. http://www.aclweb.org/anthology/P14-2050

Levy O, Goldberg Y (2014b) Neural word embedding as implicit matrix factorization. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) Advances in Neural Information Processing Systems 27, Curran Associates Inc., pp 2177–2185

Lewis D (1969) Convention: a Philosophical Study. Harvard University Press, Cambridge

Li J, Jurafsky D (2015) Do multi-sense embeddings improve natural language understanding? In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp 1722–1732. http://aclweb.org/anthology/D15-1200

Li Y, Li W, Sun F, Li S (2015) Component-enhanced Chinese character embeddings. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp 829–834

Ling W, Dyer C, Black AW, Trancoso I, Fermandez R, Amir S, Marujo L, Luis T (2015) Finding function in form: compositional character models for open vocabulary word representation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp 1520–1530. http://aclweb.org/anthology/D15-1176

Löbner S (2014) Evidence for frames from human language. In: Gamerschlag T, Gerland D, Osswald R, Petersen W (eds) Frames and Concept Types, vol 94. Studies in Linguistics and Philosophy. Springer, Cham

Lochbaum KE (1998) A collaborative planning model of intentional structure. Comput Linguist 24(4):525–572

Luo Z (2010) Type-Theoretical Semantics with Coercive Subtyping. SALT20, Vancouver

Luo Z (2012) Formal Semantics in Modern Type Theories with Coercive Subtyping. Linguist Philos

Mann WC, Thompson SA (1986) Rhetorical structure theory: description and construction of text structures. In: Kempen G (ed) Natural Language Generation: New Results in Artificial Intelligence. Springer, Dordrecht, pp 279–300

Mann WC, Thompson SA (1987) Rhetorical structure theory: a framework for the analysis of texts. Int Pragmat Assoc Pap Pragmat 1:79–105

Manning CD, Schütze H (1999) Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA

Marcu D (1997) The rhetorical parsing of unrestricted natural language texts. In: Cohen PR, Wahlster W (eds) Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Somerset, New Jersey, pp 96–103

Marcus MP, Santorini B, Marcinkiewicz MA (1993) Building a large annotated corpus of English: the Penn treebank. Comput Linguist 19(2):313–330

McCarthy J (1980) Circumscription-a form of non-monotonic reasoning. Artif Intell 13(1–2):27–39

Mercer R (1987) A default logic approach to the derivation of natural language presuppositions. PhD thesis, University of British Columbia
Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: Proceedings of Workshop at ICLR'13
Miller GA, Beckwith R, Fellbaum C, Gross D, Miller KJ (1990) Introduction to WordNet: an on-line lexical database. Int J Lexicogr 3(4):235–244
Mimno D, Wallach HM, Naradowsky J, Smith DA, McCallum A (2009) Polylingual topic models. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '09, pp 880–889. http://dl.acm.org/citation.cfm?id=1699571.1699627
Minsky M (1969) Semantic Information Processing. The MIT Press, Cambridge
Minsky M (1994) A framework for representing knowledge. In: Winston PH (ed) Psychology of Computer Vision. McGraw-Hill, New York
Mintz M, Bills S, Snow R, Jurafsky D (2009) Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: volume 2 - volume 2, Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pp 1003–1011. http://dl.acm.org/citation.cfm?id=1690219.1690287
Misawa S, Taniguchi M, Miura Y, Ohkuma T (2017) Character-based bidirectional LSTM-CRF with words and characters for Japanese named entity recognition. In: Proceedings of the First Workshop on Subword and Character Level Models in NLP, Association for Computational Linguistics, Copenhagen, Denmark
Mnih A, Kavukcuoglu K (2013) Learning word embeddings efficiently with noise-contrastive estimation. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in Neural Information Processing Systems 26, Curran Associates, Inc., pp 2265–2273. http://papers.nips.cc/paper/5165-learning-word-embeddings-efficiently-with-noise-contrastive-estimation.pdf
MUC5 (1993) Fifth message understanding conference (MUC-5). Defense Advanced Research Projects Agency, Morgan Kaufmann, San Francisco, Ca
Nagata R, Whittaker E (2013) Reconstructing an Indo-European family tree from non-native English texts. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (volume 2: short papers), Association for Computational Linguistics, Sofia, Bulgaria
Oehrle R, Bach E, Wheeler D (eds) (1988) Categorial Grammars and Natural Language Structures, vol 32. Studies in Linguistics and Philosophy. Kluwer, Dordrecht
Pelletier FJ, Asher N (1997) Generics and defaults. In: van Benthem J, ter Meulen A (eds) Handbook of Logic and Language. North Holland, Amsterdam, pp 1125–1177
Polanyi L (1985) A theory of discourse structure and discourse coherence. In: W H Eilfort PDK, Peterson KL (eds) Papers from the General Session at the 21st Regional Meeting of the Chicago Linguistics Society
Pollard C (1984) Generalized phrase structure grammars, head grammars and natural language. PhD thesis, Stanford University
Pollard C, Sag IA (1987) Information-based syntax and semantics. vol 1: fundamentals. CSLI
Prasad R, Dinesh N, Lee A, Miltsakaki E, Robaldo L, Joshi A, Webber B (2008) The penn discourse treebank 2.0. In: Calzolari N, Choukri K, Maegaard B, Mariani J, Odijk J, Piperidis S, Tapias D (eds) Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco
Pustejovsky J (1995) The Generative Lexicon. MIT Press, Cambridge
Rabin M (1990) Communication between rational agents. J Econ Theory 51:144–170
Ravichandran D, Hovy E (2002) Learning surface text patterns for a question answering system. In: Proceedings of 40th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pp 41–47. https://doi.org/10.3115/1073083.1073092. http://www.aclweb.org/anthology/P02-1006
Reiter R (1980) A logic for default reasoning. Artif Intell 13:91–132

Ruder S, Vulić I, Søgaard A (2017) A survey of cross-lingual embedding models. CoRR abs/1706.04902v2

Schank R, Abelson R (1977) Scripts, Plans, Goals, and Understanding. Lawrence Erlbaum Associates, Hillsdale, New Jersey

Schmid H (1994) Probabilistic part-of-speech tagging using decision trees. In: Proceedings of the International Conference on New Methods in Language Processing, Manchester, UK, pp 44–49

Schütze H (1992) Dimensions of meaning. In: Proceedings of Supercomputing'92, Minneapolis, pp 787–796

Searle JR (1980) Minds, brains, and programs. Behav Brain Sci 3(3):417–424

Sharp R, Surdeanu M, Jansen P, Clark P, Hammond M (2016) Creating causal embeddings for question answering with minimal supervision. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, pp 138–148. https://aclweb.org/anthology/D16-1014

Sowa JF (1984) Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, London

Steyvers M, Griffiths T (2006) Probabilistic topic models. In: Landauer T, McNamara D, Dennis S, Kintsch W (eds) Latent Semantic Analysis: A Road to Meaning. Lawrence Erlbaum

Styler W 4th, Bethard S, Finan S, Palmer M, Pradhan S, de Groen P, Erickson B, Miller T, Lin C, Savova G, Pustejovsky J (2014) Temporal annotation in the clinical domain. Trans Assoc Comput Linguist 2:143–154

Sun Y, Lin L, Tang D, Yang N, Xiaolong Wang ZJ (2014) Radical-enhanced Chinese character embedding. In: Proceedings of ICONIPS

Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - volume 2, MIT Press, Cambridge, MA, USA, NIPS'14, pp 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173

Thomason RH (ed) (1974) Formal Philosophy. Yale University Press, New Haven

Touretsky D, Horty J, Thomason R (1987) A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In: Proceedings of the International Joint Conference on Artificial Intelligence, vol 1. pp 476–482

Tourille J, Ferret O, Tannier X, Névéol A (2017) LIMSI-COT at SemEval-2017 Task 12: Neural architecture for temporal information extraction from clinical narratives. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics, Vancouver, Canada, pp 597–602. http://www.aclweb.org/anthology/S17-2098

Turian J, Ratinov LA, Bengio Y (2010) Word representations: A simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Uppsala, Sweden, pp 384–394. http://www.aclweb.org/anthology/P10-1040

van Rooij R (2004) Signalling games select horn strategies. Linguist Philos 27:493–527

van Rooij R, Schultz K (2004) Exhaustive interpretation of complex sentences. J Log Lang Inf 13(4):491–519

Veltman F (1996) Defaults in update semantics. J Philos Log 25:221–261

Voorhees EM, Harman DK (2005) TREC: Experiment and Evaluation in Information Retrieval. The MIT Press, Cambridge

Vulić I, Smet WD, Moens MF (2011) Identifying word translations from comparable corpora using latent topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT), pp 479–484

Wang L, Li Y, Lazebnik S (2016) Learning deep structure-preserving image-text embeddings. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition

Weizenbaum J (1966) Eliza-a computer program for the study of natural language communication between man and machine. Commun ACM 9(1):1–5

Yao L, Riedel S, McCallum A (2012) Unsupervised relation discovery with sense disambiguation. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics

Yarowsky D (1995) Unsupervised word sense disambiguation rivalling supervised methods. In: Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Cambridge, Massachusetts, USA

Young P, Lai A, Hodosh M, Hockenmaier J (2014) From image descriptions to visual denotations: new similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics 2:67–78. https://transacl.org/ojs/index.php/tacl/article/view/229

Yu M, Dredze M (2014) Improving lexical embeddings with semantic knowledge. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (volume 2: short papers), Association for Computational Linguistics, Baltimore, Maryland, pp 545–550. http://www.aclweb.org/anthology/P14-2089

Zhang M, Liu Y, Luan H, Sun M (2017) Earth mover's distance minimization for unsupervised bilingual lexicon induction. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, pp 1924–1935

Zhao B, Xing EP (2007) HM-BiTAM: Bilingual topic exploration, word alignment, and translation. In: NIPS

Information Retrieval and Artificial Intelligence

Mohand Boughanem, Imen Akermi, Gabriella Pasi and Karam Abdulahhad

Abstract Information Retrieval (IR) is a process involving activities related to human cognition and to knowledge management; as such, the definition of Information Retrieval Systems can benefit of the application of artificial intelligence techniques to account for the intrinsic uncertainty and imprecision that characterize the subjectivity of this task. This chapter presents a synthetic analysis of the IR task from an AI perspective and explores how AI techniques are employed within IR.

1 Introduction

Information retrieval (IR) systems (aka search engines) are widely employed in a variety of applications, among which Web search engines are the most known example. These tools are used by millions of users and became an essential part of our daily lives.

Examples of applications that benefit from Information Retrieval Systems (IRSs) are digital libraries, medical based applications, and desktop search. Regardless of the application domain, the IR task is conditioned by several important factors. First, an IRS constitute a so called pull technology, as it implies that a user proactively specifies a keyword based query to express a specific information need. However, user queries hardly capture the complexity of a user need, and the few specified keywords

M. Boughanem (✉) · I. Akermi
IRIT, Université Toulouse III, Toulouse, France
e-mail: mohand.boughanem@irit.fr

I. Akermi
e-mail: imen.akermi@irit.fr

G. Pasi
Università degli Studi di Milano-Bicocca, Milan, Italy
e-mail: pasi@disco.unimib.it

K. Abdulahhad
GESIS - Leibniz Institute for the Social Sciences, Cologne, Germany
e-mail: karam.abdulahhad@gesis.org

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_5

are often imprecise, incomplete, ambiguous, and often inadequate to express the real user intent. A user's query imposes a set of constraints on texts (we only consider textual documents in this chapter) that are written in natural language, which is known to be ambiguous. This is another important barrier that the IR system must deal with. Finally, the IR system has to decide which documents to present to the user, i.e. it has to estimate their relevance, based on an analysis of the textual content of these documents.

The above observations show some of the aspects related to the difficulty underlying the IR task. Artificial Intelligence (AI) techniques may help better tackle the IR task by addressing the incompleteness, vagueness and subjectivity intrinsic in the IR process. Possible applications of AI to IR have been extensively discussed in the literature (Jones 1983, 1991; Croft 1987; Mandl 2009), by raising two main questions: how the IR task can be addressed from an AI point of view, and how IR is considered within AI.

The purpose of this synthetic survey is to highlight different aspects related to the first question. We give in Sect. 3 an overview related to AI and IR. Then, we discuss in the subsequent sections the different AI approaches that have been employed in IR.

2 Information Retrieval: Background

An IR system aims at selecting from a huge document collection the *documents* that are deemed *relevant* to a particular *user need* expressed by means of a *query* (Salton and McGill 1986). This definition points out three key concepts: *documents*, *information need/query* and *relevance*, which can be defined as follows:

- A document (or item) constitutes a unit of retrievable information that can be selected to satisfy a user need expressed by a query. A document can be a text, an image, a video, an audio or it can be multimedia. In this chapter we only consider textual documents.
- A query is a formal representation of the user's information need. It is usually composed of a set of keywords eventually connected with Boolean operators.
- Relevance is the core notion in IR. It can be defined as a relationship between a user need and a document. This notion is very subjective and difficult to model. Several IR models have been proposed, based on a variety of theoretical frameworks namely, probability theory, linear algebra, set theory, fuzzy logic, etc. Most of these models address topical relevance, usually relying on query-document matching.

A typical IR system is composed of three main components, namely, document representation, user need representation and query-document matching. The formal representation of a document is produced by the indexing process, which usually consists in extracting a set of features, basically keywords, that characterize the content of the document. The representation of a user need is defined through the query formulation phase.

IR models are finalized at formalizing the query-document matching, and they allow one to estimate the so called topical relevance. Query expansion can also be part of the IR process, to the aim of modifying the original query by adding terms extracted from the retrieved documents or drawn out from external resources such as dictionaries, ontology, etc.

3 Artificial Intelligence for Information Retrieval

Intelligent Information Retrieval was introduced in the 1980s when AI techniques were perceived as a promising means to define effective IRSs. As a consequence, there has been a shift from the classical IRSs, i.e. those based on the Boolean model to ranking-based systems and probabilistic approaches. Therefore, various IR approaches were developed involving AI techniques that helped to better express the document content, to better learn the users needs and to more effectively formalize the concept of relevance. Consequently, several formal definitions were proposed to address the concept of Intelligent Information retrieval (IIR).

Generally, IIR systems can be described as systems using AI techniques to replicate intelligence through the IR process. Spark Jones (1983) defined an IIR system as "*a system with a knowledge base and inferential capabilities that can be used to establish connections between a request and a set of documents*". In Van Rijsbergen (1986), IIR is considered as an inference process described as "*given a document representation D and a request R, IR is the process of establishing a probability for* "$D \rightarrow$ R".

A two-fold definition was presented in Chen and Dhar (1989), Belkin and Marchetti (1989), stating that intelligence may intervene when "*users do not know what information they need before accessing the system so they have to be helped in forming the query to the information retrieval system*" (Chen and Dhar 1989), and that "*users become aware of their information need only through this process of interacting with the system*" (Belkin and Marchetti 1989).

A machine and Human-oriented perspectives were also put forward to define IIR systems. Belkin et al. (1987) consider that "*an intelligent IR system was one in which the functions of the human intermediary were performed by a program, interacting with the human user*". In Maes (1994), the authors state that "*intelligent IR is performed by a computer program (a so-called intelligent agent), which, acting on (perhaps minimal or even no explicit) instructions from a human user, retrieves and presents information to the user without any other interaction*".

According to these definitions, Cole (1998) resumes that the main goal of using AI techniques is to support IR systems in the process of assisting users to discover documents relevant to their information need by interacting with the system.

Consequently, a recurrent question that arises is: how AI techniques could benefit IR systems ? One of the answers that provides some insights for this question is stressed by Karen Sparck Jones (1991) who stated that, "*IR is seen as a search for unknown, and under-specified, information in a world of information as conveyed by*

natural language texts, it is easy to conclude that what AI discovers about the representation of knowledge, reasoning under uncertainty, and learning, will be clearly applicable to document retrieval". In Ding (2001), KSJ claims were paraphrased according to three different aspects:

- *"Knowledge representation. IRs representation of entities and relations is very weak. Concept names are not normalized, and descriptions are mere sets of independent terms without structure… Concepts and topics, term and description meanings are left implicit… The relation between terms is only association based on co-presence… While, the representation in AI is strong. There already exist various full-fledged methods and techniques to model the knowledge. Ontology can be considered as the generic term for generalizing these representation ideas.*
- *Reasoning: Reasoning in IR is also weak, looking at what is in common between descriptions and preferring one item over another because more in shared (whether as different words or, via weighting, occurrences of the same word)… The probabilistic network approach, that allows for more varied forms of search statement and matching condition, does not alter the basic style of reasoning. While development in knowledge representation of AI, especially ontology provides the backbone for reasoning and also guarantees the reasoning.*
- *Learning: Loosely speaking, the relevance feedback of IR can be considered as forms of learning. This again is very weak in IR. In this part, machine learning will link the IR and AI together to improve both sides."*

To sum up, these claims asserted that several AI areas can help handling IR tasks, particularly:

- Natural language Processing techniques (See chapter "Databases and Artificial Intelligence" of this volume) and Knowledge Representation provide tools that allow one to better represent the document content,
- Reasoning under uncertainty, e.g. by modal logic, probabilistic reasoning, fuzzy logic, can help both in the phase of query formulation and relevance assessment
- Machine Learning techniques may intervene at different levels of the IR process. Indeed, recent advances in neural networks have offered new perspectives to IR. Such approaches have been applied to handle different IR tasks such as learning document or query representations and learning the ranking model.
- Other close topics such as Metaheuristics (evolutionary computation), game theory, multi-agent systems have been applied in IR. They, generally, regard IR as an optimization problem where individuals, agents or players cooperate to realize a given task, namely, building the "best" document or query representation, retrieving the most relevant set of documents, or building an effective relevance function.

The next sections will discuss how AI topics, particularly those listed above, can help IR with respect to the three main components of the general IR process, namely document representation, information need representation and relevance modelling.

4 Document Representation

The majority of document representation models are based on single words, commonly referred to as *a bag of words* representation. Document content (resp. query content) is represented as a set of independent weighted words. This representation has several limitations due to the lexical variety of words (synonym words) and the semantic variation of words (polysemous words). This leads to a known issue called "term-mismatch" or "word-mismatch". Therefore, setting up a more sophisticated representation that can go beyond a simple bag of words has been considered as necessary since decades. This had been obvious for the pioneers of IR (Cleverdon and Keen 1966; Sparck Jones 1972; Salton 1991; Luhn 1957). In fact, they proposed to represent texts by syntactic or semantic units much more appropriate to represent the meaning of the document's components. Therefore, AI techniques, especially those related to Natural Language Processing (NLP) and Knowledge Management (KM), can be seen as natural tools that will help to better identify and extract the meanings (word senses or concepts) conveyed in the document.

Several simple NLP techniques have been explored in IR including term extraction (tokenization), word stemming, compound phrase identification, part of speech tagging (POS), chunking, word sense disambiguation and named entity recognition. All of these techniques, extensively discussed in chapter "Artificial Intelligence and Natural Language" of this volume, are somehow useful at different extents in IR (Manning et al. 2008) and help to better extract different forms of term units, including single words, phrases, word senses, topics, etc. (Li and Xu 2014). Without being exhaustive, stemming algorithms (Porter 1980; Krovetz 1993) are clearly the most used "NLP" technique in IR. They have relatively low-cost processing and often bring slight improvements in document retrieval (Harman 1991; Hollink et al. 2004). Part of speech tagging (e.g., verb, noun), have also been applied in IR for different purposes: POS-based term weighting (Lioma and Blanco 2009), disambiguation (Krovetz 1997). However, moderate improvements have been reported (Kraaij and Pohlmann 1996; Chowdhury and McCabe 1998; Lioma and Blanco 2009).

We will focus, in this section, on the two classes of approaches that have been widely investigated to cope with the term-mismatch issue, namely, compound term (phrase) indexing and concept based representation. Phrase indexing consists in indexing multiword units instead of single words. Concept (Semantic) indexing attempts to represent terms according to their meaning that might be taken from semantic resources such as thesaurus, ontology, knowledge base, etc., or derived from text corpus such as word embedding approaches (Deerwester et al. 1990; Mikolov et al. 2013).

4.1 Phrase-Based Indexing

Phrase indexing consists in representing index units by multiword units. These units can be addressed according to two classes of approaches: linguistic and statistic approaches. Linguistic-based approaches employ pure NLP techniques, including lexical, syntactic and semantic analysis and discourse processing, in order to extract meaningful phrases (i.e., phrases with certain syntactic relations Tong et al. 1997).

Several approaches, based on different linguistic clues, have been proposed and developed in IR. Such approaches include linguistic phrases (Fagan 1987a; Evans and Zhai 1996), lexical atoms (Sheridan and Smeaton 1992; Tong et al. 1997), head-modifier pairs (Strzalkowski 1995; Zhai 1997). Most of the results that have been reported showed no clear significant improvements of the retrieval performance (Fagan 1987a; Lewis 1992).

Statistical approaches are the most widespread, they mainly rely on word collocation to determine the weight of word relationships. The current IR approaches based on such representation investigate different types of collocation-based on pure statistical clues such as term proximity (Tao and Zhai 2007; Zhao and Yun 2009) and adjacent terms (inseparability) (Metzler and Croft 2005; Shi and Nie 2009). Other techniques combining linguistic and statistic approaches (Fagan 1987b; Hammache et al. 2014) have also been proposed. However, the impact of phrase-based indexing in terms of performances is quite limited. A combination with single words is often required (Hammache et al. 2014; Shi and Nie 2009). Furthermore, it has been shown that positional approaches that capture term dependency without explicitly extracting phrases are much more effective (Lv and Zhai 2009).

The conclusions that can be drawn from the reported results of phrase-based indexing are that pure NLP techniques have a limited impact on search, as statistical approaches are capable to effectively handle terms proximity without sophisticated linguistic analysis.

4.2 Semantic-Based Representation

Semantic based indexing consists in representing documents and queries according to the meanings conveyed by their terms. These meanings are obtained through external resources such as ontologies (WordNet,[1] YAGO, …), controlled vocabularies (Mesh,[2] …) or Knowledge Bases (Wikipedia, Freebase, …) (see chapter "Semantic Web" of this volume and chapter "Knowledge Engineering" of volume 1 for more details). Representing documents by means of the meaning of words in IR is also referred to as concept-based indexing. The two notions of semantic and concept-based indexing, are often mixed up; although both are based on external resources, semantic indexing uses linguistic resources, called also "light" ontologies, while con-

[1] https://wordnet.princeton.edu/.

[2] http://mesh.inserm.fr/FrenchMesh/.

cept indexing is based on formal ontological taxonomies. But, the two approaches share the same purpose and intend to represent documents and queries as a set of individual entries taken from resources. For instance, in case of WordNet, semantic indexing consists in representing a document as set of synsets (synonyms sets). A more sophisticated representation based on sub-trees extracted from WordNet has been also proposed in Baziz et al. (2005). Such representations enable word sense disambiguation (Sanderson 2000), where words are represented by their meaning, and allow retrieving documents with words that are semantically related to those of the query. The literature abounds on this topic (Krovetz and Croft 1992; Voorhees 1993, 1994; Sanderson 1994, 2000; Gonzalo et al. 1998; Moon et al. 2004; Stokoe et al. 2003; Liu et al. 2004, 2005; Fang 2008; Cao et al. 2005), and relevant surveys can be found in Sanderson (2000), Li and Xu (2014).

The results reported for such representations differ. Indeed, Sanderson (1994) and Voorhees (1994) showed that there is no significant improvement in the search performance. The work presented in Schütze and Pedersen (1995) is one of the first works showing improvements on a large collection. Other improvements have been reported in Gonzalo et al. (2014), Mihalcea and Mihalcea (2000), Baziz et al. (2005), Dinh et al. (2013), Zakos (2005), where it was noted that these representations are particularly effective in a domain-specific search environment (Li and Xu 2014) such as the medical domain (Wang and Akella 2015).

What can be noticed from most of the semantic-based approaches listed above, is that the presence of AI techniques is limited. In these works, an ontology was addressed from a linguistic perspective, neither reasoning nor inference processes are employed. The notion of inference with ontologies is rather developed in the context of the Semantic Web (see chapter "Semantic Web" of this volume), and for this reason this topic is not covered in this chapter.

Extensions based on fuzzy ontologies, where relationships between concepts are weighted, have been proposed (Miyamoto 1990). These weights indicate the relative strength of these relationships. Possibilistic ontologies have also been explored in Baziz et al. (2007), Boughanem et al. (2007). The links between concepts are estimated by two degrees, possibility and necessity (see chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1). Two types of relations have been considered in the above works, synonymy and hypernymy. The necessity degree estimates to what extent it is certain that one concept is a specialization of the other. Possibility indicates to what extent two concepts can describe the same thing. Experiments have been conducted on small collections, and moderate improvements have been reported.

To sum up, although ontology is used in most of the approaches listed above, these approaches are not employing AI techniques for reasoning and inferring new knowledge. Works relying on AI for document representation and reasoning, mainly provide formal document representations derived from logic. To this purpose, different frameworks have been used such as case frame-based representations (Mauldin 1991), rule-based systems (Vickery and Brooks 1987), logic-based representation (e.g. propositional, predicates, modal) (Fuhr 1995; Van Rijsbergen 1986; Nie 1988;

Meghini et al. 1993), and conceptual graphs (Chevallet and Chiaramella 1998). These models will be deeply discussed in the retrieval models section of this chapter.

4.3 Word Embedding Representation

The deep learning (neural networks) wave has also intervened within the NLP domain, one of the success stories for document representation is word embedding representations, such as word2vec (Mikolov et al. 2013; Pennington et al. 2014). Word embedding approaches consist in representing terms, or more generally phrases, sentences or paragraphs, according to the contexts where they appear. They all share the idea popularized by Firth (1957): "*You shall know a word by the company it keeps*". This leads to represent each term as a vector of attributes (real numbers) that captures precise syntactic and semantic word relationships.

Two classes of approaches have been used to build such representations (Onal et al. 2017), context-counting based on algebraic methods such as singular value decomposition (Deerwester et al. 1990) and context-predicting based on neural methods (Mikolov et al. 2013). These latter attempt to learn word embeddings from the raw text. One of the first neural based approaches date back to the 2000's (Bengio et al. 2003). In 2013, Mikolov et al. (2013), introduced word2vec within two different folds: the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. These models are quite similar, except that CBOW predicts target words from source context words, while the skip-gram does the inverse and predicts source context-words from the target words. Pennington et al. released GloVe (2014). Most of these approaches are based on feed-forward neural networks (Bebis and Georgiopoulos 1994). Other neural models have been employed, including convolutional neural network (Huang et al. 2013; Mitra et al. 2016; Shen et al. 2014) and recurrent neural network (Kiros et al. 2015; Wan et al. 2016).

The success of neural embeddings in NLP is mainly related to the unsupervised nature of the learning. None annotated data is needed, these representations can be learned from any collection of texts. Pre-trained vectors are available, to mention, the one provided by Google. These vectors are partially trained on part of Google News dataset (about 100 billion words). It is composed of 3 millions of words and phrases represented on 300-dimensional vectors.

From IR point of view, the word vectors that are used during a search may be obtained from any pre-trained word vectors or can be derived from the same collection where the search is performed. They have been widely applied for document-query matching or for query expansion. In general, the query and the documents are either represented as bag of word vectors or as an aggregated vector. Aggregation can be obtained through different operators such as sum or average (Vulić and Moens 2015; Mitra et al. 2016; Nalisnick et al. 2016; Le and Mikolov 2014), Non-linear combinations using Fisher Kernel (Clinchant and Gaussier 2010), k-means clustering (Ganguly et al. 2015) and maximum likelihood estimation (Zamani and Croft 2016b). The query-document relevance score is computed either by comparing query and

document word vectors, aggregated or not, using a variety of similarity metrics such as cosine or dot-product (Mitra et al. 2016; Nalisnick et al. 2016). An alternative to computing the relevance score is to incorporate the term representations into existing IR models such as Language model (Zuccon et al. 2015; Ganguly et al. 2015; Zamani and Croft 2016a; Ai et al. 2016) or BM25 (Kenter and De Rijke 2015; Rekabsaz 2016).

Word Embeddings have been also employed for query expansion. The basic approach consists in comparing query term with term embeddings of the whole collection or of the top retrieved document to find expansion candidates (Diaz et al. 2016; Roy et al. 2016; Zamani and Croft 2016a; Zheng and Callan 2015).

5 Information Need Representation

Query formulation is a crucial phase in the IR process: this is a subjective process that should be tolerant to the uncertainty that intrinsically characterizes the identification and the expression of an information need. As it has been widely advocated in the literature, IR is an interactive process by which the user aims at locating the documents useful to fulfill the needs behind his/her request. Despite the developments underlying the technologies for managing and accessing information, state of the art and commercial search engines are still mainly based on keyword-based query formulation, which seldom makes use of knowledge resources to face the problem of words' disambiguation.

The complexity of natural languages, with their nuances and their subjective usage is still far from being effectively captured by computer applications. Moreover the intended semantics of the few keywords specified in a user query should be disambiguated depending on both the user and the query context.

To cope with uncertainty a possibility is to allow the user to imprecisely or vaguely represent his/her information needs. In this context the application of Fuzzy Set Theory has been finalized at modelling a tolerance to uncertainty in query formulation, by means of the definition of flexible query languages. In particular, flexible query languages have been defined as generalizations of the Boolean query language. Two main kinds of generalizations have been proposed: (1) to associate numeric or linguistic weights to query terms; (2) to introduce linguistic quantifiers to aggregate (weighted) query terms.

A query term weight expresses the importance of a term as descriptor of the users needs, and it is formally defined as a flexible constraint on the index term weights. By this fuzzy extension, the structure of a Boolean query is maintained, by allowing weighted query terms to be aggregated by the AND, OR connectives and negated by the NOT operator. In this way, the exact matching of the Boolean model is relaxed to a partial matching; in fact, the query evaluation mechanism applies a fuzzy decision process that evaluates the degree of satisfaction of the query constraints by each document representation, by applying a partial matching function. In the context of Fuzzy Set Theory, the connectives AND and OR are defined as aggregation operators

belonging to the classes of T-norms and T-conorms respectively. Usually, the AND is defined as the min aggregation operator, and the OR as the max aggregation operator.

The first fuzzy models proposed the definition of numeric query term weights, in the range [0, 1]. The flexible constraint identified by a query term weight depends on its semantics; in the literature different semantics have been proposed, which have introduced distinct fuzzy generalizations of the Boolean model (Yager 1988; Kraft and Buell 1983; Bordogna et al. 1991; Kraft et al. 1999; Bordogna and Pasi 2001; Boughanem et al. 2007).

The three main semantics that have been proposed for query term weights are: the relative importance semantics (query weights express the relative importance of pairs of terms in a query), the threshold semantics (a query weight expresses a threshold on index term weights), and the ideal index term weight semantics (a query weight expresses the perfect index term weight). The choice amongst the three proposed query weight semantics implies a distinct modeling of the retrieval function evaluating a query against documents representations.

To overcome the problem of imposing to the user the unnatural choice of a numeric value, thus forcing her/him to quantify a qualitative concept of importance, some recent models proposed in the literature have introduced linguistic query weights, based on the concept of linguistic variable (Bordogna and Pasi 1993; Kraft et al. 1999). By this linguistic extension of the Boolean query language, query terms are expressed by means of words such as important, and very important. Besides, linguistic query term weights express flexible constraints on the index term weights. As previously outlined, a second generalization of the Boolean query language has concerned the definition of linguistic quantifiers as aggregation operators. This proposal has come to improve query formulation by going beyond the usage of the AND and OR connectives (Bordogna and Pasi 2005). In fact, when the AND is used for aggregating the keywords specified in a user query, a document indexed by all keywords but one is not retrieved, thus causing the possible rejection of useful items. The opposite behavior characterizes the aggregation by OR. The use of linguistic quantifiers (formally defined within Fuzzy Set Theory) was proposed to allow more expressive and more natural query formulations. Linguistic quantifiers, such as at least 2 and most, specify in fact more flexible selection strategies. Linguistic quantifiers have been formally defined as averaging aggregation operators, the behavior of which lies between the behavior of the AND and the OR connectives, which correspond to the *all* and the *at least one* linguistic quantifiers. By adopting linguistic quantifiers, the requirements of a complex Boolean query can be more easily and intuitively formulated. For example, when desiring that *at least 2* out of three selection conditions a, b, c be satisfied, one should formulate the following Boolean query:
(a AND b) OR (a AND c) OR (b AND c)
which can be replaced by a simpler one: *at least 2(a, b, c)*.

In Bordogna and Pasi (1995), a generalization of the Boolean query language that allows one to personalize search in structured documents, was proposed; both content-based selection constraint, and soft constraints on the document structure can be expressed. The atomic component of the query (basic selection criterion) is defined as follows: t in Q preferred sections, in which t is a search term expressing a

content-based selection constraint, and Q is a linguistic quantifier such as all, most, or at least k. Q expresses a part of the structure-based selection constraint. It is assumed that the quantification refers to the sections that are semantically meaningful to the user. Q is used to aggregate the significance degrees of t in the desired sections.

6 Retrieval Models: Relevance Modelling

Relevance is the most important notion in IR and one of the fundamental issues is to define the formal and the theoretical frameworks allowing the interpretation of this notion. The majority of the IR models consider relevance as a matching problem between query and document characteristics, often represented as a set of weighted terms (phrases). Probabilistic models including BM25 (Robertson and Walker 1994), language models (Ponte and Croft 1998; Lavrenko and Croft 2017; Zhai 2008), information theory-based models (Amati and Van Rijsbergen 2002; Clinchant and Gaussier 2010), and algebraic models such as vector space model (Salton et al. 1975), are currently the most widespread and most performing models.

However, there are other theoretical frameworks, more related to AI, that have been used to interpret the notion of relevance. This includes, logic (Propositions, Modal, Description, …) (Van Rijsbergen 1986; Crestani et al. 2003; Abdulahhad 2014), fuzzy logic (Damiani et al. 2007; Boughanem et al. 2009, 2007), inferential and beliefs models (Turtle and Croft 2017; Silva et al. 2000), and optimization methods such as evolutionary computation [genetic algorithms (Kim and Zhang 2003; Vrajitoru 2013), swarm intelligence (Kennedy and Eberhart 1995)], game theory (Raifer et al. 2017; Zhai 2016), multi-agent systems (Enembreck et al. 2004; Trifa et al. 2017).

Retrieval models were also addressed by Machine Learning techniques. The first work tackling learning to rank dates back to 2000. Since 2013, the Neural networks trend have also been inspiring the IR tasks. The surprising results, obtained in vision and image retrieval, gave real opportunities to the document retrieval communities. We will list, in the following, some IR approaches based on these theoretical frameworks.

6.1 Logic-Based Models

These models assume that the retrieval process has an inferential nature. For example, the direct term-based comparison, between a document d discussing "violin" and a user query q entailing "fiddle", will lead to a mismatch. However, based on the knowledge that "violin" and "fiddle" are synonymous, it is possible to infer that d is a possible answer to q. Therefore, using only classical (bag of words) IR models, is not able to solve such issue. On the other hand, using formal logics, which are basically inference systems and well adapted tools for knowledge representation, to

model the retrieval process, supposes to make it more intelligent (i.e. closer to the way how a human-being expert decides about relevance).

The use of logic in IR dates back to Cooper (1971), where the relevance is seen as an inference process between a document d and a query q. The retrieval consists of finding the documents that imply the query, denoted $d \rightarrow q$ where d and q are normally logical sentences in the underlying logic. In the literature, logic-based IR models adopted six stands (interpretations) of the $\rightarrow$ operator between d and q (Sebastiani 1998). This strict view of inference allows only binary decisions, where most formal logics allow only True/False decisions. Thus, Van Rijsbergen (1986; 1989) proposed the notion of Logical Uncertainty Principle (LUP) which allows a nuanced formulation of the implication by associating a degree of uncertainty to it, denoted $U(d \rightarrow q)$ (see also chapter "Constraint Reasoning" of Volume 2 for more details).

The main issue that has been addressed in this line of research, is to define the theoretical framework for translating the queries, the documents, the implication and the uncertainty U. It is worth mentioning, in this context, that most logic-based IR models differentiate between the two tightly related notions, namely matching (represented via $\rightarrow$) and ranking (represented via U). This distinguishing allows for a finer grained analysis.

Several frameworks have been proposed and adapted to IR, namely modal logic, description logic, conceptual graphs, etc.. In the same way, uncertainty has been considered in different forms, including fuzzy logic, probability theory, logical imaging, belief revision, etc.(see chapters "Knowledge Representation: Modalities, Conditionals and Nonmonotonic Reasoning", "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1 and chapter "Automated Deduction" of Volume 2 for details about some of these logics). We list in the following some logic-based IR models. This part is largely inspired by Crestani et al. (2003), Abdulahhad (2014), Lalmas (1998), which provide much more details than those listed below. We present the models according to the formal logic, that is used to represent the different components of the implication, and also according to the mathematical theory that is used to estimate uncertainty.

Propositional logic: Many IR models use propositional logic as a logical framework to represent the retrieval process. In Losada and Barreiro (2001, 2003), both d and q are logical sentences, and the IR implication $d \rightarrow q$ is the logical consequence $d \models q$.[3] The uncertainty is estimated using Belief Revision. They particularly used Dalal's operator for document ranking. Abdulahhad et al. (2017) use opted for the same choices to model d, q, and $d \rightarrow q$. However, instead of using Belief Revision, they make use of the lattice structure that can be constructed between logical sentences in order to have a probabilistic estimation for U.

Modal logic: Modal logic extends classical propositional and predicate logic to include modality operators, namely *necessity* and *possibility*. In this context, two mathematical frameworks, namely Kripke's Possible Worlds (PW) semantics (Kripke

[3] $\models$ is a meta-language symbol, where $s_1 \models s_2$ means that in any interpretation if s_1 is true then s_2 is also true.

1963) and Logical Imaging, have been used to build IR models. PW assumes that Worlds (formal interpretations) are connected through accessibility relationships. The two modalities for a logical sentence s refer to the possibility to reach a possible world where s is true starting from the current possible world and following the accessibility relations between worlds. Logical Imaging evaluates the process of moving probabilities from the worlds where a given sentence is false to the most similar worlds where it is true. Propositional modal logic was used by Nie (1988, 1989), where documents are possible worlds or interpretations and queries are logical sentences. According to Nie, a document d is relevant to a query q iff q is true in d, or in a world accessible from d. Therefore, uncertainty is seen as the cost of the path that is needed to move from the original document d, where q is not true, to a document d', where d' is accessible from d and q is true in it. However, Crestani et al. (Crestani and van Rijsbergen (1995), Crestani (1998)) assume that each term is a possible world and both documents and queries are logical sentences. A document (resp. query) is true in a given term (world) t iff t appears in that document (resp. query). Logical Imaging is then used to rank documents, where terms' scores are first relocated from the terms that do not appear in the document to the most similar terms inside the document according to the accessibility relations, then the relevance value $U(d \rightarrow q)$ is estimated based only on the terms that appear in d. Other extensions that evaluate the accessibility between two possible worlds have also been proposed in a fuzzy framework (Nie et al. 1995).

Conceptual graph: Conceptual Graph formalism of Sowa (1983) has been used in IR (Chiaramella and Chevallet 1992; Chevallet and Chiaramella 1998; Amati and Ounis 2000). It is a graphical formalism that is equivalent to first-order logic. In this IR model documents and queries are represented by conceptual graphs (i.e. logical sentences), the retrieval decision is carried out by conceptual graph operations to establish a projection (i.e. material implication) between d and q. The uncertainty is the cost of these operations.

Description/Terminological logic: Description Logic (DL) is a family of languages to represent knowledge. It is widely used in Semantic Web. It is more expressive than propositional logic and less than first-order logic, but it has more efficient reasoning than first-order logic. In Meghini et al. (1993), the query is a concept and the document could be a concept or an individual. If the document is an individual, then the retrieval decision is to check if the individual d is an instance of the concept q. Otherwise, the retrieval decision is to check if the concept d is subsumed by the concept q. In Meghini et al., the relevance is binary. It has been then extended in Sebastiani (1994), Meghini and Straccia (1996) to include probabilities to fit the LUP of Van Rijsbergen (1986; 1989). Other extensions have been proposed to estimate uncertainty using the notion of possibility (Qi and Pan 2008).

Probabilistic Datalog: Datalog is a predicate logic that has been developed in the database field. Probabilistic Datalog is an extension of Datalog using probability. More precisely, predicates are associated with probabilities, denoted αg where g is a classical predicate and α is the probability that g is true. Probabilistic Datalog has been used in IR (Fuhr 1995; Rölleke and Fuhr 1996), where documents are represented as a set of probabilistic predicates of the form $\alpha term(t, d)$ that expressing

the document d is indexed by the term t, and α indicates the probability that d is about t. Queries are written as Boolean expressions, and the retrieval decision is seen as an inference rule.

Probabilistic Argumentation Systems: is a logical framework that extends propositional logic by a probabilistic mechanism to express uncertainty. It is able to express both the qualitative and quantitative uncertainty. Picard (1999), Picard and Savoy (2000) have proposed an IR model based on such logic. Documents and queries are represented as a set of weighted rules indicating document-term aboutness, inter-term, and inter-document relations, where weights indicate the strength of the implications. The relevance is seen as the degree to which the document supports the query.

Others: Other families of formal logics have been used to model IR process. Situation theory was adopted by Lalmas and van Rijsbergen (1993), and Huibers (1994) to build an IR model where the document d is a situation and the query q is an infon or a set of infons. An infon is an atomic information carrier, and it refers to the information that a particular relation holds / does not hold between a set of objects. Accordingly, d is relevant to q iff d supports q. Abductive reasoning (Thiel and Müller 1996) and default logic (Hunter 1997) are also used. This later is used to represent semantic relations between objects, e.g. synonymy, polysemy, etc.

Although formal logics make the retrieval process more intelligent, where formal logics are powerful inference and knowledge representation systems, the use of formal logics to model the IR process is not cost-less. Most logic-based IR models are too complex to have operational instances of them. However, some recent studies were able to build operational logic-based IR systems (Abdulahhad et al. 2017; Zuccon et al. 2009; Losada and Barreiro 2003).

6.2 *Fuzzy Models*

Fuzzy Set Theory has been applied to IR since the 70s, to the aim of modeling both the vagueness/uncertainty in the formulation of an information need and the subjectivity of the notion of relevance. Fuzzy sets were initially applied to information retrieval as a means to generalize the Boolean retrieval model (Bordogna and Pasi 1995; Miyamoto 1990; Buell 1985; Bookstein 1980). As it was outlined in Sect. 5, an outcome of the proposed generalizations was to enable flexible query formulation, by allowing the specification of both numeric and linguistic query term weights, interpreted as constraints on the document representation formally expressed as a fuzzy subset of index terms (Fox and Sharan 1986; Molinari and Pasi 1996; Herrera-Viedma 2001).

The first fuzzy generalization the the Boolean IR model has consisted in simply extending the document representation, by maintaining the Boolean query language. By representing a document as a fuzzy subset of index terms, instead of a classical set, index term weights can be considered (e.g. normalized $tf * idf$ weights), and the Boolean query evaluation mechanism can produce an RSV (relevance score) for each

document, thus allowing a ranking of the proposed results. One of the first models proposing this extension is the MMM (Min, Max, and Mixed) model introduced in Fox and Sharan (1986). The adaptation is quite simple, the document is seen as a fuzzy subset of the index terms in the collection (dictionary), where the term weight represents the degree of membership of a term in a document. Then, the evaluation of a Boolean query relies on the interpretation of the AND and OR connectives as conjunctive aggregation operators, generally the min and max operators respectively.

Subsequent generalizations of the Boolean model have been proposed, to the aim of also extending the Boolean query language (beside generalizing the formal document representation). As mentioned in Sect. 5 two kinds of extensions have been defined: (1) the association of (numeric or linguistic) weights to the query terms, (2) the generalization of the AND, OR connectives (by means of linguistic quantifiers, as shown in Sect. 5). In particular, different interpretations of query terms weights have given origin to distinct generalizations of the Boolean retrieval model. As shortly introduced in Sect. 4, the three semantics associated with query term weights are: relative importance (the weights express the relative importance among terms), threshold (the query term weight expresses a threshold constraint on the index term weights) and the ideal index term weight semantics (Yager 1988; Sanchez 1989; Kraft and Buell 1983; Bordogna et al. 1992; Boughanem et al. 2007; Baziz et al. 2006).

As outlined in Sect. 5, to help users in qualifying the importance of query terms as descriptors of their needs, the numeric query weights have been generalized to linguistic query weight, by maintaining their semantics. Formally, these weights are defined as values of the linguistic variable Importance (e.g., *important*, *very important* etc.), which still specify constraints on the index term weights (Bordogna and Pasi 1993; Kraft et al. 1999).

The other aspect related to the extension of the Boolean query language concerns the connectives employed to aggregate the different search criteria, i.e. query terms. Basically, in the Boolean query evaluation process, aggregation consists in evaluating a document on each term separately, and then aggregating the according to the Boolean structure of the query. When generalizing the document representation to a fuzzy document representation, the aggregation process must account for index term weights (or for scores expressing the satisfaction of the constraint imposed by a query term weight, in the case of generalized Boolean queries). The AND and OR connectives are associated with conjunctive and disjunctive aggregation operators respectively, such as t-norms for AND and t-conorms for OR (Yager 1988; Dubois and Prade 1985; De Baets and Fodor 1997). Linguistic variants of aggregation operators enabling to relax AND (all) and OR (at least 1), such as *most* or *at least k* have also been proposed and used in IR (Hayashi et al. 1992; Sanchez 1989). To this purpose ordered weighted aggregation (OWA) operators have been introduced (Yager 1988; Bordogna and Pasi 1995; Marrara et al. 2017).

There exist two noticeable refinements of the MIN operation,called *discrimin* and *leximin* (Dubois et al. 1997). They allow to distinguish between values to be aggregated having the same minimal value. These operators have been applied to

IR in Boughanem et al. (2007), Baziz et al. (2006). An interesting survey about aggregation in IR can be found in Marrara et al. (2017).

Other approaches based on Possibility Theory (See chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1) have been defined. In particular, Loiseau et al. (2004) used fuzzy pattern matching (Dubois et al. 1988), to formulate and evaluate flexible queries on documents represented by fuzzy sets. In order to estimate the relevance of a document to a query, also called compatibility, the possibility and the necessity measures were used. The possibility metric estimates to what extent it is possible that a query q and a document d refer to the same value (terms). It represents the intersection of the fuzzy set of values compatible with q with the fuzzy set of possible values of d. The necessity metric measures to what extent it is certain that the value corresponding to d is compatible with q. It is computed as an inclusion degree of the possible values for d into the set q of values compatible with the query terms The compatibility is evaluated by means of a possibilistic ontology that allow to compare the compatibility of terms even if they are not similar.

Other approaches have been defined within the framework of Possibility Theory (Boughanem et al. 2009, 2007). Several surveys on fuzzy IR can be found in Kraft et al. (1999), Tamir et al. (2015), Kraft et al. (2015), Pasi (2009), Kraft and Colvin (2017), Marrara et al. (2017).

6.3 Bayesian Networks

Bayesian inference networks provide a probabilistic formalism for describing inference relations with uncertainty (See chapter "Belief Graphical Models for Uncertainty Representation and Reasoning" of Volume 2). Several IR models have been proposed (Turtle and Croft 2017; Ribeiro and Muntz 1996; Silva et al. 2000; De Campos et al. 2002; Lee et al. 2009) where nodes represent either documents, terms or queries variables. The links indicate the causality between nodes, the relevance is related to the probability of logically inferring the query from document representations or conversely (Van Rijsbergen 1986). The two most known models are the inference networks model of Turtle and Croft (2017) and the Belief model of Ribeiro-Neto and Muntz (Ribeiro and Muntz 1996; Silva et al. 2000). In these models, document, index terms, and query are represented by Boolean variables and the relevance is seen either as the inference of the query from the documents (Turtle and Croft 2017), or the deduction of relevant documents given a query (Ribeiro and Muntz 1996). Thus, Belief networks can generalize Boolean, vector space, probabilistic and inference models. Other extensions based on Bayesian networks have been proposed either for optimizing the computation of conditional probabilities (Bruza and van der Gaag 1994; Indrawan et al. 1996; Fung and Del Favero 1995), or integrating dependence between term pairs (De Campos et al. 2003; Crestani et al. 2003) or document pairs (De Campos et al. 2002), or dealing with heterogeneous

documents (Crestani et al. 2003; Denoyer and Gallinari 2003), or with tweet search (Jabeur et al. 2012).

Possibilistic framework has also been applied in IR to better characterize relevance. These models also use possibility and necessity, as relevance metrics instead of a unique probability metric. This allows one to better reflect the subjectivity of the actual relevance. Such model has been proposed in Boughanem et al. (2009), it is inspired by the Turtle model (Turtle and Croft 2017), but it employs possibilities instead of probabilities. The relevance of a document given a query is measured by two degrees: the necessity and the possibility. The possibility degree is convenient to filter documents out from retrieved documents and the necessity degree is useful for document relevance confirmation.

6.4 *Machine Learning Based Models: Learning To Rank, Deep Learning*

Although the theoretical frameworks underlying the traditional IR models differ, they all combine the same relevance signals, such as tf (term frequency), idf (inverse document frequency), document length. However, if the number of signals increases to reach hundreds of signals, which is actually the case for search engines, these models fail and are unable to process such amount of signals. Machine learning techniques provide a way to handle such issues, although on their side they require annotated data, which are often not available.

The use of machine learning, particularly neural networks, dates back to the 1990's. Thee early works (Belew 1987; Kwok 1989; Boughanem SDC 1992) are based on spreading activation networks, often composed of two layers. The search is carried out by propagating forward the entry (the query) from the term layer into the document layer, with eventually, one more step back to the term layer to facilitate learning. The first models that were built with effective ability to learn from hundreds of features and combine them, date back to the 2000s. They are known as, Learning to rank (LTR) approaches, their goal is to learn the ranking function over a set of hand-crafted features composed of tens or even hundreds of characteristics, extracted from documents and/or queries (Liu 2009; Li 2011). Such features include tf, idf, BM25 scores, occurrence of query term in document title, in anchor text, document length, PageRank, number of unique words, document trust, etc. The learned model is then used in the testing phase.

Several machine learning models including support vector machines (Herbrich 2000; Nallapati 2004; Yue et al. 2007), neural networks (Burges et al. 2005; Tsai et al. 2007), and boosting (Wu et al. 2010), were developed to support IR tasks (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2). The main issues, that arise in LTR, include training data creation, feature construction and the machine training model. The use of a particular model depends

on the size and the type of the training data, and on the training objective (the type of the desired output). Liu (2009) categorized three types of objectives:

- Pointwise where the output is a query-document relevance, which can be represented as a degree of relevance, a binary relevance (relevant vs. irrelevant) or multiple ordered categories: Perfect $\succ$ Excellent $\succ$ Good $\succ$ Fair $\succ$ Bad)(Crammer and Singer 2002; Nallapati 2004; Shashua and Levin 2003; Li et al. 2008).
- Pairwise is based on document preference, document d_1 is more relevant than document d_2. (Burges et al. 2005; Freund et al. 2003; Wu et al. 2010),
- Listwise is based on a list of documents ranked according to their relevance. Its main objective is to optimize the ranking metrics (MAP, NDCG) (Yue et al. 2007; Tsai et al. 2007; Xia et al. 2008).

Unlike LTR models that require hand-crafted features, deep learning approaches have been used to automatically learn the useful features to model relevance. This class of IR neural models is called *Interaction focused models* as it attempts to extract salient features from the interaction between a query and a document (or a set of documents).

Neural architectures that have been used to handle this task include MultiLayer Perceptron (MLP), convolutional neural networks (CNN) and recurrent models.

Generally, MLP are used to enable non linear combination of inputs (entries). They help aggregate different input word vectors into a single representation vector (Le and Mikolov 2014), and map a sparse vector to a low dimensional representation vector (Huang et al. 2013). CNN networks are used to learn representation vectors from raw text through a sequence of convolutional and pooling layers (Shen et al. 2014; Hu et al. 2014; Mitra et al. 2017). CNN defines a set of linear filters (layers) able to extract features by detecting regularities of inputs having spatial constraints such as images and texts. Convolutional layers are typically followed by pooling layers that perform aggregation. Recurrent neural models are also widely used in IR for their ability to represent sequential inputs, such as continuous word sequence (Kiros et al. 2015; Wan et al. 2016), and their memorization aspect, as they allow one to remember the different information present in the input data while processing them.

Most of the proposed models in literature start with using a NN model, usually CNN or RNN, to extract the salient features which are then given as input to an MLP network to be aggregated or to learn relevance. Typically, these models operate according to the network input which might be of different forms. It can take the form of term vectors put forth by word embedding approaches (refer to Sect. 4.3), or interaction matrix generated by comparing windows of text from the query and the document. The terms within each window can be represented as one-hot vector (Jozefowicz et al. 2016; Kim et al. 2015; Huang et al. 2013) or as word embeddings (Hu et al. 2014).

The approaches using interaction matrix have been addressed for short text matching (Lu and Li 2013; Yin and Schütze 2015; Pang et al. 2016) and for long documents ranking as well (Pang et al. 2016; Mitra et al. 2017). The Deep Structured Semantic Models presented in Huang et al. (2013) were the first to introduce a NN

based approach for ad-hoc retrieval. The proposed models were trained by maximizing the conditional likelihood of the clicked documents given a query by using click through data. Shen et al. (2014) proposed the Convolutional Deep Structured Semantic Models, C-DSSM as an extension to the DSSM for documents/query matching by combining CNN with max-pooling. However, this kind of models fails when dealing with rare terms and search intents. Indeed, a good neural IR model should incorporate both lexical and semantic matching signals (Mitra et al. 2017).

Lu and Li (2013) developed a deep matching method called DeepMatch that allows one to model the matching between two objects from heterogeneous domains. The proposed model was applied in two tasks: finding relevant answers for a given question and matching tweets with comments. Likewise, Guo et al. (2016) proposed a deep relevance matching model (DRMM) for ad-hoc retrieval that employs three relevance matching factors: Exact matching signals, Query terms importance and Diverse matching requirement. Their model is based on the interaction-focused models and uses a joint deep architecture at the query term level for relevance matching. Recently, Zamani et al. (2018) explored how neural models addressed ranking documents with multiple document fields. The proposed model handles short text fields like document's title and long text fields like document's body. They found that it is more effective to learn separate embedding spaces to match the different document's fields against the query rather than opting for a common embedding space. This can be explained by the fact that the document's fields can correspond to different aspects of the query and thus, it would be better to consider comparing with separate representations of the query text.

Despite of the major improvements achieved by neural models operating with supervised data, one of the main challenges is to learn how to handle IR tasks with weak supervised or unsupervised data. There are some recent works that attempted to address this issue. Dehghani et al. (2017) proposed a "Pseudo-Labeling" approach for query-dependent ranking that creates its own training data set employing existing unsupervised methods. The weak supervised signals generated are then used to train a neural retrieval model. MacAvaney et al. (2017) presented an approach that generates weak supervision training data for neural IR models and considers negative training examples. The proposed approach is applied on a news corpus where article headlines are extracted as pseudo-queries and articles' content as pseudo-documents. The human relevance judgments are replaced by a similarity metric that measures the interactions between the pseudo-queries and the pseudo-documents.

Applying deep learning to IR tasks is currently one of the hottest topic in information retrieval field. There are more than fifty papers that have been published in high venue conferences and journals. Several interesting surveys have been published (Onal et al. 2017; Mitra and Craswell 2017).

6.5 Evolutionary Computation

Several IR systems turned to using evolutionary algorithms in order to improve the search performance and to reduce the time required to answer complex queries.

Evolutionary algorithms like Genetic algorithms (GAs) (Holland 1992), Ant Colony (Colorini et al. 1991), Artificial Bee colony (Karaboga 2005) and Particle Swarm Optimization (Kennedy and Eberhart 1995) are bio-inspired methods that have been proposed as a way of finding optimal solutions for complex problems in a much shorter time compared to the time required by evaluating all possible solutions. These algorithms are extensively discussed in chapter "Meta-Heuristics and Artificial Intelligence" on Volume 2.

6.5.1 Evolutionary Algorithms : Genetic Algorithm

Genetic algorithms (GAs), initially introduced by Holland (1992), are stochastic optimization algorithms inspired from natural selection and genetics mechanisms. They start with a population of potential solutions randomly chosen. Based on their relative fitness (performances), a new population of potential solutions is created using simple evolutionary operators: *selection*, *crossing* and *mutation*. This process is repeated until we reach a "satisfactory" solution.

Genetic algorithms were proposed by several works (Vrajitoru 2013; Chen 1995; Yang et al. 1993; Gordon 1991) as a solution to IR issues, like document indexing and query reformulation. Analogically, the search space of a genetic algorithm, within an IR system, involves a set of documents' descriptors which are composed of the terms belonging to each document. The *genes* are considered as the terms' weights and an *individual* is represented as the concatenation of all the document's descriptors. Thus, the main purpose of the genetic algorithm in this context, is to create at least one new *individual* whose performance will be greater than that of its *parents*. The work presented in De Almeida et al. (2007) addressed ranking strategies, from a genetic programming perspective, that combine several term weighting functions and adapt to each document collection. The proposed approach proved to be effective as it outperforms the traditional weighting functions and improves the retrieval precision.

Tamine et al. (2003) made use of genetic algorithmic to develop a query reformulation (optimization) process involving the niching technique (Goldberg and Corruble 1994), which retrieves for the same query, relevant documents that have relatively dissimilar descriptors. In Kim and Zhang (2003), the authors proposed a genetic based mining method to determine the significant tags and their weights for document retrieval. Araujo and Pérez-Iglesias (2010) developed a query expansion approach using genetic algorithm with a fitness function based on the user's relevance judgments. They believe that using a genetic algorithm to select the terms maximizing the average precision, for each query, can enhance the retrieval process. Likewise, in Sathya and Simon (2010), the authors use GA to obtain the best terms' combination

from a set of keywords extracted by a document crawler. The output generated by the GA is then applied to an IR system.

The work presented in Al-Khateeb et al. (2017) has also used GA for query reformulation and expansion. Unlike traditional IR systems, instead of using a single query, the authors use Wordnet to extract synonyms of the query's keywords and thus put forward a population of queries generated from the original query. They consider that such approach allows to expand the search space.

6.5.2 Metaheuristic and Swarm Intelligence

Swarm Intelligence based Metaheuristics are founded on the collective and social behavior of some species like ants and bees, forming Swarm Intelligence algorithms i.e. Particle Swarm Optimization(PSO), Ant Colony Optimization(ACO) and Artificial Bee Colony (ABC). When dealing with IR issues, these algorithms proved their worth and have empirically demonstrated their effectiveness.

Particle Swarm Optimization is an evolutionary technique that uses a population of candidate solutions to develop an optimal solution to the problem. The members of the population, particles, are distributed randomly in the search space, having each a random velocity. In Bindal and Sanyal (2012), the authors proposed a PSO-based approach for query optimization that learns the query's terms significance using the documents contexts. It determines the optimal query vector that improves the IR system effectiveness. Therefore, a particle stands for a query vector and the fitness function is represented as the cosine similarity between a query and the top-k documents retrieved for the original query. An enhanced PSO algorithm was also introduced by Khennak and Drias (2017) for query expansion and aimed to determine the most suitable expanded query, rather than extracting the best expanded keywords (Sathya and Simon 2010). In order to overcome the huge number of the expanded query candidates, the authors turned to an accelerated version of the PSO algorithm called APSO which deals with this issue as a combinatorial optimization problem.

Ant Colony Optimization (ACO) algorithm (Colorini et al. 1991) is based on the behavior of ants seeking a path between their colony and a food source using pheromone trails. The original idea has since diversified to solve a broader class of problems and several algorithms have emerged, drawing on various aspects of ant behavior. Chawla (2013) proposed an ACO based approach for personalized Web search which considers the ant pheromone as *the information scent* and the set of users play the ant's role. The pages clicked by other users for a given query are used as the *the information scent*, i.e. pheromone, that helps to enrich the search space of a given user for the same query. ACO algorithm was also applied for Web page ranking (Chawla 2017) as it addresses finding the optimal ranking of clicked URLs from an optimization perspective.

The Artificial Bee Colony (ABC) algorithm (Karaboga 2005; Karaboga and Basturk 2008) is a population-based, naturalistic-inspired algorithm based on the bees foraging. In Abdullah and Hadi (2014), the authors put to the test an ACO based approach for Web IR and proved that such approach helps to cope with the huge

volume of information as it prunes the search space by exclusion and thus improves the query processing and the response time. Hassan and Hadi (2016) opted for ACO to address Word Sense Disambiguation (WSD) in IR, using the simplified lesk algorithm. They showed that an ACO based approach yields a very high response time and an accurate relevance compared to the traditional algorithms.

7 IR Approaches Based on Other AI Frameworks

IR issues were extensively addressed within other theoretic frameworks like Multi-agent systems, Game theory and Decision making, in order to provide a better search experience for the user. In the following, we put forth some of the approaches that were proposed to tackle IR tasks within these frameworks.

Multi-agent systems (MASs) are considered as an important alternative to the traditional IR models as they proved to yield better results by providing scalability and load balancing, by using agents for the different IR tasks (search, filter, rank,etc.). Enembreck et al. (2004) proposed to distribute the retrieving process over several agents: a Personal Agent that manages the user's favorite websites, a Library Agent for document indexing and a Filter Agent which retrieves the required information from the Search Agents and filter it based on the user's profile. They proved that such distributed process help improve the search performance. The work proposed in Trifa et al. (2017), addressed personalization within IR from a MAS perspective, integrating a Web scraping agent that tracks the user's activities on the Web and a crawling agent which collects information from social networks. These two agents are used to predict the user's search intentions.

Game theory, as a branch of mathematics used in several scientific domains (see chapter "Games in Artificial Intelligence" of Volume 2), is perceived as an analytic method that is used to model the behavior of rational players who defend their interests in well-defined situations. It consists in identifying the actors and the strategies undertaken. There are several works that addressed IR tasks from a game theoretic point of view. Raifer et al. (2017) considered game theory to analyze publishers' behavior regarding their documents ranking on the Web. They described that, as a "*ranking competition between documents' authors (publishers) for certain queries*". They believe that the modelling of the publishers behavior from a game theoretic perspective helps address the post-ranking process in retrieval models. The work presented in Zhai (2016), proposed a game-theoretic formulation to optimize the search engine performance on a search session and not just for an individual query. The retrieval process consists of a search engine and a user who stands for a player in a cooperative game, with the aim to help the user satisfy her/his information need with a minimum use effort and operation cost. Hubert et al. (2018) proposed an unsupervised ranking model inspired by real-life game and sport competition principles. Documents compete against each other in tournaments using features as evidences of relevance. Tournaments are

modeled as a sequence of matches involving pairs of documents matches. Once a tournament is ended, documents are ranked according to their number of won matches during the tournament.

Decision Making is relatively close to the IR domain as it intervenes within several tasks, starting from the choice of the query terms to the information display. Indeed, the fact that the user has to decide about the query's terms in order to have the desired information, can be considered as a decision making problem and several works were proposed to assist the user with query reformulation (Phillips-Wren and Forgionne 2004). Hosanagar (2011) addresses several IR issues from an optimal operational decision making perspective for distributed IR by considering the user's preferences and the performance history of the distributed sources. He proposed a utility-theoretic framework associating the waiting time cost, user's decision strategies and the information value. Moulahi et al. (2014) proposed, iAggregator, a fuzzy-based operator for multidimensional relevance aggregation, inspired from the Choquet Integral Operator (Choquet 1954). This latter has been extensively used in multicriteria decision-making problems. The authors adapted this operator, not widely used in IR, to evaluate multicriteria relevance aggregation on a tweet search task.The criteria considered were: topicality, recensy and authority (chapters "Multicriteria Decision Making", "Decision under Uncertainty" and "Collective Decision Making" of Volume 1 discuss in extensive details the different aspects of decision making.)

8 Conclusion

The role of AI in IR has been discussed by several authors for decades. They all assumed that the impact of AI remains limited especially for adhoc IR. In this chapter, we attempted to give an overview of the AI topics that have been mostly used in IR. We first addressed the NLP topic with its ability to improve documents' representation through its accurate text analysis models. Then, came the logic wave in the late 80's and early 90's, which basically involves inference systems and it is well adapted tools for knowledge representation.

Fuzzy models were introduced in the 70's but were mainly developed in the 90's. These models allowed a flexible formulation of queries. Meanwhile, evolutionary-based approaches and simplistic neural networks models, not involving learning techniques, were proposed. However, despite of the consequent number of the works based on these models, the improvements drawn out in terms of performance, were not significant enough to validate these models out of an academic frameworks.

It was until the 2000's, that we could see Machine Learning techniques getting involved with IR tasks and particularly the ranking process, "Learning to Rank". The ability of these models to handle hundreds of features shed a real interest on them, especially for the Web search engines. However, these models, applied to other adhoc

tasks, could not compete in terms of performance with the traditional models, since these later do not require a learning phase.

Recently, we have been experiencing the Deep Learning trend especially for document representation. Here again, the results reported do not clearly show the impact of these models,unlike to what has been observed in the field of Image Retrieval. Indeed, thanks to these neural models, there have been considerable advances in terms of performances in this research field (Zhao et al. 2017).

To sum up, as far as IR models have been proved to be effective for a particular task of information management, namely, document retrieval, the impact of AI for this type of tasks is still limited compared to statistical methods. One of the reasons observed by Lewis and Sparck-Jones (1996) is that "*Statistical IR has picked some of the fruits of the tree, and what is left is much harder*". Another explanation pointed out by Karen Sparck-Jones (1999) is "*that they work because, in situations where information demand, and hence supply, is underspecified, the right strategy is to be broadly indicative, rather than aggressively analytic (as in decision trees)*". Therefore, AI is deemed especially useful for tasks that require fine-grained text analysis, to mention, Opinion Mining, Question Answering, Entity Retrieval and Relation Retrieval. Likewise, complex IR tasks, involving more that the usual retrieval modelling task, will necessarily require AI tools (Yang et al. 2016). Tasks, like conversational search, demand natural language understanding. IR systems are no longer related only to the typical search process. They have to be able to explain the queries' answers and handle users' questions. Information credibility is an important aspect ensuring that the retrieved information is trustworthy. Reasoning models can be the best resort to such issues.

References

Abdulahhad K (2014) Information retrieval (IR) modeling by logic and lattice. Application to conceptual IR. Theses, Université de Grenoble

Abdulahhad K, Chevallet JP, Berrut C (2017) Logics, lattices and probability: the missing links to information retrieval. Comput J 60(7):995–1018

Abdullah HS, Hadi MJ (2014) Artificial bee colony based approach for web information retrieval. Eng Technol J 32(5):899–909

Ai Q, Yang L, Guo J, Croft WB (2016) Analysis of the paragraph vector model for information retrieval. In: Proceedings of the 2016 ACM international conference on the theory of information retrieval, ACM, pp 133–142

Al-Khateeb B, Al-Kubaisi AJ, Al-Janabi ST, (2017) Query reformulation using wordnet and genetic algorithm. In: Annual conference on new trends in information and communications technology applications (NTICT) (2017), IEEE, pp 91–96

De Almeida HM, Gonçalves MA, Cristo M, Calado P (2007) A combined component approach for finding collection-adapted ranking functions based on genetic programming. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 399–406

Amati G, Ounis I (2000) Conceptual graphs and first order logic. Comput J 43(1):1–12

Amati G, Van Rijsbergen CJ (2002) Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans Inf Syst (TOIS) 20(4):357–389

Araujo L, Pérez-Iglesias J (2010) Training a classifier for the selection of good query expansion terms with a genetic algorithm. In: IEEE congress on evolutionary computation (CEC), IEEE, pp 1–8

Baziz M, Boughanem M, Aussenac-Gilles N (2005) Conceptual indexing based on document content representation, vol CoLIS'05. Springer, Berlin, Heidelberg, pp 171–186

Baziz M, Boughanem M, Prade H, Pasi G (2006) A fuzzy logic approach to information retrieval using an ontology-based representation of documents. In: Sanchez E (ed) Fuzzy logic and the semantic web, capturing intelligence, vol 1. Elsevier, Amsterdam, pp 363–377

Baziz M, Boughanem M, Pasi G, Prade H (2007) An information retrieval driven by ontology from query to document expansion. In: Large Scale Semantic Access to Content (Text, Image, Video, and Sound), Le Centre des Hautes Etudes Internationales D'Informatique Documentaire, pp 301–313

Bebis G, Georgiopoulos M (1994) Feed-forward neural networks. IEEE Potentials 13(4):27–31

Belew RK (1987) A connectionist approach to conceptual information retrieval. In: Proceedings of the 1st international conference on artificial intelligence and law, ACM, New York, NY, USA, ICAIL '87, pp 116–126

Belkin NJ, Marchetti PG (1989) Determining the functionality features of an intelligent interface to an information retrieval system. In: Proceedings of the 13th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 151–177

Belkin NJ, Brooks HM, Daniels PJ (1987) Knowledge elicitation using discourse analysis. Int J Man-Mach Stud 27(2):127–144

Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. J Mach Learn Res 3(Feb):1137–1155

Bindal AK, Sanyal S (2012) Query optimization in context of pseudo relevant documents. In: 3rd Italian Information Retrieval (IIR) workshop, EPFL-CONF-174006

Bookstein A (1980) Fuzzy requests: an approach to weighted boolean searches. J Assoc Inf Sci Technol 31(4):240–247

Bordogna G, Pasi G (1993) A fuzzy linguistic approach generalizing boolean information retrieval: a model and its evaluation. J Am Soc Inf Sci 44(2):70

Bordogna G, Pasi G (1995) Linguistic aggregation operators of selection criteria in fuzzy information retrieval. Int J Intell Syst 10(2):233–248

Bordogna G, Pasi G (2001) An ordinal information retrieval model. Int J Uncertain Fuzziness Knowl-Based Syst 09(supp01):63–75

Bordogna G, Pasi G (2005) Personalised indexing and retrieval of heterogeneous structured documents. Inf Retr 8(2):301–318

Bordogna G, Carrara P, Pasi G (1991) Query term weights as constraints in fuzzy information retrieval. Inf Process Manage 27(1):15–26

Bordogna G, Carrara P, Pasi G (1992) Extending boolean information retrieval: a fuzzy model based on linguistic variables. In: 1992 IEEE international conference on fuzzy systems, IEEE, pp 769–776

Boughanem M, Loiseau Y, Prade H (2007) Refining aggregation functions for improving document ranking in information retrieval. In: International conference on scalable uncertainty management, Springer, pp 255–267

Boughanem M, Brini A, Dubois D (2009) Possibilistic networks for information retrieval. Int J Approx Reason 50(7):957–968

Boughanem SDC M (1992) A connexionist model for information retrieval. In: Proceeding of DEXA, pp 260–265

Bruza PD, van der Gaag LC (1994) Index expression belief networks for information disclosure. Int J Expert Syst 7(2):107–138

Buell DA (1985) A problem in information retrieval with fuzzy sets. J Assoc Inf Sci Technol 36(6):398–401

Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005) Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on machine learning, ACM, pp 89–96

Cao G, Nie JY, Bai J (2005) Integrating word relationships into language models. In: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 298–305

Chawla S (2013) Personalised web search using aco with information scent. Int J Knowl Web Intell 4(2–3):238–259

Chawla S (2017) Web page ranking using ant colony optimisation and genetic algorithm for effective information retrieval. Int J Swarm Intell 3(1):58–76

Chen H (1995) Machine learning for information retrieval: neural networks, symbolic learning, and genetic algorithms. J Assoc Inf Sci Technol 46(3):194–216

Chen H, Dhar V (1989) Online query refinement on information retrieval systems: a process model of searcher/system interactions. In: Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 115–133

Chevallet JP, Chiaramella Y (1998) Experiences in information retrieval modelling using structured formalisms and modal logic. Information retrieval: uncertainty and logics. Springer, Berlin, pp 39–72

Chiaramella Y, Chevallet JP (1992) About retrieval models and logic. Comput J 35(3):233–242

Choquet G (1954) Theory of capacities. Annales de l'institut Fourier 5:131–295

Chowdhury A, McCabe MC (1998) Improving information retrieval systems using part of speech tagging. Tech rep

Cleverdon CW, Keen M (1966) Factors determining the performance of indexing systems; volume 2, test results, aslib cranfield research project. Tech rep

Clinchant S, Gaussier E (2010) Information-based models for ad hoc ir. In: Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval, ACM, pp 234–241

Cole C (1998) Intelligent information retrieval: diagnosing information need. Part i. the theoretical framework for developing an intelligent ir tool. Inf Process Manag 34(6):709–720

Colorini A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant colonies. The 1st european conference on artificial life. Pans, France

Cooper WS (1971) A definition of relevance for information retrieval. Inf Storage Retr 7(1):19–37

Crammer K, Singer Y (2002) Pranking with ranking. In: Advances in neural information processing systems, pp 641–647

Crestani F (1998) Logical imaging and probabilistic information retrieval. In: Information Retrieval: uncertainty and logics. Springer, Berlin, pp 247–279

Crestani F, van Rijsbergen CJ (1995) Information retrieval by logical imaging. J Doc 51(1):3–17

Crestani F, De Campos LM, Fernández-Luna JM, Huete JF (2003) A multi-layered bayesian network model for structured document retrieval. In: European conference on symbolic and quantitative approaches to reasoning and uncertainty. Springer, pp 74–86

Croft WB (1987) Approaches to intelligent information retrieval. Inf Process Manage 23(4):249–254

Damiani E, Marrara S, Pasi G (2007) Fuzzyxpath: using fuzzy logic an ir features to approximately query xml documents. In: International fuzzy systems association world congress. Springer, pp 199–208

De Baets B, Fodor J (1997) On the structure of uninorms and their residual implicators. In: 18th Proceedings Linz seminar on fuzzy set theory, pp 81–87

De Campos LM, Fernández-Luna JM, Huete JF (2002) A layered bayesian network model for document retrieval. In: European conference on information retrieval. Springer, pp 169–182

De Campos LM, Fernández-Luna JM, Huete JF (2003) Improving the efficiency of the bayesian network retrieval model by reducing relationships between terms. Int J Uncertain Fuzziness Knowl-Based Syst 11(supp01):101–116

Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391

Dehghani M, Zamani H, Severyn A, Kamps J, Croft WB (2017) Neural ranking models with weak supervision. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 65–74

Denoyer L, Gallinari P (2003) A belief networks-based generative model for structured documents. an application to the xml categorization. In: International workshop on machine learning and data mining in pattern recognition. Springer, pp 328–342

Diaz F, Mitra B, Craswell N (2016) Query expansion with locally-trained word embeddings. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers), Association for computational linguistics, pp 367–377

Ding Y (2001) Ir and ai: the role of ontology. In: International conference of Asian digital libraries, India

Dinh D, Tamine L, Boubekeur F (2013) Factors affecting the effectiveness of biomedical document indexing and retrieval based on terminologies. Artif Intell Med 57(2):155–167

Dubois D, Prade H (1985) A review of fuzzy set aggregation connectives. Inf Sci 36(1–2):85–121

Dubois D, Prade H, Testemale C (1988) Weighted fuzzy pattern matching. Fuzzy Sets Syst 28(3):313–331

Dubois D, Fargier H, Prade H (1997) Beyond min aggregation in multicriteria decision:(ordered) weighted min, discri-min, leximin. The ordered weighted averaging operators. Springer, Berlin, pp 181–192

Enembreck F, Barthès JP, Ávila BC (2004) Personalizing information retrieval with multi-agent systems. In: International workshop on cooperative Information Agents. Springer, pp 77–91

Evans DA, Zhai C (1996) Noun-phrase analysis in unrestricted text for information retrieval. In: Proceedings of the 34th annual meeting on association for computational linguistics, Association for computational linguistics, pp 17–24

Fagan J (1987a) Automatic phrase indexing for document retrieval. In: Proceedings of the 10th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 91–101

Fagan JL (1987b) Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods. PhD thesis

Fang H (2008) A re-examination of query expansion using lexical resources. In: Proceedings of ACL-08: HLT, pp 139–147

Firth JR (1957) A synopsis of linguistic theory, 1930-1955. Studies in linguistic analysis

Fox EA, Sharan S (1986) A comparison of two methods for soft boolean operator interpretation in information retrieval. Tech. rep. Tech., Department of. Computer Science, Blacksburg, VA: Virginia

Freund Y, Iyer R, Schapire RE, Singer Y (2003) An efficient boosting algorithm for combining preferences. J Mach Learn Res 4(Nov):933–969

Fuhr N (1995) Probabilistic dataloga logic for powerful retrieval methods. In: Proceedings of the 18th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 282–290

Fung R, Del Favero B (1995) Applying bayesian networks to information retrieval. Commun ACM 38(3):42–50

Ganguly D, Roy D, Mitra M, Jones GJ (2015) Word embedding based generalized language model for information retrieval. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 795–798

Goldberg DE, Corruble V (1994) Algorithmes génétiques: exploration, optimisation et apprentissage automatique. Ed. Addison-Wesley, France

Gonzalo J, Verdejo F, Chugur I, Cigarrin J (1998) Indexing with wordnet synsets can improve text retrieval. In: Workshop on usage Of WordNet in natural language processing systems, pp 38–44

Gonzalo J, Li H, Moschitti A, Xu J (2014) Semantic matching in information retrieval. In: Proceedings of the 37th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 1296–1296

Gordon MD (1991) User-based document clustering by redescribing subject descriptions with a genetic algorithm. J Am Soc Inf Sci 42(5):311

Guo J, Fan Y, Ai Q, Croft WB (2016) A deep relevance matching model for ad-hoc retrieval. In: Proceedings of the 25th ACM international on conference on information and knowledge management, ACM, pp 55–64

Hammache A, Boughanem M, Ahmed-Ouamer R (2014) Combining compound and single terms under language model framework. Knowl Inf Syst 39(2):329–349

Harman D (1991) How effective is suffixing? J Am Soc Inf Sci 42(1):7

Hassan A, Hadi M (2016) Sense-based information retrieval using artificial bee colony approach. Int J Appl Eng Res 11(15):8708–8713

Hayashi I, Nomura H, Yamasaki H, Wakami N (1992) Construction of fuzzy inference rules by ndf and ndfl. Int J Approx Reason 6(2):241–266

Herbrich R (2000) Large margin rank boundaries for ordinal regression. Advances in large margin classifiers. MIT Press, Oxford, pp 115–132

Herrera-Viedma E (2001) Modeling the retrieval process for an information retrieval system using an ordinal fuzzy linguistic approach. J Assoc Inf Sci Technol 52(6):460–475

Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Oxford

Hollink V, Kamps J, Monz C, De Rijke M (2004) Monolingual document retrieval for european languages. Inf Retr 7(1–2):33–52

Hosanagar K (2011) Usercentric operational decision making in distributed information retrieval. Inf Syst Res 22(4):739–755

Hu B, Lu Z, Li H, Chen Q (2014) Convolutional neural network architectures for matching natural language sentences. In: Advances in neural information processing systems, pp 2042–2050

Huang PS, He X, Gao J, Deng L, Acero A, Heck L (2013) Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM international conference on Conference on information and knowledge management, ACM, pp 2333–2338

Hubert G, Pitarch Y, Pinel-Sauvagnat K, Tournier R, Laporte L (2018) Tournarank: when retrieval becomes document competition. Inf Process Manag 54(2):252–272

Huibers TWC (1994) Situations, a general framework for studying information retrieval, vol 1994. Unknown Publisher

Hunter A (1997) Using default logic for lexical knowledge. Qualitative and quantitative practical reasoning. Springer, Berlin, pp 322–335

Indrawan M, Ghazfan D, Srinivasan B (1996) Using bayesian networks as retrieval engines. In: Proceedings of the text retrieval conference TREC'96

Jabeur LB, Tamine L, Boughanem M (2012) Active microbloggers: Identifying influencers, leaders and discussers in microblogging networks. In: International symposium on string processing and information retrieval. Springer, pp 111–117

Jones KS (1983) Intelligent retrieval. Proceedings of Informatics

Jones KS (1991) The role of artificial intelligence in information retrieval. J Am Soc Inf Sci 42(8):558

Jones KS (1999) Information retrieval and artificial intelligence. Artif Intell 114(1–2):257–281

Jozefowicz R, Vinyals O, Schuster M, Shazeer N, Wu Y (2016) Exploring the limits of language modeling. arXiv:160202410

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer engineering department

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (abc) algorithm. Appl Soft Comput 8(1):687–697

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: 1995 Proceedings of IEEE international conference on neural networks, vol 4, pp 1942–1948

Kenter T, De Rijke M (2015) Short text similarity with word embeddings. In: Proceedings of the 24th ACM international on conference on information and knowledge management, ACM, pp 1411–1420

Khennak I, Drias H (2017) An accelerated pso for query expansion in web information retrieval: application to medical dataset. Appl Intell 1–16

Kim S, Zhang BT (2003) Genetic mining of html structures for effective web-document retrieval. Appl Intell 18(3):243–256

Kim Y, Jernite Y, Sontag D, Rush AM (2015) Character-aware neural language models. In: AAAI, pp 2741–2749

Kiros R, Zhu Y, Salakhutdinov RR, Zemel R, Urtasun R, Torralba A, Fidler S (2015) Skip-thought vectors. In: Advances in neural information processing systems, pp 3294–3302

Kraaij W, Pohlmann R (1996) Viewing stemming as recall enhancement. In: Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 40–48

Kraft DH, Buell DA (1983) Fuzzy sets and generalized boolean retrieval systems. Int J Man-Mach Stud 19(1):45–56

Kraft DH, Colvin E (2017) Fuzzy information retrieval. Synthesis lectures on information concepts, retrieval, and services. Morgan & Claypool Publishers, San Rafael

Kraft DH, Bordogna G, Pasi G (1999) Fuzzy set techniques in information retrieval. Fuzzy sets in approximate reasoning and information systems. Springer, Berlin, pp 469–510

Kraft DH, Colvin E, Bordogna G, Pasi G (2015) Fuzzy information retrieval systems: a historical perspective. Fifty years of fuzzy logic and its applications. Springer, Berlin, pp 267–296

Kripke SA (1963) Semantic analysis of modal logic I: Normal modal and propositional calculi. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9:67–96

Krovetz R (1993) Viewing morphology as an inference process. In: Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 191–202

Krovetz R (1997) Homonymy and polysemy in information retrieval. In: Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the European chapter of the association for computational linguistics, Association for computational linguistics, Stroudsburg, PA, USA, ACL '98, pp 72–79

Krovetz R, Croft WB (1992) Lexical ambiguity and information retrieval. ACM Trans Inf Syst 10(2):115–141

Kwok KL (1989) A neural network for probabilistic information retrieval. In: Proceedings of the 12th annual international ACM SIGIR conference on research and development in information retrieval, ACM, New York, NY, USA, SIGIR '89, pp 21–30

Lalmas M (1998) Logical models in information retrieval: introduction and overview. Inf Process Manag 34(1):19–33

Lalmas M, van Rijsbergen K (1993) A logical model of information retrieval based on situation theory. In: 14th information retrieval colloquium. Springer, pp 1–13

Lavrenko V, Croft WB (2017) Relevance-based language models. In: ACM SIGIR Forum, ACM, vol 51, pp 260–267

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning, pp 1188–1196

Lee H, Grosse R, Ranganath R, Ng AY (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th annual international conference on machine learning, ACM, pp 609–616

Lewis DD (1992) Representation and learning in information retrieval. PhD thesis, University of Massachusetts at Amherst

Lewis DD, Jones KS (1996) Natural language processing for information retrieval. Commun ACM 39(1):92–101

Li H (2011) Learning to rank for information retrieval and natural language processing. Synth Lect Hum Lang Technol 4(1):1–113

Li H, Xu J et al (2014) Semantic matching in search. Found Trends® Inf Retrieva 7(5):343–469
Li P, Wu Q, Burges CJ (2008) Mcrank: learning to rank using multiple classification and gradient boosting. In: Advances in neural information processing systems, pp 897–904
Lioma C, Blanco R (2009) Part of speech based term weighting for information retrieval. In: European conference on information retrieval. Springer, pp 412–423
Liu S, Liu F, Yu C, Meng W (2004) An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 266–272
Liu S, Yu C, Meng W (2005) Word sense disambiguation in queries. In: Proceedings of the 14th ACM international conference on information and knowledge management, ACM, pp 525–532
Liu TY et al (2009) Learning to rank for information retrieval. Found Trends® Inf Retr 3(3):225–331
Loiseau Y, Prade H, Boughanem M (2004) Qualitative pattern matching with linguistic terms. AI Commun 17(1):25–34
Losada DE, Barreiro A (2001) A logical model for information retrieval based on propositional logic and belief revision. Comput J 44(5):410–424
Losada DE, Barreiro A (2003) Propositional logic representations for documents and queries: a large-scale evaluation. In: European conference on information retrieval, Springer, pp 219–234
Lu Z, Li H (2013) A deep architecture for matching short texts. In: Advances in neural information processing systems, pp 1367–1375
Luhn HP (1957) A statistical approach to mechanized encoding and searching of literary information. IBM J Res Dev 1(4):309–317
Lv Y, Zhai C (2009) Positional language models for information retrieval. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 299–306
MacAvaney S, Hui K, Yates A (2017) An approach for weakly-supervised deep information retrieval. In: Workshop on neural information retrieval (Neu-IR '17) at SIGIR 2017
Maes P et al (1994) Agents that reduce work and information overload. Commun ACM 37(7):30–40
Mandl T (2009) Artificial intelligence for information retrieval. In: Encyclopedia of artificial intelligence, IGI Global, pp 151–156
Manning CD, Raghavan P, Schtze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge, UK
Marrara S, Pasi G, Viviani M (2017) Aggregation operators in information retrieval. Fuzzy Sets Syst 324:3–19
Mauldin ML (1991) Retrieval performance in ferret a conceptual information retrieval system. In: Proceedings of the 14th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 347–355
Meghini C, Straccia, (1996) A relevance terminological logic for information retrieval. In: Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 197–205
Meghini C, Sebastiani F, Straccia U, Thanos C (1993) A model of information retrieval based on a terminological logic. In: Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 298–307
Metzler D, Croft WB (2005) A markov random field model for term dependencies. In: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 472–479
Mihalcea R, Moldovan D (2000) Semantic indexing using wordnet senses. In: Proceedings of the ACL-2000 workshop on recent advances in natural language processing and information retrieval: held in conjunction with the 38th annual meeting of the association for computational linguistics - volume 11, Association for computational linguistics, Stroudsburg, PA, USA, RANLPIR '00, pp 35–45
Mikolov T, Chen K, Corrado G (2013) Efficient estimation of word representations in vector space. Dean J CoRR abs/1301.3781

Mitra B, Craswell N (2017) An introduction to neural information retrieval. Found Trends® Inf Retr :1–120
Mitra B, Nalisnick ET, Craswell N, Caruana R (2016) A dual embedding space model for document ranking. CoRR abs/1602.01137
Mitra B, Diaz F, Craswell N (2017) Learning to match using local and distributed representations of text for web search. In: Proceedings of the 26th international conference on world wide web, international world wide web conferences steering committee, pp 1291–1299
Miyamoto S (1990) Information retrieval based on fuzzy associations. Fuzzy Sets Syst 38(2):191–205
Molinari A, Pasi G (1996) A fuzzy representation of html documents for information retrieval systems. In: 1996 Proceedings of the fifth ieee international conference on fuzzy systems, IEEE, vol 1, pp 107–112
Moon J, Shon T, Seo J, Kim J, Seo J (2004) An approach for spam e-mail detection with support vector machine and n-gram indexing. In: International symposium on computer and information sciences, Springer, pp 351–362
Moulahi B, Tamine L, Yahia SB (2014) iaggregator: multidimensional relevance aggregation based on a fuzzy operator. J Assoc Inf Sci Technol 65(10):2062–2083
Nalisnick E, Mitra B, Craswell N, Caruana R (2016) Improving document ranking with dual word embeddings. In: Proceedings of the 25th international conference companion on world wide web, international world wide web conferences steering committee, pp 83–84
Nallapati R (2004) Discriminative models for information retrieval. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 64–71
Nie J (1988) An outline of a general model for information retrieval systems. In: Proceedings of the 11th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 495–506
Nie J (1989) An information retrieval model based on modal logic. Inf Process Manag 25(5):477–491
Nie JY, Brisebois M, Lepage F (1995) Information retrieval as counterfactual. Comput J 38(8):643–657
Onal KD, Zhang Y, Altingovde IS, Rahman MM, Karagoz P, Braylan A, Dang B, Chang HL, Kim H, McNamara Q et al (2017) Neural information retrieval: at the end of the early years. Inf Retr J 1–72
Pang L, Lan Y, Guo J, Xu J, Cheng X (2016) A study of matchpyramid models on ad-hoc retrieval. In: ACM SIGIR workshop on neural information retrieval (Neu-IR)
Pasi G (2009) Fuzzy models. Encyclopedia of database systems. Springer, Berlin, pp 1205–1209
Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
Phillips-Wren GE, Forgionne GA (2004) Intelligent decision making in information retrieval. In: International conference on intelligent information and engineering systems. Springer, pp 103–109
Picard J (1999) Logic as a tool in a term matching information retrieval system. In: Proceedings of the workshop on logical and uncertainty models for information systems, pp 77–90
Picard J, Savoy J (2000) A logical information retrieval model based on a combination of propositional logic and probability theory. In: Soft computing in information retrieval, Springer, pp 225–258
Ponte JM, Croft WB (1998) A language modeling approach to information retrieval. In: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 275–281
Porter MF (1980) An algorithm for suffix stripping. Program 14(3):130–137
Qi G, Pan JZ (2008) A tableau algorithm for possibilistic description logic $\mathcal{DL}$. In: Asian semantic web conference, Springer, pp 61–75

Raifer N, Raiber F, Tennenholtz M, Kurland, (2017) Information retrieval meets game theory: The ranking competition between documents' authors. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 465–474

Rekabsaz N (2016) Enhancing information retrieval with adapted word embedding. In: Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 1169–1169

Ribeiro BA, Muntz R (1996) A belief network model for ir. In: Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 253–260

van Rijsbergen CJ (1989) Towards an information logic. In: ACM SIGIR forum, ACM, vol 23, pp 77–86

Robertson SE, Walker S (1994) Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval, Springer-Verlag New York, Inc., pp 232–241

Rölleke T, Fuhr N (1996) Retrieval of complex objects using a four-valued logic. In: Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 206–214

Roy D, Paul D, Mitra M, Garain U (2016) Using word embeddings for automatic query expansion. In: ACM SIGIR workshop on neural information retrieval

Salton G (1991) Developments in automatic text retrieval. Science 253(5023):974–980

Salton G, McGill MJ (1986) Introduction to modern information retrieval. McGraw-Hill Inc, New York, NY, USA

Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620

Sanchez E (1989) Importance in knowledge systems. Inf Syst 14(6):455–464

Sanderson M (1994) Word sense disambiguation and information retrieval. In: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval, Springer-Verlag New York, Inc., pp 142–151

Sanderson M (2000) Retrieving with good sense. Inf Retr 2(1):49–69

Sathya SS, Simon P (2010) A document retrieval system with combination terms using genetic algorithm. Int J Comput Electr Eng 2(1):1

Schütze H, Pedersen JO (1995) Information retrieval based on word senses, pp 161–175

Sebastiani F (1994) A probabilistic terminological logic for modelling information retrieval. In: SIGIR94, Springer, pp 122–130

Sebastiani F (1998) On the role of logic in information retrieval. Inf Process Manag 34(1):1–18

Shashua A, Levin A (2003) Ranking with large margin principle: two approaches. In: Advances in neural information processing systems, pp 961–968

Shen Y, He X, Gao J, Deng L, Mesnil G (2014) Learning semantic representations using convolutional neural networks for web search. In: Proceedings of the 23rd international conference on world wide web, ACM, pp 373–374

Sheridan P, Smeaton AF (1992) The application of morpho-syntactic language processing to effective phrase matching. Inf Process Manag 28(3):349–369

Shi L, Nie JY (2009) Integrating phrase inseparability in phrase-based model. In: Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval, ACM, New York, NY, USA, SIGIR '09, pp 708–709

Silva I, Ribeiro-Neto B, Calado P, Moura E, Ziviani N (2000) Link-based and content-based evidential information in a belief network model. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp pp 96–103

Sowa JF (1983) Conceptual structures: information processing in mind and machine

Sparck Jones K (1972) A statistical interpretation of term specificity and its application in retrieval. J Doc 28(1):11–21

Stokoe C, Oakes MP, Tait J (2003) Word sense disambiguation in information retrieval revisited. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval, ACM, pp 159–166

Strzalkowski T (1995) Natural language information retrieval. Inf Process Manag 31(3):397–417

Tamine L, Chrisment C, Boughanem M (2003) Multiple query evaluation based on an enchanced genetic algorithm. Inf Process Manag 39(2):215–231

Tamir DE, Rishe ND, Kandel A (2015) Fifty years of fuzzy logic and its applications, vol 326. Springer, Berlin

Tao T, Zhai C (2007) An exploration of proximity measures in information retrieval. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 295–302

Thiel U, Müller A (1996) Why was this item retrieved? new ways to explore retrieval results. Information retrieval and hypertext. Springer, Berlin, pp 181–201

Tong X, Zhai C, Milic-Frayling N, Evans DA (1997) Evaluation of syntactic phrase indexing. In: The fifth text retrieval conference (TREC-5)

Trifa A, Sbaï AH, Chaari WL (2017) Evaluate a personalized multi agent system through social networks: web scraping. In: 2017 IEEE 26th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE), IEEE, pp 18–20

Tsai MF, Liu TY, Qin T, Chen HH, Ma WY (2007) Frank: a ranking method with fidelity loss. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 383–390

Turtle H, Croft WB (2017) Inference networks for document retrieval. ACM SIGIR forum, ACM, vol 51, pp 124–147

Van Rijsbergen CJ (1986) A non-classical logic for information retrieval. Comput J 29(6):481–485

Vickery A, Brooks HM (1987) Plexus-the expert system for referral. Inf Process Manag 23(2):99–117

Voorhees EM (1993) Using wordnet to disambiguate word senses for text retrieval. In: Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 171–180

Voorhees EM (1994) Query expansion using lexical-semantic relations. In: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval, Springer-Verlag New York, Inc., pp 61–69

Vrajitoru D (2013) Large population or many generations for genetic algorithms? Soft computing in information retrieval: techniques and applications, vol 50, p 199

Vulić I, Moens MF (2015) Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 363–372

Wan S, Lan Y, Guo J, Xu J, Pang L, Cheng X (2016) A deep architecture for semantic matching with multiple positional sentence representations. In: AAAI, pp 2835–2841

Wang C, Akella R (2015) Concept-based relevance models for medical and semantic information retrieval. In: Proceedings of the 24th ACM international on conference on information and knowledge management, ACM, New York, NY, USA, CIKM '15, pp 173–182

Wu Q, Burges CJ, Svore KM, Gao J (2010) Adapting boosting for information retrieval measures. Inf Retr 13(3):254–270

Xia F, Liu TY, Wang J, Zhang W, Li H (2008) Listwise approach to learning to rank: theory and algorithm. In: Proceedings of the 25th international conference on machine learning, ACM, pp 1192–1199

Yager RR (1988) On ordered weighted averaging aggregation operators in multicriteria decision-making. IEEE Trans Syst Man Cybern 18(1):183–190

Yang GH, Sloan M, Wang J (2016) Dynamic information retrieval modeling, vol 8(3). Synthesis lectures on information concepts, retrieval, and services, pp 1–144

Yang J, Korfhage RR, Rasmussen E (1993) Query improvement in information retrieval using genetic algorithms: a report on the experiments of the trec project. In: Proceedings of the 1st text retrieval conference, pp 31–58

Yin W, Schütze H, Xiang B, Zhou B (2015) Abcnn: attention-based convolutional neural network for modeling sentence pairs. arXiv preprint arXiv:151205193

Yue Y, Finley T, Radlinski F,Joachims T (2007) A support vector method for optimizing average precision. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, ACM, pp 271–278

Zakos J (2005) A novel concept and context-based approach for web information retrieval. PhD thesis

Zamani H, Croft WB (2016a) Embedding-based query language models. In: Proceedings of the 2016 ACM international conference on the theory of information retrieval, ACM, pp 147–156

Zamani H, Croft WB (2016b) Estimating embedding vectors for queries. In: Proceedings of the 2016 ACM international conference on the theory of information retrieval, ACM, pp 123–132

Zamani H, Mitra B, Song X, Craswell N, Tiwary S (2018) Neural ranking models with multiple document fields. In: Proceedings of the eleventh ACM international conference on web search and data mining, ACM, pp 700–708

Zhai C (1997) Fast statistical parsing of noun phrases for document indexing. In: Proceedings of the fifth conference on applied natural language processing, Association for computational linguistics, pp 312–319

Zhai C (2008) Statistical language models for information retrieval. Synthesis lectures on human language technologies, vol 1(1), pp 1–141

Zhai C (2016) Towards a game-theoretic framework for text data retrieval. IEEE Data Eng Bull 39(3):51–62

Zhao B, Feng J, Wu X, Yan S (2017) A survey on deep learning-based fine-grained object classification and semantic segmentation. Int J Autom Comput 14(2):119–135

Zhao J, Yun Y (2009) A proximity language model for information retrieval. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 291–298

Zheng G, Callan J (2015) Learning to reweight terms with distributed representations. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 575–584

Zuccon G, Azzopardi L, van Rijsbergen CJ (2009) Revisiting logical imaging for information retrieval. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '09, pp 766–767

Zuccon G, Koopman B, Bruza P, Azzopardi L (2015) Integrating and evaluating neural word embeddings in information retrieval. In: Proceedings of the 20th Australasian document computing symposium, ACM, p 12

Semantic Web

Jérôme Euzenat and Marie-Christine Rousset

Abstract The semantic web aims at making web content interpretable. It is no less than offering knowledge representation at web scale. The main ingredients used in this context are the representation of assertional knowledge through graphs, the definition of the vocabularies used in graphs through ontologies, and the connection of these representations through the web. Artificial intelligence techniques and, more specifically, knowledge representation techniques, are put to use and to the test by the semantic web. Indeed, they have to face typical problems of the web: scale, heterogeneity, incompleteness, and dynamics. This chapter provides a short presentation of the state of the semantic web and refers to other chapters concerning those techniques at work in the semantic web.

1 Introduction

The idea of using knowledge representation on the worldwide web appeared rapidly after the development of the web. Systems like SHOE (Luke et al. 1997) and Ontobroker (Fensel et al. 1998) integrated formal knowledge representation in web pages while Ontoserver (Farquhar et al. 1995) and HyTroeps (Euzenat 1996) used the web to browse and edit knowledge.

In 1998, Tim Berners-Lee described the principles of what he called the 'semantic web'. He already had launched, within the W3C,[1] the development of the RDF language whose initial goal was to annotate web pages. At that time, he wrote '*A Semantic Web is not Artificial Intelligence*', but later he added that it was '*Knowledge Representation goes Global*' (Berners-Lee 1998).

[1] *WorldWide Web Consortium*: the organisation in charge of recommending web technologies.

J. Euzenat (✉)
Université Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
e-mail: Jerome.Euzenat@inria.fr

M.-C. Rousset
Université Grenoble Alpes and Institut Universitaire de France, CNRS, Inria,
Grenoble INP, LIG, 38000 Grenoble, France
e-mail: Marie-Christine.Rousset@imag.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_6

In 2000, this idea had not taken off nor been subject to ambitious developments. The US DARPA launched the DAML (*DARPA Agent Markup Language*) program directed by James Hendler, the initiator of SHOE. Right afterwards, was held a seminar on '*Semantics for the Web*' (Fensel et al. 2003) which steered the EU OntoWeb thematic network, itself leading to the Knowledge Web network of excellence. Such efforts have involved many artificial intelligence researchers in the development of semantic web technologies (Euzenat 2002).

The semantic web is often presented as a 'web for machines': it aims at representing knowledge on the web with formal languages that can be processed by machines. For that purpose, it is natural to use techniques developed in artificial intelligence.

Indeed, the main target application for the semantic web is information retrieval (chapter "Artificial Intelligence and Natural Language" of this Volume): being able to retrieve precise information on the web because documents are annotated with 'semantic metadata'. Although this goal seemed remote in 2000, it is nowadays far more tangible. In the interval, we witnessed the use of of Open Graph by Facebook, the definition of the lightweight ontology schema.org by four of the main search engines (Bing, Google, Yahoo, Yandex), and the use by Google of the *knowledge graph* providing structured answers to queries instead of lists of documents.

Since then, many other types of applications have emerged, each based on the annotation of resources using the semantic web knowledge representation languages. Such applications are referred to as applications of semantic technologies. In particular, this covers:

- Semantic web services in which web service interfaces (input/output) are semantically annotated,
- Semantic peer-to-peer systems in which shared resources are semantically annotated,
- Semantic social networks in which social relations and people profiles are semantically annotated,
- Semantic desktop in which personal information (agenda, address book, etc.) is semantically expressed,
- Semantic web of things in which sensors, effectors and the information they exchange are semantically annotated,
- The web of data in which data sources are expressed and linked in semantic web languages (see Sect. 2.2).

The semantic web has thus spread out of its initial playground. It is now a broad experimentation field in which various problems are investigated such as trust in peer-to-peer systems, statistical data integration, or smart city monitoring. As web technologies, semantic technologies have the potential to progressively affect all computer developments (Janowicz et al. 2015).

We focus on the relationships between artificial intelligence techniques and semantic web technologies. For that purpose, we shall consider three problems at the heart of the semantic web: knowledge representation (Sect. 2), reasoning with knowledge and data (Sect. 3), and relating data and knowledge sources (Sect. 4). This organisation may seem arbitrary. Indeed, it is not organised chronologically,

e.g., the definition of an ontology may occur before or after that of data, and reasoning may occur within a simple data graph or across a network of ontologies. However, each section corresponds to one type of activity with a well-identified set of techniques. Each relies on artificial intelligence techniques, mostly from knowledge representation and reasoning, and could take advantage of other techniques described in this book. We can only survey them: there exist more extensive treatments of these (Antoniou and van Harmelen 2008; Hitzler et al. 2009; Abiteboul et al. 2011).

2 Publishing Data on the Web

The first issue raised on the semantic web is the expression of factual data on the web. To address it, a data expression language, adapted to the characteristics of the web, is required. The RDF (*Resource Description Framework*) language has been developed and recommended by the W3C and adopted in practice.

2.1 RDF: Simple Conceptual Graphs

RDF (Lassila and Swick 1999; Cyganiak et al. 2014) is a formal language for expressing assertion of relations between 'resources'. It is first used to annotate documents written in non structured languages, or as interface between data sources expressed in languages with a comparable semantics, e.g., data bases.

An RDF graph is a set of triples ⟨subject, predicate, object⟩. The three elements may be IRIs (*Internationalized Resource Identifiers* generalising URIs, i.e., *Uniform Resource Identifiers* (Berners-Lee et al. 1998) themselves comprising URLs), literals, e.g., character strings or dates, or variables, also called 'blanks'.

The use of IRIs is specific to semantic web technologies, with respect to artificial intelligence practice. IRIs behave as simple identifiers. However, they are generally associated with a name space identifying their provenance (vocabulary or data source). For instance, `foaf:Person` identifies the `Person` concept in the friend-of-a-friend name space `http://xmlns.com/foaf/0.1/` shortened as `foaf:`. Thus an RDF graph may use different vocabularies with very little risks of conflict or ambiguity.

This set of triples can be represented in a natural way as a graph, or more precisely as a directed labelled multigraph, in which the elements appearing as subject or object are vertices and each triple is represented by an edge between the subject and object labelled by the predicate (see Fig. 1). Such a graph may be encoded as an RDF/XML (Gandon and Schreiber 2014) or N-triple (Carothers and Seaborne 2014) document.

Figure 1 displays an RDF graph fragment. The IRIs of some resources belong to specific vocabularies: FOAF, RDF, rel. The literals are represented by a rectangle, such as "Pierre". Non-labelled vertices represent variables, also called blanks. Intu-

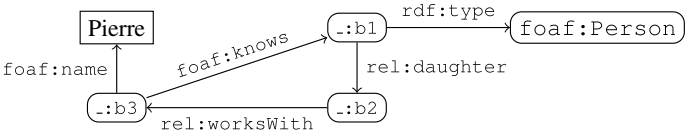


Fig. 1 A RDF graph: Pierre knows a parent of one of his colleagues. Blank nodes are identified by ids in the _ name space

itively, this graph may be understood as: 'Someone called Pierre knows a parent of one of his colleagues'. This intuitive semantics is not sufficient for an automated treatment; hence, RDF has been given a formal semantics.

The semantics of an RDF graph is defined in model theory (Hayes and Patel-Schneider 2014). RDF is thus a proper logics. This semantics has this peculiarity that predicates, which naturally correspond to classical dyadic predicates, may also be considered as resources. Hence, the triple ⟨`rdf:type`, `rdf:type`, `rdf:Property`⟩ is legitimate in RDF and can be interpreted (it indeed means that `rdf:type` denotes a predicate). This is achieved by interpreting triples in two steps: a first step associates a denotation to each IRI used and a second step interprets those used in predicate position as binary relations. This is the main specificity of the RDF semantics with respect to that of first-order logics.

Some RDF graphs may be translated as formulas in a positive (without negation), conjunctive, existential and function-free first-order logic. To each triple $\langle s, p, o\rangle$ corresponds the atomic formula $p(s, o)$, such that p is a predicate name, and o and s are constants if these elements are IRIs or literals, and variables otherwise. A graph is translated as the existential closure of the conjunction of atomic formulas associated to its triples. Hence, the graph of Fig. 1 is translated by:

$$\exists ?b1, ?b2, ?b3;\ \texttt{rdf:type}(?b1, \texttt{foaf:Person}) \wedge \texttt{rel:daughter}(?b1, ?b2) \wedge \texttt{rel:worksWith}(?b2, ?b3) \wedge \texttt{foaf:name}(?b3, "Pierre") \wedge \texttt{foaf:knows}(?b3, ?b1)$$

The models of such a formula are isomorphic to those of the direct semantics of graphs. This logical translation of RDF illustrates the proximity with other representations: logics, of course, as well as positive Datalog (see chapter "Databases and Artificial Intelligence" of this Volume) or conceptual graphs (see chapter "Reasoning with Ontologies" of Volume 1).

RDF is a simple and stable language. It fulfils adequately its function on the semantic web. It benefits from a good software support in many programming languages.

RDF 1.1 has acknowledged the notion of *RDF Dataset* which is a collection of RDF graphs and that of *named graphs* which are graphs identified by IRIs. This identification may be exploited to assert statements (triples) about graphs and hence triples. Such statements may be about the origin of the graph or the confidence in

what it asserts. This function was previously fulfilled by '*quads*', i.e., triples which were usually tagged by a IRI.

It has also embraced various formats used for expressing RDF graphs: beyond RDF/XML and n-triples. The new well-defined formats are RDFa —enabling the embedding of RDF within HTML—, Turtle, N-Quads —extending ntriples by quads—, and JSON-LD. These formats are usually supported by RDF data management systems, so-called *triple stores*.

2.2 The Web of Data

In 2006, from the observation that the semantic web development was lagging due to the lack of resources, Tim Berners-Lee put forth a methodology for publishing large quantities of data on the web so that they can help the semantic web (Berners-Lee 2006).

This defined the '*web of data*' through four principles for publishing data on the web, summarised as (Berners-Lee 2006; Heath and Bizer 2011):

1. Resources are identified by IRIs.
2. IRIs are dereferenceable, i.e., can be looked up on the web.
3. When a IRI is dereferenced, a description of the identified resource should be returned, ideally adapted through content negotiation.
4. Published web data sets must contain links to other web data sets.

Although not explicitly specified, linked data sources are more usable if they are published with semantic web technologies: IRIs for identifying resources, RDF for describing them, OWL for defining the used vocabularies, SPARQL for accessing data, and asserting links between sources through the `owl:sameAs` predicate.

The 'five stars rating', to be interpreted as incremental steps, has been introduced for measuring how much of this usability is achieved, namely:

⋆ Publish data on the web in any format, e.g., a scan of a table in PDF;

⋆⋆ Use structured data formats, e.g., a table in Excel instead of its scan in PDF;

⋆⋆⋆ Use non-proprietary formats, e.g., CSV instead of Excel, such that users have direct access to the raw data;

⋆⋆⋆⋆ Use universal formats to represent data, such as RDF, which encapsulates both syntax and semantics;

⋆⋆⋆⋆⋆ Link data to other data sets on the web, thereby providing context.

The first three stars are easy to reach and these already enable some data reuse. However, humans still have to handle all the semantic issues related to integration. In

```
SELECT ?n, ?z
FROM Figure 1
WHERE {
     ?x rel:daughter ?y .
     ?y rel:worksWith ?z .
     ?z foaf:knows ?x .
     OPTIONAL { ?z foaf:name ?n }
}
```

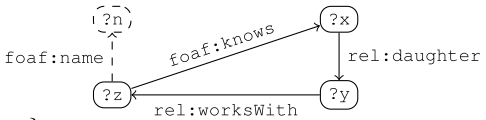


Fig. 2 A SPARQL query and the corresponding graph pattern (the optional part is displayed in dashed lines)

order to have data that is more easily discoverable and interoperable, it is necessary to reach the fourth and the fifth stars.

The web of data is thus a huge RDF graph reachable through the HTTP protocol like the web. Very quickly it developed around DBpedia (Bizer et al. 2009), a massive extraction of Wikipedia in RDF. Many efforts linked their resources to DBpedia. Soon after, several governments encouraged the publication of their (open) data in RDF. Since then, libraries, museums, research institutions of various kinds joined the linked open data cloud. In 2016, it contained 9960 data sources providing more than 150 billion triples using 576 vocabularies or ontologies.[2]

2.3 *Querying RDF with SPARQL*

The RDF recommendation (Hayes and Patel-Schneider 2014) defines the RDF semantics, so that developers can assess the soundness and completeness of their inference mechanisms. Such a mechanism is however discussed in W3C documents in direct relation with simple conceptual graphs (see chapter "Reasoning with Ontologies" of Volume 1). RDF consequence can be tested by looking for a labelled graph homomorphism. Although finding a homomorphism is an NP-complete problem, efficient algorithms are available for implementing it.

A major way to deal with RDF graphs consists of answering structured queries, as opposed to testing if a formula is a consequence. SPARQL (Prud'hommeaux and Seaborne 2008; Harris and Seaborne 2013) is the recommended query language for RDF. It is inspired by the SQL relational data base query language. It returns the values of variables that make a graph pattern consequence of a given graph. For instance, the query of Fig. 2 returns the answer: $\{\langle$"$Pierre$", _:b3$\rangle\}$ from the graph of Fig. 1.

SPARQL is based on the graph pattern concept (RDF graphs containing variables) composed through classical operators (conjunction, disjunction, optional conjunction, filtering). The answers are provided by graph homomorphisms between the queried graph and the graph patterns. These homomorphisms are then com-

[2]Sources: http://stats.lod2.eu/ and http://lov.okfn.org.

bined by operators reflecting the semantics of the composition, like in SQL (Pérez et al. 2009). SPARQL is thus a hybrid language between conceptual graphs (see chapter "Reasoning with Ontologies" of Volume 1) and data bases.

Beside the SELECT query form, it is possible to use a CONSTRUCT query form which returns a graph. As relational SELECT queries extract a relation from relations, a SPARQL CONSTRUCT query extracts a graph from graphs.

The following query is a rewriting of that of Fig. 2 as a CONSTRUCT query: instead of returning a set of homomorphisms (or assignments to the SELECT variables which satisfy the pattern), it will return an RDF graph made of the CONSTRUCT graph pattern whose variables are instantiated with the resulting homomorphisms.

```
CONSTRUCT {
    ?z rdf:knowsTheFatherOf ?y .
    ?z foaf:name ?n }
FROM Fig. 1
WHERE {
    ?z foaf:knows/rel:daughter ?y .
    ?y rel:workWith+ ?z .
    OPTIONAL { ?z foaf:name ?n }
}
```

The query also uses paths which have been introduced in SPARQL 1.1. Triple patterns may use regular path, such as the sequence of `foaf:knows` and `rel:daughter` or the unbounded sequence of `rel:workWith` predicate instead of atomic predicates. This leads to suppress any unnecessary variables from the initial graph pattern.

SPARQL 1.1 (Harris and Seaborne 2013) also introduced the opportunity to combine queries and sources. In particular, queries can integrate subqueries, evaluate a query part against a specific named graph (with the FROM NAMED and GRAPH constructs), and evaluate federated queries, i.e., queries against several data sources whose results are combined (with the SERVICE construct).

2.4 Beyond SPARQL: Streams and Navigation

Many applications in sensor networks, smart cities or the web of things are generating streams of RDF triples or quads, i.e., in this context, RDF triples labelled by their occurrence time. One may want to query such streams for detecting particular events occurring and alerting some monitor about it. Because the expressiveness of triples is very poor, these queries rely on graph patterns. The difficulty is that applications should be able to deal quickly with large amounts of real time data. For that purpose, they can take advantage of techniques defined in artificial intelligence for rule compilation (RETE, TREAT), as well as more sophisticated techniques for recognising chronicles (Dousson et al. 1993). This has to be combined with time windowing techniques matching events only related to closer events.

The C-SPARQL language has been designed for that purpose (Barbieri et al. 2010) and theoretical studies for stream reasoning are under development (Beck et al. 2015).

The path facilities of SPARQL 1.1 are a good way to specify navigational queries and to evaluate queries against the semantic web in a '*follow your nose*' fashion, i.e., by dereferencing reachable IRIs and querying their own data sources. This querying mode is well adapted to the web of data (Sect. 2.2).

However, SPARQL queries are usually evaluated against one particular graph specified in the FROM or GRAPH clauses. The new federated query feature also requires that graph patterns be evaluated against one specific server. Thus the SPARQL recommendation does not specify the semantics of evaluating queries against the web. Moreover, the availability of the web is unpredictable and the evaluation would require, in principle, to look-up a potentially infinite number of data sources. To better specify the semantics of such queries evaluated over the web, *reachability-based* and *context-based query semantics* have been defined (Hartig and Pirró 2016). They characterise the set of answers to a query by answers with respect to reachable data sources or those selected from the IRIs instantiating the paths. They also identify *web-safe queries* which can be evaluated according to the new semantics and return complete answers.

3 A Little Knowledge Representation Goes a Long Way

The title of this section is an adaptation of James Hendler's catch phrase 'A little semantics goes a long way'.

Once able to express data on the web, the next problem is the definition of the vocabulary used in the RDF graphs. This requires listing the terms used to describe data as well as expressing the axioms constraining the use of such terms allowing a machine to interpret them properly. Such vocabularies are defined as *ontologies* (see chapters "Reasoning with Ontologies" and "Knowledge Engineering" of Volume 1). We will consider below ontologies as sets of axioms in a specific logic. Several such logics have been defined.

3.1 RDFS

RDFS (for RDF Schema (Brickley and Guha 2004)) extends the RDF language by more precisely describing the resources used for labelling the graphs. In RDFS, this extension is seen as the introduction of a specific vocabulary (a set of IRIs) carrying a precise semantics. RDF already introduced a few keywords in the `rdf:` name space for structuring knowledge:

- ⟨ex:Sonia rdf:type ex:Employee⟩ asserts that the resource ex:Sonia is an instance of the class ex:Employee;
- ⟨rel:worksWith rdf:type rdf:Property⟩ tells that rel:worksWith is a predicate, i.e., a resource used for labelling edges.

RDFS is expressed as RDF triples using a few more keywords in the rdfs: name space, such as:

- ⟨ex:Employee rdf:type rdfs:Class⟩ indicating that the resource ex:Employee as for type rdfs:Class, it is thus a class.
- ⟨ex:Employee rdfs:subClassOf foaf:Person⟩ meaning that ex:Employee is a subclass of foaf:Person, thus each instance of ex:Employee is also an instance of foaf:Person, like ex:Sonia;
- ⟨ex:worksWith rdfs:range ex:Employee⟩ asserts that any resource used as the extremity of an edge labelled by rel:worksWith is an instance of the ex:Employee class.

The RDFS semantics extends that of RDF by considering some IRIs as identifying classes and interpreting them in two steps, as properties (Hayes and Patel-Schneider 2014). ρDF (Pérez et al. 2009) is the subset of RDFS that concentrate on the purely ontology description features.

However, RDFS only provides limited mechanisms to specify classes further. Its expressiveness is limited to that of frame languages, hence other languages have been introduced to address more demanding needs.

3.2 OWL: Description Logics on the Web

The OWL language (Horrocks et al. 2003; Dean and Schreiber 2004; Beckett 2009) is dedicated to class and property definitions. Inspired from description logics (see chapter "Reasoning with Ontologies" of Volume 1), it provides constructors to constrain them precisely. The OWL syntax, based on RDF, introduces a specific vocabulary in the owl: name space, but contrary to RDFS, this is not sufficient to define the full OWL syntax, such as the owl:Restriction imbrication presented below. Hence, not all RDF graphs using this vocabulary are necessarily valid OWL ontologies: further constraints have to be satisfied.

The semantic of OWL constructors is defined through model theory (Patel-Schneider et al. 2004; Motik et al. 2009c) and directly follows description logics. It is more precise than the semantics assigned to the RDF graphs representing OWL ontologies: more can be deduced from the ontology.

OWL allows one to finely describe relations, e.g., the relation daughter, is here defined as a subrelation of the inverse of hasParent whose codomain is of female gender:

```
<owl:ObjectProperty rdf:about="rel:daughter">
  <owl:subPropertyOf>
```

```
    <owl:ObjectProperty>
      <owl:inverseOf rdf:resource="rel:hasParent"/>
    </owl:ObjectProperty>
  </owl:subPropertyOf>
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="foaf:gender"/>
      <owl:hasValue>female</owl:hasValue>
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>
```

We provide the main OWL constructors and their intuitive semantics (see Motik et al. 2009b for a formal definition):

- RDF keywords (`rdf:type`, `rdf:Property`) and RDFS' (`rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:range`, `rdfs:domain`) are used with the same semantics.
- `owl:Class` is a new (meta) class.
- `owl:sameAs` and `owl:differentFrom` are used to assert that two resources are equal or different.
- `owl:inverseOf` asserts that a property p is the converse of a property p' (in this case, the triple $\langle s\ p\ o\rangle$ entails $\langle o\ p'\ s\rangle$); other characteristics may be assigned to properties such as reflexivity (`owl:ReflexiveProperty`), transitivity (`owl:TransitiveProperty`), symmetry (`owl:SymmetricProperty`) or functionality (`owl:FunctionalProperty`).
- `owl:allValuesFrom` associates a class c to a relation p. This defines the class of objects x such that if $\langle x\ p\ y\rangle$ holds, then y belong to c (this is a universally quantified role in description logics). `owl:someValuesFrom` encodes existentially quantified roles.
- `owl:minCardinality` (resp. `owl:maxCardinality`) defines the class of objects related to at least (resp. at most) a specific number of objects through a given property. A qualified version of these constructors constrains, in addition, that these objects belong to a particular class.
- `owl:oneOf` defines a class in comprehension by enumerating the set of its instances.
- `owl:hasValue` constrains a property to have a particular individual as value.
- `owl:disjointWith` asserts that two classes cannot have a common instance.
- `owl:unionOf`, `owl:intersectionOf` and `owl:complementOf` define a class as the disjunction, the conjunction or the negation of other classes.
- `owl:hasSelf` defines the class of objects related to themselves through a specific relation.
- `owl:hasKey` asserts that a set of properties is a key for a class, i.e., that two distinct instances cannot share the same values for these properties.
- `owl:propertyChainAxiom` composes several relations and properties to obtain a new relation or property.

We have not mentioned all constructors. Many of them can be trivially implemented by using the cited ones, e.g., `owl:equivalentClass` asserting that two classes are equivalent can be expressed with two `rdfs:subClassOf` assertions. OWL also uses data types that are not considered here. They are however important as they can lead to inconsistency.

OWL 2 interprets differently and independently a IRI if it has to be considered as a class or a property. This constraint does not apply to OWL full.

3.3 Expressiveness/Efficiency Trade-Off

The expressiveness/efficiency trade-off has always been present in the work on description logics (see chapter "Reasoning with Ontologies" of Volume 1). It is possible to adopt a unified encoding of description logics preserving the opportunity to choose the one corresponding to a particular trade-off (Euzenat and Stuckenschmidt 2003). This attitude has been adopted in OWL 1 (Dean and Schreiber 2004) through only three languages:

- OWL Lite covers a subset of the available constructors and their use may be restricted, e.g., cardinality constraints can only use 0 or 1. It corresponds to the $\mathcal{SHIF}$ description logics for which optimized reasoners based on tableau methods have been implemented and used in practice with a reasonable performances, despite that the reasoning problems such as subsumption checking or satisfiability checking are known to be ExpTime-complete (Baader et al. 2003).
- OWL DL extends OWL Lite by covering all constructors including number restrictions, but enforcing specific constraints, e.g., a resources cannot be both a class and a property. It corresponds to the $\mathcal{SHOIN}$ description logics for which the reasoning problems are decidable but with high complexity (NExpTime-complete). Nevertheless, like for OWL-Lite, practical reasoners such as FaCT++ (Tsarkov and Horrocks 2006), Pellet (Sirin et al. 2007) or HermiT (Glimm et al. 2014) have acceptable performances on most OWL DL ontologies built in practice.
- OWL Full is a semantic extension of both OWL DL and RDF(S) and cannot be translated into a description logic language. In addition, the fact that arbitrary roles can be used in number restrictions would make entailment (and thus subsumption) in the underlying logic undecidable (Horrocks et al. 2000).

However, the various usages of these languages may have led to a different organisation. For instance, some applications may be more easily dealt with a version of OWL DL without disjunction, with intuitionistic negation and such that classes may be considered as instances. Such a language does not exist because OWL has not been defined modularly.

Hence, some languages have been studied independently from the W3C in order to offer efficient reasoning services. First, description logic research has proposed languages for answering efficiently to queries modulo ontologies. DL-Lite (Calvanese et al. 2007; Artale et al. 2009) is a subfamilly of OWL which contains no disjunction,

limited constraints on properties and a restricted negation (see chapter "Reasoning with Ontologies" of Volume 1). In consequence, conjunctive queries in DL-Lite may be evaluated efficiently by rewriting them into SQL queries and using existing database management systems. This family features a language, DL-Lite$_{\mathcal{R}}$, more expressive than ρDF, for which reasoning is tractable.

Another approach (Voltz 2004; Hustadt et al. 2007) consists of determining a subset of OWL whose expressiveness is lower than that of OWL Lite and to complement it with a rule language in such a way that its complexity remains polynomial.

OWL 2 integrated these approaches by introducing the notion of language profiles (Motik et al. 2009a). In addition to the languages of OWL 1, it provides profiles that correspond to tractable description logics in which subsumption checking can be performed in polynomial time:

OWL 2 EL suppresses all universal quantification from OWL DL, i.e., it forbids universal quantification, cardinality restrictions, disjunction, negation and relation properties as well as inverse role, but it preserves role composition. If the ontology is made only of concept definitions, i.e., if general concept inclusions are forbidden, the EL knowledge bases correspond to simple conceptual graphs (see chapter "Reasoning with Ontologies" of Volume 1). General concept inclusions in the ontology correspond to existential rules that have been studied both in Datalog$^{+}_{-}$ (Calì et al. 2009a) and as an extension of conceptual graphs with rules (see chapter "Reasoning with Ontologies" of Volume 1);

OWL 2 QL corresponds to DL-Lite$_{\mathcal{R}}$;

OWL 2 RL suppresses any construction which may generate unknown objects and direct the axioms so that, only some expressions (existential) are

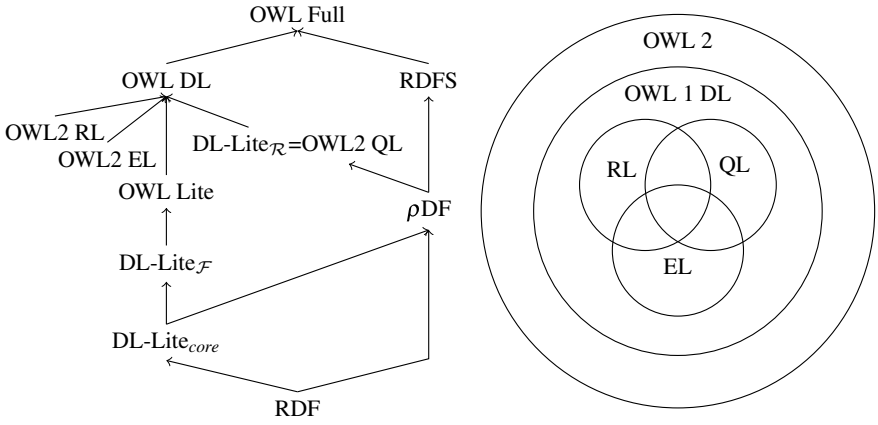


Fig. 3 A hierarchy of ontology languages for the semantic web. Arrows and set inclusion represent the expressibility order between these different languages

authorised in the superclass and others (universals) are expressed in subclass of a `rdfs:subClassOf` axiom.

Figure 3 offers two views of the relations between the main languages defined around OWL and RDFS.

3.4 Reasoning

Reasoning is at the core of many tasks raised by semantic web applications, such as data interlinking, answering queries through ontologies, data consistency checking. Efficient reasoning tools is a decisive criterion for the adoption of a particular language for expressing ontological constraints. Such tools may be embedded in query systems in order to process sophisticated queries and link and enrich the returned answers (see Sect. 3.5).

However, the problem of determining if an RDF graph is the consequence of another is already an NP-complete problem. Efficient algorithms have been developed to compute the graph homomorphisms which correspond to this problem, based on backtracking methods pioneered in constraint networks (see chapter "Constraint Reasoning" of Volume 2).

In the semantic web, several factors make the problem more difficult:

- One may want to achieve reasoning on the whole semantic web, i.e., all the RDF and OWL information available on the web (see Sect. 2.4).
- These graphs and the reasoning to be performed depend on ontologies for which reasoning problems may be of high complexity or even undecidable.
- Last, but not least, this information is distributed and heterogeneous.

Research on reasoning with the semantic web languages are being pursued in several directions:

- developing efficient decision procedures for the underlying description logics (Horrocks and Sattler 2007; Motik et al. 2009d), and
- isolating OWL fragments for which efficient procedures are known.

In the semantic web setting, the size of the ontologies is usually much smaller than the size of the data described using these ontologies. Therefore, it is of utter importance to distinguish reasoning tasks concerning the ontological part only (such as subsomption test or query rewriting) from reasoning tasks involving data (such as instance checking or answering queries). In the latter case, measuring the data complexity is more meaningful for evaluating the practical performances of the reasoning algorithms than measuring the combined complexity that takes into account the size of the ontology and possibly the size of the query in addition to the size of the data. For instance, answering conjunctive queries by query rewriting in DL-Lite has an exponential combined complexity in the worst-case but its data complexity is in AC_0, a highly parallelizable subclass of LOGSPACE, just as in standard relational data bases from which we can take advantage of all the optimizations available.

3.5 *Querying Modulo Ontologies*

The definition of SPARQL through graph homomorphisms does not make it easy to apply it with other semantics than that of simple RDF graphs. Indeed, without correspondence results between consequence and homomorphisms for such languages, query evaluation is disconnected from their semantics. This happens if one queries RDF graphs without taking into account their RDFS Schema or OWL ontologies. SPARQL could have been grounded on the notion of logical consequences instead of graph homomorphism (Alkhateeb et al. 2009). With such a foundation, any ontology language using model theory for expressing its semantics could be used for interpreting queries.

Various attempts have been made to deal with this situation. For instance, it is possible to answer a SPARQL query with respect to an ontology written in ρDF or DL-Lite through simple rewriting of the query as a PSPARQL (Alkhateeb et al. 2008), SPARQL 1.1 or SQL query. Similar results have been obtained by restricting SPARQL itself (Pérez et al. 2010). Similarly, the SPARQL-DL language (Sirin and Parsia 2007) evaluates a fragment of SPARQL (conjunctive queries with limited use of variables) with respect to an OWL DL ontology.

SPARQL 1.1 introduced the notion of SPARQL entailment regime (Glimm and Ogbuji 2013) which replaces graph homomorphism by the notion of logical entailment. Hence, the queried data may be interpreted under OWL or RDFS semantics among others. It also introduces further constraints on the types of authorised query pattern, expected answers, or what to do in case of inconsistency.

Independently from SPARQL, other works have extended Datalog to deal with queries written in DL-Lite$_{\mathcal{R}}$ (hence in ρDF) (Calì et al. 2009b) or studied the complexity of such queries when introducing the SPARQL disjunction operator (Artale et al. 2009). This work has sparked the new domain of ontology-based data access (see chapter "Reasoning with Ontologies" of Volume 1) in which such an ontology is used for extracting more conveniently data from relational data bases. This, in particular, contributes exposing traditional data bases as SPARQL endpoints.

3.6 *Rules*

Concerning rules, the situation is less clear than for other semantic web languages. The W3C has recommended a '*rule interchange format*' (RIF Boley et al. 2013) whose goal is to offer both logic programming rules (chapter "Logic Programming" of Volume 2) and '*business rules*', a kind of rule triggering an action when a condition is satisfied. Two dialects, sharing part of their syntax (RIF-Core), have been defined: BLD for '*Basic Logic Dialect*' and PRD for '*Production Rule Dialect*'. RIF has a syntax different from those provided by other languages. In particular, the RDF syntax of RuleML and SWRL (Horrocks et al. 2004) has not been retained for RIF.

From a theoretical standpoint, all work developed in logic programming applies naturally to the semantic web and can be transposed in the RIF-BLD dialect. Moreover, some connection between rules and ontologies have been specified (de Bruijn and Welty 2013).

Combining description logics and Datalog has been studied (Levy and Rousset 1998). In the semantic web context, work has started from less expressive fragments: DLP ('*description logic programs*' (Grosof et al. 2003)) is a minimal language combining Horn clauses and description logics. It cannot express all OWL Lite, nor all logic programming. However, it could be adopted to reason at large scale. DLP has been the inspiration for OWL 2 RL.

A closer integration of description logics (chapter "Reasoning with Ontologies" of Volume 1) and of '*answer set programming*' (chapter "Logic Programming" of Volume 2), which reintroduces aspects such as negation-as-failure and the closed world assumption, has also been considered (Eiter et al. 2008; Motik and Rosati 2010). Other works have attempted the definition of description logics on a logic programming language basis (Hustadt et al. 2007, 2008).

From the conceptual graph side, logical rules have been studied as an extension of simple conceptual graphs (see chapter "Reasoning with Ontologies" of Volume 1, Baget et al. 2011) which can be directly applied to RDF given the proximity of these formalisms.

3.7 Robust Inference

A typical property of the web is the huge quantity of available information. The semantic web is no exception. Unfortunately, it also contains erroneous, outdated, redundant or incomplete pieces of information. A human user is, in general, able to detect these problems and to overcome them. It is thus necessary to develop reasoning modes taking the semantic web into account, i.e., which remain as faithful as possible to the recommended languages, without being disrupted by errors found in the data sources. In summary, robust reasoners are needed.

Various techniques may be adapted to the semantic web context to achieve robustness: paraconsistent logics or statistical models, intuitionistic logics, etc. Some proposals use non monotonic logics (Phrer et al. 2010) to counter inconsistency. Anytime reasoning or reasoning under resource constraints might be used to overcome the huge size of the web. However, this aspect is not very developed.

Other works aim to reduce the impact of errors in ontologies. They only take into account axioms that extend an ontology by defining new concepts and not those that constrain the meaning of existing ones (Hogan et al. 2009). This is a radical option to obtain conservative extensions (see Sect. 4.3).

4 Dealing with Heterogeneity

The semantic web being grounded in the same principles of autonomy and independence as the web, ontologies used by various sources of knowledge may be different. It is thus necessary to reconcile them and to express relations between these ontologies.

4.1 From Alignments to Networks of Ontologies

The first problem is the description of relations between data or ontologies, i.e., that an individual in a data source is the same as another in another source or that a class in one ontology is more general than another class in another ontology. In general, such a relation, called a correspondence expresses the relation between two entities of two different ontologies. For instance, one may want to express that the property `daughter` is more specific than the property `child`:

$$\langle \texttt{rel:daughter}, \sqsubseteq, \texttt{my:child} \rangle$$

Such correspondences, when they are generated by ontology matching systems are often assigned a confidence measure.

However, correspondences may go beyond the ontology language. This is especially useful when the ontology languages are not very expressive, e.g., SKOS. For instance, the former correspondence could have been expressed as a daughter to be equivalent to the conjunction of the converse of the parent relation whose domain is Female:

$$\langle \texttt{rel:daughter}, \equiv, \texttt{my:parent}^{-1} \sqcap \texttt{domain}(\texttt{my:Female}) \rangle$$

The EDOAL language (David et al. 2011) provides description logics constructors within the alignment language and can thus express such correspondences. It offers four types of features:

- Constructions of entities from other entities can be expressed through algebraic operators (here $\sqcap$).
- Restrictions can be expressed on entities in order to narrow their scope (here `domain(my:Female)`).
- Transformations of property values can be specified, e.g., concatenation of first and last name to give the name or conversion from m/s to miles-per-hour.
- Link keys can be defined for expressing conditions under which, instances of the aligned entities should be considered equivalent (see Sect. 4.5).

A set of such correspondences between the two same ontologies is called an alignment (Euzenat and Shvaiko 2013). More generally, it may be necessary to express networks of ontologies involving several ontologies and alignments between these.

The Distributed Ontology Language (DOL) (Mossakowski et al. 2015) has been designed for the purpose of dealing in a unified manner with ontologies defined in different languages. Such languages are considered in the categorical formalisms of institutions and operations are defined to use them together. In the DOL framework, alignments and networks of ontologies are syntactic objects. Alignments are interpreted as ontologies which relate two other ontologies. DOL is becoming an Object Management Group standard and is supported by the Hets tool.

4.2 Semantics of Alignments

There exists a consensus on the necessity to express alignments between ontologies. However, contrary to ontology languages, there is no recommendation for expressing alignments, networks of ontologies and their semantics.

A first solution is simply to use OWL to express ontologies and alignments. This is called the reduced semantics (Euzenat and Shvaiko 2013). OWL offers the `owl:sameAs` relation to express that two individuals are the same. This is particularly used in the web of data (Sect. 2.2) for expressing link sets very often kept separated from the data sets. It also offers `owl:equivalentClass` and `rdfs:subClassOf` to express the equivalence or subsumption between two classes (the same exists for properties). However, using OWL in this context constrains to merge data sources in order to exploit them: the alignments between ontologies do not exist on their own.

The same remarks apply to the SKOS language (Miles and Bechhofer 2009) designed for expressing thesauri and which offers relations between terms of different thesauri, such as `skos:broadMatch` and `skos:exactMatch`.

Several proposals have been made to preserve this ontology/alignment duality. C-OWL (Bouquet et al. 2004) is a language for expressing oriented alignments, called mappings, between ontology expressed in OWL. A correspondence expresses how the information from one ontology may be imported to another ontology, but not the other way. Its originality is that networks of ontologies are considered as such and their semantics is provided on top of OWL semantics preserving the locality of each ontology (contrary to the previous solution). The semantics of C-OWL is grounded on that of distributed description logics (Borgida and Serafini 2003).

The equalising semantics (Zimmermann and Euzenat 2006) is an alternative to the distributed semantics which does not require that the ontologies be expressed in OWL. It also preserves the interpretations of the various ontologies and projects these interpretations in a common domain of interpretation in which the alignment relations can be interpreted. Global reasoning on networks of ontologies, e.g., composing alignments, can thus be performed.

The DOL framework (Mossakowski et al. 2015) supports these three semantics, and the heterogeneity of ontological formalisms, at least on simple alignments.

4.3 Reasoning in Networks of Ontologies

Once ontologies are connected through alignments, methods for reasoning with them have to be developed. This can concern several types of problems:

Consistency test: determining if a network of ontologies, or a sub-part of it, has a model;
Entailment test: determining if a statement or a correspondence is entailed by a network of ontologies;
Query answering: determining variable assignments that satisfy a formula.

Solving such problems often requires specific techniques. These are widely open problems which have found only few concrete developments. They raise, in particular, strong scalability problems.

First, most OWL reasoners are able to reason with imported ontologies. However, gathering the ontologies becomes less possible when data has to be considered in addition to axioms. Furthermore, this approach cannot be adopted when data is only accessible through SPARQL endpoints.

In the domain of semantic peer-to-peer systems, peers offer data expressed in their own ontologies related by alignments (usually sets of subsumption statements). SomeWhere (Adjiman et al. 2006) has developed an original approach to consistency testing and query answering. It uses propositional reasoning about the ontology structure in order to determine query evaluation plans, then it evaluates queries from peer to peer.

DRAGO (Serafini and Tamilin 2005) is a distributed reasoner for C-OWL. Ontologies are described in OWL DL. It is implemented as a distributed tableau reasoner in which the construction of a model for one peer may recursively require the satisfaction tests of formulas obtained through correspondences by other peers.

DRAOn (Zimmermann and Le Duc 2008) is a distributed reasoner determining the consistency of a network of ontologies in the equalising semantics (Zimmermann and Euzenat 2006). It works independently of the logics used in each peer: their only requirement is that they support the $\mathcal{ALC}$ logic. However, the alignment language is reduced to the subsumption and the disjunction of concepts and properties.

One of the important aspects of these three reasoners is that they operate in a decentralised manner: each peer is autonomous and has to communicate with others.

More localised attempts have concentrated on reasoning with only two aligned ontologies in order to repair them (Meilicke et al. 2009; Jimenez Ruiz et al. 2009). These approaches use the reduced semantics (Sect. 4.1) to identify correspondences leading to inconsistency or incoherence and return a coherent alignment. They typically choose a consistent sub-alignment satisfying some criterion, i.e., they suppress correspondences causing incoherence. Alcomo (Meilicke et al. 2009) can compute optimal repairs which minimally modify the alignment when several inconsistency sources are detected. LogMap (Jimenez Ruiz et al. 2009) decomposes equivalence correspondences into two subclass correspondences. Hence, it is possible to suppress only one of these when repairing. Alignment repair or debugging is directly

related to belief revision (see chapter "Belief Revision, Belief Merging and Information Fusion" of Volume 1). Revision techniques have been developed for networks of ontologies as well (Euzenat 2015).

The notion of conservative extension has been defined for description logics (Ghilardi et al. 2006; Lutz et al. 2007). It means that, when an ontology is extended, e.g., by connecting it to another ontology through an alignment, no additional statement using the signature of the initial ontology can be deduced. This somewhat defeats the purpose of networks of ontologies since their goal is rather to benefit from the knowledge exposed in other locations. However, it may be useful to ensure that ontologies are loosely connected self-sufficient modules. Moreover, reasoning can be implemented efficiently with conservative extensions since all formulas expressible with the vocabulary of an ontology can be derived from this ontology alone. Unfortunately, deciding if an ontology or a network, is a conservative extension of another ontology is hard and even non decidable for $\mathcal{ALCQIO}$ which is the heart of OWL DL (Ghilardi et al. 2006).

4.4 Ontology Matching

Ontology matching is the task of finding an alignment between two ontologies (Euzenat and Shvaiko 2013). Numerous algorithms have been designed to fulfil this task exploiting techniques developed in various domains, including artificial intelligence. Ontology matching approaches may be organised in:

- content-based approaches which rely on ontology content. These may be based on:
  - Labels used to name or comment the entities found in the ontologies. Techniques may be inspired from computational linguistics (chapter "Artificial Intelligence and Natural Language" of this Volume) or information retrieval (chapter "Artificial Intelligence and Information Retrieval" of this Volume).
  - Relationships between entities, and prominently the subsumption relation. Techniques from graph theory may be used.
  - Concept extensions, i.e., their instances when available. In this case, machine learning (chapter "Statistical Computational Learning" of Volume 1), data analysis or deduplication techniques are used.
  - Semantics of ontologies which may be used for expanding alignments or testing their consistency (Sect. 4.3). Then, techniques from automated reasoning (chapter "Automated Deduction" of Volume 2) and constraint programming (chapter "Constraint Reasoning" of Volume 2) may be used.
- context-based approaches which rely on relationships that the ontologies have with other resources. These may be:

– a corpus of documents annotated by one or both of the ontologies on which statistical machine learning (chapter "Statistical Computational Learning" of Volume 1) may be used;
– one or several ontologies used as background knowledge for deducing relations between entities through automated reasoning (chapter "Automated Deduction" of Volume 2);
– user feedback when providing positive or negative assessment on correspondences;
– the relation with specific resources such as DBPedia, WordNet or UMLS.

However, none of these approaches works satisfactorily in all situations. Hence, systems use several such techniques together. It is not possible to present these different types of systems (see Euzenat and Shvaiko 2013). We briefly consider below those using machine learning, a typical artificial intelligence technique.

Ontology matching is an inductive rather than a deductive task. There are no rules for deducing alignments, only heuristics to finding them. It is thus natural to reuse machine learning techniques to match ontologies. Two types of techniques may contribute: data mining (chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2), which isolates regularities in large quantity of data, and learning from example (chapter "Statistical Computational Learning" of Volume 1), which induces correspondence patterns from examples of correspondences.

Learning from examples is less frequent because, as mentioned, matching cannot easily be generalised. It can, however, be used in two cases: first, when a user is able to show some simple alignments from which the system can learn how to match, and, second, when there are references alignments from which it is possible to learn algorithm parameters which provide the best results (Ehrig et al. 2005).

Data mining algorithms infer the similarity between two concepts from the similarity between their instances, or the probability that the instances of one class be instances of another one. This may be based on symbolic techniques such as formal concept analysis (Stumme and Mädche 2001; Kalfoglou and Schorlemmer 2003) (chapter "Formal Concept Analysis: From Knowledge Discovery to Knowledge Processing" of Volume 2), decision trees (Xu and Embley 2003) or numeric techniques such as neural networks (Mao et al. 2010), Bayes networks (Pan et al. 2005), association rule extraction (David 2007) or a mix of such approaches (Doan et al. 2004).

4.5 Data Interlinking

Data interlinking consists of finding representations of the same entity in two (or more) data sets and linking these representation through the `owl:sameAs` predicate. The resulting triple is called a link and a set of links across the same two data sets a link set. It thus plays a very important role for the web of data (Sect. 2.2).

This task may be considered as very close to ontology matching: ontologies are replaced by data sets and alignments by link sets. Like ontology matching this is an inductive task. However, ontology matching is confronted with two ontologies from which it is difficult to extract regularities, though data interlinking faces huge data sets in which regularities can be found. Hence they use different types of techniques. These two activities are rather complementary: link sets may be exploited by extension-based matching techniques and alignments may guide the objects and features to compare within two data sets.

Data interlinking may be divided in two broad types of approaches:

- Similarity-based approaches use a similarity measure to compare entities and, if they are similar enough, they are considered as the same. The similarity either compares different properties of RDF descriptions or project them in a vector space in which similarity is computed.
- Key-based approaches isolate sufficient conditions, called keys or link keys, for two resources to be the same and use them to find same entities. Keys are an extension of relational data base keys which must be associated with an alignment if the data sources do not use the same ontologies (Symeonidou et al. 2014). Link keys express directly in the same structure identification constraints across ontologies (Atencia et al. 2014).

Both types of approaches may use machine learning or statistical (Capadisli et al. 2015) techniques in order to induce "linkage rules" exploiting similarity or link keys. If a sample of links is provided, supervised methods may be used to learn them; otherwise algorithms have to resort on a measure of the quality of the generated links (Atencia et al. 2014).

Linkage rules are used to generate the actual links. Link keys may be translated into SPARQL queries. Silk can express and process similarities to generate links against large data sets (Volz et al. 2009). LIMES also uses similarities in metric spaces and focusses on exploiting them efficiently (Ngonga Ngomo and Auer 2011). Linkage rules have also been expressed in probabilistic Datalog (Al-Bakri et al. 2016) or description logics (Gmati et al. 2016). Hence, reasoning can be interleaved with data interlinking.

5 Perspectives

Research on efficient inference with semantic web languages (OWL, SPARQL), is very active. It is at the crossroad between data bases, in which the huge amount of data rules, and knowledge representation and reasoning (see chapter "Automated Deduction" of Volume 2), in which expressiveness prevails. These traditional problems, never dealt with at such a scale, are combined with distributed system problems such as communication, latency and caches. It is thus an important source of problems for artificial intelligence, and specifically for automated reasoning. From the representation standpoint, semantic web languages are not very expressive and oriented

towards data expression. There exist work for providing non monotonic (Eiter et al. 2008) or fuzzy (Stoilos et al. 2010) versions of semantic web representations.

The semantic web has reached some kind of maturity. Its technologies are used in various domains and could be in more. Thus users are requesting for making semantic web technologies more reliable. Methods coming from programming languages are used for providing guarantees. This is the case, for instance, of work on SPARQL query containment which aims at ensuring that a query can have an answer or that a data source can indeed provide an answer to a query before actually evaluating the query (Chekol et al. 2012). There are also work on supporting debugging of knowledge expressed on the web, such as pinpointing used to narrow down the search for inconsistency causes in ontological reasoning or alignment repair (Schlobach 2005).

Another evolution of the semantic web goes out of the strict realm of artificial intelligence: it is the involvement of users and society as a whole. This trend has now pervaded all computing activities, in particular, through crowdsourcing various kinds of tasks. This applies to query systems in which users can provide feedback on the quality of answers and interactive ontology matching in which matchers ask users for specific help.

Finally, the success of the semantic web should trigger work on its evolution. Indeed, a large, uncontrollable mass of knowledge used directly by machines cannot be stopped for maintenance. It has to continuously evolve without hiccup. Hence specific mechanisms should be designed in order to support coherently evolving ontology, data, alignments and links.

6 Conclusion

As the web, the semantic web is not a well-defined object. Both may be reduced to the technologies of which they are made: URL, HTTP, HTML on the one hand, IRI, RDF, OWL, SPARQL and alignments on the other hand. In both cases, such a description would miss a lot.

Although the semantic web has initially been designed around these technologies, as time passes, they are used to encompass more diverse objects: from minimal RDF snippets to encyclopaedic data bases, from simple schemas to foundational ontologies, from path queries against the web to elaborate queries against triple stores, from tiny devices to large back-ends. Embracing all these situations threatens the unity of semantic technologies. It is thus remarkable that the semantic web ecosystem remains grounded on these same technologies.

For artificial intelligence research, the semantic web is a wonderful application field. As we tried to show it, many techniques originating from knowledge representation (chapter "Reasoning with Ontologies" of Volume 1), automated reasoning (chapters "Automated Deduction" and "Reasoning with Propositional Logic - From SAT Solvers to Knowledge Compilation" of Volume 2), constraint programming (chapter "Constraint Reasoning" of Volume 2), and machine learning

(chapter "Statistical Computational Learning" of Volume 1) are contributing to the semantic web. Another topic not discussed in this chapter is the filling of the semantic web through data and knowledge acquisition using techniques from knowledge engineering (chapter "Knowledge Engineering" of Volume 1), computational linguistic (chapter "Artificial Intelligence and Natural Language" of this Volume) and machine learning (chapter "Statistical Computational Learning" of Volume 1).

This is a demanding application field: the scale is gigantic, users are difficult. Moreover, artificial intelligence techniques are confronted with specific constraints. For instance, the distribution of resources over the network requires specific reasoning and query evaluation techniques and soon social constraints may force new developments.

This is also a gratifying application field: each technique which will find a place in semantic web technologies will have millions of users.

References

Abiteboul S, Manolescu I, Rigaux P, Rousset MC, Senellart P (2011) Web data management. Cambridge University Press, Cambridge

Adjiman P, Chatalic P, Goasdoué F, Rousset MC, Simon L (2006) Distributed reasoning in a peer-to-peer setting: application to the semantic web. J Artif Intell Res 25:269–314

Al-Bakri M, Atencia M, David J, Lalande S, Rousset M (2016) Uncertainty-sensitive reasoning for inferring sameAs facts in linked data. In: Proceedings of the 22nd European conference on artificial intelligence (ECAI), The Hague (NL), pp 698–706

Alkhateeb F, Baget JF, Euzenat J (2008) Constrained regular expressions in SPARQL. In: Arabnia H, Solo A (eds) Proceedings of the international conference on semantic web and web services (SWWS), Las Vegas (NV US), pp 91–99

Alkhateeb F, Baget JF, Euzenat J (2009) Extending SPARQL with regular expression patterns (for querying RDF). J Web Semant 7(2):57–73

Antoniou G, van Harmelen F (2008) A semantic web primer, 2nd edn. The MIT Press, Cambridge

Artale A, Calvanese D, Kontchakov R, Zakharyaschev M (2009) The DL-Lite family and relations. J Artif Intell Res 36:1–69

Atencia M, David J, Euzenat J (2014) Data interlinking through robust linkkey extraction. In: Proceedings of the 21st European conference on artificial intelligence (ECAI), Prague (CZ), pp 15–20

Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) (2003) The description logic handbook: theory, implementation, and applications. Cambridge University Press, Cambridge

Baget J, Leclère M, Mugnier M, Salvat E (2011) On rules with existential variables: walking the decidability line. Artif Intell 175(9–10):1620–1654

Barbieri DF, Braga D, Ceri S, Della Valle E, Grossniklaus M (2010) C-SPARQL: a continuous query language for RDF data streams. Int J Semant Comput 4(1):3–25

Beckett D (2009) OWL 2 web ontology language document overview. Recommendation, W3C. http://www.w3.org/TR/owl2-overview/

Beck H, Dao-Tran M, Eiter T, Fink M (2015) LARS: a logic-based framework for analyzing reasoning over streams. In: Proceedings of the 29th conference on artificial intelligence (AAAI), Austin (TX US), pp 1431–1438

Berners-Lee T (1998) Design issue: what the semantic web can represent. http://www.w3.org/DesignIssues/RDFnot.html

Berners-Lee T (2006) Design issue: linked data. http://www.w3.org/DesignIssues/LinkedData
Berners-Lee T, Fielding R, Masinter L (1998) Uniform resource identifiers (URI): generic syntax. RFC 2396, IETF. http://www.ietf.org/rfc/rfc2396.txt
Bizer C, Lehmann J, Kobilarov G, Auer S, Becker C, Cyganiak R, Hellmann S (2009) DBpedia: a crystallization point for the web of data. J Web Semant 7(3):154–165
Boley H, Hallmark G, Kifer M, Paschke A, Polleres A, Reynolds D (2013) RIF core dialect. Recommendation, W3C. http://www.w3.org/TR/rif-core/
Borgida A, Serafini L (2003) Distributed description logics: assimilating information from peer sources. J Data Semant I:153–184
Bouquet P, Giunchiglia F, van Harmelen F, Serafini L, Stuckenschmidt H (2004) Contextualizing ontologies. J Web Semant 1(1):325–343
Brickley D, Guha R (2004) RDF vocabulary description language 1.0: RDF schema. Recommendation, W3C. http://www.w3.org/TR/rdf-schema/
Calì A, Gottlob G, Lukasiewicz T (2009a) A general datalog-based framework for tractable query answering over ontologies. In: International conference on principles of database systems (PODS), pp 77–86
Calì A, Gottlob G, Lukasiewicz T (2009b) A general datalog-based framework for tractable query answering over ontologies. In: Proceedings of the 28th ACM principle of database systems conference (PODS), Providence (RI US), pp 77–86
Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J Autom Reason 39(3):385–429
Capadisli S, Auer S, Ngonga Ngomo A (2015) Linked SDMX data: path to high fidelity statistical linked data. Semant Web 6(2):105–112
Carothers G, Seaborne A (2014) RDF 1.1 N-triples – a line-based syntax for an RDF graph. Recommendation, W3C. http://www.w3.org/TR/n-triples/
Chekol M, Euzenat J, Genevès P, Layaïda N (2012) SPARQL query containment under SHI axioms. In: Proceedings of the 26th American national conference on artificial intelligence (AAAI), Toronto (ONT CA), pp 10–16
Cyganiak R, Wood D, Lanthaler M (2014) RDF 1.1 concepts and abstract syntax. Recommendation, W3C. http://www.w3.org/TR/rdf11-concepts/
David J, Guillet F, Briand H (2007) Association rule ontology matching approach. Int J Semant Web Inf Syst 3(2):27–49
David J, Euzenat J, Scharffe F, Trojahn C (2011) The Alignment API 4.0. Semant Web J 2(1):3–10
de Bruijn J, Welty C (2013) RIF RDF and OWL compatibility. Recommendation, W3C. http://www.w3.org/TR/rif-rdf-owl/
Dean M, Schreiber G (eds) (2004) OWL web ontology language reference. Recommendation, W3C. http://www.w3.org/TR/owl-ref/
Doan AH, Madhavan J, Domingos P, Halevy A (2004) Ontology matching: a machine learning approach. In: Staab S, Studer R (eds) Handbook on ontologies. Springer Verlag, Berlin (DE), chap 18, pp 385–404
Dousson C, Gaborit P, Ghallab M (1993) Situation recognition: representation and algorithms. In: Proceedings of the 13th international joint conference on artificial intelligence (IJCAI), Chambéry (FR), pp 166–174
Ehrig M, Staab S, Sure Y (2005) Bootstrapping ontology alignment methods with APFEL. In: Proceedings of the 4th international semantic web conference (ISWC), Galway (IE). Lecture notes in computer science, vol 3729. pp 186–200
Eiter T, Ianni G, Lukasiewicz T, Schindlauer R, Tompits H (2008) Combining answer set programming with description logics for the semantic web. Artif Intell 172(12–13):1495–1539
Euzenat J (1996) Hytropes: a www front-end to an object knowledge management system. In: Proceedings of the 10th demonstration track on knowledge acquisition workshop (KAW), Banff (CA), pp (62)1–12

Euzenat J (ed) (2002) Research challenges and perspectives of the semantic web. EU-NSF Strategic report, ERCIM, Sophia Antipolis (FR). http://www.ercim.org/EU-NSF/semweb.html
Euzenat J (2015) Revision in networks of ontologies. Artif Intell 228:195–216
Euzenat J, Shvaiko P (2013) Ontology matching, 2nd edn. Springer, Heidelberg
Euzenat J, Stuckenschmidt H (2003) The 'family of languages' approach to semantic interoperability. In: Omelayenko B, Klein M (eds) Knowledge transformation for the semantic web, IOS Press, Amsterdam (NL), pp 49–63
Farquhar A, Fikes R, Pratt W, Rice J (1995) Collaborative ontology construction for information integration. Technical Report 63, Knowledge system laboratory, Stanford university, Stanford (CA, US). ftp://ksl.stanford.edu/pub/KSL_Reports/KSL-95-63.ps
Fensel D, Decker S, Erdmann M, Studer R (1998) Ontobroker: the very high idea. In: Proceedings of the 11th international Florida artificial intelligence research society conference (FLAIRS), Sanibel Island (FL, US), pp 131–135
Fensel D, Hendler J, Lieberman H, Wahlster W (eds) (2003) Spinning the semantic web: bringing the World Wide Web to its full potential. The MIT Press, Cambridge
Gandon F, Schreiber G (2014) RDF 1.1 XML syntax. Recommendation, W3C. http://www.w3.org/TR/rdf-syntax-grammar/
Ghilardi S, Lutz C, Wolter F (2006) Did I damage my ontology? a case for conservative extensions in description logics. In: Proceedings of the 10th international conference on principles of knowledge representation and reasoning, Lake District (UK), pp 187–197
Glimm B, Ogbuji C (2013) SPARQL 1.1 entailment regimes. Recommendation, W3C. http://www.w3.org/TR/sparql11-entailment
Glimm B, Horrocks I, Motik B, Stoilos G, Wang Z (2014) HermiT: an OWL 2 reasoner. J Autom Reason 53(3):245–269
Gmati M, Atencia M, Euzenat J (2016) Tableau extensions for reasoning with link keys. In: Proceedings of the 11th international workshop on ontology matching, Kobe (JP), pp 37–48
Grosof B, Horrocks I, Volz R, Decker S (2003) Description logic programs: combining logic programs with description logic. In: Proceedings of the 12th World Wide Web conference, Budapest (HU), pp 48–57
Harris S, Seaborne A (2013) SPARQL 1.1 query language. Recommendation, W3C. http://www.w3.org/TR/sparql11-query
Hartig O, Pirró G (2016) SPARQL with property paths on the web. Semant Web J
Hayes P, Patel-Schneider P (2014) RDF semantics. Recommendation, W3C. http://www.w3.org/TR/rdf11-mt/
Heath T, Bizer C (2011) Linked data: evolving the web into a global data space. Morgan & Claypool, Milton Keynes
Hitzler P, Krötzsch M, Rudolph S (2009) Foundations of semantic web technologies. Chapman & Hall/CRC, London
Hogan A, Harth A, Polleres A (2009) Scalable authoritative OWL reasoning for the web. Int J Semant Web Inf Syst 5(2):49–90
Horrocks I, Sattler U (2007) A tableau decision procedure for SHOIQ. J Autom Reason 39(3):249–276
Horrocks I, Sattler U, Tobies S (2000) Practical reasoning for very expressive description logics. Log J IGPL 8(3):239–264
Horrocks I, Patel-Schneider P, van Harmelen F (2003) From SHIQ and RDF to OWL: the making of a web ontology language. J Web Semant 1(1):7–26
Horrocks I, Patel-Schneider P, Boley H, Tabet S, Grosof B, Dean M (2004) SWRL: a semantic web rule language combining OWL and RuleML. http://www.w3.org/Submission/SWRL/
Hustadt U, Motik B, Sattler U (2007) Reasoning in description logics by a reduction to disjunctive datalog. J Autom Reason 39(3):351–384
Hustadt U, Motik B, Sattler U (2008) Deciding expressive description logics in the framework of resolution. Inf Comput 206(5):579–601

Janowicz K, van Harmelen F, Hendler J, Hitzler P (2015) Why the data train needs semantic rails. AI Mag 36(1):5–14

Jimenez Ruiz E, Cuenca Grau B, Horrocks I, Berlanga R (2009) Ontology integration using mappings: towards getting the right logical consequences. In: Proceedings of the 6th European semantic web conference (ESWC), Hersounisous (GR). Lecture notes in computer science, vol 5554, pp 173–188

Kalfoglou Y, Schorlemmer M (2003) Ontology mapping: the state of the art. Knowl Eng Rev 18(1):1–31

Lassila O, Swick R (1999) Resource description framework (RDF) model and syntax specification. Recommendation, W3C. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Levy A, Rousset MC (1998) Combining Horn rules and description logics in CARIN. Artif Intell 104(1–2):165–209

Luke S, Spector L, Rager D, Hendler J (1997) Ontology-based web agents. In: Proceedings of the 1st international conference on autonomous agents, pp 59–66

Lutz C, Walther D, Wolter F (2007) Conservative extensions in expressive description logics. In: Proceedings of the 20th international joint conference on artificial intelligence, Hyderabad (IN), pp 453–458

Mao M, Peng Y, Spring M (2010) An adaptive ontology mapping approach with neural network based constraint satisfaction. J Web Semant 8(1):14–25

Meilicke C, Stuckenschmidt H, Tamilin A (2009) Reasoning support for mapping revision. J Log Comput 19(5):807–829

Miles A, Bechhofer S (2009) SKOS simple knowledge organization system: reference. Recommendation, W3C. http://www.w3.org/TR/skos-reference

Mossakowski T, Codescu M, Neuhaus F, Kutz O (2015) The distributed ontology, modeling and specification language - DOL. In: Koslow A, Buchsbaum A (eds) The road to universal logic: Festschrift for the 50th birthday of Jean-Yves Béziau volume II. Springer, pp 489–520

Motik B, Rosati R (2010) Reconciling description logics and rules. J ACM 57(5):

Motik B, Cuenca Grau B, Horrocks I, Wu Z, Fokoue A, Lutz C (2009a) OWL 2 web ontology language profiles. Recommendation, W3C. http://www.w3.org/TR/owl2-profiles/

Motik B, Patel-Schneider P, Cuenca Grau B (2009b) OWL 2 web ontology language direct semantics. Recommendation, W3C. http://www.w3.org/TR/owl2-direct-semantics/

Motik B, Patel-Schneider P, Parsia B (2009c) OWL 2 web ontology language structural specification and functional-style syntax. Recommendation, W3C. http://www.w3.org/TR/owl2-syntax/

Motik B, Shearer R, Horrocks I (2009d) Hypertableau reasoning for description logics. J Artif Intell Res 36:165–228

Ngonga Ngomo A, Auer S (2011) LIMES - a time-efficient approach for large-scale link discovery on the web of data. In: Proceedings of the 22nd international joint conference on artificial intelligence (IJCAI), Barcelona (ES), pp 2312–2317

Pan R, Ding Z, Yu Y, Peng Y (2005) A Bayesian network approach to ontology mapping. In: Proceedings of the 4th international semantic web conference (ISWC), Galway (IE). Lecture notes in computer science, vol 3729, pp 563–577

Patel-Schneider P, Hayes P, Horrocks I (2004) OWL web ontology language semantics and abstract syntax. Recommendation, W3C. http://www.w3.org/TR/owl-absyn/

Pérez J, Arenas M, Gutierrez C (2009) Semantics and complexity of SPARQL. ACM Trans Database Syst 34(3):16

Pérez J, Arenas M, Gutierrez C (2010) nSPARQL: a navigational language for RDF. J Web Semant 8(4):255–270

Prud'hommeaux E, Seaborne A (2008) SPARQL query language for RDF. Recommendation, W3C. http://www.w3.org/TR/rdf-sparql-query

Phrer J, Heymans S, Eiter T (2010) Dealing with inconsistency when combining ontologies and rules using DL-programs. In: Proceedings of the 7th extended semantic web conference (ESWC), Hersounissos (GR), pp 193–197

Schlobach S (2005) Debugging and semantic clarification by pinpointing. In: Proceedings of the 2nd European semantic web conference (ESWC), Heraklion (GR), pp 226–240

Serafini L, Tamilin A (2005) DRAGO: distributed reasoning architecture for the semantic web. In: Proceedings of the 2nd European semantic web conference (ESWC), Hersounisous (GR). Lecture notes in computer science, vol 3532, pp 361–376

Sirin E, Parsia B (2007) SPARQL-DL: SPARQL query for OWL-DL. In: Proceedings of the 3rd OWL experiences and directions workshop (OWLED), Innsbruck (AT)

Sirin E, Parsia B, Grau BC, Kalyanpur A, Katz Y (2007) Pellet: a practical OWL-DL reasoner. J Web Semant 5(2):51–53

Stoilos G, Stamou G, Pan J (2010) Fuzzy extensions of owl: logical properties and reduction to fuzzy description logics. Int J Approx Reason 51(6):656–679

Stumme G, Mädche A (2001) FCA-Merge: bottom-up merging of ontologies. In: Proceedings of the 17th international joint conference on artificial intelligence (IJCAI), Seattle (WA, US), pp 225–234

Symeonidou D, Armant V, Pernelle N, Saïs F (2014) Sakey: scalable almost key discovery in RDF data. In: Proceedings of the 13th international semantic web conference (ISWC), Riva del Garda (IT), pp 33–49

Tsarkov D, Horrocks I (2006) FaCT++ description logic reasoner: system description. In: Proceedings of the 3rd international joint conference (IJCAR), pp 292–297

Voltz R (2004) Web ontology reasoning with logic databases. Ph.D. thesis, Universitt Fridericiana zu Karlsruhe

Volz J, Bizer C, Gaedke M, Kobilarov G (2009) Discovering and maintaining links on the web of data. In: Proceedings of the 8th international semantic web conference (ISWC), Chantilly (VA, US). Lecture notes in computer science, vol 5823, pp 650–665

Xu L, Embley D (2003) Discovering direct and indirect matches for schema elements. In: Proceedings of the 8th international conference on database systems for advanced applications (DASFAA), Kyoto (JP), pp 39–46

Zimmermann A, Euzenat J (2006) Three semantics for distributed systems and their relations with alignment composition. In: Proceedings of the 5th conference on international semantic web conference (ISWC), Athens (GA, US), pp 16–29

Zimmermann A, Le Duc C (2008) Reasoning on a network of aligned ontologies. In: Proceedings of the 2nd international conference on web reasoning and rule systems (RR), Karlsruhe (DE), pp 43–57

Artificial Intelligence and Bioinformatics

Jacques Nicolas

Abstract The chapter shines a light on the strong links shared by Artificial intelligence and Bioinformatics since many years. Bioinformatics offers many NP-hard problems that are challenging for Artificial intelligence and we introduce a selection of them to illustrate the vitality of the field and provide a gentle introduction for people interested in its research questions. Together with the framing of questions, we point to several achievements and progresses made in the literature with the hope it can help the bioinformatician, bioanalyst or biologist to have access to state of the art methods.

1 Introduction

The links between Artificial intelligence and Bioinformatics are long-standing and strong. J. Lederberg, a professor of genetics from Stanford interested in exobiology, pointed out in the early 1960's the need for large biological equipments to be assisted by sophisticated programs (Hundley et al. 1963). At a time when personal computers did not exist and the amount of data was measured in Kbytes, J. Lederberg shared working relationships with E. Feigenbaum and J. McCarthy and expressed visionary ideas in this small NASA report:

> In the experimental sciences ... applications will involve searching through data accumulations like spectra and other physical properties, the experimenter forming generalizing hypothesis and using the computer to test them against the file.
>
> Experienced insight into the role of the human and machine components of these systems should be invaluable in the further development of artificial intelligence, the programming of machines to simulate as far as possible those human cognitive processes that we begin to understand.These generalizations of human intellectual capability, and their delegation to machines, continue to refine the strategic role of the human element and to give it increasing leverage in the solution of complex problems.

J. Nicolas (✉)
Univ. Rennes, Inria, CNRS, IRISA, Rennes, France
e-mail: jacques.nicolas@inria.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_7

At the same time, Artificial intelligence was looking at complex, real-world problems to stimulate this science and get some measurable impact of its outcomes. The DENDRAL project, an expert system recognizing organic compounds from mass spectrometry data, was one of the first outcomes of these efforts (Lindsay et al. 1993). It introduced task-specific knowledge about a problem field as a source of heuristics, and paved the way for proteomics studies. One of the first success twenty years later was Protean, a system that aimed to interpret NMR data to determine protein three-dimensional structures (Hayes-Roth et al. 1986). It also initiated the development of realistic Machine Learning applications with Meta-Dendral, which was a system that learnt the rules necessary for Dendral from (spectra, structure) pairs (Feigenbaum and Buchanan 1993). Note that surprisingly enough, the assistance for the interpretation of mass spectra remains in high demand in biology, whether for studying small objects like peptides, glycans or various metabolites as well as for the study of larger molecular complexes. This mutual interest has not decreased over time and it is not an exaggeration to say that the contributions or the need for Artificial Intelligence may be found everywhere in bioinformatics.

1.1 A Major Application Field for Artificial Intelligence

Artificial Intelligence takes a close interest in the development of bioinformatics for at least two reasons. The first one dates back to the very beginning of Artificial Intelligence, when living organisms were used as a source of inspiration for the design of computational methods that were more robust to the treatment of real data, including the tolerance of uncertain and imprecise data, auto-adaptation and learning skills (see also chapter "When Artificial Intelligence and Computational Neuroscience Meet" of this volume). Most of these methods form the basis of what is called soft computing: neural networks (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2), evolutionary algorithms such as genetic algorithms or evolution strategies, immunological computation and various optimization techniques such as swarm or ant colony optimization (see chapter "Meta-Heuristics and Artificial Intelligence" of Volume 2), and uncertainty logics such as fuzzy set or possibility theory (see chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1). The second one dates back to the beginning of the century, when Biology became a data-intensive field of science, with the emergence of numerous instruments scanning life at the molecular level.

In fact, Biology is not just a science of data, it is also a science of knowledge. From this point of view, there is a clear gradient from Physics to Chemistry then Biology in the complexity of structures and systems. First of all, living organisms store information in a discrete way in complex molecules and contain many symbolic machines to decipher the corresponding code (e.g. polymerases, ribosomes, and spliceosome). But life may also be characterized by the presence of many context-dependent causal influences related to biological functions (Walker and Davies 2013). A cell state is determined not only by its genetic code but also by its history (cells

have a memory) and its environment, which strongly impact the expression of genes: at any time in its life, only a small fraction of molecules in a cell will react. Biology is certainly the science *par excellence* of relations and dependencies, in the sense that it studies many different types of objects related by a number of different interactions appearing at different levels in a hierarchical structure of organization. The famous paradigm coined by R. Kowalski, "Algorithm = Logic+ Control" (Kowalski 1979) could be rephrased as "Life = Logic+ Control" to emphasize the importance of these two components in all forms of life (although the control component does not simply define the computing strategy in this case but can have a feedback effect on the logic itself). It is thus a perfect application field of Artificial Intelligence for studies in knowledge representation, reasoning and machine learning. Moreover, Bioinformatics tackles numerous NP-hard problems that are interesting challenges for Artificial Intelligence.

Roughly speaking, the four main research tracks in Bioinformatics relate to statistics, algorithms on words ("stringology"), modeling (data bases, knowledge representation and system modeling) and combinatorial and optimization problems. The first one addresses issues that originate from the treatment of noisy observation data or from global effects of populations of variable elements like in population genetics. Although these questions are of interest to Artificial Intelligence, the contributions in this domain clearly derive from established methods in statistics. The second track led to many developments since sequences are the main source of data. Even expression data that measure the quantity of certain compounds are provided now in the form of sets of sequences. In this domain, the size of data—often many gigabytes—makes it essential to find linear analysis algorithms. Indexation techniques play a major role in achieving this goal. The last two tracks fully concern Artificial Intelligence techniques and still offer many interesting research challenges.

There were two possible ways to conduct a review on Artificial Intelligence and Bioinformatics: making a review, for each AI subfield, of existing or possible applications in bioinformatics, or making a review of problems in bioinformatics that are or would be interesting to AI people. The first choice would have been significantly imbalanced since Machine Learning and optimization in general occur in an overwhelming majority of papers (Baldi and Brunak 2001; Bhaskar et al. 2006; Inza et al. 2010; Mitra et al. 2008; Zhang and Rajapakse 2009). Machine learning generally faces hard problems in Bioinformatics cumulating small sample size, individual variability, high dimensionality and complex relations between data. For this reason and probably also because biology is a science that is experienced in cross-checking various hints to draw robust conclusions, ensemble learning (e.g. bagging, boosting or random forests, see chapter "Statistical Computational Learning" of Volume 1) has been particularly studied in this context (Yang et al. 2010). It has been applied to various problems and showed interesting results each time, either for gene or protein expression data (Gertheiss and Tutz 2009; Liu et al. 2010; Piao et al. 2012; Wu et al. 2003), the prediction of regulatory elements (Gordon et al. 2005; Lihu and Holban 2015; Wang et al. 2009) and a large range of other applications such as the analysis of interaction data (Nagi et al. 2017), protein structure prediction (Bouziane et al.

2015) or automatic function annotation (Galiez et al. 2016; Schietgat et al. 2010; Smitha and Reddy 2016; Yang et al. 2016a).

As usual, applications are a driving force for methodological developments and interesting extensions of the ensemble approach have been proposed. In particular, the inclusion of feature selection in this framework has been identified as an important issue, since it is subject to instability (Awada et al. 2012; Okun and Priisalu 2007; Pes et al. 2017). Many studies are available on this subject, ranging from a bootstrap strategy sampling and ranking data before feature extraction (Wald et al. 2012; Yang et al. 2011) to the integration of several testing methods or selection methods (Brahim and Limam 2017; Sheela and Rangarajan 2017). From the point of view of ensemble prediction techniques, Support Vector Machines (see chapter "Statistical Computational Learning" of Volume 1) have been often used, either during feature selection (Jong et al. 2004) or with different kernels (Guan et al. 2008) or combined with bagging for instance (Gordon et al. 2005). In general, Random Forests seem more adapted to high-dimension data, a frequently occurring case in Bioinformatics (Diaz-Uriarte 2007; Lertampaiporn et al. 2014; Pashaei et al. 2017) and to providing explanations to the predictions, an increasing concern in recent literature. Bagging and boosting seem more adapted to black box predictors and small sample size and can be usefully supplemented by transfer learning techniques (Mei and Zhu 2014).

This chapter aims to give a more precise view of the major current or emerging bioinformatics challenges that are of interest to Artificial Intelligence in order to encourage further works on this topic. The abundance of papers in the field prevents an exhaustive presentation of all the various topics addressed. We have made a selection that we hope is representative of the variety of themes, trying each time to give the basics necessary to understand a specific problem, a formalization of this problem and a few achievements and progresses in relation to this problem. People interested in further reading could consult general reviews like (Hassanien et al. 2008, 2013) or articles that are oriented towards a particular subfield of AI such as agents (Keedwell and Narayanan 2005; Merelli et al. 2007) or knowledge discovery (Holzinger et al. 2014). For the field of Bioinformatics itself many introductory texts exist, see for instance (Ramsden 2004; Singh 2015). Before reviewing the research work and getting to the heart of the subject, we begin with a brief introduction to the different types of data processed in bioinformatics.

1.2 Bioinformatics: Analyzing Life Data at the Molecular Level

Living cells are made of a large variety of molecules of different types performing different functions. Apart from water, the main constituents are biopolymers, i.e., molecules forming long chains of elementary components linked by covalent bonds, which can be considered at an abstract level as texts on fixed alphabets. There are four main classes of biopolymers: DNA, RNA, proteins and glycans.

The best-known kind of macromolecule is made of DNA and holds genetic information. DNA is made of four components (bases) represented by four letters (A, T, G, and C) that match together to form in most cases a double strand (A with T and C with G being the canonical matches). The DNA text is highly structured and includes regions coding for genes and others coding for the regulation of these genes. This structure differs depending on the absence of a nucleus in cells (bacteria and archaea) or the containment of DNA in a nucleus (eukaryotes). The order of magnitude of genome lengths is 10^6 bp (letters) for bacterial genomes and 10^9 bp for eukaryotes. Basically an organism's DNA is the same in all its cells but DNA is not a long quiet river: it can be modified by so-called epigenetic factors like DNA methylation or histone modifications. It is also well known that DNA can acquire mutations that cause genetic differences (polymorphism) between individuals. The most common are single nucleotide point mutations called SNPs (one SNP every 1000 bases for human individuals). However, mutations can occur throughout life on any cell's DNA and are then transmitted to the daughter cell lineage. All these transformations can potentially favor the appearance of diseases like cancer. An ancient but still topical challenge is the classification of species and the reconstruction from their DNA of their evolution scheme from ancestral species (see Sect. 6). A main challenge on DNA is the annotation problem, which consists in discovering the functional content encoded by the nucleotide sequences: where are the genes and other genomic units and can we relate them to known elements? For simple organisms it is routinely achieved by laboratories, using off-the-shelf software pipelines, but it remain a complex task for higher eukaryotic organisms (see Sect. 3). A more recent challenge is the representation and management of all variations that occur between individuals or even individual cells, a challenge that is also central to RNA.

The second macromolecule, RNA, also made of four nucleotides (A, U, G and C), has been particularly studied in the last fifteen years and shown a key role in the regulation processes. RNA is thought to have been a precursor molecule to life on earth (Robertson and Joyce 2012). RNA is a biochemical mediator having both the capacity to store information as DNA and to act as a biochemical catalyst like proteins. The primitive quasi-life forms of viruses are made of RNA. It is in most cases a single-strand folded onto itself, giving rise to various structures in space related to their function. A number of RNA types exist (a typical eukaryotic cell can contain ten of thousands of RNA species) that are built from DNA through a process called transcription. In a given cell type, a specific part of the DNA is expressed in RNA, generally with multiple copies. One of the most important RNA species is messenger RNA (mRNA), expressed and matured from regions of the DNA called (protein) genes and acting as an intermediate for the synthesis of proteins (through a process called translation). It is estimated that in the human genome 85% of DNA is transcribed in RNA and 3% of DNA encodes protein-coding genes. To give an order of magnitude of cell complexity, approximately 350,000 mRNA molecules are present in a single mammalian cell, made up of approximately 12,000 variants with a typical length of 2 kb. The abundance of each variant can range from 10,000 to a few copies. Other types of RNA are used for gene expression regulation, translation or

RNA processing. The RNA level of expression of thousands of genes in a sample of cells can be measured simultaneously (a process called gene expression profiling) for different conditions and at different time points. A major challenge of RNA analysis is the determination of differentially expressed genes across experimental conditions, development stages, or healthy/pathological conditions (Han et al. 2015). Beyond expression of individual genes, advanced studied are looking for gene sets (Huang et al. 2008) (see Sect. 2) or even gene networks, either to represent co-expression networks or differentially expressed biological pathways (Khatri et al. 2012) (see Sect. 5). Another challenge on RNA is the search of expressed gene variants that are variations from the same DNA coding region that produce alternative transcripts (alternative splicing) depending on the environment (see Sect. 3).

The falling cost of sequencing technology for these two types of nucleic acid chains is having a marked impact on the biological research community. Essentially, sequencing projects have generally become small-scale affairs that are now carried out by individual laboratories.

Proteins, the third type of macromolecule are the main actors of the cell, performing a large range of functions ranging from cell structure (e.g. collagen) to metabolic reaction catalysis (enzymes), transport, cell signaling, and DNA replication. Proteins control the continuous component of life dynamics: essentially all biochemical reactions have continuous rates depending on protein expression. The building blocks of proteins are amino acids (AA, 20 amino acids in the genetic code) that are bonded together by peptide bonds. When two amino acids combine to form a peptide, water is removed and what remains of each amino acid is called a (amino-acid) residue. The order of magnitude of protein lengths is 10^3 AA. Proteins fold into 3D-structures called tertiary structures that depend on their amino acid sequence and determine their function. A protein tertiary structure is slightly flexible and may adopt several conformations but it may be described at a more abstract level in terms of simpler secondary structure elements (mainly alpha helices and beta sheets, the rest being generally classified as random coils) or, on the contrary, be assembled into dimeric structures or large protein complexes that function as a single entity. We know that the structure of proteins is crucial to understanding their role. Unlike nucleic polymers, even the sequence of proteins is hard to obtain since proteins are often transformed by post-translation modifications (e.g. the most common modification, glycosylation is estimated to occur in greater than 70% of the eukaryotic proteins, see the next paragraph on polysaccharides). Protein structures can only be obtained through timely and costly experiments and a long-standing challenge in bioinformatics is to predict this structure from the sequence (see Sect. 4). Another particularly active field of research because of its importance for human health is the development of new drugs (see Sect. 7).

A more recent topic from the point of view of bioinformatics relates to nonribosomal peptides (NRP), which are small chains of amino-acids (size less than 50 AA) that are essentially produced by microorganisms, using specialized large (complexes of) peptide-synthetase enzymes instead of ribosomes. Their characteristic is to use

a large alphabet (more than 300 variations of amino-acids have been identified, compared to the 20 AA in aminoacids) and have a linear, branching and/or cyclical structure. These natural peptides have many biological functions (antibiotics, immunosuppressants, antifungals, toxins) that are of high interest for medicine and biotechnology. In fact, there is also a largely unexplored space of ribosomal peptides that share many of the structural and enzymatic features of NRPs, paving the way for a discipline in its own right called peptidomics (Dallas et al. 2015).

The last type of biopolymer are polysaccharides, which are carbohydrate structures made of long chains of monosaccharide elements often called single sugars (mainly 4 types but many hundreds structural isomers and variants have been identified), typically joined with covalent glycosidic bonds. Polysaccharides are often a reserve of energy for organisms (e.g. starch may be converted into glucose) or can have a structural role (e.g. chitin used in composite materials forming the exoskeleton of insects or the shells of crustaceans, or cellulose used as a dietary fiber). They are crucial to the functioning of multi-cellular organisms and are implied in protein folding, cell-cell interaction, immune response (antigen-antibody interaction) and epithelium protection, serving as mediators (or carriers) of many information transmission processes. Their structure can be linear or branched. Polysaccharides are particularly important in biology when associated with other elements and are referred to as glycans in this context. At an abstract level, glycans can be modeled by ordered labeled trees. The main glycans are found associated with proteins (glycoproteins or proteoglycans) and lipids (glycolipids). The modification of some protein amino acids after their translation by glycans is called glycosylation. The identification of glycan structures (sequencing) remains a complex experimental process, as is the identification of glycosylated sites in a protein. Unlike other biopolymers, progress in glycobiology is recent (Frank and Schloissnig 2010). Due to the importance of glycans in practical applications (drug targets, biofuels, alternatives to petroleum-based polymers), there is, however, no doubt that this field will experience major developments in the coming years.

When the entire set (or a large set) of elements of a certain type is available through high-throughput experiments, they are called omics data. For the four types described, the corresponding terms are genome, transcriptome, proteome and glycome. One could add another high-throughput source of information, the bibliome (Grivell 2002; Sarkar 2015), since millions of documents are published in Biology and can be considered as raw data out of reach of manual treatment (see next section).

Many biopolymers work by interacting with other biopolymers. For instance the regulation of gene expression is ensured by many mechanisms involving RNA (e.g., microRNA are small non-coding RNA molecules that interact with mRNA to regulate gene expression after transcription) or proteins (e.g. histones are proteins interacting with DNA in a structure called chromatine that organizes the very long DNA chain of a chromosome into structural units). Proteins can act on DNA by attaching to a specific sequence of DNA adjacent to a gene that they regulate (transcription factor) or by generating complementary DNA (cDNA) from an RNA template (reverse transcriptase, a mechanism used by retroviruses). Protein kinases modify other proteins

and are essential for the transmission of external signals within cells. All these interactions between numerous compounds occur within networks at different levels (in a compartment inside the cell, in the cell or in the extracellular matrix surrounding the cells, in a tissue or in a microbiota) and at different time scales and a great deal of research addresses this subject (see Sects. 2 and 5).

2 Data and Knowledge Management

In molecular biology, the flow of data from multiple high-throughput observation devices combined with the results of (semi-)automated analyses of these observations is collected in many databases (the journal Nucleic Acids Research, which publishes each year a catalogue of such databases in a special issue, lists for instance 152 databases in 2017, 54 of which are new (Galperin et al. 2017). The difficulties of analyzing observations to obtain functional knowledge about the organisms studied are not discussed here. The complex process of functional annotation of data is indeed more a matter of software engineering than artificial intelligence. An exception is the use of multiple agents to ease this task (Jiang and Ramachandran 2010). Predictions made by machine learning methods are treated in other sections and the next section for instance deals with the annotation of genes.

This section is about data and knowledge management. The maintenance of all these databases quickly became a nightmare due to the rapid increase in data with different levels of quality, which requires frequent releases, changes in database schema and an updating process that grows generally quadratically with the number of organisms. Moreover, the question of integration of all these sources of data is a central concern for biology. It is managed through the creation of ontologies, the representation of graphs linking heterogeneous data and, to a small extent, through automated reasoning (Blake and Bult 2006). The stake here is to support and leverage the transition from a world of isolated islands of expertise knowledge to a world of inter-related domains.

In terms of data integration, the requirements are:

- Identify entities (unambiguously);
- Describe them (i.e., their properties and their relations with other entities) and ensure each element of a description is itself identifiable;
- Combine descriptions from multiple places (typically different aspects of the same entity);
- Support semantically rich querying and reasoning on descriptions.

Over the last decade, Semantic Web technologies such as RDF, SPARQL and OWL from the W3C have provided the infrastructure supporting the Linked Open Data Initiative (Bizer et al. 2009) (see chapter "Semantic Web" of this volume). This has

played a key role for integrating bioinformatics data (Bellazzi 2014; Cannata et al. 2008; Post et al. 2007).

2.1 Information Extraction in Biological Literature

Despite strong and growing efforts to normalize data formats and collect observations in databases, a large amount of information relevant to biology research is still recorded as free text in journal articles and in comment fields of databases (Hirschman et al. 2002). Since interactions are ubiquitous in biology, retrieving interactions between identified components is a very important task. This assumes that different naming variants of a same component have first been recognized and it is far from easy due to early lax practices: use of common names (e.g. *bag, can, cat, or, six, top* are all gene or protein names), use of synonyms (e.g. *can* as well as *n214* are aliases for *nucleoporin 214*), ambiguous names (e.g. the string *cat* may refer to enzyme *Chloramphenicol acetyltransferase* in bacteria, or a gene for the *catalase* enzyme in Human), or terms used in other biological contexts (e.g., *shotgun*, which can refer both to *DE-cadherin* gene or to a technique used for sequencing long DNA strands) (Chen et al. 2005).

An interesting recent development concerns the rapprochement between two communities, namely expert curators who read each publication to extract the relevant information in a standardized computationally tractable format and specialists in text-mining applied to biologically relevant problems. Manual curation is a very complex task that will likely need human experts over the long term, but it does not scale with the growth of biomedical literature (estimated in 2016 at more than 3 publications per minute for the main database of citations for biomedical literature Pubmed, which comprises more than 27 million citations). In 2007, the team working on RegulonDB, a carefully curated database on transcriptional regulation in E. coli, showed that 45% of the knowledge content could be extracted using rule-based natural language processing and that it allows for an additional 0.15% new rules to be proposed, of which a quarter was subsequently shown to be accurate (Rodríguez-Penagos et al. 2007). The BioGRID database, which systematically curates the biomedical literature for genetic and protein interaction data has been involved in a coordinated effort with the BioCreative (Critical Assessment of Information Extraction in Biology) initiative to produce the BioC-BioGRID corpus, which contains a test collection of 120 full text articles with both biological entity annotations (gene and species) and molecular interaction annotations (protein and genetic interactions) (Islamaj Doğan et al. 2017). This collaborative work allowed for guidelines for building text-mining systems that assist curation of biological databases to be proposed and fosters further research in this area. We extracted from the previous paper a list of four tasks in full text analysis that are particularly relevant in this context:

The curator assistance problem Given the full text of a research paper in biology, solve the following tasks:

- Recognition of evidence described in the current article vs information taken from other articles;
- Recognition of evidence which is supported by experimental data vs hypothetical or vague statements;
- Distinction between statements describing the layout vs statements describing the results of an experiment;
- Recognition of negative statements.

The OntoGene system is a state-of-the-art text mining system for the detection of entities and relationships from various items such as genes, proteins, chemicals but also drugs and diseases, implemented as web services for more flexibility. It provides standard text pre-processing tasks including identification of sections, sentence splitting, tokenization lemmatization and stemming, and can optionally perform syntactic analysis using a dependency parser (see chapter "Artificial Intelligence and Natural Language" of this volume for more information on natural language analysis). It includes a module for entity recognition, using a rule-based approach, and disambiguation, using HMM-based learning from noisy data. This last point is crucial since it is hard to obtain a specialized dictionary in every domain. OntoGene has been applied to RegulonDB, a manually curated resource about the regulation of gene expression in E.coli (Rinaldi et al. 2017).

Going beyond the state of the art will imply also taking into account the variety of forms in which knowledge is available in papers. As was mentioned in Rodríguez-Penagos et al. (2007), texts are not always sufficient and tables and figures and their captions contain what human curators need to generate relevant information. The recognition of evidence in particular can be greatly enhanced with such data. In this respect, the development of efficient and robust figure extraction methods (Clark and Divvala 2016), able to scale to large databases of documents such as semantic scholar is certainly good news when it comes to fostering these studies.

2.2 Biological Ontologies

As Stevens et al. noted, biology has a strong need for knowledge management tools (Stevens et al. 2000),

> Much of biology works by applying prior knowledge [...] to an unknown entity, rather than the application of a set of axioms that will elicit knowledge. In addition, the complex biological data stored in bioinformatics databases often require the addition of knowledge to specify and constrain the values held in that database.

The knowledge we are focusing on is mostly symbolic. It should typically support comparison, generalization, association and deduction (Bechhofer et al. 2013).

Such knowledge is typically represented in ontologies, which Bard defines as (Bard and Rhee 2004) "formal representations of areas of knowledge in which the essential terms are combined with structuring rules that describe the relationship between the terms". In the early 1960s, the National Library of Medicine proposed a controlled vocabulary, Medical Subject Headings (MeSH), for the purpose of indexing journal articles and books in the life sciences and facilitate searching. It consists of hierarchically organized sets of terms that permits searching at various levels of specificity. the MEDLINE/PubMed article database and the NLM's catalog of book holdings. Available since a few years in RDF format, it currently contains over 28,000 descriptors accessible via 90,000 entry terms. Since then, a large number of ontologies have been developed in biology. A repository like Bioportal (Whetzel et al. 2011) was referencing 685 ontologies and 95M direct annotations in 2017. Moreover, a huge amount of data annotated by ontologies are now available via public SPARQL endpoints like the EBI RDF platform (Jupp et al. 2014), which is built on the OpenLink triple store technology and allows program access to these data.

The Gene Ontology (GO) is probably the best example of a significant development in biological ontologies (Ashburner et al. 2000), the number of papers citing this article (more than 20000) being a good indication of its importance. GO is both a standard terminology for describing the function of genes and gene products and a corpus of evidence-based GO annotations for gene products. In 2016, it contained more than 40,000 terms and 90,000 relations, over 600,000 annotations with an experimental evidence extracted from 140,000 papers, and is also regularly revised by a dedicated team and requested by the scientific community. GO consists of three independent ontologies in the form of directed acyclic graphs (DAG): "molecular function" describing activities (e.g. catalytic or binding activities) at the molecular level, "biological process" giving programs accomplished by these activities, and "cellular component" where the function occurs. Each concept includes a term (recommended name), an identifier, definition (explanation and references) and synonyms. The DAG is essentially a tree with a few children having several parents. The relationship of child to parent can be either "is_a", "part_of", "regulates" or "occurs_in".

It is associated with GO tools such as browsing, SQL querying, and the Term Enrichment Service to find terms that are significantly more present in a set of product genes than by chance. GO is also supported by external tools such as Blast2GO (Götz et al. 2008), dedicated to high-throughput functional annotation of genomic sequences. For the annotation of a new sequence, Blast2GO looks for homologous sequences in sequence databases with the comparison tool Blast, transfers the annotation of these homologous sequences and applies various rules to enhance the final annotations given relationships between the three subontologies and other databases on protein domains and pathways, and using natural language text mining to simplify or structure the annotations. Since the management of ontologies has a strong technological component and uses a lot of engineering work, there are of course generic tools that are applied to GO as for instance ontologyX (Greene et al. 2017), a package for integrating ontologies in the R environment. As a data provider of growing importance, GO is not used solely for annotation purpose but as a source of

features for prediction purpose. For instance, the issue of predicting the subcellular location of a protein has been addressed by machine learning methods including GO terms as discriminant features of protein locations (Li et al. 2012; Mei 2012; Wan et al. 2014). The idea in Mei (2012) is to retrieve proteins homologous to the target sequence by looking for matches against the InterPro protein signature databases (InterproScan), to extract their GO terms in the three subontologies, then to learn a kernel for a SVM to transfer the appropriate GO terms on the target protein and use them for predicting the location. The interest of using GO or other ontologies through transfer learning to enhance predictions has been confirmed in other studies such as the prediction of the associations between human genes and phenotypes based on human protein–protein interaction network (Petegrosso et al. 2017).

In recent years, the tendency has in fact been to transform the ontology into a true knowledge base by integrating other sources (for instance the database of molecular interactions IntAct), by adding axioms and biological reasoning abilities, and by working on more elaborated representations as for instance for the description of biochemical pathways (The GO Consortium 2017). There is thus a strong opportunity for the IA community to transfer and test some tools in this domain (see chapter "Reasoning with Ontologies" of Volume 1). The Web Ontology Language (OWL) is used in advanced versions of GO that include "has_part" and "occurs_in" relations and propose a fully axiomatized content giving access to cross ontology relationships. These other Open Biological Ontologies come from OBO, a consortium with an editorial committee that ensures coordinated development in biological and medical sciences of ontologies. They are designed to be interoperable and logically well-formed and to incorporate accurate representations of biological reality (Smith et al. 2007). Thus, GO includes, for instance, links to (small) Chemical Entities of Biological Interest (Chebi Hastings et al. 2013) and a multi-species anatomy ontology (Uberon, Mungall et al. 2012). From the point of view of the integration of different ontologies, a central problem is that of *alignment*, where one tries to match the entities with each other (Shvaiko and Euzenat 2013). From the point of view of knowledge representation, a form of negation has been introduced in GO: when a gene product is expected to have a certain activity but it is known from experiments that it is not the case, it is emphasized with a *Not* qualifier (The GO Consortium 2017). Although there are currently very few negative annotations in GO, this is a clear advance with respect to reasoning. The full exploitation of biological knowledge expressed in OWL needs highly efficient reasoners on the underlying descriptive logics. A good example of such a framework is Aber-OWL (Hoehndorf et al. 2015), which uses the ELK reasoner. The main fulltext index of the scientific literature in Biology, PubMed, is built on top of Aber-OWL, and give an ontology-based access to more than 27 M citations for biomedical literature from MEDLINE, life science journals, and online books.

The field of biological ontologies provides a large scale experimental field for researches at the crossroads of Semantic Web and Knowledge Bases. It is led by a dynamic community that proposes many research issues and we will just point at two of them to conclude this section. *Tracking the inconsistencies* in such complex knowledge bases subject to many types of updates (new terms, obsoletion, new

name for a term, term merge, etc.) is a research task of high importance. In Chen et al. (2007) authors propose an ontology-based framework to detect inconsistencies in biological databases. This task is approached as a quality control problem in Ghisalberti et al. (2010). In Park et al. (2011), as many as 27 databases are used to check GO and, besides syntactic errors, semantic inconsistencies are checked concerning redundancy and use of species-specific definitions. A more global Belief Revision approach (See chapter "Belief Revision, Belief Merging and Information Fusion" of Volume 1) is proposed in Deagustini et al. (2016), where consolidations operators are built satisfying a fixed set of properties, based on kernel contraction and restoration and performed by the application of incision functions that select formulas to delete (conflicts). Although developed for Datalog± ontologies, these operators can be applied to Description Logics and it seems to be an interesting research direction for bio-ontologies.

As coined in Matentzoglu et al. (2017), building ontologies using OWL is a difficult, error-prone process and these errors have to be made explicit to authors. A general technique that seems to give good results in limited experiments is to improve the understanding of correct and efficient authoring actions by providing entailment set changes. In its perspectives, the GO consortium has also announced moving towards the *introduction of relations* between annotations for the function description of gene products in the context of a larger biological process. The new model, called LEGO for "Logical Extension of the Gene Ontology" is a neat progress towards the study of causality in biological networks. The LEGO formalism will define how different traditional GO annotations can be combined into a larger 'model' of gene and system function. Preliminary studies have been presented in Huntley et al. (2014). The idea is to associate to standard annotations (a pair single gene product GP-single GO term) a relational extension of the form $Relation(Entity)$ depending on the gene product, where $Relation$ concern either a chemical (molecular relation) or any other entity like a cell type (contextual relation). This extension is interpreted internally as a relation $Relation(GP, Entity)$.

3 Gene and Non-coding RNA Prediction

Gene prediction is a task that occurs at the beginning of a new genome annotation, just after completing DNA sequencing. It refers to the process of identifying the regions of genomic DNA that encode protein-coding genes or RNA genes. The prediction of protein genes of prokaryotic genomes (bacteria and archaea) is relatively easy since they usually appear without interruption in the sequence, as a single block named open reading frame (ORF). A remaining difficulty is the prediction of ribosomal frameshift events, a particular mechanism present in all organisms including viruses, which causes a shift by one or two nucleotides when translating the mRNA code, thereby changing the ORF and the protein code. It is generally treated through HMM prediction (Antonov and Borodovsky 2010; Kislyuk et al. 2009), but the difficulty to obtain experimentally validated frameshifts has reduced opportunities for model

learning. The recently developed ribosome profiling technique (Ribo-Seq) that allows precise mapping of the locations of translating ribosomes on mRNAs should boost interest in research on this topic (Michel et al. 2012).

For protein of eukaryotic genomes, genes are made of a mosaic of blocks and an important subtask is the search of possible isoforms called splicing variations made of the combination of specific coding parts called exons, which are built by a special editing process during or just after transcription (transformation of the DNA code in a RNA molecule). Alternative splicing occurs for instance in half of the human genes, largely increasing the diversity of proteins and their specificity in different tissues. A gene is in this case the set of all the exons appearing in these isoforms. Splicing uses faint signals that are not completely understood and the prediction of alternative splicing variants remains a primary challenge in gene prediction. Moreover, part of the remaining sequence (called introns) can be used in some rare variants. In fact, the definition of a gene has evolved significantly with discoveries and it is likely that much remains to be discovered in this area and requires new developments in bioinformatics. Examples of recent advances in gene knowledge include chimeric mRNAs that are produced by joining exons from two or more different gene loci (Lai et al. 2015).

Historically, the development of gene prediction algorithms was based solely on the DNA sequence since the technology was still limited with respect to RNA sequencing. In order to have access to transcribed sequences, people were generating Expressed Sequence Tag (EST), short sequences of DNA synthesized from mRNA by special enzymes making the reverse transformation. EST are still in use for genetic studies of populations (simple sequence repeat (SSR) markers are ideal for this purpose). Since then, the RNA-Seq technology has given far more efficient access to transcribed sequences, introducing a small revolution for gene discovery, and other hints, such as mass spectrometry data, are also available to help finding protein-coding genes. Another evolution is due to the accumulation of genome sequences that enables the transfer of knowledge about these genomes to building new ones. We briefly review these aspects, starting with the analysis of DNA sequences. A review of practical tools is available in Hoff and Stanke (2015).

Main gene finders, such as GeneMark, GeneID, GlimmerHMM, AUGUSTUS, and SNAP, have been developed to recognize specific features in the DNA sequences (translation start site, sequence composition, splice site patterns, etc.) whose cumulated presence is signaling the presence of genes. They also frequently include the prediction of functional elements closely associated to genes, such as regulatory regions. In most cases HMMs form the basis for modeling these patterns, with tools having pre-built HMM models for several model species, which are then adjusted for a new genome by training them on a user-provided subset of known genes. The recent tendency is to include more complete learning capacities in integrated frameworks: see for instance WebAugustus (Hoff and Stanke 2015) and SnowyOWL (Reid et al. 2014) for Augustus and the work of Lomsadze et al. for Genmark (Lomsadze et al. 2005, 2014) (Genmark-ET and ES).

The technology used to acquire RNA sequences has made major improvements over recent years, firstly through high-throughput sequencing of short reads (e.g.

sequences of length 150), then much longer sequences (e.g. of length 1500). Despite these improvements, the state-of-the-art of transcript-based approaches for gene recognition is still unsatisfying, and there is surely room for advanced AI techniques in the analysis of these new data. This includes the combinatorial issue of assembling full transcripts from the fragments due to the diversity of situations (a giant jigsaw puzzle), the difficulty in quantifying the expression levels (solving a system of linear Diophantine equations) and the difficulty in recognizing and discarding long untranslated regions that may contain fragments close to protein-like coding sequences. The integration of RNA-seq data in the training of gene finders has been proposed in Hoff et al. (2015); Lomsadze et al. (2014). In Pertea et al. (2015), transcript reconstruction and quantification are solved simultaneously, the quantification question being stated as a maximal flow issue on an alternative splice graph of overlapping fragments that authors solve with a specific breadth first search algorithm.

In the domain of health, individual variations (mutations) that occur in genes are known to be a major factor of diseases directly or indirectly, and the increasing access to individual whole-genome sequences could be the vector of a deep change in care strategy generally referred to as "precision medicine" (Aronson and Rehm 2015). As mentioned in the introduction to this section, splicing is an essential feature of eukaryotes. It is estimated that at least 90% of human genes have splicing variants. Some mutations directly impact exons and their translation into viable proteins, but missplicing due to mutations in introns is also an important source of human diseases. It is possible in some cases to align the RNA sequences on a reference genome and this strategy is often chosen in the case of the human genome and does not use artificial intelligence. The state-of-the-art in this domain has reached a high level of sensitivity (Medina et al. 2016), using the combination of an indexing method (Burrows-Wheeler Transform) in order to obtain a first set of high-quality mapped reads and a constrained dynamic programming search (Smith-Waterman) for resolving more difficult splicing variants. However, the problem remains difficult in the absence of a reference genome or with highly altered variants (tumour tissue sampling).

The prediction of splice sites (frontiers between introns and exons) include donor (exon/intron boundary) and acceptor (intron/exon boundary) splice site recognition and the recognition of a particular structure called branchpoint element. It generally occurs on a window of 100–150 nucleotides sliding on the sequence. Many machine learning methods have been used for the donor and acceptor prediction tasks, including Neural Networks, Decision Trees, HMM, and Support Vector Machines, or a combination of them. For instance, Wei et al. (2013) trains a first order Markov model to generate sequence features on the conditional probability of presence of each aminoacid within the site and outside the site. These and other features (e.g. trinucleotide compositions) are first subject to a feature selection step, then used to build a SVM model from a training set. It has been slightly refined by Pashaei et al. in a series of papers showing the difficulty to choose methods that have simultaneously a high accuracy, use a few parameters and are efficient (Pashaei et al. 2016a,b; Pashaei et al. 2017; Pashaei and Aydin 2017). They first introduced a boosting algo-

rithm (AdaBoostM1 with C4.5 for the weak learner), then introduced second order Markov models, then replaced the SVM and boosting method by a bagging algorithm (Random Forest classifier) and came back to Adaboost with another feature encoding scheme.

At a higher level, alternative splicing leads to multiple transcripts from a single gene, using different pairs of donor/acceptor splice sites. These splicing variants can be recovered by using EST data or RNA sequences (Pirola et al. 2012). The basis for the representation of all variants is a *splicing graph*, i.e. a DAG whose vertices are blocks (maximal sequences of exons fragments that always appear together in all variants) and edges correspond to adjacency in at least one variant (Lacroix et al. 2008). From a set of sequencing reads, it is possible to obtain a weighted graph, i.e.the number of reads(abundance) supporting each edge. Then the *transcriptome reconstruction problem* consists to find for each possible path (and thus each variant) in the splice graph an abundance compatible with these data and the possible errors in sequencing. It is an integer linear programming issue that has not always a solution but that seems to be solvable in practice if the list of all variants is given (Lacroix et al. 2008). A variant of this graph has been proposed in Beretta et al. (2014) where it is the vertices rather than the edges that are weighted, each weight being the size of the sequence associated to a block. Authors are looking for a minimal-weight splicing graph having the same set of k-mers (string of length k) than the set of reads.

4 Protein Structure Prediction and Computational Protein Design

The issue of predicting the structure of a protein from the mere observation of its sequence is one of the oldest challenges in bioinformatics and still warrants much study. Given a protein sequence and some fixed environmental conditions, the folding in space of this sequence is a deterministic process that leads to a specific conformation, up to small vibrations. The best source of protein structures, PDB (Protein Data Bank), contained 42,000 distinct proteins with known 3D structures in 2017. In contrast, the main source of protein sequences, UniProtKB, contained more than 500,000 reviewed entries and almost 100 M automatically annotated sequences. This gives an idea of the gap between experimental capacities and the needed knowledge of structure annotations.

The structure prediction research community is very dynamic both because it is a fundamental research issue relating to understanding the basic building blocks of life from a structural and functional point of view and also because it has, since 1994, been centered around a very challenging competition, a kind of "Olympic Games" known as CASP (Critical Assessment of protein Structure Prediction, http://predictioncenter.org). Targets for structure prediction are structures about to be solved by X-ray crystallography or NMR spectroscopy and that will be available in the Protein Data Bank at the end of the competition. A hundred teams are participating

in about ten prediction categories and results are published in the journal Proteins (Moult et al. 2018).

Since the prediction of 3D structures is a very complex issue, a number of subproblems have been defined. At the lowest level, one can look for secondary structures, where each position in the sequence is assigned a conformational state between a small number of possibilities (generally 3 classes, alpha helix, beta strand or coil). Other predictions can relate to physical measures such as solvent accessibility or localization information such as transmembrane regions or protein subcellular localization. It can also relate to particular bonds like disulfide bonds or hydrophobic interactions. Finally, a very useful intermediary level of representation for protein structures is made by *contact maps*, which point to close positions within the structure (distance less than a given threshold).

4.1 Secondary Structure Prediction, A Benchmark Model for Structural Bioinformatics

Secondary structure prediction (SSP) is the first step to understanding the 3D structure of a protein and is essential in the kinetics of protein folding. It is certainly one of the bioinformatics issues on which the highest number of machine learning methods has been tested, due to an early effort to standardize secondary structure states and accuracy measures and propose benchmark data sets. In Yang et al. (2016b), 266 methods were counted between 1973 and 2016, a research effort that underscores the importance of bioinformatics problems for machine learning development. Currently, most methods are used to predict several structural parameters and, particularly, solvent accessibility.

Secondary structures are local regular conformations spontaneously formed as an intermediate during protein folding. They are defined by specific hydrogen bonding arrangements between the amino and carboxyl groups in the backbone of the molecule. The prediction is generally made on 3 classes (alpha helices, beta sheets and random coils, see Sect. 1) and achieved on a window of about 15 aminoacids sliding on the protein sequence. The current tendency is to work on 8 classes. The interest of ensemble learning has been pointed out very early on this problem (Guermeur et al. 1999), for instance by using Multivariate Linear Regression to combine the scores of multiple classifiers.

Generally, a two-step procedure is used: one for a raw prediction of the class, and one for smoothing this prediction over the sequence, taking into account correlations between several positions. Decisive progress then came from the introduction of more domain knowledge into the prediction process. First, instead of only considering the protein sequences, the fact that all proteins are related through evolution has been taken into account by looking for known proteins similar to the one studied (Rost and Sander 1993). It not only served to improve the prediction score but also to define a position-specific reliability index pointing at regions of the proteins with

high confidence predictions. Once similar proteins have been collected, a multiple alignment of their sequences provides information about the importance of each position that can be used in the classification procedure, for instance in the form of a profile HMM (Cuff and Barton 2000; Eddy 1998). Together with an ensemble strategy, it has given birth to three important prediction servers based on neural networks, PSIPRED (McGuffin et al. 2000), Distill (Baú et al. 2006), and JPred (Drozdetskiy et al. 2015).

The current state of the art introduces another source of knowledge, the PDB, which has sufficiently grown to propose a large set of non-redundant protein chains (5800) with a known 3D structure and thus a known secondary structure. A simple strategy has thus been proposed in Magnan and Baldi (2014), where the basic prediction uses a bidirectional recursive neural network and the sequence regions with sufficient similarity with some of these chains are assigned with their majority secondary structure.

As stated in this paper, the problem seems virtually solved with such a strategy, but it is not the end of the story. Using existing structures as templates does not convey any information on the logics of folding, although it must exist since the number of possible structures seems very small as compared to the variety of sequences and the size of the conformation space. Any progress in prediction methods for SSP may benefit other more difficult prediction problems on protein structure, but this requires going beyond case-based reasoning to discover the folding logic. If no, it seems likely that the complete 3D prediction problem will remain hard to solve. This is why people continue to measure accurately the contribution of prediction methods without the input of structural information and even without the use of homology with other sequences.

The contribution of ensemble methods is reviewed for various classifiers in Bouziane et al. (2015). Standard deep learning methods do not seem to provide a significant gain with respect to other methods as is shown in Spencer et al. (2015), which uses a belief network made of Restricted Boltzman Machines. In contrast, dedicated deep learning network architecture has been recently developed giving better results (Wang et al. 2017b, 2016c). Studied in Wang et al. (2016c) and available on the server RaptorX (Wang et al. 2016a), DeepCNF (Deep Convolutional Neural Fields) is a mix between a Conditional Random Field (input and output layers) and a deep Convolutional Neural Network. An in-depth analysis of the next required steps in SSP is provided in Yang et al. (2016b). DeepCNF makes it possible to take into account the correlations between secondary structures at different positions and to look at longer range correlations in protein sequences, which is a crucial point for SSP improvement. Indeed, beta sheets can link distant regions in proteins and it is a severe difficulty for window-based methods (the prediction error rate increases almost linearly with the number of non local contacts in proteins). In the same vein, another study has introduced the architecture LSTM-BRNN (Long Short Term Memory Bidirectional Recurrent Neural Network), and made it available on the server Spider3 with promising results. Another point is to consider meaningful features for SSP, such as intrinsically disordered (so-called chameleon sequences) or semi-disordered regions (Zhang et al. 2013), exposed parts of the protein (greater

solvent accessibility in contrast with the hydrophobic core) (Faraggi et al. 2012), capping regions (boundaries of helices and sheets) (Midic et al. 2005) or proline conformation (proline is a particular aminoacid that is responsible for conformational heterogeneity) (Song et al. 2006). The prediction can also be focused on more detailed secondary structures (8 states) or particular super-secondary structures like beta hairpins (see Xia et al. 2010 for an SVM and ensemble-based approach), a motif of two consecutive beta strands that looks like a hairpin and are important in folding kinetics.

Our personal conclusion is that expertise and knowledge has become central to this problem, and it could be interesting to progressively integrate the relations found by a statistical approach in a reasoning framework. Early work along these lines, for instance in Inductive Logic Programming, which achieved wholly satisfactory results when it was proposed (Muggleton et al. 1992), certainly warrants new studies.

4.2 Folding in Space

The next problems after SSP in structural bioinformatics are the prediction of the *backbone structure of a protein*, which can be represented by a series of values for two angles (dihedral or torsion angles), and the prediction of *contacts*, typically at a distance <8 Å. Indeed, SSP provides a rough classification into a few states that lacks precision, particularly at the interface between two secondary structures. A more relevant prediction level for the characterization or the multiple alignment of sequences on 3D structures is the level of local conformational parameters. Important atoms in the backbone of the protein are a central carbon, Cα, attached to the nitrogen N of an amine group, to the carbon C of a carboxyl group, and to a side chain specific to each aminoacid. Angles as well as distances are measured on Cα and sometimes C and N positions. These problems are much harder than the SSP problem and there is still room for significant improvements. A comparison of methods for predicting dihedral angles of a protein has been proposed in 2014 (Singh et al. 2014), showing the best results for the method SPINE-X (Faraggi and Kloczkowski 2017). This method combines a discrete classifier and continuous real-value prediction of torsion angles trough Conditional Random Fields and is characterized by a six-step training approach that predicts successively the secondary structure, residue solvent accessibility, and torsion angles. The comparison paper also shows that part of the performance of the method is due to a simple representation trick that shifts the angles in order to have an easier separability of the trimodal distributions of angles along the three main SSP states.

Crossing results of multiple predictions of related structural parameters to enhance the prediction of one of them, possibly iteratively, is a general tendency that can be found in state-of-the-art studies (Heffernan et al. 2015; Li et al. 2017) using deep learning methods. Deep learning allows for easy integration of multiple features and, in this case, recurrent networks and introducing a prediction of the contact number gave the best result. Another interesting recent direction is to discretize

(partition) the space of torsion angles into characteristic regions before prediction based on the observation (Ramachandran plot) that there are strong preferences on the possible dihedral angle pairs (Gao et al. 2017). This idea is at the basis of *structural alphabets* (Offmann et al. 2007; Pandini et al. 2010), made of a complete set of elementary fragments extracted from the analysis of known structures and sufficient to describe all the conformations of these structures. For instance (De Brevern et al. 2000) proposed a structural alphabet of 16 elements describing the conformations of fragments of five residue length, learned in two steps using the principle of Kohonen Self-Organized Maps. Protein backbone reconstruction is achieved by a bayesian probabilistic approach, considering sequence windows of size 15. Maupetit et al. (2006) proposed Sabbacc, an encoding of the protein trace in a hidden Markov model-derived structural alphabet of 155 elements describing the conformations of fragments of four residue length. Protein backbone reconstruction is achieved by a greedy algorithm assembling the alphabetical fragments in order to minimize an energy-based score.

The most detailed reduced representation of a protein structure is its *contact map*. A residue contact map is a graph that shows aminoacids with Cβ (the carbon in the side chain attached to Cα) at a distance less than 8 Å in the three-dimensional structure (Cheng and Baldi 2007). This information is useful for machine learning since it is invariant to rotations and translations and it helps a lot to retrieve the 3D structure (Adhikari et al. 2015; Pietal et al. 2015; Vendruscolo et al. 1997; Wang et al. 2016b). Typically, around 5% of residue pairs are in contact in a protein and the number of contacts in a protein is only linear in its length. The Confold method (Adhikari et al. 2015) uses the iterative scheme described for torsion angle prediction to build a rough 3D model from secondary structures and contacts using a distance geometry algorithm, which then serves as an input to refine the contact and secondary structure prediction and produce a second refined model.

As stated in Wuyun et al. (2016), most approaches in contact prediction are based on multiple sequence alignment (see Sect. 6) and indices like mutual information between positions in the alignment (Dunn et al. 2008), since there are strong evolutionary constraints for maintaining the protein structures. Machine learning methods may also be used to predict contacts from features extracted from multiple alignments.

A more recent tendency consists of combining the two types of methods and removing unlikely contact prediction corresponding to indirect coupling pairs obtained by transitivity or adding predictions that are likely to be missed with respect to typical patterns of contacts in proteins (Jones et al. 2015; Skwark et al. 2014). Skwark et al. (2014) presents PconsC2, a pipeline that uses first an ensemble method on 8 multiple alignments and 2 contact prediction methods, then corrects the prediction with deep learning through a 11×11 window providing information about the neighborhood of predicted contacts around each residue pair and a 5-layer feed-forward stack of random forest learners. The same type of approach appears in MetaPSICOV (Jones et al. 2015), using this time two stages of classic feed-forward networks, the first one learning contact pairs from 672 features extracted from multiple alignments and including an evaluation of the quality of this alignment, and the

second one using from among 731 features a window 11×11 matrix of contact pairs predicted by the first stage. Physical constraints that reflect the sparsity of contacts and the presence of particular secondary structures can be added to evolutionary constraints to further reduce the search space. Interestingly enough, a method proposed in Wang and Xu (2013) first predicts the probability of any two residues forming a contact learning Random Forest built on about 300 features extracted from multiple alignments. In the second stage, it uses integer programming to check a set of hard and soft (that can be relaxed) constraints, trying to maximize the sum of probabilities minus the sum of penalties generated by violated soft constraints.

Among the most recent methods, a qualitative leap has been made in the study (Wang et al. 2017a) by modeling the contact map issue as a pixel-level labeling problem and using a deep learning model concatenating two deep residual neural networks. It simultaneously predicts the label of all entries in the contact matrix. Also far from the state-of-the-art, another recent work (Chapman et al. 2017) is interesting because it introduces a particular representation of the contact map problem as learning a logical circuit (deterministic Markov network), whose inputs are binarized features on the sequences and final outputs are a negative and a positive contact decision. The logic gates are elements of a genetic algorithm using point mutation, duplication and deletion as evolutionary operators and accuracy as fitness function. This preliminary work paves the way for more logical approaches to the problem.

Finally, the ultimate and most complex step in structural studies is the prediction of the *three-dimensional native structure of proteins*. Since the work of Anfinsen in 1973, it is regarded as a free energy minimization problem in a space of possible conformations, a hypothesis that seems to be verified with very few exceptions. This problem is too hard to be solved for most proteins, which have a high number of atoms and a corresponding space that can reach millions of degrees of freedom. In practice all methods work on so-called "coarse-grained" models, by reducing the number of considered atoms or generating pseudo-atoms abstracting some groups of atoms (Blaszczyk et al. 2014). One must distinguish between the number of degrees of freedom used for the representation of each aminoacid (typically from 1 to 3) and the representation itself. Often, the backbone of Cα atom positions is used as a degree of freedom and the other atoms or pseudo-atoms of the representation calculated from these positions. For instance, the CABS model uses a representation tracing Cα, Cβ, the center of mass of the side chain, and a virtual atom placed at the center of each bond between Cα, the last three values being computed from three consecutive Cα positions. Another interesting but less used possibility, the SICHO model, represents a backbone of pseudo-atoms tracing the side chain centers and calculates the Cα positions from three consecutive side chain positions. One of the finest models, Rosetta, represents proteins with 3 dihedral angles for each aminoacid, using a library of low energy backbone-dependent side-chain conformations called rotamers to constrain the set of possible side chains.

All these coarse grain models can be applied to numerous applications in structural biology. It is out of the scope of this chapter to provide a detailed account of all these applications, but we have chosen to present a specific folding abstract framework,

called lattice protein folding, that allows us to rapidly test artificial intelligence methods without the need for an in-depth involvement in the physical, chemical and biochemical notions underlying this framework.

The protein folding in lattice models problem Coarse grain models have been developed in many studies within a discretized space, a grid or more generally a lattice (Mann and Backofen 2014). The lattice folding issue consists in finding a path of minimal score in the lattice such that (a) each residue is restricted to be placed on vertices of a lattice; (b) each vertex is associated with at most one residue (self-avoiding walks); and (c) a residue has to be placed in the neighborhood of the previous residue in the protein, this neighborhood being defined by a set of predetermined vectors. The cost function is generally derived from considerations on the free energy of the resulting molecule but is much simpler to compute than in continuous models. We are pointing to two types of lattice that are representative of the variety of models, the first one having been extensively studied:

The HP lattice: One of the simplest models of protein folding is the hydrophobic-hydrophilic (HP) model, which abstracts aminoacids in simply two states, hydrophobic (H, nonpolar) or hydrophilic (P, polar), and places them on a 2D square or a 3D cubic lattice. The score (energy function) is simply based on hydrophobicity: it is typically the number of non-bonding H in contact.

High-resolution lattices: Some authors argue that the main factor responsible for an observed folding are the constraints between side chains. For this reason, the SICHO model has been developed, taking the side chains as vertices and including a high number of possible interactions between these side chains (Feig et al. 2000). The CABS lattice is the most refined lattice that has been proposed (Koliński et al. 2004). It has 3 interaction centers for each amino acid and a basis of 800 vectors for the Cα neighborhood.

A number of lattice models have been proposed that can represent more or less finely the natural folding of proteins, in particular with respect to secondary structures. Among the main difficulties of the search is the detection of symmetries leading to equivalent conformations (Gan et al. 2008). Even in this simplest HP case, the problem is known to be NP-complete (Berger and Leighton 1998). Approximation algorithms exist however and folding in the cubic lattice may be achieved in linear time, for instance with an approximation ratio of $3/8$ (Newman and Ruhl 2004). A variety of local search methods have been tried as well as constraint methods (see chapter "Constraint Reasoning" of Volume 2), which have proven to be very interesting in this respect (Backofen and Will 2006; Mann et al. 2008). A description of exact methods may be found in the review (Mann and Backofen 2014). Many variants of the HP model exist that enable for instance diagonals (triangular lattice), work in an hexagonal lattice (Shaw et al. 2014), or in a face-centered cubic lattice (fcc, one of the best models in this category (Pokarowski et al. 2003; Shatabda et al.

2014). Other moderate resolution lattices exist, using a larger basis of vectors for the definition of the neighborhood. For instance, the "chess knight" model in 3D uses vectors $(\pm 2, \pm 1, 0)$. The 210 and 310 models use respectively 56 and 90 vectors.

Despite being one of the oldest models, the HP model continues to appear regularly in literature. For instance, in Doğan and Ölmez (2015) the problem is stated as a robot path planning problem where each amino-acid in the sequence is consecutively added to form continuous and self-avoiding amino-acid chains on the lattice. A new reinforcement learning method (see chapter "Reinforcement Learning" of Volume 1) is applied to this planning problem where such methods are known to perform well. Authors use for this purpose a compact state space representation and a distributed Q-learning algorithm (Ant-Q). In Dubey et al. (2017), the authors propose an enhanced energy function for the square lattice model taking into account other interactions than H-H ones. A mixed integer programming formulation is presented in Yanev et al. (2017), together with an exact algorithm and two heuristic algorithms. A swarm optimization algorithm and a Tabu search are combined in Guo et al. (2017). For a review of constraint programming in structural bioinformatics, see (Barahona and Krippahl 2008). Due to its complexity, the CABS model is closer to the continuous dynamic approaches. It can only be used in integrated environments that are predicting various structural properties like the presence of secondary structures and produce the corresponding constraints to restrain the general model (Blaszczyk et al. 2013; Gront et al. 2006).

We conclude this section by mentioning other important applications and point to some artificial intelligence techniques applied in this field.

The *homology modeling of protein structure* and the *protein threading problem* look both for the alignment of a protein sequence (the target) on a protein structure (the template). The first method, as in the case of sequence-sequence alignments, uses an homologous protein of known structure for the template. Protein threading deals with the harder case where there is no hypothesis of the existence of a homologous protein. In this case, one relies on the fact that the number of natural foldings is very limited and that a library of core templates is available, usually consisting of a set of segments separated by fixed or variable lengths. For each template, the best alignment is obtained by optimizing an objective function scoring the compatibility between sequences and between positions in the template. The complexity of the problem depends on the chosen structural model and the amount of homologous sequences available. Finding the optimal alignment is NP-hard in the general case where there are gaps of varying length between the segments, and where the objective function includes interactions between neighboring amino acids in the structure. Many methods are based on extensions of the dynamic programming approach used for sequence alignment, adding constraints from knowledge on amino-acid preferences with respect to neighbors, mutations, solvent accessibility or secondary structures for instance. A typical program of protein threading is RaptorX (Peng and Xu 2010), which uses a collection of regression trees to determine the scoring function of each alignment state in a Conditional Random Fields model that predict the state (match or gap) of each position in the alignment, and uses a neural network to rank the

different alignments on the target and give a measure of quality for each alignment (Källberg et al. 2012).

The *protein docking problem* is a 3D matching problem: it entails finding minimal free energy conformations of a protein complex made of a protein receptor and a generally small molecule (a drug) called ligand or another protein. As in the previous problems, an efficient approach is template-based modeling, which uses the knowledge of an already known complex to guide the conformation search of the new complex (Xue et al. 2017). For instance, case-based reasoning (see chapter "Case-Based Reasoning, Analogical Reasoning, Interpolation" of Volume 1) has been used in Ghoorah et al. (2013) for this problem.

The *protein folding pathway prediction* problem looks at the analysis of the folding kinetics of a protein with a known 3D structure. It has been represented by roadmaps, which are graphs of conformations where each edge indicates a possible transition between conformations (Moll et al. 2008). Roadmap-based methods were originally developed in robotics for collision-free robot motion planning and vastly extended and adapted for folding.

Finally, *computational protein design* is the process of designing new protein sequences with a fold close to a target protein structure. The ultimate goal is protein engineering, i.e., the design of new molecules adapted to target new materials or new functions, like for instance enzymes that are critical components in bioengineering and biomedical applications (Coluzza 2017). Rational protein design makes massive use of libraries of rotamers and can be seen as an optimization problem based on complex energy functions. In this domain, the exact solving method at the basis of many developments is a special dominance search algorithm (dead-end elimination), followed by an A* algorithm. Linear Programming, Quadratic Programming, Weighted Partial MaxSAT and Graphical Model optimization can be used to solve this problem, but a special form of Weighted Constraint Satisfaction formulation (Cost Function Network, see chapter "Valued Constraint Satisfaction Problems" of Volume 2) proves to be very efficient for this task (Allouche et al. 2014; Traoré et al. 2016).

5 Network Modelling

As emphasized in a recent editorial of a special issue of PLOS Computational Biology on biological networks (Ideker and Nussinov 2017), networks are everywhere in biology from molecular interaction circuits or modules to ecosystems, and have become a major mode of analysis in bioinformatics studies. Networks make it possible to understand biological entities at the system level, explaining diseases or drug effects as a cumulative result of small effects of individual genes of a cell for instance. This global understanding is out of reach of scientists from the mere observation of components for relatively small systems due to non linearity/additivity of regulations or interactions and it is even harder in current developments, which are designed to model genome-scale systems. It is thus crucial to assist biologists in this task not

only through the development of simulators, but also for the analysis of a network behavior in plain intelligible language, i.e., by checking properties and finding causal explanations.

At a higher level of organization, the interactions within bacterial consortia or in host-microbial symbiosis are and will certainly be the subject of a growing number of studies. Networks are present as observed data and as an abstract graph data structure (associated with a set of dedicated methods) that represents physical structures, as well as the dynamics of living components. If one considers the kinetics of components, the time scale may vary a lot depending of the type of interaction. For instance, it can range from milliseconds to seconds between the first and last stages of a signaling cascade (activation pathway from cell surface receptors to molecules controlling a cell function such as cell division) from seconds to minutes for metabolic reactions (biochemical reactions occurring in a cell) and reach hours for regulation processes (e.g. cell mechanisms used to increase or decrease the production of specific gene products). Signaling, regulation and metabolic networks are the three main types of networks studied in cells.

The representation of networks has used many formalisms that may be characterized as discrete (qualitative) or continuous models. A broad review of the work in this area can be found in the next chapter (see chapter "Artificial Intelligence in Biological Modelling" of this volume). We will focus here on discrete modeling since artificial intelligence methods are more directly applicable in this case. The early works of R. Thomas and S. Kauffman in the 1970's have demonstrated the value of logical modeling for representing gene regulation mechanisms in cells, by seeing them as discrete dynamical systems. The gene expression levels may be abstracted using Boolean variables (e.g. active or not active), time using logical steps and the expression changes using logical functions over the set of interacting genes. This simple but powerful framework is formalized in the following definition:

Definition 1 Boolean Networks are made of a graph $G = \{g_i, i = 1, n\}$ of Boolean variables. An edge $e_{ij} \in G$ represents the fact that variable g_i is one of the inputs to variable g_j. Each node is associated with a logical formula (a transition function) giving the output value of its Boolean variable with respect to the value of its input variables. A state is a vector of values over the complete set of variables. The state transition graph (STG) is the graph of all possible transitions between states, based on an update rule. Using formulas attached to variables to compute their new value, two main update rules are used in practice: the synchronous rule, which updates all variables simultaneously, and the asynchronous rule, which updates one variable at a time.

It is possible to extend this framework to multi-valued variables in order to take into account the possibility of multiple thresholds.

Numerous studies have used this formalism on biological applications. Several reviews are available on this topic (Abou-Jaoudé et al. 2016; Albert and Thakar 2014) and on modeling the dynamics of cellular networks (Le Novere 2015). Some repositories are emerging on the web (Chelliah et al. 2015; Klarner et al. 2017) and

an informal consortium, CoLoMoTo (Consortium for Logical Models and Tools) (Abou-Jaoudé et al. 2016) is working on the definition of standards.

The automatic extraction of networks from scientific literature has already been mentioned in Sect. 2. It is also possible to automatically refine such an initial prior knowledge network using experimental data, such as gene expression data for gene regulation networks (Lim et al. 2016) or phosphoproteomics data for signaling networks (Videla et al. 2015). The first paper uses a form of swarming hill climbing strategy, whereas the second one produces all possible solutions through a combinatorial approach coded in Answer Set Programming (ASP). For metabolic networks, the reconstruction of the set of biochemical reactions in a newly sequenced organism can be inferred from the sequenced and annotated genomes (Karpe et al. 2011). It works by recognizing in the new genome the enzymes catalyzing each reaction, then using the knowledge of previously known pathways from other organisms. To predict unknown or alternative pathways, it is possible to reason on atomic transfers (Boyer and Viari 2003).

Once constructed, the metabolic network is usually too large to be analyzed directly. A general technique to reduce it is based on the description of the properties that one wishes to keep between the initial network and the reduced network and then to automatically infer the minimum possible reduced network. This was proposed in Röhl and Bockmayr (2017), where the authors developed a mixed-integer linear programming (MILP) approach for computing this reduction.

The analysis of networks may be achieved from a topologic, static point of view, or from a kinetics-centered, dynamic, point of view.

In metabolic networks, static analysis includes the search for two close concepts, elementary modes and minimal cut sets (Acuna et al. 2009).

Elementary flux modes (EFMs) analyze networks from a pathway-oriented perspective. They are the minimal sub-networks (with respect to set inclusion of reactions) that can function (stoichiometrically and thermodynamically feasible) in steady state. These modes can help reveal the capabilities/objectives of a cell metabolic network, that is, the matching between phenotype and genotype. The standard approach to solve this problem is to reduce it to the enumeration of extremal rays in a pointed polyhedral cone, a standard problem in computational geometry. Unfortunately, the number of EFMs can increase exponentially with the network size. An interesting question is then how to navigate into the solution space instead of enumerating the solutions. To this end, the authors of Martin et al. (2016) allow the user to add/remove boolean constraints on the solutions that interest them. They use then a Satisfiability modulo Theory solver, CDC4, to solve both the boolean constraints on reactions occurring in the solutions and linear constraints taking into account stoichiometric data and steady state fluxes.

A minimal cut set (MCS) is a minimal (irreducible) set of reactions in the network whose inactivation will definitely lead to a failure in certain network functions (see the paragraph on perturbation analysis at the end of this section) (Klamt 2006). It helps to identify crucial, fragile parts in the network structure and to select suitable targets for repressing undesired metabolic functions.

Studying the dynamics of a given Boolean network entails three main issues on the associated state transition graph $STG = (S, T)$:

The state transition graph search problem

Attractors: Find all minimal subsets of states $A \subseteq S$ that are trap domains, i. e., such that the transition from an element of A yields an element of A. As an important particular case, stable or steady states are attractors reduced to a singleton and do not depend on the update scheme. The others are called cyclic attractors. The detection of stable or cyclic attractors is NP-hard (Akutsu et al. 2012).

Reachability: Check the absence/presence of specified trajectories in STG, i.e., such that there are paths in STG whose elements s_i belong to a specified subset S_i of S.

Perturbation analysis: Check the effect of fixing some variable values or some logical function values on the attractors and reachability properties.

Stable state attractors have been shown to correspond to identified cell differentiation states, a good example being the study of the regulatory and signaling networks associated with Th-cell subtypes differentiation (Naldi et al. 2010). Cyclic attractors have been shown in the cell cycle networks of Yeast or Mammals (see e.g. Barberis et al. 2017; Traynard et al. 2016). Attractors provide, more generally speaking, pointers to the possible steady functioning modes of the studied systems, either in normal conditions or under degraded conditions. They are also a way of checking the quality of a model by comparing attractors with the observed behavior of the biological system.

The detection of stable or cyclic attractors is NP-hard (Akutsu et al. 2012). A number of algorithms have been proposed to find attractors of a Boolean network, mostly using (reduced-order) binary decision diagrams. Quite naturally, a SAT-based bounded model checking method has been experimented in the case of synchronous networks (Dubrova and Teslenko 2011). Basically, the idea is to search by unfolding the transition relation for paths of bounded length p in the STG, to increase p if there is a path that is not a circuit and to mark the variables that are part of a circuit as attractors and exclude them from possible paths.

In practice, the knowledge of the formula associated with each gene and of the update scheme may be incomplete. The system ASP-G (Mushthofa et al. 2014) uses a higher level language for describing the network and the update scheme (with predicates such as activates, inhibited or changed) and relies on Answer Set Programming solvers to search efficiently the STG based on the previous SAT-based approach. The authors use incremental solving on the path length and, once a solution is found, the attractor is removed from the search space by adding a new constraint. In Abdallah et al. (2017), ASP is also used for more ambitious work extended to both synchronous and asynchronous update modes and considering automata networks instead of Boolean networks, a framework that enables multi-valued domains for vari-

ables and where more sophisticated update rules like non-deterministic synchronous mode can be introduced. In its current state, the program works in incremental mode (looking for paths of fixed size) and can produce several times the same attractor described by several cycles.

Another interesting approach (Klarner et al. 2015) looks at minimal and maximal trap domains represented by partial states (with free variables) instead of attractors and computes them efficiently from the set of prime implicants of the Boolean formulas. An insightful paper by K. Inoue has established a deeper relationship between Boolean networks and Logic programs (Inoue 2011) (Logic programming is described in chapter "Logic Programming" of Volume 2). Stable sates of a network may indeed be characterized as supported models of the corresponding logic program, a correspondence that allows a trivial coding of their search.

The reachability problem is basically searching trajectories in a state transition graph of size 2^n, where n is the number of Boolean variables and is a perfect application field for model checking and temporal logics. All studies in this field formalize this graph with a Finite State Transition System (*FSTS*) that is easily mapped into a Kripke structure. This framework is very general and can be applied far beyond the case of Boolean networks. In the field of biological systems, fine simulation of their dynamics by piecewise linear differential equations can, in particular, be considered. The principle is to associate with states hyper-rectangles in the concentration space (Batt et al. 2008) and a tool like GNA allows us to generate and export the $FSTS$ from differential models (Batt et al. 2012). See Carrillo et al. (2012), Brim et al. (2013) for reviews of existing tools. Once discretized, the state space can be explored via a logic allowing branching time (states with more than one immediate future) like CTL (Computational Tree Logic) or the μ-calculus, a practical issue being that it helps biologists to formulate their questions. This generic question in model checking has been addressed in Monteiro et al. (2008) through the development of a specific language of patterns with place-holders that allows the use of predefined state descriptors (e.g. increases or isSteadyState) and predefined types of questions (e.g. possibility of occurrence of a pattern or if-then statements). Specialized model checkers have been developed for the study of genetic regulation networks such as Antelope (Arellano et al. 2011), which addresses the important question of not just checking but exhibiting the states with a given property, using a Hybrid CTL with state variables. The size of the state space remains a crucial parameter for model checkers and the current state of the art is still too limited to query all the known models.

The third problem on Boolean networks, perturbation analysis, has been mainly applied to Probabilistic Boolean Networks, a stochastic extension of the standard framework where the transition function is replaced by a set of possible functions with an associated probability distribution. One of the main objectives of the analysis is focused on intervening in biological cell dynamics in order to alter the gene regulatory network or the signaling network and avoid undesirable cellular states, particularly in the search for a therapeutic strategy (e.g. to counteract the development of cancerous cells). The difficulty is how to bypass the inherent living system's robustness that uses many redundant pathways, while avoiding side effects and thus looking to minimize

the necessary changes. Finding an optimal control strategy leading to a desired state by changing some variable values is NP-hard (Akutsu et al. 2007). It is also possible to act on the transition functions by altering the transition probabilities or flipping a minimum number of values in the truth table (Xiao and Dougherty 2007). People are often using quantitative approaches to solve this problem by finely tuning the system's kinetic behavior and the use of Artificial Intelligence techniques on this problem is less well developed.

The problem has been set within a three-valued logical framework (to represent knock-out, knock-in and no intervention operations) in Samaga et al. (2010). It defines *intervention problems* made of a set of pairs (G, C), where G is a goal made of desired values for some target species (e.g. genes or reactions) and C equates to environment constraints setting some other species to fixed values. The issue is then to find (subset-)minimal intervention sets (MIS), i.e., a set of values for a set of species S such that all goals G are satisfied in their context C and at least one goal is not satisfied if an element of S is removed. The authors introduce a dedicated breadth-first search algorithm and emphasize the importance of preprocessing to reduce the dimension of a practical problem by finding classes of equivalence containing interventions having the same effect on target species. An ASP encoding is proposed in Kaminski et al. (2013) to enumerate all MIS for real-world signaling networks, showing that negation by default and recursive definition of reachability are valuable tools for searching for larger intervention sets and potentially solving the unbounded problem where the size of the intervention sets is not bounded. In practice, the ideal network is not known and it would be interesting to look for solutions that are compatible with several alternative networks explaining the same system.

Apart from signaling networks, metabolic networks have also been studied from the point of view of control, the interventions consisting of deleting reactions and/or regulating the reaction fluxes. One of the main industrial opportunities of such control is the optimized production of some target compounds by microbial organisms, a process called metabolic engineering, which is a research axis of synthetic biology. The MIS problem is transformed into a very similar Minimal Cut Set problem (MCS) (von Kamp and Klamt 2014) or Regulatory MCS (RegMCS) (Mahadevan et al. 2015) and Mixed Integer Linear Programming models have been reported in these publications in relation to solving MCS and RegMCS.

In contrast to the previous problem, the use of perturbations to learn the network has been the subject of many publications. Very few studies address both problems, one exception being the toolbox caspo (Videla et al. 2017), which proposes functions dedicated to each problem.

6 Understanding Evolution

The study of evolution is certainly a main topic of interest for biologists and bioinformatics has had a huge impact in this field since the advent of sequencing techniques. It is not merely of interest to evolutionary biologists and for the study of biodiver-

sity: evolution is a fundamental mechanism that helps to solve hard problems and obtain reliable answers at the level of populations but also in respect of structural or functional biological issues. It is thus also a key issue for bioinformatics. The term *Evolution* refers essentially in biology to changes in the inheritable traits of populations which occur over generations. Since easy access to the molecular content of living things, evolution can be considered at a much finer level than before, and evolution can even be observed at the individual level. A particularly important recent application relates to the evolutionary process in cancer. A tumor is an evolutionary process. It starts from a single cell and evolves with an anarchic development, including somatic mutations. Technology allows us now to sample a tumor and try to retrieve the history of tumoral cells, a combinatorial problem that may be treated by a phylogenetic approach (Caravagna et al. 2016; Malikic et al. 2015; Popic et al. 2015; Schwartz and Schäffer 2017).

The most cited papers in all research fields, as stated in news published in the journal Nature in 2014 on the top 100 papers cited since 1900, are, together with the Sangers Sequencing method and amplification methods, two papers on phylogenetics and two programs in this field. Moreover, phylogeny makes extensive use of sequence comparison and programs in this field are well represented with two versions of Blast and two versions of Clustal being cited (the multiple sequence alignment method on ClustalW is ranked 10th). It is why we start this section with a description of multiple alignment.

6.1 Multiple Sequence Alignment

At the core of many problems involving sequences in biology (sequence assembly, functional or structural annotation, homology search and phylogeny) lies the issue of sequence alignment. Its goal is to line up the letters in several sequences in order to exhibit a maximal similarity between letters at the same position. It may concern protein, DNA or RNA sequences, which are under a selective pressure. Indeed, this problem stems from the fact that all species originate from a common ancestor: sequences are assumed to be on the leaves of some unknown common evolutionary tree and thus share common characteristics that have become blurred by various mutation and insertion/deletion events. This is the reason why multiple sequence alignment and phylogeny are closely related, although other aspects like structural or functional properties may be taken into account. An alignment of a set of sequences helps to recover the evolutionary tree of the species they come from and, conversely, a known or assumed evolutionary tree (the guide-tree) helps to recover a relevant alignment of sequences. In fact, a growing number of authors are trying to build sequence alignment and phylogeny simultaneously (Ng et al. 2017). Note that the issue of sequence conservation modeling is not reduced to multiple alignment. If one is interested in sequence annotation or functional prediction, multiple alignment can usefully be extended to a pattern recognition problem (HMM profile) or even to grammatical inference studies (automata), working with a more expressive syntactic

model of conservation (Coste 2016). Moreover, multiple alignment takes into account evolution events such as point mutations, insertions and deletions, but more complex events can occur in a genome such as duplications, inversions, or recombinations, and there is still a lot of work to be done to address all these sources of variability.

Formally speaking, the Multiple Sequence Alignment (MSA) issue is as follows:

Definition 2 Given a set of sequences $S = s_1, \ldots, s_n$ on a finite alphabet Σ, an MSA of S is a set of sequences $A = \{a_1, \ldots, a_n\}$ on $\Sigma \cup \{\text{“}-\text{”}\}$, where “ $-$ ” is a new letter representing insertion/deletion events (gaps) in the aligned sequences. Moreover, all elements of A have the same length and all a_i are equal to s_i up to the deletion of the “ $-$ ” characters. Given a cost or score function c on pairs of sequences in an MSA, the MSA issue looks for an MSA A optimizing the value of a function of c.

A frequent choice in this general definition is the Sum of Pairs criterion (SP) that minimizes the sum $\Sigma_{i<j}\, c(a_i, a_j)$, where c is typically defined as a sum along the alignment positions of a scoring of letters at these positions and the gaps receive a special treatment with an affine function. In this setting and its variants, MSA is an NP-complete problem and, for this reason, using a standard dynamic programming approach provides an exact solution in practice only for a very small number of sequences, typically no more than 3. Artificial intelligence took an early interest in this problem since it can be reduced to finding a shortest path in a huge graph joining the possible alignment positions of characters in each sequence and a branch and bound algorithm can be applied (Gupta et al. 1995). Variants of the A* algorithm (see chapter “Heuristically Ordered Search in State Graphs” of Volume 2) have been developed by Ikeda et al. and Yoshizumi et al. both for the exact and the approximated case (Ikeda and Imai 1999; Yoshizumi et al. 2000). This gave rise to a number of papers on space-efficient or faster heuristics (Korf et al. 2005; Schroedl 2005; Zhou and Hansen 2004), or the recursive best first search MREC enabling the exact optimal alignment of up to 11 sequences (Koshino et al. 2006). Recent works are focusing on solutions using external disk space, adapted to the best first search order, and multi-threaded computation (Hatem and Ruml 2013; Sundfeld et al. 2017), pushing further the limits of exact methods.

A number of suboptimal approaches have been developed and are still to be developed, for instance to scale to large sets of sequences (million) or large sequences (whole genomes). A basis of almost all these approaches is the use of a progressive strategy, starting from pairwise alignments and trying to combine them in the best ordering. Following the works of Korostensky and Gonnet, it is possible to define the progressive alignment using a circular sum measure where each sequence is aligned with exactly two sequences (Gonnet et al. 2000; Korostensky and Gonnet 1999). Multiple alignment is reduced this way to a Traveling Salesman Problem, where the goal is to find the circular ordering of sequences minimizing the sum. Although this approach has some relevance with respect to evolution, it appears to have not been pursued apart from a small piece of work in Abu-Srhan and Al Daoud (2013). The progressive strategy is often combined with an iterative strategy, where solutions are

progressively refined in order to improve the alignment. Stochastic optimization is useful in this case. In Omar et al. (2005), a genetic algorithm is used for the progressive part and a simulated annealing algorithm is used for the iterative part. Many aligners have been developed but none outperforms the other in all cases, depending on the properties of sequences (presence of domains, partial sequences, intrinsically unstructured regions, alternatively regions with known 2D/3D structure, etc.). A natural approach is then to propose meta-methods running a number of algorithms in parallel and choosing the best alignment in the different results (Muller et al. 2010). More recently, the concept of assisted multiple alignment has emerged as an important issue for more efficient and more relevant alignments. AlexSys is an expert system in protein multiple sequence alignment that learns rules predicting for each method if it is suited or not to the sequences to be aligned (Aniba et al. 2010).

6.2 Building Phylogenetic Trees

Given a set of species (or taxonomic units), each one being usually represented by a subset of its gene sequences, molecular phylogenetic studies try to infer a phylogenetic tree that reflects the actual lineages of species during evolution. Multiple alignments are just one (important) source of data for building phylogenetic trees. Numerous other sources of information are used to build, compare and reconcile trees. The following definition is adapted from Brooks et al. (2007), Erdem (2011), Miranda et al. (2014)

Definition 3 A phylogeny is a septuple $P = (V, E, F, C, D_C, v, \mathscr{L})$, where (V, E) is a graph commonly describing a rooted binary tree, F is the set of terminal nodes of the graph (leaves of the tree), C is a set of qualitative attributes, the characters, with domains D_C, v is a function giving the value of each attribute for each terminal node in F, and $\mathscr{L}$ is an optional function that provides a real length for each edge. Variants exist with unrooted trees, non binary trees, or even phylogenetic networks that take into account the possibility of a reticulate evolution due to the exchange of genes between species (horizontal transfer).

The characters may be binary and, moreover, cladistic (the values are ordered during evolution from an ancestral state to derived states). When data are made of genetic sequences, characters are positions in a multiple alignment with at least two different letters (called SNPs). These mutation positions are generally binary characters. Nodes of the tree may be considered as states of the evolution process and edges as transformations, such as mutations. Considering only its structure (without function $\mathscr{L}$), a general problem is to build a tree on a set of species that is correct with respect to a given set of characters.

The phylogeny inference problem Given a set F of taxonomic units and a triple C, D_C, v providing character values for each element in F, decide if there exists a phylogeny that fulfills one of these criteria:

- k-compatibility: at least k characters must be compatible with the tree. A character is compatible with a phylogeny tree if the set of all vertices having the same value for this character forms a subtree.
- k-parsimony: the tree may be mapped to a rectilinear Steiner tree with a size at most k. In such a tree, edges have an integer length that is positive or zero. The paths between two elements of F have a length that equals the Manhattan distance between the vectors of their characters.
- d-goodness-of-fit: Assumes the analysis of characters to be summarized with a symmetric dissimilarity matrix D between pairs of elements of F. The tree must have a goodness-of-fit at least d with respect to D. Given a phylogenetic tree with a function $\mathscr{L}$ associating a value with each of its edges, it is possible to build a symmetric matrix P giving the path length between pairs of elements of F. The goodness-of-fit is the Euclidean distance between P and D (i.e. the Frobenius norm of the difference of the two matrices).

All these problems and variants have been proved NP-complete by Day and Sankoff. Apart from the decision problems, people often look for optimization versions, for instance looking for a tree with a maximum number of compatible characters. A topological criterion may also be used to build the tree from unrooted trees produced on subsets of taxonomic units, typically quartets of species.

The whole setting may be a bit more complex since a number of assumptions have to be added to obtain realistic trees, the main one being about the evolution of characters.

In a tree, starting from a state where all characters are considered absent, a character may be gained once and for all (perfect phylogeny), gained once and then lost at most once (persistent phylogeny), gained several times but never lost (Camin-Sokal criterion), etc. Constraint modeling frameworks can adjust with great flexibility to these many different criteria. Integer Linear Programming has been successful in providing different models, depending on the type of data to be processed (Sridhar et al. 2008) (perfect phylogenies, maximum parsimony), (El-Kebir et al. 2015) (perfect phylogenies from tumor multisample sequences), (Gusfield 2015) (persistent phylogenies), (Bonizzoni et al. 2017) (incomplete perfect phylogenies on tumoral sequences). Answer set programming has been used in Brooks et al. (2007) (perfect phylogenies), (Kavanagh et al. 2006) (Camin-Sokal criterion) and (Wu et al. 2007) (maximum quartet consistency).

In practice, even with these assumptions, a number of equivalent solutions may exist and it is fundamental to put the experts in the decision loop. This is why in recent years algorithmics is not the sole concern of evolutionary bioinformatics. There is a growing interest in a knowledge-based approach in this field since it brings together

a large community and many results that may be partially contradictory have to be integrated. The ability to build complex queries and check combinatorial properties has shed light on logical approaches. In particular, if one considers a phylogenetic tree as a transition system (by adding loops on the leaves), it is possible to apply a temporal logic to make queries on the tree properties, the state properties or a mix of both. Requeno et al. (2013) have thus developed a model checking framework where it is possible to use CTL on phylogenetic trees. The possibility to obtain not only verifications but also counter-examples if formulas are not satisfied is important in a practical interactive context where the user makes an intensive use of queries to mine the trees (e.g. checking if there are back mutations in the tree can be used to detect these mutational events). This framework has been extended to the treatment of quantitative information through the use of stochastic logics (Requeno and Colom 2016). This allows to introduce in queries some probabilities and an explicit time. It is of significant interest if one wants to test models of evolution and to compute maximum likelihood estimations for trees in this context.

Other logical frameworks such as Answer Set Programming have been used. For instance, the supertree construction problem, which consists of building a tree that is maximally consistent with a set of trees built on overlapping sets of species, has been encoded as an ASP model in Koponen et al. (2015). A web service interface API has been developed in ASP for TreeBASE, a relational database designed to manage and explore information on phylogenetic relationships (Le et al. 2012) and a toolkit has been developed for the alignment, consistency checking and inconsistency repair of taxonomies, using various reasoning systems (first order, Answer Set and dedicated provers) (Chen et al. 2013; Franz et al. 2015). Note that the alignment of trees may be used to build a phylogeny with a divide and conquer approach in order to improve search efficiency. For instance, Ford et al. (2015) describes a method splitting the set of characters into subsets for which the search for a perfect phylogeny is possible and then use the 'anchor' trees built on these subsets to constrain the search of the whole tree.

7 Drug Discovery

High throughput screening (HTS) refers to a set of techniques aiming at identifying biologically active molecules that exhibit useful properties among elements of a large database of chemical compounds. The selection of these candidate molecules together with an accurate prediction of their molecular activity is an important economic issue. In particular, such databases are used by the chemical industry and have a major value in pharmacology for developing new drugs and reduce the need for animal testing. They can contain millions of components. The key problem is to establish *structure-activity relationships* (SAR), i.e., to predict a biological activity from molecular descriptors and some knowledge on physico-chemical properties of chemicals. If we try to predict a degree of activity, we will talk about QSAR (quantitative SAR) and when activity is replaced by other physicochemical prop-

erties, we will talk about SPR (structure-property relationships). This question is at the crossroads of chemoinformatics and bioinformatics. Solutions are based on the assumption that "similar" molecules generally share a similar activity but it is far from being so simple in reality, some minor structural changes being enough to completely change the activity of a molecule. This problem is generally referred as the *activity cliff* (Cruz-Monteagudo et al. 2014; Dimova and Bajorath 2016).

A fundamental notion in biochemistry is that of receptor/ligand interaction. A ligand is a substance that forms a complex with a biomolecule. Ligand binding to a receptor protein changes the 3D conformation and thus the functional state of this protein. When the structure of the target protein is known, the most commonly used approach is molecular docking (see Sect. 4) and the approach is called structure-based drug design (SBDD). There are now a number of drugs whose development was heavily influenced by SBDD, such as HIV protease inhibitors (Kitchen et al. 2004). A recent review of work in this area is available in Ferreira et al. (2015). When the structure is unknown, the approach is called ligand-based drug design (LBDD) and the research described in the rest of this section mostly falls within this approach. In fact, most recent methods try to associate similarities of both ligands and receptor protein by concatenating their descriptors for learning classifiers. This computational chemogenomic approach is particularly useful for a case-based/analogical reasoning approach (see chapter "Case-Based Reasoning, Analogical Reasoning, Interpolation" of Volume 1), although it does not appear to have been formalized in these terms (Brown et al. 2013). It applies either when searching for a new ligand on the basis of similarity with other ligands having known targets, or vice versa when searching for a new target on the basis of similarity with proteins having known active ligands.

Biological activity is generally dose-dependent and can be described by many parameters. In pharmacology, it is represented by two types of attributes, the activity of the target and the toxicity of the drug, which is itself described along four dimensions reflecting the life cycle of the substance and the different aspects of its transformation in the organism such as bio-availability or biodegradability (ADME: "absorption, distribution, metabolism, and excretion"). If all data and attributes can be turned into numerical or ordered values, it is possible to build mathematical functions that can predict the activities of new chemicals. Historically, the structure-activity study was based on simple models where the degree of activity was assumed to be a linear function of the measured properties (e.g. hydrophobicity) on chemical compounds. Simple statistical methods such as linear regression were typically applied for such studies. Since then, machine learning techniques have played an increasingly important role in this field (Lavecchia 2015). In its simplest form, the drug discovery challenge may be formalized as the following machine learning issue:

Definition 4 Given a set of graphs labeled as positive or negative instances of molecular compound fragments with a given property, build a predictor for this property enabling the classification of new compounds.

The tested property can take various forms: activity (active/nonactive or a degree), drug-likeness, ADME property (absorption, solubility or permeability, metabolic stability, etc.), toxicity.

A number of open-source or commercial rule-based systems have been developed and are still used to solve this prediction problem in various domains such has skin sensitization, hepatotoxicity, or carcinogenic compounds (Raies and Bajic 2016). Knowledge-based expert systems are routinely used to predict potential chemical toxicity, on the basis of qualitative evidence. For instance, Toxtree (Benigni and Bossa 2008) operates with a a manually designed set of rules for evaluating the mutagenic/carcinogenic potential of chemicals. At a finer level, predicting xenobiotic metabolism (the way an organism degrades a chemical compound that it does not naturally produce in several metabolites) seems more challenging since a lot of false-positive can be produced (Judson 2014).

Besides expert systems, which assume the existence of a large knowledge base, there are of course automated prediction methods. The main methods in this field are Bayesian methods and SVM. The most recent ones use deep learning. Bayesian networks (see chapter "Belief Graphical Models for Uncertainty Representation and Reasoning" of Volume 2) have been used in Abdo et al. (2010, 2014). Authors propose to train a Bayesian belief network and use it in Abdo et al. (2014) to infer the activity class of a target compound. In the network, there is a terminal node for the target compound and other terminal nodes for sets of compounds known tho share some activity. The root nodes of the network represent the presence of specific fragments (substructures) in compounds. The calculation of conditional probabilities is adapted to the graph structure of chemical data and the target molecule is assigned to the most similar class based on the presence of common fragments. SVM (see chapter "Statistical Computational Learning" of Volume 1) have been used with Gaussian as well as simple linear kernels (Hinselmann et al. 2011). More specific kernels have been designed (Vert and Jacob 2008), in particular graph-kernels that work on labeled graphs (Mahé and Vert 2009). Decision trees have also been used with some success, particularly Random Forests. For instance members of Pfizer showed that RF can produce results as good as SVM for predicting the relationship between the chemical structure of a compound and its metabolic stability (Sakiyama et al. 2008). More recently, they have proved to be interesting for toxicology prediction (Tox21 challenge, Banerjee et al. 2016). RF have also been used for the protein-ligand docking problem in Ballester and Mitchell (2010). As for protein structure prediction (see Sect. 4), deep learning has allowed to decrease the necessity to select optimal descriptors, although to the detriment of explainability of predictions. A good review of deep learning approaches in this field is available in Gawehn et al. (2016). Note that the search space is huge when crossing available chemical compounds and target proteins and the incompleteness of databases is a serious concern for all these methods (Mestres et al. 2008). For this reason, a number of authors stress the importance of *active learning* to better select the relevant part of databases or relevant experiments in order to transfer the available knowledge to new cases of protein-ligand association (Naik et al. 2016; Reker et al. 2016, 2017; Wei et al. 2015). In Reker et al. (2017), authors use Random Forests for the learning component and propose a 'curiosity' criterion to select incrementally the relevant interactions in the database of known interactions. The curiosity measure selects each time the interaction for which there is the least consensus among the decision trees when

classifying the interaction as active or not. This strategy, which is compatible with incremental selection of the most interesting experiments to refine the knowledge base, is also proving effective for machine learning using 10 to 20 percent of the database.

Except when drug discovery is based on deep learning techniques, the preprocessing phase is itself a hard problem, since it is necessary to extract from chemical databases the structural fragments that will be used as instances in the previous problem. This leads to a data mining problem:

Definition 5 Given a set of graphs representing molecular compounds, build a set of frequently occurring subgraphs.

For a review of the multiple descriptors that have been used to represent molecular data, see Sawada et al. (2014). The structural and physicochemical fragments at the origin of the biological behavior of chemicals are often called *structural alerts* in the literature. One of the early AI approach for this problem is the system CASE (and later MultiCASE) (Rannug et al. 1991), which considers structural subunits containing less than 10 connected heavy atoms and learns if they are active by measuring their occurrence probability with respect to a binomial distribution. Since then, the graph data mining approach has been prevalent (Sherhod et al. 2014; Takigawa and Mamitsuka 2013). The search space of frequent subgraphs is explored either with a Breadth-First strategy (Apriori approach) or with a Depth-First strategy (Pattern-Growth approach). The AGM method (Inokuchi et al. 2000) uses the Apriori approach and works on canonical forms of graph adjacency matrices. It incrementally increases the size of matrices, merging at step $k + 1$ frequent matrices of size k resulting from step k. An interesting extension of this track of research proposes to integrate Apriori with the Version Space framework in order to produce the most specific and general molecular fragments corresponding to toxic compounds (De Raedt and Kramer 2001; Helma et al. 2002). This is in the full continuity of the founding work on the Meta-Dendral system that we mentioned in the introduction to this chapter. Methods using the Pattern-Growth approach are described in Lepailleur et al. (2013). From frequent atoms, they build increasingly larger frequent molecules by adding new bonds.The authors highlight the interest of searching for discriminating patterns (emerging patterns, jumping patterns) by considering frequency ratios (growth rate) in addition to frequencies, and especially most specific discriminating patterns (closed or representative pruned molecular patterns).

In Shao et al. (2015), the issue is set as mining discriminant subgraphs from graph data with multiple labels and it is shown how produced subgraphs can be applied to drug adverse effect prediction problem.

Among recent works, the notion of Pareto dominance with respect to a set of user-preference measures has proved to be of high importance for the selection of useful patterns (called skypatterns). The work described in Ugarte et al. (2017) proposes a static method whose efficiency is based on a condensed representation of patterns and a dynamic method whose effectiveness is based on improved pruning through iterative use of the patterns produced to refine dominance constraints. It is applied to the search of toxic chemical fragments, using as preference measures frequency,

growth rate and chemical aromaticity. Authors have used the DynCSP framework of dynamical constraint solving (Verfaillie and Jussien 2005) for posting new constraints from current sky patterns and the Gecode toolkit (Schulte and Stuckey 2008) for the implementation.

8 Glycobiology

As for proteins, understanding the biological functions of carbohydrates (glycans) and relating them to their structure remains experimentally difficult and classification, machine learning or data mining methods are needed to propose general models or predictors of these functions. Unlike proteins, the information in databases (Kanehisa 2017; Pérez et al. 2015; Tiemeyer et al. 2017) are rooted trees with ordered children (up to 5) and not simply sequences, a characteristic that introduces interesting challenges. Moreover, the basic units of glycans (the nodes of the tree, monosaccharides) exist in numerous derived forms (e.g. more than 100 in bacteria). The root is a specific sugar that binds to its environment (cell or protein). The edges are made of several types of sugar bonds (say a dozen). The total number of glycans is estimated to be in the order of hundreds of thousands. This renders the representation of carbohydrates a difficult problem, more difficult than the analysis of trees that appear in RNA folding secondary structures.

Ontologies have started to be developed for this field. For instance, GlycoRDF (Ranzinger et al. 2015) proposes a standard OWL ontology that gives access for a number of glycomics databases to an RDF representation of various data ranging from publications relating to glycan structures to experimental datasets. Glycomics offers a nice setting to compare different technologies for graph databases or knowledge bases. For instance, RDF and Property Graph representations have been compared in respect of the glycan substructure search issue (Alocci et al. 2015), showing a clear advantage for RDF representations.

The physical recognition of glycan structures is currently treated by tandem mass spectrometry (MS/MS) and liquid chromatography. Two approaches are possible to infer a glycan structure from its MS/MS spectrum. In the simplest case we can use a large curated database of already known structures together with their spectra and develop a matching program for the annotation of a new spectrum. Sparql queries through RDF technology can be sufficient in this case. The other approach (de novo sequencing) tries to assign structures to peaks of the spectrum without any database. This requires machine learning methods to help peak assignment. The Glyfon program (Kumozaki et al. 2015) builds from a spectrum a graph of possible monosaccharide assignments and their links to other peaks using information on mass difference between two peaks and searches the space of all assignments compatible with a realistic glycan structure using an integer programming approach with Lagrangian relaxation. The parameters of the objective function are learned through a structured SVM, a task made tricky by the availability of a training set of structure-spectrum pairs, but not the corresponding residue-peak pairs.

One of the main demands in glycomics bioinformatics tasks is the data mining of glycan structure databases to classify glycans and discriminate the classes on the basis of the structural patterns that they contain (Mamitsuka 2011). Genetic programming has been adapted in Miyahara and Kuboyama (2014) to learn glycan motifs through the use of tag tree patterns. An interesting adaptation of SVM classifiers has been proposed in Yamanishi et al. (2007), who introduce tree kernels for glycans. In practice, a tree kernel measures the similarity between two trees by counting the number of common subtrees, possibly with the same size and/or the same depth, and a powerful restriction is to consider only subtrees that are close with respect to the sibling relation (co-rooted trees).

Since motifs are as important as the identification of classes, the choice of features has to be compatible with a feature selection method in order to extract the high-scoring subtrees. Authors have applied this work with success to the task of predicting the blood origin of glycans among leukemic and non-leukemic blood cell types and finding a glycan motif typical of leukemic cells. Instead of using a dedicated kernel and then extracting features, some authors have tried to directly produce the relevant attributes through pattern mining in glycan structures, the presence of each frequent subtree becoming a binary attribute. A method is proposed in Takigawa et al. (2010), which claims better results on a mixed set of existing and randomly synthesized glycans than the previous method. Frequent subtrees are extracted using two criteria, the search of subtrees that are significantly more frequent than the tree they come from and the search of significant subtrees with respect to a Fisher test using a control dataset.

A more ambitious approach for analyzing the structure of glycans is to use formal grammars. This way, one can not only discover motifs (associated to non-terminals of the grammar) but also the hierarchical relations they share (the rules of the grammar). T. Akutsu has introduced elementary ordered tree grammars for this purpose (Akutsu 2010), where production rules use trees with edges labeled either using terminal or non-terminal symbols and one leaf in the tree may be tagged to indicate where another tree may be attached. Grammars are restricted to Chomsky's normal form (two non-terminals on the right-hand side). The idea is then to use grammar-based compression as a criterion to find interesting structures: the problem is to find the smallest grammar that generates exactly a given tree. An integer programming model is proposed in Zhao et al. (2010), with a small experimentation on glycan trees labeled with the types of monosaccharides and the use of Cplex as a solver. This work is extended in Zhao et al. (2015) to take into account multiple trees, this time using glycans labeled with the glycosyl transferases that enable the linkage of monosaccharides in the construct (a small experiment on RNA secondary structures is also provided). An interesting point in this extension is that a grammar could directly reflect the construction process of the molecule.

In this respect, glycobiology offers a specific application field for pathway reconstruction techniques (see Sect. 5). Indeed, the formation of each glycan structure results from dedicated biochemical pathways using polymerization reactions catalyzed by specific enzymes. One important question is thus to associate one or several genes corresponding to these enzymes to reactions that progressively transform

a glycan structure. This can be studied in bacteria thanks to knockout experiments observing the effect of discarding a gene on the produced structures and the presence/absence of genes in related strains, as it is used in Sternberg et al. (2013). This paper demonstrates the application of inductive logic programming (Progol) to learn the gene-rule associations. Authors emphasize the interesting fact that aside from using some background knowledge on biochemistry (pathways, decomposition of glycan) and strain serotypes, it is necessary to introduce some speculative assumptions to better score the competing hypotheses. As in other areas of Artificial intelligence, it appears that the formalization of preferences has been key to successful predictions.

It seems that there is still scope for other research on this problem, using other techniques, and to our knowledge, no grammatical inference method has been applied so far to look at a grammar generalizing a positive training set of glycan structures and possibly rejecting a negative training set. The last study also points to the interest in the development of preference reasoning and possibly preference learning to help select from the predictions a reasonable subset of hypotheses that will be the subject of experimental testing.

There are also probabilistic approaches that have proposed extensions of Hidden Markov Models for the treatment of ordered trees. Among the most interesting ones, Ordered Tree Markov Model (OTMM) considers dependencies between parents and their first child and dependencies between ordered children (Ueda et al. 2005), and profilePTSMM (Probabilistic Sibling-dependent Tree Markov Model) considers two different types of transition dependencies between parents and all their children and dependencies between ordered children, together with the introduction of match/delete and insert states as in profile HMM (Aoki-Kinoshita 2015; Aoki-Kinoshita et al. 2006).

Another common task in glycomics is the prediction of the glycosylation state of proteins. There are four types of glycosylation, the main ones being O-linked and N-linked glycosylation, then C-linked glycosylation. It is known to occur on particular sites in the protein, partially characterized by short sequence motifs. The issue is to predict the glycosylation type and the sites.

The glycosylation site prediction problem

- Given a type of glycosylation, given a set P of proteins with known sequence and glycosylation sites and some optional background knowledge providing functional features or annotations for any protein,
- Build a classifier that can predict the glycosylation sites of this type in a new protein.

Of course, it is possible to state the problem as a three- or four-class problem instead of building one classifier for each type.

As in many bioinformatics applications, the difficulty is to find a trade-off between the number of parameters of the learned models and the relative scarcity of available data. Several predictors are often combined to this end.

For example, in Senger and Karim (2008) a set of recurrent Elman networks are trained to predict the major presence of a certain type of glycans in N-linked glycosylation of proteins, the training data being provided by predictors of the secondary structure and the accessibility state of these proteins from their sequence. In Chauhan et al. (2013), using the same type of information, the three major types of glycosylation are predicted with an SVM-based approach, using a Gaussian RBF kernel and a carefully selected non-redundant dataset. Authors have chosen this approach after testing numerous methods available in the Weka machine learning toolkit (namely Random Forest, Logistic Model Trees, various types of SVM in libsvm and with Sequential Minimal Optimization SMO, Bayesian network and naive Bayes). The philosophy of the best methods is to use a maximum number of attributes, including derived attributes that result from auxiliary predictors, and to add a feature selection stage to avoid overfitting. Among state-of-the-art methods at the time of this review, GlycoMine (Li et al. 2015) makes use of a knowledge base extracted from a number of databases of protein features (Gene ontology, Kegg, Pfam, Uniprot, etc.) and uses a feature selection procedure based both on mutual information and information gain.

It is likely that many problems studied on sequences will have an extension on glycan trees. It is thus a new field of study and application of AI techniques developed for sequences to these more complex structures. For instance, algorithms have been developed recently for glycan multiple alignments (Hosoda et al. 2017) and it would be interesting to check ideas developed on sequence multiple alignment (see Sect. 6) in this new context.

9 Conclusion

Bioinformatics is a field full of incomplete data, knowledge expertise and NP–complete problems, and is as such a playground offering many opportunities for Artificial intelligence studies. This exciting interdisciplinary field comes at a cost. The first difficulty is to cope with the rapidly advancing technology. It is not always easy to distinguish short term problems that will be rapidly obsolete thanks to the next generation technology from more fundamental issues that are created by accessing a new kind of data. Two significant trends seem, however, to be emerging in this field.

First, there is an extensive use of weighted sequence data to cover all omics observations, giving both access to their qualitative and quantitative content. A weight may be a quality score that reflects the probability that a letter at a given position in the sequence was correctly observed by the sequencer (this tends to be standardized in file formats like FastQ, which codes letters as well as quality by ASCII characters) or an abundance (read count) that reflects the degree of expression of an element of the sequence in the observed sample. Sequencing is no longer reduced to the analysis of DNA and is now applied as a high-throughput technology for the identification of

chromosomal 3D structures, the observation of epigenetic factors like DNA methylation or histone modification, the analysis of coding and non-coding types of RNA, translation efficiency of proteins (MGlincy and Ingolia 2017), and all the interactions like the interactions between DNA and proteins (Soon et al. 2013). This unification from the technological side is good news for all researchers in Bioinformatics since it provides some stability to their results.

Secondly, access to individual observations, which started with genomes of individual organisms and culminates now in the development of single cell analysis (Baron and Yanai 2017; Bock et al. 2016; Gawad et al. 2016; Yuan et al. 2017), is a clear breakthrough in molecular biology. Understanding the pool of variations that lead to diseases, analyzing a microbial community and the exchanges that occur between its elements and tracking the embryonic development of a pool of cells, are all attainable with the technological developments. It opens the door to personalized medicine and to the rational representation and understanding of populations at all levels. Most likely, new types of intelligent models and methods are needed and will emerge to address these new challenges.

Once the raw data have been preprocessed, what makes studies successful is a deep understanding of the peculiarities of a particular biological question. The devil is in the detail. These peculiarities are key to addressing the complexity and push forward the boundaries of feasibility, but it is also a guarantee that relevant solutions for the biologist will be proposed. Of course, it is possible to work with standard benchmarks and build on the accomplishments of predecessors. But there are plenty of opportunities to participate in biological discoveries. The only advice I could address to AI people eager to start out in this field and contribute to these discoveries is to keep their focus on a specific set of questions and to ensure that there is a biologist expert on these questions in the loop.

Besides standard biology, an alternative and much more prospective route to cooperation between computer science and biology is synthetic biology, a field interested by life engineering, trying to design living components with simplified, fully controlled behaviors that can be assembled. It is not only a question of introducing a new technology: since experiments can be better controlled, it provides key to an in-depth understanding of living systems. This could be inspiring for AI in its goal of better understanding the components of intelligence. Neurons are no more the sole interesting cells in this respect. It has been shown by T. Nakagaki for instant that even primitive systems like ciliates or slime are able to memorize and have learning capacities. Some authors start to think of BI (Bio-artificial Intelligence) after AI (Nesbeth et al. 2016) by looking at ways to implement learners with synthetic gene and protein networks.

Acknowledgements I would like to thank authors of the French version of this chapter who offered me a primary material of quality to start this English version: F. Coste, C. Nédellec, Th. Schiex and J.-P. Vert. Thanks also to O. Dameron and F. Coste for their proofreading of the manuscript.

References

Abdallah EB, Folschette M, Roux O, Magnin M (2017) ASP-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks. Algorithms Mol Biol 12(1):20

Abdo A, Chen B, Mueller C, Salim N, Willett P (2010) Ligand-based virtual screening using Bayesian networks. J Chem Inf Model 50(6):1012–1020

Abdo A, Leclère V, Jacques P, Salim N, Pupin M (2014) Prediction of new bioactive molecules using a Bayesian belief network. J Chem Inf Model 54(1):30–36. PMID: 24392938

Abou-Jaoudé W, Traynard P, Monteiro PT, Saez-Rodriguez J, Helikar T, Thieffry D, Chaouiya C (2016) Logical modeling and dynamical analysis of cellular networks. Front Genet 7

Abu-Srhan A, Al Daoud E (2013) A hybrid algorithm using a genetic algorithm and cuckoo search algorithm to solve the traveling salesman problem and its application to multiple sequence alignment. Int J Adv Sci Technol 61:29–38

Acuna V, Chierichetti F, Lacroix V, Marchetti-Spaccamela A, Sagot M-F, Stougie L (2009) Modes and cuts in metabolic networks: complexity and algorithms. Biosystems 95(1):51–60

Adhikari B, Bhattacharya D, Cao R, Cheng J (2015) CONFOLD: residue-residue contact-guided ab initio protein folding. Proteins: Struct Funct Bioinform 83(8):1436–1449

Akutsu T (2010) A bisection algorithm for grammar-based compression of ordered trees. Inf Process Lett 110(18–19):815–820

Akutsu T, Hayashida M, Ching W-K, Ng MK (2007) Control of boolean networks: hardness results and algorithms for tree structured networks. J Theor Biol 244(4):670–679

Akutsu T, Kosub S, Melkman AA, Tamura T (2012) Finding a periodic attractor of a Boolean network. IEEE/ACM Trans Comput Biol Bioinform 9(5):1410–1421

Albert R, Thakar J (2014) Boolean modeling: a logic-based dynamic approach for understanding signaling and regulatory networks and for making useful predictions. Wiley Interdiscip Rev: Syst Biol Med 6(5):353–369

Allouche D, André I, Barbe S, Davies J, de Givry S, Katsirelos G, O'Sullivan B, Prestwich S, Schiex T, Traoré S (2014) Computational protein design as an optimization problem. Artif Intell 212:59–79

Alocci D, Mariethoz J, Horlacher O, Bolleman JT, Campbell MP, Lisacek F (2015) Property graph vs RDF triple store: a comparison on glycan substructure search. PLOS ONE 10(12):1–17

Aniba MR, Poch O, Marchler-Bauer A, Thompson JD (2010) Alexsys: a knowledge-based expert system for multiple sequence alignment construction and analysis. Nucleic Acids Res 38(19):6338

Antonov I, Borodovsky M (2010) Genetack: frameshift identification in protein-coding sequences by the viterbi algorithm. J Bioinform Comput Biol 8(03):535–551

Aoki-Kinoshita KF (2015) Analyzing glycan-binding patterns with the ProfilePSTMM tool. Springer, New York, pp 193–202

Aoki-Kinoshita KF, Ueda N, Mamitsuka H, Kanehisa M (2006) ProfilePSTMM: capturing tree-structure motifs in carbohydrate sugar chains. Bioinformatics 22(14):e25–e34

Arellano G, Argil J, Azpeitia E, Benítez M, Carrillo M, Góngora P, Rosenblueth DA, Alvarez-Buylla ER (2011) Antelope: a hybrid-logic model checker for branching-time Boolean GRN analysis. BMC Bioinformatics 12(1):490

Aronson SJ, Rehm HL (2015) Building the foundation for genomics in precision medicine. Nature 526(7573):336–342

Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT et al (2000) Gene ontology: tool for the unification of biology. Nat Genet 25(1):25–29

Awada W, Khoshgoftaar TM, Dittman D, Wald R, Napolitano A (2012) A review of the stability of feature selection techniques for bioinformatics data. In: 2012 IEEE 13th International conference on information reuse and integration (IRI). IEEE, pp 356–363

Backofen R, Will S (2006) A constraint-based approach to fast and exact structure prediction in three-dimensional protein models. Constraints 11(1):5–30

Baldi P, Brunak S (2001) Bioinformatics: the machine learning approach. MIT Press, Cambridge

Ballester PJ, Mitchell JBO (2010) A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. Bioinformatics 26(9):1169–1175
Banerjee P, Siramshetty VB, Drwal MN, Preissner R (2016) Computational methods for prediction of in vitro effects of new chemical structures. J Cheminformatics 8(1):51
Barahona P, Krippahl L (2008) Constraint programming in structural bioinformatics. Constraints 13(1):3–20
Barberis M, Todd RG, van der Zee L (2017) Advances and challenges in logical modeling of cell cycle regulation: perspective for multi-scale, integrative yeast cell models. FEMS Yeast Res 17(1)
Bard JB, Rhee SY (2004) Ontologies in biology: design, applications and future challenges. Nat Rev Genet 5(3):213
Baron M, Yanai I (2017) New skin for the old rna-seq ceremony: the age of single-cell multi-omics. Genome Biol 18(1):159
Batt G, De Jong H, Page M, Geiselmann J (2008) Symbolic reachability analysis of genetic regulatory networks using discrete abstractions. Automatica 44(4):982–989
Batt G, Besson B, Ciron P-E, de Jong H, Dumas E, Geiselmann J, Monte R, Monteiro PT, Page M, Rechenmann F, Ropers D (2012) Genetic network analyzer: a tool for the qualitative modeling and simulation of bacterial regulatory networks. Springer, New York, pp 439–462
Baú D, Martin AJ, Mooney C, Vullo A, Walsh I, Pollastri G (2006) Distill: a suite of web servers for the prediction of one-, two- and three-dimensional structural features of proteins. BMC Bioinform 7(1):402
Bechhofer S, Buchan I, De Roure D, Missier P, Ainsworth J, Bhagat J, Couch P, Cruickshank D, Delderfield M, Dunlop I et al (2013) Why linked data is not enough for scientists. Future Gener Comput Syst 29(2):599–611
Bellazzi R (2014) Big data and biomedical informatics: a challenging opportunity. Yearb Med Inform 9(1):8
Benigni R, Bossa C (2008) Structure alerts for carcinogenicity, and the Salmonella assay system: a novel insight through the chemical relational databases technology. Mutat Res/Rev Mutat Res 659(3):248–261
Beretta S, Bonizzoni P, Vedova GD, Pirola Y, Rizzi R (2014) Modeling alternative splicing variants from RNA-seq data with isoform graphs. J Comput Biol 21(1):16–40
Berger B, Leighton T (1998) Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. J Comput Biol 5(1):27–40
Bhaskar H, Hoyle DC, Singh S (2006) Machine learning in bioinformatics: a brief survey and recommendations for practitioners. Comput Biol Med 36(10):1104–1125. Intelligent technologies in medicine and bioinformatics intelligent technologies in medicine and bioinformatics
Bizer C, Heath T, Berners-Lee T (2009) Linked data-the story so far. Int J Semant Web Inf Syst 5(3):1–22
Blake JA, Bult CJ (2006) Beyond the data deluge: data integration and bio-ontologies. J Biomed Inform 39(3):314–320. Biomedical ontologies
Blaszczyk M, Jamroz M, Kmiecik S, Kolinski A (2013) CABS-fold: server for the de novo and consensus-based prediction of protein structure. Nucleic Acids Res 41(W1):W406–W411
Blaszczyk M, Gront D, Kmiecik S, Ziolkowska K, Panek M, Kolinski A (2014) Coarse-grained protein models in structure prediction. Springer, Berlin, pp 25–53
Bock C, Farlik M, Sheffield NC (2016) Multi-omics of single cells: strategies and applications. Trends Biotechnol 34
Bonizzoni P, Ciccolella S, Della Vedova G, Soto M (2017) Beyond perfect phylogeny: multisample phylogeny reconstruction via ILP. In: Proceedings of the 8th ACM international conference on bioinformatics, computational biology, and health informatics, ACM-BCB '17, ACM, New York, USA, pp 1–10
Bouziane H, Messabih B, Chouarfia A (2015) Effect of simple ensemble methods on protein secondary structure prediction. Soft Comput 19(6):1663–1678
Boyer F, Viari A (2003) Ab initio reconstruction of metabolic pathways. Bioinformatics 19(suppl_2):ii26–ii34

Brahim AB, Limam M (2017) Ensemble feature selection for high dimensional data: a new method and a comparative study. Adv Data Anal Classif 1–16
Brim L, Češka M, Šafránek D (2013) Model checking of biological systems. In: Formal methods for dynamical systems. Springer, Berlin, pp 63–112
Brooks DR, Erdem E, Erdoğan ST, Minett JW, Ringe D (2007) Inferring phylogenetic trees using answer set programming. J Autom Reason 39(4):471–511
Brown JB, Niijima S, Okuno Y (2013) Compound protein interaction prediction within chemogenomics: theoretical concepts, practical usage, and future directions. Mol Inform 32(11–12):906–921
Cannata N, Schröder M, Marangoni R, Romano P (2008) A semantic web for bioinformatics: goals, tools, systems, applications. BMC Bioinform 9(4):S1
Caravagna G, Graudenzi A, Ramazzotti D, Sanz-Pamplona R, De Sano L, Mauri G, Moreno V, Antoniotti M, Mishra B (2016) Algorithmic methods to infer the evolutionary trajectories in cancer progression. Proc Natl Acad Sci 113(28):E4025–E4034
Carrillo M, Góngora PA, Rosenblueth DA (2012) An overview of existing modeling tools making use of model checking in the analysis of biochemical networks. Front Plant Sci 3
Chapman SD, Adami C, Wilke CO, KC DB (2017) The evolution of logic circuits for the purpose of protein contact map prediction. PeerJ 5:e3139
Chauhan JS, Rao A, Raghava GPS (2013) In silico platform for prediction of N-, O- and C-glycosites in eukaryotic protein sequences. PLOS ONE 8(6):1–10
Chelliah V, Juty N, Ajmera I, Ali R, Dumousseau M, Glont M, Hucka M, Jalowicki G, Keating S, Knight-Schrijver V, Lloret-Villas A, Natarajan KN, Pettit J-B, Rodriguez N, Schubert M, Wimalaratne SM, Zhao Y, Hermjakob H, Le Novère N, Laibe C (2015) BioModels: ten-year anniversary. Nucleic Acids Res 43(D1):D542–D548
Chen L, Liu H, Friedman C (2005) Gene name ambiguity of eukaryotic nomenclatures. Bioinformatics 21(2):248–256
Chen Q, Chen Y-PP, Zhang C (2007) Detecting inconsistency in biological molecular databases using ontologies. Data Min Knowl Discov 15:275–296
Cheng J, Baldi P (2007) Improved residue contact prediction using support vector machines and a large feature set. BMC Bioinform 8(1):113
Chen M, Yu S, Franz N, Bowers S, Ludäscher B (2013) Euler/X:a toolkit for logic-based taxonomy integration. Technical report 1306, 22nd International workshop on functional and (Constraint) logic programming, Technische Berichte des Instituts fur Informatik der Christian-Albrechts-Universitat zu Kiel
Clark C, Divvala S (2016) Pdffigures 2.0: mining figures from research papers. In: 2016 IEEE/ACM joint conference on digital libraries (JCDL). IEEE, pp 143–152
Coluzza I (2017) Computational protein design: a review. J Phys: Condens Matter 29(14):143001
Coste F (2016) Learning the language of biological sequences. Springer, Berlin, pp 215–247
Cruz-Monteagudo M, Medina-Franco JL, Perez-Castillo Y, Nicolotti O, Cordeiro MND, Borges F (2014) Activity cliffs in drug discovery: Dr. Jekyll or Mr. Hyde? Drug Discov Today 19(8):1069–1080
Cuff JA, Barton GJ (2000) Application of multiple sequence alignment profiles to improve protein secondary structure prediction. Proteins: Struct Funct Bioinform 40(3):502–511
Dallas DC, Guerrero A, Parker EA, Robinson RC, Gan J, German JB, Barile D, Lebrilla CB (2015) Current peptidomics: applications, purification, identification, quantification, and functional analysis. PROTEOMICS 15(5–6):1026–1038
De Brevern A, Etchebest C, Hazout S (2000) Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks. Proteins: Struct Funct Bioinform 41(3):271–287
De Raedt L, Kramer S (2001) The levelwise version space algorithm and its application to molecular fragment finding. In: Proceedings of the 17th international joint conference on artificial intelligence - volume 2, IJCAI'01. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 853–859

Deagustini CAD, Martinez MV, Falappa MA, Simari GR (2016) Datalog+-ontology consolidation. J Artif Intell Res 56:613–656
Diaz-Uriarte R (2007) GeneSrF and varSelRF: a web-based tool and R package for gene selection and classification using random forest. BMC Bioinform 8(1):328
Dimova D, Bajorath J (2016) Advances in activity cliff research. Mol Inform 35(5):181–191
Doğan B, Ölmez T (2015) A novel state space representation for the solution of 2D-HP protein folding problem using reinforcement learning methods. Appl Soft Comput 26:213–223
Drozdetskiy A, Cole C, Procter J, Barton GJ (2015) Jpred4: a protein secondary structure prediction server. Nucleic Acids Res 43(W1):W389–W394
Dubey SP, Kini NG, Balaji S, Kumar MS (2017) Protein structure prediction on 2D square HP lattice with revised fitness function. In: 2017 International conference on advances in computing, communications and informatics (ICACCI), pp 1732–1736
Dubrova E, Teslenko M (2011) A SAT-based algorithm for finding attractors in synchronous Boolean networks. IEEE/ACM Trans Comput Biol Bioinform 8(5):1393–1399
Dunn S, Wahl L, Gloor G (2008) Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. Bioinformatics 24(3):333–340
Eddy SR (1998) Profile hidden Markov models. Bioinformatics (Oxford, England) 14(9):755–763
El-Kebir M, Oesper L, Acheson-Field H, Raphael BJ (2015) Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. Bioinformatics 31(12):i62–i70
Erdem E (2011) Applications of answer set programming in phylogenetic systematics. Springer, Berlin, pp 415–431
Faraggi E, Kloczkowski A (2017) Accurate prediction of one-dimensional protein structure features using SPINE-X. Springer, New York, pp 45–53
Faraggi E, Zhang T, Yang Y, Kurgan L, Zhou Y (2012) SPINE X: improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. J Comput Chem 33(3):259–267
Feigenbaum EA, Buchanan BG (1993) Dendral and meta-dendral: roots of knowledge systems and expert system applications. Artif Intell 59(1):233–240
Feig M, Rotkiewicz P, Kolinski A, Skolnick J, Brooks CL (2000) Accurate reconstruction of all-atom protein representations from side-chain-based low-resolution models. Proteins: Struct Funct Bioinform 41(1):86–97
Ferreira LG, dos Santos RN, Oliva G, Andricopulo AD (2015) Molecular docking and structure-based drug design strategies. Molecules 20(7):13384–13421
Ford E, St. John K, Wheeler WC, (2015) Towards improving searches for optimal phylogenies. Syst Biol 64(1):56–65
Frank M, Schloissnig S (2010) Bioinformatics and molecular modeling in glycobiology. Cell Mol Life Sci 67(16):2749–2772
Franz NM, Chen M, Yu S, Kianmajd P, Bowers S, Ludäscher B (2015) Reasoning over taxonomic change: exploring alignments for the Perelleschus use case. PLOS ONE 10(2):1–34
Galiez C, Magnan CN, Coste F, Baldi P (2016) VIRALpro: a tool to identify viral capsid and tail sequences. Bioinformatics 32(9):1405–1407
Galperin MY, Fernández-Suárez XM, Rigden DJ (2017) The 24th annual nucleic acids research database issue: a look back and upcoming changes. Nucleic Acids Res 45(D1):D1
Gan X, Kapsokalivas L, Albrecht AA, Steinhöfel K (2008) A symmetry-free subspace for ab initio protein folding simulations. Bioinformatics research and development. Springer, Berlin, pp 128–139
Gao Y, Wang S, Deng M, Xu J (2017) Real-value and confidence prediction of protein backbone dihedral angles through a hybrid method of clustering and deep learning. bioRxiv
Gawad C, Koh W, Quake SR (2016) Single-cell genome sequencing: current state of the science. Nat Rev Genet 17(3):175
Gawehn E, Hiss JA, Schneider G (2016) Deep learning in drug discovery. Mol Inform 35(1):3–14
Gertheiss J, Tutz G (2009) Supervised feature selection in mass spectrometry-based proteomic profiling by blockwise boosting. Bioinformatics 25(8):1076–1077

Ghisalberti G, Masseroli M, Tettamanti L (2010) Quality controls in integrative approaches to detect errors and inconsistencies in biological databases. J Integr Bioinform 7(3):52–64
Ghoorah AW, Devignes M-D, Smaïl-Tabbone M, Ritchie DW (2013) Protein docking using case-based reasoning. Proteins: Struct Funct Bioinform 81(12):2150–2158
Gonnet GH, Korostensky C, Benner S (2000) Evaluation measures of multiple sequence alignments. J Comput Biol 7(1–2):261–276
Gordon JJ, Towsey MW, Hogan JM, Mathews SA, Timms P (2005) Improved prediction of bacterial transcription start sites. Bioinformatics 22(2):142–148
Götz S, García-Gómez JM, Terol J, Williams TD, Nagaraj SH, Nueda MJ, Robles M, Talón M, Dopazo J, Conesa A (2008) High-throughput functional annotation and data mining with the Blast2GO suite. Nucleic Acids Res 36(10):3420–3435
Greene D, Richardson S, Turro E (2017) OntologyX: a suite of R packages for working with ontological data. Bioinformatics 33(7):1104–1106
Grivell L (2002) Mining the bibliome: searching for a needle in a haystack? EMBO Rep 3(3):200–203
Gront D, Kmiecik S, Koliński A, Meinke JH, Zimmermann MT, Mohanty S, Hansmann UHE (2006) High throughput method for protein structure prediction. In: NIC workshop 2006: from computational biophysics to system biology, vol 34. John von Neumann institute for computing, Julich, pp 79–82
Guan Y, Myers CL, Hess DC, Barutcuoglu Z, Caudy AA, Troyanskaya OG (2008) Predicting gene function in a hierarchical context with an ensemble of classifiers. Genome Biol 9(S1):S3
Guermeur Y, Geourjon C, Gallinari P, Deléage G (1999) Improved performance in protein secondary structure prediction by inhomogeneous score combination. Bioinformatics 15(5):413–421
Guo Y, Tao F, Wu Z, Wang Y (2017) Hybrid method to solve HP model on 3D lattice and to probe protein stability upon amino acid mutations. BMC Syst Biol 11(4):93
Gupta SK, Kececioglu JD, Schäffer AA (1995) Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. J Comput Biol 2(3):459–472
Gusfield D (2015) Persistent phylogeny: a galled-tree and integer linear programming approach. In: Proceedings of the 6th ACM conference on bioinformatics, computational biology and health informatics, BCB '15, ACM, New York, USA, pp 443–451
Han Y, Gao S, Muegge K, Zhang W, Zhou B (2015) Advanced applications of RNA sequencing and challenges. Bioinform Biol Insights 9s1:BBI.S28991
Hassanien A-E, Milanova MG, Smolinski TG, Abraham A (2008) Computational intelligence in solving bioinformatics problems: reviews, perspectives, and challenges. Springer, Berlin, pp 3–47
Hassanien AE, Al-Shammari ET, Ghali NI (2013) Computational intelligence techniques in bioinformatics. Comput Biol Chem 47:37–47
Hastings J, de Matos P, Dekker A, Ennis M, Harsha B, Kale N, Muthukrishnan V, Owen G, Turner S, Williams M, Steinbeck C (2013) The chebi reference database and ontology for biologically relevant chemistry: enhancements for 2013. Nucleic Acids Res 41(D1):D456–D463
Hatem M, Ruml W (2013) External memory best-first search for multiple sequence alignment. In: 27th AAAI conference on artificial intelligence
Hayes-Roth B, Buchanan BG, Lichtarge O, Hewitt M, Altman RB, Brinkley JF, Cornelius C, Duncan BS, Jardetzky O (1986) PROTEAN: deriving protein structure from constraints. In: AAAI, pp 904–909
Heffernan R, Paliwal K, Lyons J, Dehzangi A, Sharma A, Wang J, Sattar A, Yang Y, Zhou Y (2015) Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. Sci Rep 5:11476
Helma C, Kramer S, Luc DR (2002) The molecular feature miner MOLFEA. In: Proceedings of the Beilstein workshop 2002: molecular informatics: confronting complexity; Beilstein Institut, pp 79–93

Hinselmann G, Rosenbaum L, Jahn A, Fechner N, Ostermann C, Zell A (2011) Large-scale learning of structure- activity relationships using a linear support vector machine and problem-specific metrics. J Chem Inf Model 51(2):203–213

Hirschman L, Park JC, Tsujii J, Wong L, Wu CH (2002) Accomplishments and challenges in literature data mining for biology. Bioinformatics 18(12):1553–1561

Hoehndorf R, Slater L, Schofield PN, Gkoutos GV (2015) Aber-owl: a framework for ontology-based data access in biology. BMC Bioinform 16(1):26

Hoff KJ, Lange S, Lomsadze A, Borodovsky M, Stanke M (2015) Braker1: unsupervised rna-seq-based genome annotation with genemark-et and augustus. Bioinformatics 32(5):767–769

Hoff K, Stanke M (2015) Current methods for automated annotation of protein-coding genes. Curr Opin Insect Sci 7(Supplement C):8–14. Insect genomics * Development and regulation

Holzinger A, Dehmer M, Jurisica I (2014) Knowledge discovery and interactive data mining in bioinformatics-state-of-the-art, future challenges and research directions. BMC Bioinform 15(6):I1

Hosoda M, Akune Y, Aoki-Kinoshita KF (2017) Development and application of an algorithm to compute weighted multiple glycan alignments. Bioinformatics 33(9):1317–1323

Huang DW, Sherman BT, Lempicki RA (2008) Systematic and integrative analysis of large gene lists using David bioinformatics resources. Nat Protoc 4(1):44

Hundley L, Lederberg J, Levinthal E (1963) Multivator- a biochemical laboratory for Martian experiments. Technical report NSG-81-60, NASA

Huntley RP, Harris MA, Alam-Faruque Y, Blake JA, Carbon S, Dietze H, Dimmer EC, Foulger RE, Hill DP, Khodiyar VK, Lock A, Lomax J, Lovering RC, Mutowo-Meullenet P, Sawford T, Van Auken K, Wood V, Mungall CJ (2014) A method for increasing expressivity of gene ontology annotations using a compositional approach. BMC Bioinform 15(1):155

Ideker T, Nussinov R (2017) Network approaches and applications in biology. PLOS Comput Biol 13(10):1–3

Ikeda T, Imai H (1999) Enhanced A* algorithms for multiple alignments: optimal alignments for several sequences and k-opt approximate alignments for large cases. Theor Comput Sci 210(2):341–374

Inokuchi A, Washio T, Motoda H (2000) An apriori-based algorithm for mining frequent substructures from graph data. In: Zighed DA, Komorowski J, Żytkow J (eds) Principles of data mining and knowledge discovery. Springer, Berlin, pp 13–23

Inoue K (2011) Logic programming for Boolean networks. In: Proceedings of the twenty-second international joint conference on artificial intelligence - volume volume two, IJCAI'11. AAAI Press, pp 924–930

Inza I, Calvo B, Armañanzas R, Bengoetxea E, Larrañaga P, Lozano JA (2010) Machine learning: an indispensable tool in bioinformatics. Humana Press, Totowa, pp 25–48

Islamaj Doğan R, Kim S, Chatr-aryamontri A, Chang CS, Oughtred R, Rust J, Wilbur WJ, Comeau DC, Dolinski K, Tyers M (2017) The bioc-biogrid corpus: full text articles annotated for curation of protein-protein and genetic interactions. Database 2017(1):baw147

Jiang S-Y, Ramachandran S (2010) Assigning biological functions to rice genes by genome annotation, expression analysis and mutagenesis. Biotechnol Lett 32(12):1753–1763

Jones DT, Singh T, Kosciolek T, Tetchner S (2015) Metapsicov: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. Bioinformatics 31(7):999–1006

Jong K, Marchiori E, Sebag M, Van Der Vaart A (2004) Feature selection in proteomic pattern data with support vector machines. In: Proceedings of the 2004 IEEE symposium on computational intelligence in bioinformatics and computational biology, 2004, CIBCB'04. IEEE, pp 41–48

Judson PN (2014) Knowledge-based approaches for predicting the sites and products of metabolism, chapter 12. Wiley-Blackwell, pp 293–318

Jupp S, Malone J, Bolleman J, Brandizi M, Davies M, Garcia L, Gaulton A, Gehant S, Laibe C, Redaschi N, Wimalaratne SM, Martin M, Le Novère N, Parkinson H, Birney E, Jenkinson AM

(2014) The EBI RDF platform: linked open data for the life sciences. Bioinformatics 30(9):1338–1339
Källberg M, Wang H, Wang S, Peng J, Wang Z, Lu H, Xu J (2012) Template-based protein structure modeling using the RaptorX web server. Nat Protoc 7(8):1511–1522
Kaminski R, Schaub T, Siegel A, Videla S (2013) Minimal intervention strategies in logical signaling networks with ASP. Theory Pract Log Program 13(4–5):675–690
Kanehisa M (2017) Kegg glycan. In: A practical guide to using glycomics databases. Springer, Berlin, pp 177–193
Karpe PD, Latendresse M, Caspi R (2011) The pathway tools pathway prediction algorithm. Stand Genomic Sci 5(3):424
Kavanagh J, Mitchell D, Ternovska E, Maňuch J, Zhao X, Gupta A (2006) Constructing Camin-Sokal phylogenies via answer set programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 452–466
Keedwell E, Narayanan A (2005) Intelligent bioinformatics: the application of artificial intelligence techniques to bioinformatics problems. Wiley, New York
Khatri P, Sirota M, Butte AJ (2012) Ten years of pathway analysis: current approaches and outstanding challenges. PLOS Comput Biol 8(2):1–10
Kislyuk A, Lomsadze A, Lapidus AL, Borodovsky M (2009) Frameshift detection in prokaryotic genomic sequences. Int J Bioinform Res Appl 5(4):458–477
Kitchen DB, Decornez H, Furr JR, Bajorath J (2004) Docking and scoring in virtual screening for drug discovery: methods and applications. Nat Rev Drug Discov 3(11):935
Klamt S (2006) Generalized concept of minimal cut sets in biochemical networks. Biosystems 83(2–3):233–247
Klarner H, Bockmayr A, Siebert H (2015) Computing maximal and minimal trap spaces of Boolean networks. Nat Comput 14(4):535–544
Klarner H, Streck A, Siebert H (2017) Pyboolnet: a python package for the generation, analysis and visualization of Boolean networks. Bioinformatics 33(5):770–772
Koliński A et al (2004) Protein modeling and structure prediction with a reduced representation. Acta Biochim Pol 51
Koponen L, Oikarinen E, Janhunen T, Säilä L (2015) Optimizing phylogenetic supertrees using answer set programming. Theory Pract Log Program 15(4–5):604–619
Korf RE, Zhang W, Thayer I, Hohwald H (2005) Frontier search. J ACM 52(5):715–748
Korostensky C, Gonnet G (1999) Near optimal multiple sequence alignments using a traveling salesman problem approach. In: SPIRE/CRIWG, pp 105–114
Koshino M, Murata H, Shirayama M, Kimura H (2006) Applying the various optimal solution search methods to multiple sequence alignments and performance evaluation. Syst Comput Jpn 37(11):1–10
Kowalski R (1979) Algorithm = logic + control. Commun ACM 22(7):424–436
Kumozaki S, Sato K, Sakakibara Y (2015) A machine learning based approach to de novo sequencing of glycans from tandem mass spectrometry spectrum. IEEE/ACM Trans Comput Biol Bioinform 12(6):1267–1274
Lacroix V, Sammeth M, Guigo R, Bergeron A (2008) Exact transcriptome reconstruction from short sequence reads. In: International workshop on algorithms in bioinformatics. Springer, pp 50–63
Lai J, An J, Seim I, Walpole C, Hoffman A, Moya L, Srinivasan S, Perry-Keene JL, Wang C, Lehman ML et al (2015) Fusion transcript loci share many genomic features with non-fusion loci. BMC Genomics 16(1):1021
Lavecchia A (2015) Machine-learning approaches in drug discovery: methods and applications. Drug Discov Today 20(3):318–331
Le Novere N (2015) Quantitative and logic modelling of gene and molecular networks. Nature Rev Genet 16(3):146
Le T, Nguyen H, Pontelli E, Son TC (2012) ASP at work: an ASP implementation of PhyloWS. In: Dovier A, Costa VS (eds) Technical communications of the 28th international conference on

logic programming (ICLP'12), volume 17 of Leibniz international proceedings in informatics (LIPIcs), pp 359–369, Dagstuhl. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Germany
Lepailleur A, Poezevara G, Bureau R (2013) Automated detection of structural alerts (chemical fragments) in (eco)toxicology. Comput Struct Biotechnol J 5(6):e201302013
Lertampaiporn S, Thammarongtham C, Nukoolkit C, Kaewkamnerdpong B, Ruengjitchatchawalya M (2014) Identification of non-coding RNAs with a new composite feature in the hybrid random forest ensemble algorithm. Nucleic Acids Res 42(11):e93
Li L, Zhang Y, Zou L, Li C, Yu B, Zheng X, Zhou Y (2012) An ensemble classifier for eukaryotic protein subcellular location prediction using gene ontology categories and amino acid hydrophobicity. PLOS One 7(1):1–12
Li F, Li C, Wang M, Webb GI, Zhang Y, Whisstock JC, Song J (2015) Glycomine: a machine learning-based approach for predicting n-, c-and o-linked glycosylation in the human proteome. Bioinformatics 31(9):1411–1419
Li H, Hou J, Adhikari B, Lyu Q, Cheng J (2017) Deep learning methods for protein torsion angle prediction. BMC Bioinform 18(1):417
Lihu A, Holban Ş (2015) A review of ensemble methods for de novo motif discovery in ChIP-Seq data. Brief Bioinform 16(6):964–973
Lim CY, Wang H, Woodhouse S, Piterman N, Wernisch L, Fisher J, Göttgens B (2016) BTR: training asynchronous Boolean models using single-cell expression data. BMC Bioinform 17(1):355
Lindsay RK, Buchanan BG, Feigenbaum EA, Lederberg J (1993) Dendral: a case study of the first expert system for scientific hypothesis formation. Artif Intell 61(2):209–261
Liu H, Liu L, Zhang H (2010) Ensemble gene selection for cancer classification. Pattern Recognit 43(8):2763–2772
Lomsadze A, Ter-Hovhannisyan V, Chernoff YO, Borodovsky M (2005) Gene identification in novel eukaryotic genomes by self-training algorithm. Nucleic Acids Res 33(20):6494–6506
Lomsadze A, Burns PD, Borodovsky M (2014) Integration of mapped RNA-Seq reads into automatic training of eukaryotic gene finding algorithm. Nucleic Acids Res 42(15):e119
Magnan CN, Baldi P (2014) SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. Bioinformatics 30(18):2592–2597
Mahadevan R, von Kamp A, Klamt S (2015) Genome-scale strain designs based on regulatory minimal cut sets. Bioinformatics 31(17):2844–2851
Mahé P, Vert J-P (2009) Graph kernels based on tree patterns for molecules. Mach Learn 75(1):3–35
Malikic S, McPherson AW, Donmez N, Sahinalp CS (2015) Clonality inference in multiple tumor samples using phylogeny. Bioinformatics 31(9):1349–1356
Mamitsuka H (2011) Glycoinformatics: data mining-based approaches. CHIMIA Int J Chem 65(1):10–13
Mann M, Backofen R (2014) Exact methods for lattice protein models. Bio-Algorithms Med-Syst 10(4):213–225
Mann M, Will S, Backofen R (2008) CPSP-tools – Exact and complete algorithms for high-throughput 3D lattice protein studies. BMC Bioinformatcis 9(1):230
Martin M, Dague P, Pérès S, Simon L (2016) Minimality of metabolic flux modes under Boolean regulation constraints. In: 12th International workshop on constraint-based methods for bioinformatics WCB'16, Toulouse, France
Matentzoglu N, Vigo M, Jay C, Stevens R (2017) Inference inspector: improving the verification of ontology authoring actions. Web semantics: science services and agents on the World Wide Web
Maupetit J, Gautier R, Tufféry P (2006) SABBAC: online structural alphabet-based protein backbone reconstruction from alpha-carbon trace. Nucleic Acids Res 34(suppl_2):W147–W151
McGuffin LJ, Bryson K, Jones DT (2000) The psipred protein structure prediction server. Bioinformatics 16(4):404–405
Medina and Tárraga, J., Martínez, H., Barrachina, S., Castillo, M. I., Paschall, J., Salavert-Torres, J., Blanquer-Espert, I., Hernández-García, V., Quintana-Ortí, E. S., and Dopazo, J (2016) Highly sensitive and ultrafast read mapping for RNA-seq analysis. DNA Res 23(2):93–100

Mei S (2012) Multi-label Multi-Kernel Transfer Learning for Human Protein Subcellular Localization. PLOS ONE 7(6):1–12
Mei S, Zhu H (2014) AdaBoost based multi-instance transfer learning for predicting proteome-wide interactions between salmonella and human proteins. PLOS ONE 9(10):1–12
Merelli E, Armano G, Cannata N, Corradini F, d'Inverno M, Doms A, Lord P, Martin A, Milanesi L, Möller S, Schroeder M, Luck M (2007) Agents in bioinformatics, computational and systems biology. Brief Bioinform 8(1):45
Mestres J, Gregori-Puigjane E, Valverde S, Sole RV (2008) Data completeness-the Achilles heel of drug-target networks. Nature Biotechnol 26(9):983
MGlincy NJ, Ingolia NT (2017) Transcriptome-wide measurement of translation by ribosome profiling. Methods 126:112–129. Post-transcriptional regulation of gene expression
Michel AM, Choudhury KR, Firth AE, Ingolia NT, Atkins JF, Baranov PV (2012) Observation of dually decoded regions of the human genome using ribosome profiling data. Genome Res 22(11):2219–2229
Midic U, Dunker AK, Obradovic Z (2005) Improving protein secondary-structure prediction by predicting ends of secondary-structure segments. In: Proceedings of the 2005 IEEE symposium on computational intelligence in bioinformatics and computational biology, CIBCB'05. IEEE, pp 1–8
Miranda M, Lynce I, Manquinho V (2014) Inferring phylogenetic trees using pseudo-Boolean optimization. AI Commun 27(3):229–243
Mitra S, Datta S, Perkins T, Michailidis G (2008) Introduction to machine learning and bioinformatics. CRC Press, Boca Raton
Miyahara T, Kuboyama T (2014) Learning of glycan motifs using genetic programming and various fitness functions. J Adv Comput Intell Intell Inform 18(3):401–408
Moll M, Schwarz D, Kavraki LE (2008) Roadmap methods for protein folding. Protein Struct Predict 219–239
Monteiro PT, Ropers D, Mateescu R, Freitas AT, de Jong H (2008) Temporal logic patterns for querying dynamic models of cellular interaction networks. Bioinformatics 24(16):i227–i233
Moult J, Fidelis K, Kryshtafovych A, Schwede T, Tramontano A (2018) Critical assessment of methods of protein structure prediction (CASP)—round xii. Proteins: Struct Funct Bioinform 86:7–15
Muggleton S, King RD, Stenberg MJ (1992) Protein secondary structure prediction using logic-based machine learning. Protein Eng Des Sel 5(7):647–657
Muller J, Creevey CJ, Thompson JD, Arendt D, Bork P (2010) Aqua: automated quality improvement for multiple sequence alignments. Bioinformatics 26(2):263–265
Mungall CJ, Torniai C, Gkoutos GV, Lewis SE, Haendel MA (2012) Uberon, an integrative multi-species anatomy ontology. Genome Biology 13(1):R5
Mushthofa M, Torres G, Van de Peer Y, Marchal K, De Cock M (2014) ASP-G: an ASP-based method for finding attractors in genetic regulatory networks. Bioinformatics 30(21):3086–3092
Nagi S, Bhattacharyya DK, Kalita JK (2017) Complex detection from ppi data using ensemble method. Netw Model Anal Health Inform Bioinform 6(1):3
Naik AW, Kangas JD, Sullivan DP, Murphy RF (2016) Active machine learning-driven experimentation to determine compound effects on protein patterns. eLife 5:e10047
Naldi A, Carneiro J, Chaouiya C, Thieffry D (2010) Diversity and plasticity of th cell types predicted from regulatory network modelling. PLOS Comput Biology 6(9):1–16
Nesbeth DN, Zaikin A, Saka Y, Romano MC, Giuraniuc CV, Kanakov O, Laptyeva T (2016) Synthetic biology routes to bio-artificial intelligence. Essays Biochem 60(4):381–391
Newman A, Ruhl M (2004) Combinatorial problems on strings with applications to protein folding. In: Farach-Colton M (ed) LATIN 2004: theoretical informatics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 369–378
Ng C-T, Li C, Fan X (2017) A fast algorithm for reconstructing multiple sequence alignment and phylogeny simultaneously. Current Bioinform 12:329–348

Offmann B, Tyagi M, de Brevern AG (2007) Local protein structures. Current Bioinform 2(3):165–202

Okun O, Priisalu H (2007) Random forest for gene expression based cancer classification: overlooked issues. Pattern Recognit Image Anal 483–490

Omar M, Salam R, Abdullah R, Rashid N (2005) Multiple sequence alignment using optimization algorithms. Int J Comput Intell 1(2):81–89

Pandini A, Fornili A, Kleinjung J (2010) Structural alphabets derived from attractors in conformational space. BMC Bioinform 11(1):97

Park YR, Kim J, Lee HW, Yoon YJ, Kim JH (2011) Gochase-ii: correcting semantic inconsistencies from gene ontology-based annotations for gene products. BMC Bioinform 12(1):S40

Pashaei E, Ozen M, Aydin N (2017) Splice site identification in human genome using random forest. Health Technol 7(1):141–152

Pashaei E, Aydin N (2017) Frequency difference based DNA encoding methods in human splice site recognition. In: 2017 international conference on computer science and engineering (UBMK). IEEE, pp 586–591

Pashaei E, Ozen M, Aydin N (2016a) Splice sites prediction of human genome using adaboost. In: 2016 IEEE-EMBS international conference on biomedical and health informatics (BHI). IEEE, pp 300–303

Pashaei E, Yilmaz A, Aydin N (2016b) A combined SVM and Markov model approach for splice site identification. In: 2016 6th international conference on computer and knowledge engineering (ICCKE). IEEE, pp 200–204

Peng J, Xu J (2010) Low-homology protein threading. Bioinformatics 26(12):i294–i300

Pérez S, Sarkar A, Rivet A, Breton C, Imberty A (2015) Glyco3d: a portal for structural glycosciences. In: Glycoinformatics. Springer, pp 241–258

Pertea M, Pertea GM, Antonescu CM, Chang T-C, Mendell JT, Salzberg SL (2015) Stringtie enables improved reconstruction of a transcriptome from rna-seq reads. Nat Biotechnol 33(3):290–295

Pes B, Dessì N, Angioni M (2017) Exploiting the ensemble paradigm for stable feature selection: a case study on high-dimensional genomic data. Inf Fus 35(Supplement:C):132–147

Petegrosso R, Park S, Hwang TH, Kuang R (2017) Transfer learning across ontologies for phenome-genome association prediction. Bioinformatics 33(4):529–536

Piao Y, Piao M, Park K, Ryu KH (2012) An ensemble correlation-based gene selection algorithm for cancer classification with gene expression data. Bioinformatics 28(24):3306–3315

Pietal MJ, Bujnicki JM, Kozlowski LP (2015) GDFuzz3D: a method for protein 3D structure reconstruction from contact maps, based on a non-euclidean distance function. Bioinformatics 31(21):3499–3505

Pirola Y, Rizzi R, Picardi E, Pesole G, Della Vedova G, Bonizzoni P (2012) Pintron: a fast method for detecting the gene structure due to alternative splicing via maximal pairings of a pattern and a text. BMC Bioinform 13(5):S2

Pokarowski P, Kolinski A, Skolnick J (2003) A minimal physically realistic protein-like lattice model: designing an energy landscape that ensures all-or-none folding to a unique native state. Biophys J 84(3):1518–1526

Popic V, Salari R, Hajirasouliha I, Kashef-Haghighi D, West RB, Batzoglou S (2015) Fast and scalable inference of multi-sample cancer lineages. Genome Biology 16(1):91

Post LJG, Roos M, Marshall MS, van Driel R, Breit TM (2007) A semantic web approach applied to integrative bioinformatics experimentation: a biological use case with genomics data. Bioinformatics 23(22):3080–3087

Raies AB, Bajic VB (2016) In silico toxicology: computational methods for the prediction of chemical toxicity. Wiley Interdiscip Rev: Comput Mol Sci 6(2):147–172

Ramsden JJ (2004) Bioinformatics: an introduction, Volume 21 of Computational biology. Springer

Rannug U, Sjögren M, Rannug A, Gillner M, Toftgård R, Gustafsson J-Å, Rosenkranz H, Klopman G (1991) Use of artificial intelligence in structure-affinity correlations of 2,3,7,8-tetrachlorodibenzo-p-dioxin (TCDD) receptor ligands. Carcinogenesis 12(11):2007–2015

Ranzinger R, Aoki-Kinoshita KF, Campbell MP, Kawano S, Lütteke T, Okuda S, Shinmachi D, Shikanai T, Sawaki H, Toukach P, Matsubara M, Yamada I, Narimatsu H (2015) GlycoRDF: an ontology to standardize glycomics data in RDF. Bioinformatics 31(6):919–925

Reid I, O'Toole N, Zabaneh O, Nourzadeh R, Dahdouli M, Abdellateef M, Gordon PM, Soh J, Butler G, Sensen CW, Tsang A (2014) Snowyowl: accurate prediction of fungal genes by using RNA-seq and homology information to select among ab initio models. BMC Bioinform 15(1):229

Reker D, Schneider P, Schneider G (2016) Multi-objective active machine learning rapidly improves structure-activity models and reveals new protein-protein interaction inhibitors. Chem Sci 7:3919–3927

Reker D, Schneider P, Schneider G, Brown J (2017) Active learning for computational chemogenomics. Futur Med Chem 9(4):381–402

Requeno JI, Colom JM (2016) Evaluation of properties over phylogenetic trees using stochastic logics. BMC Bioinform 17(1):235

Requeno JI, de Miguel Casado G, Blanco R, Colom JM (2013) Temporal logics for phylogenetic analysis via model checking. IEEE/ACM Trans Comput Biol Bioinform (TCBB) 10(4):1058–1070

Rinaldi F, Lithgow O, Gama-Castro S, Solano H, López-Fuentes A, Muñiz Rascado LJ, Ishida-Gutiérrez C, Méndez-Cruz C-F, Collado-Vides J (2017) Strategies towards digital and semi-automated curation in RegulonDB. Database 2017:bax012

Robertson MP, Joyce GF (2012) The origins of the RNA world. Cold Spring Harb Perspect Biol 4(5):a003608

Rodríguez-Penagos C, Salgado H, Martínez-Flores I, Collado-Vides J (2007) Automatic reconstruction of a bacterial regulatory network using natural language processing. BMC Bioinform 8(1):293

Röhl A, Bockmayr A (2017) A mixed-integer linear programming approach to the reduction of genome-scale metabolic networks. BMC Bioinform 18(1):2

Rost B, Sander C (1993) Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol 232(2):584–599

Sakiyama Y, Yuki H, Moriya T, Hattori K, Suzuki M, Shimada K, Honma T (2008) Predicting human liver microsomal stability with machine learning techniques. J Mol Graph Model 26(6):907–915

Samaga R, Kamp AV, Klamt S (2010) Computing combinatorial intervention strategies and failure modes in signaling networks. J Comput Biol 17(1):39–53

Sarkar IN (2015) Mining the bibliome. In: Translational informatics. Springer, pp 75–96

Sawada R, Kotera M, Yamanishi Y (2014) Benchmarking a wide range of chemical descriptors for drug-target interaction prediction using a chemogenomic approach. Mol Inform 33(11–12):719–731

Schietgat L, Vens C, Struyf J, Blockeel H, Kocev D, Džeroski S (2010) Predicting gene function using hierarchical multi-label decision tree ensembles. BMC Bioinform 11(1):2

Schroedl S (2005) An improved search algorithm for optimal multiple-sequence alignment. J Artif Intell Res 23:587–623

Schulte C, Stuckey PJ (2008) Efficient constraint propagation engines. Trans Program Lang Syst 31(1):2:1–2:43

Schwartz R, Schäffer AA (2017) The evolution of tumour phylogenetics: principles and practice. Nat Rev Genet 18(4):213

Senger RS, Karim MN (2008) Prediction of n-linked glycan branching patterns using artificial neural networks. Math Biosci 211(1):89–104

Shao Z, Hirayama Y, Yamanishi Y, Saigo H (2015) Mining discriminative patterns from graph data with multiple labels and its application to quantitative structure-activity relationship (qsar) models. J Chem Inf Model 55(12):2519–2527 PMID: 26549421

Shatabda S, Newton MAH, Sattar A (2014)Constraint-based evolutionary local search for protein structures with secondary motifs. In: Pham D-N, Park S-B (eds) PRICAI 2014: trends in artificial intelligence. Springer International Publishing, pp 333–344

Shaw DL, Islam AS, Rahman MS, Hasan M (2014) Protein folding in HP model on hexagonal lattices with diagonals. BMC Bioinform 15(2):S7

Sheela T, Rangarajan L (2017) Combination of feature selection methods for the effective classification of microarray gene expression data. In: Santosh K, Hangarge M, Bevilacqua V, Negi A (eds) Recent trends in image processing and pattern recognition: first international conference, RTIP2R 2016, Bidar, India, 16–17 Dec 2016, Revised Selected Papers. Springer, Singapore, pp 137–145

Sherhod R, Judson PN, Hanser T, Vessey JD, Webb SJ, Gillet VJ (2014) Emerging pattern mining to aid toxicological knowledge discovery. J Chem Inf Model 54(7):1864–1879

Shvaiko P, Euzenat J (2013) Ontology matching: state of the art and future challenges. IEEE Trans Knowl Data Eng 25(1):158–176

Singh GB (2015) Introduction to bioinformatics. In: Fundamentals of bioinformatics and computational biology. Springer, pp 3–10

Singh H, Singh S, Raghava GPS (2014) Evaluation of protein dihedral angle prediction methods. PLOS ONE 9(8):1–9

Skwark MJ, Raimondi D, Michel M, Elofsson A (2014) Improved contact predictions using the recognition of protein like contact patterns. PLOS Comput Biology 10(11):1–14

Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, Goldberg LJ, Eilbeck K, Ireland A, Mungall CJ et al (2007) The obo foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnol 25(11):1251–1255

Smitha SKN, Reddy SN (2016) Amyloid motif prediction using ensemble approach. Current Bioinform 11(3):357–365

Song J, Burrage K, Yuan Z, Huber T (2006) Prediction of cis/trans isomerization in proteins using psi-blast profiles and secondary structure information. BMC Bioinform 7(1):124

Soon WW, Hariharan M, Snyder MP (2013) High-throughput sequencing for biology and medicine. Mol Syst Biology 9(1):640

Spencer M, Eickholt J, Cheng J (2015) A deep learning network approach to ab initio protein secondary structure prediction. IEEE/ACM Trans Comput Biol Bioinform 12(1):103–112

Sridhar S, Lam F, Blelloch GE, Ravi R, Schwartz R (2008) Mixed integer linear programming for maximum-parsimony phylogeny inference. IEEE/ACM Trans Comput Biol Bioinform 5(3):323–331

Sternberg MJ, Tamaddoni-Nezhad A, Lesk VI, Kay E, Hitchen PG, Cootes A, van Alphen LB, Lamoureux MP, Jarrell HC, Rawlings CJ, Soo EC, Szymanski CM, Dell A, Wren BW, Muggleton SH (2013) Gene function hypotheses for the campylobacter jejuni glycome generated by a logic-basedApproach. J Mol Biol 425(1):186–197

Stevens R, Goble CA, Bechhofer S (2000) Ontology-based knowledge representation for bioinformatics. Brief Bioinform 1(4):398–414

Sundfeld D, Razzolini C, Teodoro G, Boukerche A, de Melo ACMA (2017) Pa-star: a disk-assisted parallel a-star strategy with locality-sensitive hash for multiple sequence alignment. J Parallel Distrib Comput

Takigawa I, Mamitsuka H (2013) Graph mining: procedure, application to drug discovery and recent advances. Drug Discov Today 18(1):50–57

Takigawa I, Hashimoto K, Shiga M, Kanehisa M, Mamitsuka H (2010) Mining patterns from glycan structures, pp 13–24

The GO Consortium (2017) Expansion of the gene ontology knowledgebase and resources. Nucl Acids Res 45(D1):D331–D338

Tiemeyer M, Aoki K, Paulson J, Cummings RD, York WS, Karlsson NG, Lisacek F, Packer NH, Campbell MP, Aoki NP et al (2017) GlyTouCan: an accessible glycan structure repository. Glycobiology 27(10):915–919

Traoré S, Roberts KE, Allouche D, Donald BR, André I, Schiex T, Barbe S (2016) Fast search algorithms for computational protein design. J Comput Chem 37(12):1048–1058

Traynard P, Fauré A, Fages F, Thieffry D (2016) Logical model specification aided by model-checking techniques: application to the mammalian cell cycle regulation. Bioinformatics 32(17):i772–i780

Ueda N, Aoki-Kinoshita KF, Yamaguchi A, Akutsu T, Mamitsuka H (2005) A probabilistic model for mining labeled ordered trees: capturing patterns in carbohydrate sugar chains. IEEE Trans Knowl Data Eng 17(8):1051–1064

Ugarte W, Boizumault P, Crémilleux B, Lepailleur A, Loudni S, Plantevit M, Raïssi C, Soulet A (2017) Skypattern mining: from pattern condensed representations to dynamic constraint satisfaction problems. Artificial Intell 244:48–69. Combining constraint solving with mining and learning

Vendruscolo M, Kussell E, Domany E (1997) Recovery of protein structure from contact maps. Fold Des 2(5):295–306

Verfaillie G, Jussien N (2005) Constraint solving in uncertain and dynamic environments: A survey. Constraints 10(3):253–281

Vert J-P, Jacob L (2008) Machine learning for in silico virtual screening and chemical genomics: new strategies. Comb Chem High Throughput Screen 11(8):677–685

Videla S, Saez-Rodriguez J, Guziolowski C, Siegel A (2017) caspo: a toolbox for automated reasoning on the response of logical signaling networks families. Bioinformatics 33(6):947–950

Videla S, Guziolowski C, Eduati F, Thiele S, Gebser M, Nicolas J, Saez-Rodriguez J, Schaub T, Siegel A (2015) Learning Boolean logic models of signaling networks with ASP. Theor Comput Sci 599(Supplement C):79 – 101. Advances in computational methods in systems biology

von Kamp A, Klamt S (2014) Enumeration of smallest intervention strategies in genome-scale metabolic networks. PLoS Comput Biology 10(1):e1003378

Wald R, Khoshgoftaar TM, Dittman D, Awada W, Napolitano A (2012) An extensive comparison of feature ranking aggregation techniques in bioinformatics. In: 2012 IEEE 13th international conference on information reuse and integration (IRI). IEEE, pp 377–384

Walker SI, Davies PC (2013) The algorithmic origins of life. J R Soc Interface 10(79):20120869

Wan S, Mak M-W, Kung S-Y (2014) HybridGO-Loc: mining hybrid features on gene ontology for predicting subcellular localization of multi-location proteins. PLOS ONE 9(3):1–12

Wang Z, Xu J (2013) Predicting protein contact map using evolutionary and physical constraints by integer programming. Bioinformatics 29(13):i266–i273

Wang X, Xuan Z, Zhao X, Li Y, Zhang MQ (2009) High-resolution human core-promoter prediction with CoreBoost_HM. Genome Res 19(2):266–275

Wang S, Li W, Liu S, Xu J (2016a) RaptorX-Property: a web server for protein structure property prediction. Nucl Acids Res 44(W1):W430–W435

Wang S, Li W, Zhang R, Liu S, Xu J (2016b) CoinFold: a web server for protein contact prediction and contact-assisted protein folding. Nucl Acids Res 44(W1):W361–W366

Wang S, Sun S, Li Z, Zhang R, Xu J (2017a) Accurate de novo prediction of protein contact map by ultra-deep learning model. PLOS Comput Biology 13(1):1–34

Wang Y, Mao H, Yi Z (2017b) Protein secondary structure prediction by using deep learning method. Knowl Based Syst 118:115–123

Wang S, Peng J, Ma J, Xu J (2016c) Protein secondary structure prediction using deep convolutional neural fields. Scientific Rep 6(18962)

Wei D, Zhang H, Wei Y, Jiang Q (2013) A novel splice site prediction method using support vector machine. J Comput Inf Syst 9(20):8053–8060

Wei K, Iyer R, Bilmes J (2015) Submodularity in data subset selection and active learning. In: Bach F, Blei D (eds) Proceedings of the 32nd international conference on machine learning, volume 37 of Proceedings of machine learning research, Lille, France, PMLR, pp 1954–1963

Whetzel PL, Noy NF, Shah NH, Alexander PR, Nyulas C, Tudorache T, Musen MA (2011) Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. Nucleic Acids Res 39(suppl$_2$):W541–W545

Wu B, Abbott T, Fishman D, McMurray W, Mor G, Stone K, Ward D, Williams K, Zhao H (2003) Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. Bioinformatics 19(13):1636–1643

Wu G, You J-H, Lin G (2007) Quartet-based phylogeny reconstruction with answer set programming. IEEE/ACM Trans Comput Biol Bioinform 4(1)

Wuyun Q, Zheng W, Peng Z, Yang J (2016) A large-scale comparative assessment of methods for residue–residue contact prediction. Brief Bioinform bbw106

Xia J-F, Wu M, You Z-H, Zhao X-M, Li X-L (2010) Prediction of β-hairpins in proteins using physicochemical properties and structure information. Protein Pept Lett 17(9):1123–1128

Xiao Y, Dougherty ER (2007) The impact of function perturbations in Boolean networks. Bioinformatics 23(10):1265–1273

Xue LC, Rodrigues JP, Dobbs D, Honavar V, Bonvin AM (2017) Template-based protein-protein docking exploiting pairwise interfacial residue restraints. Brief Bioinform 18(3):458–466

Yamanishi Y, Bach F, Vert J-P (2007) Glycan classification with tree kernels. Bioinformatics 23(10):1211–1216

Yanev N, Traykov M, Milanov P, Yurukov B (2017) Protein folding prediction in a cubic lattice in hydrophobic-polar model. J Comput Biology 24(5):412–421

Yang P, Ho JW, Yang YH, Zhou BB (2011) Gene-gene interaction filtering with ensemble of filters. BMC Bioinform 12(1):S10

Yang P, Humphrey SJ, James DE, Yang YH, Jothi R (2016a) Positive-unlabeled ensemble learning for kinase substrate prediction from dynamic phosphoproteomics data. Bioinformatics 32(2):252–259

Yang Y, Gao J, Wang J, Heffernan R, Hanson J, Paliwal K, Zhou Y (2016b). Sixty-five years of the long march in protein secondary structure prediction: the final stretch? Brief Bioinform bbw129

Yang P, Hwa Yang JY, Zhou BB, Zomaya AY (2010) A review of ensemble methods in bioinformatics. Current Bioinform 5(4):296–308

Yoshizumi T, Miura T, Ishida T (2000) A* with partial expansion for large branching factor problems. In: AAAI/IAAI, pp 923–929

Yuan G-C, Cai L, Elowitz M, Enver T, Fan G, Guo G, Irizarry R, Kharchenko P, Kim J, Orkin S, Quackenbush J, Saadatpour A, Schroeder T, Shivdasani R, Tirosh I (2017) Challenges and emerging directions in single-cell analysis. Genome Biol 18(1):84

Zhang Y, Rajapakse JC (2009) Machine learning in bioinformatics, vol 4. Wiley, New York

Zhang T, Faraggi E, Li Z, Zhou Y (2013) Intrinsically semi-disordered state and its role in induced folding and protein aggregation. Cell Biochem Biophys 67(3):1193–1205

Zhao Y, Hayashida M, Akutsu T (2010) Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures. BMC Bioinform 11(11):S4

Zhao Y, Hayashida M, Cao Y, Hwang J, Akutsu T (2015) Grammar-based compression approach to extraction of common rules among multiple trees of glycans and rnas. BMC Bioinform 16(1):128

Zhou R, Hansen EA (2004) Space-efficient memory-based heuristics. In: AAAI, pp 677–682

Artificial Intelligence in Biological Modelling

François Fages

Abstract Systems Biology aims at elucidating the high-level functions of the cell from their biochemical basis at the molecular level. A lot of work has been done for collecting genomic and post-genomic data, making them available in databases and ontologies, building dynamical models of cell metabolism, signalling, division cycle, apoptosis, and publishing them in model repositories. In this chapter we review different applications of AI to biological systems modelling. We focus on cell processes at the unicellular level which constitutes most of the work achieved in the last two decades in the domain of Systems Biology. We show how rule-based languages and logical methods have played an important role in the study of molecular interaction networks and of their emergent properties responsible for cell behaviours. In particular, we present some results obtained with SAT and Constraint Logic Programming solvers for the static analysis of large interaction networks, with Model-Checking and Evolutionary Algorithms for the analysis and synthesis of dynamical models, and with Machine Learning techniques for the current challenges of infering mechanistic models from temporal data and automating the design of biological experiments.

1 Introduction

"C'est aux algorithmes du monde vivant que s'intéresse aujourd'hui la biologie." François Jacob.

In the early history of Computer Science, the biological metaphor played an important role in the design of the first models of computation based on neural networks and finite state machines. The Boolean model of the behaviour of nervous systems given by McCulloch and Pitts in 1943 turned out to be the model of a finite state machine (McCulloch and Pitts 1943). This model of events in nerve nets was reworked mathematically in the mid 50s by Kleene who created the theory of finite

F. Fages (✉)
Inria Saclay – Ile de France, 1 rue Honoré d'Estienne d'Orves, Campus de l'École Polytechnique, 91120 Palaiseau, France
e-mail: Francois.Fages@inria.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_8

automata (Kleene 1956), and later on, by Von Neumann in the mid 60's with the theory of self-replicating automata (Neumann 1966).

In return for Biology, that logical formalism was applied in the early 70's by Glass and Kaufman (1973) and Thomas (1973, 1981, 1991), Thomas and D'Ari (1990) to the analysis of Gene Networks and the prediction of cell qualitative behaviours. In particular, the existence of positive circuits in the influence graph of a gene network was conjectured by Thomas, and later proved in Remy et al. (2008), Ruet (2016), Soulé (2003), Soliman (2013), to be a necessary condition for the existence of multiple steady states which interestingly explains cell differentiation for genetically identical cells (Thomas and Kaufman 2001; Naldi et al. 2010; Sánchez et al. 2008). Similarly, the existence of negative circuits is a necessary condition for genetic oscillations and homeostasis (Snoussi 1998), Some sufficient conditions for multi-stability have also been given in Feinberg's Chemical Reaction Network Theory (Feinberg 1977) and implemented in some tools such as the "Kineticist's Workbench" at MIT AI lab (Eisenberg 1991).

Nowadays, with the progress made on SAT solving, Model-Checking and Constraint Logic Programming (see chapters "Logic Programming"–"Valued Constraint Satisfaction Problems" of Volume 2), the logical modelling of biological regulatory networks is particularly relevant to reasoning on cell processes, and not only on gene networks, but also on RNA and protein networks for the study of a variety of cell processes such as the cell division cycle control (Fauré and Thieffry 2009; Traynard et al. 2016a), cell signalling (Grieco et al. 2013), metabolism regulation, and more generally for the study of interaction systems at different scales from unicellular to multicellular, tissues and ecosystems.

This research belongs to a multidisciplinary domain, called *Systems Biology* (Ideker et al. 2001) which emerged at the end of the 90's with the end of the Human Genome Project, to launch a similar effort on post-genomic data (RNA and protein interactions) and the molecular interaction mechanisms that implement signalling modules and decision processes responsible for cell behaviours. A lot of work has been done for collecting genomic and post-genomic data, making them available in databases and ontologies (Ashburner et al. 2000; Kanehisa and Goto 2000), building dynamical models of cell metabolism (Herrgård et al. 2008), signalling, division cycle, apoptosis, and publishing them in model repositories (le Novère et al. 2006).

The biological data about cell processes are however more and more quantitative, and not only about the mean of cell populations, but also more precisely about single cells tracked over days under the microscope. The advances made in the last two decades in molecular biology with high throughput technologies, have thus made crucial the need for automated reasoning tools to help

- analyzing both qualitative and quantitative data about the concentration of molecular compounds over time,
- aggregating knowledge on particular cell processes,
- building phenomenological and mechanistic models, either qualitative or quantitative,
- learning dynamical models from temporal data,

- designing biological experiments
- and automating those experiments.

It thus makes a lot of sense now to go beyond qualitative insights, toward quantitative predictions, by developing quantitative models in, either deterministic (e.g. Ordinary Differential Equations, ODE) or stochastic (e.g. Continuous-Time Markov Chains, CTMC) formalisms, and calibrating models accurately according to experimental data. On this route, Quantitative Biology pushes the development of AI techniques for reasoning both qualitatively and quantitatively about analog and hybrid analog/digital systems, taking also into account continuous time, continuous concentration values and continuous control mechanisms,

In this chapter, we review some applications of AI techniques to biological systems modelling. We mainly focus on cell processes at the unicellular level which constitutes most of the work achieved in the last two decades in the domain of computational systems biology. We also focus on a *logical paradigm for systems biology* based on temporal logics (Clarke et al. 1999) with the following identifications:

$$\begin{gathered}
\text{biological model} = \text{transition system } K \\
\text{dynamical behavior specification} = \text{temporal logic formula } \phi \\
\text{model validation} = \text{model-checking } K,\ s\ \models ?\ \phi \\
\text{model reduction} = \text{submodel-checking } K'? \subset K,\ K',\ s\ \models\ \phi \\
\text{model prediction} = \text{valid formula enumeration } K,\ s\ \models\ \phi ? \\
\text{static experiment design} = \text{symbolic model-checking } K,\ s ?\ \models\ \phi \\
\text{model inference} = \text{constraint solving } K ?,\ s\ \models\ \phi \\
\text{dynamic experiment design} = \text{constraint solving } K ?,\ s ?\ \models\ \phi
\end{gathered}$$

This approach allows us to link biological systems to formal transition systems (either discrete or continuous), and biological modelling to program verification and synthesis from behavioural specifications. This chapter is organized in that perspective. The next section reviews some formal languages for modelling biochemical interaction networks, namely reaction systems and influence systems, and their representation by logic programs. The following section presents the successful use of SAT and Constraint Logic Programming tools, for solving NP-hard static analysis problems on biological models, such as the detection of Petri Net invariants, and the detection of model reduction relationships within large model repositories, often with better performance than with dedicated tools. Section 4 reviews some temporal logic languages used for modelling the (imprecise) behaviour of biological systems, both qualitatively and quantitatively. Section 5 presents some model-checking methods and evolutionary algorithms for constraint reasoning on dynamical models and the crucial problem of parameter search in high dimension. Finally Sect. 6 is dedicated to Machine Learning methods for automating model building and biological experiment design, that probably constitutes the main challenge now in Computational Systems Biology, and an important promise of AI.

2 Modelling Biochemical Interaction Networks

The Systems Biology Markup Language (SBML) (Hucka et al. 2003, 2008) provides a common exchange format for modelling biochemical interaction systems using essentially reactions or influences, events, and various annotations for linking the objects to external databases and ontologies. SBML has made possible the exchange of models between modellers, and the building of model repositories such as BioModels (le Novère et al. 2006)[1] or KEGG (Kanehisa and Goto 2000). BioModels currently contains 612 manually curated models, 873 non curated, and 150000 models imported from other pathway resources, including 2641 models of whole genome metabolisms. This flat list of models can be accessed through the Gene Ontology[2] which defines a set of concepts used to describe gene function, and relationships between these concepts (see chapter "Reasoning with Ontologies" of Volume 1). It classifies functions along three aspects:

- molecular function, i.e. molecular activities of gene products,
- cellular component where gene products are active,
- biological process pathways and larger processes made up of the activities of multiple gene products.

SBML is nowadays supported by a majority of modelling tools such as Copasi (Hoops et al. 2006), or Biocham[3] (Calzone et al. 2006b; Fages and Soliman 2008b) used in the examples below, and graphical editors such as Cell Designer (Funahashi et al. 2008). In this section we present the basic formalisms of reaction and influence systems with some details, in order to explain in the following sections various automated reasoning tools that have been used to reason about them and build predictive models of biological processes.

2.1 Reaction Systems

2.1.1 Syntax

In this chapter, unless explicitly noted, we will denote by capital letters (e.g. S) sets or multisets, by bold letters (e.g., $\mathbf{x}$) vectors and by small roman or Greek letters elements of those sets or vectors (e.g. real numbers, functions). For a multiset M, $\mathrm{Set}(M)$ will denote the set obtained from the support of M, and brackets like $M(i)$ will denote the multiplicity in the multiset (usually the stoichiometry). $\geq$ will denote the pointwise order for vectors, multisets and sets (i.e. inclusion).

[1] http://biomodels.net.

[2] http://geneontology.org.

[3] http://lifeware.inria.fr/biocham4.

We give here general definitions for directed reactions with inhibitors (Fages et al. 2015). A *reaction* over molecular species $S = \{x_1, \ldots, x_s\}$ is a quadruple (R, M, P, f), also noted below in Biocham syntax (`f for R / M => P`), where R is a multiset of *reactants*, M a set of *inhibitors*, P a multiset of *products*, all composed of elements of S, and $f : \mathbb{R}^s \to \mathbb{R}$ is a mathematical function over molecular species concentrations, called the *rate function*. A *reaction system* $\mathscr{R}$ is a finite multiset of reactions.

It is worth noting that a molecular species in a reaction can be both a reactant and a product (i.e. a catalyst), or both a reactant and an inhibitor (e.g. Botts–Morales enzymes). Such molecular species are not distinguished in SBML and are both called reaction *modifiers*. Unlike SBML, we consider directed reactions only (reversible reactions being represented here by two reactions) and enforce the following compatibility conditions between the rate function and the structure of a reaction: a reaction (R, M, P, f) over molecular species $\{x_1, \ldots, x_s\}$ is *well formed* if the following conditions hold:

1. $f(x_1, \ldots, x_s)$ is a partially differentiable function, non-negative on $\mathbb{R}^s_+$;
2. $x_i \in R$ if and only if $\partial f / \partial x_i(\mathbf{x}) > 0$ for some value $\mathbf{x} \in \mathbb{R}^s_+$;
3. $x_i \in M$ if and only if $\partial f / \partial x_i(\mathbf{x}) < 0$ for some value $\mathbf{x} \in \mathbb{R}^s_+$.

A reaction system is well formed if all its reactions are well formed. This is the case for instance of reaction systems with mass action law kinetics which take as rate functions the product of the concentration of the reactants with some constant rate parameter.

Example 1 The mathematical prey-predator model of Lotka–Volterra provides a model for oscillating chemical reactions between a proliferating prey molecular species A and a predator molecule B which can be represented by the following well-formed reaction system with mass action law kinetics:

```
k1*A for A => 2*A.
k2*A*B for A+B => 2*B.
k3*B for B => _.
```

$k1$ is the rate constant for the autocatalysis of A, $k2$ the rate constant for the transformation of A into B by B, and $k3$ the degradation rate constant for B. Note that in this example, the reactions have no inhibitors. If the synthesis of A were competing with the synthesis of another molecular species C (due to some hidden mechanism), this could be represented with two synthesis reactions with mutual inhibitors as follows:

```
k2*A/(k4+C) for A / C => 2*A.
k5*C/(k6+A) for C / A => 2*C.
```

2.1.2 Hierarchy of Semantics

The dynamics of a reaction system can be defined in multiple ways, either qualitatively or quantitatively, in different formalisms. However, those multiple interpretations of a same reaction system $\mathscr{R}$ can be formally related by abstraction relationships in the framework of abstract interpretation (Cousot and Cousot 1977) to form a

hierarchy of semantics corresponding to different abstraction levels to reason about them (Fages and Soliman 2008a).

The *differential semantics* associates a time varying concentration to each molecular species, and an Ordinary Differential Equation (ODE) system to the reactions, by summing for each molecular variable the rate functions multiplied by the stoichiometric coefficients of the reactions that modify it, i.e. for $1 \leq j \leq s$

$$\frac{dx_j}{dt} = \sum_{(R_i, M_i, P_i, f_i) \in \mathscr{R}} (P_i(j) - R_i(j)) \times f_i$$

It is worth noting that in this interpretation, the inhibitors are supposed to decrease the reaction rate, but do not prevent the reaction to proceed.

In Example 1, we get the classical Lotka–Volterra equations

$$dB/dt = k1 * A * B - k3 * B$$

$$dA/dt = k2 * A - k1 * A * B$$

and the well-known oscillations between the concentrations of preys and predators, as shown in Fig. 1 left.

The *stochastic semantics* associates to each molecule its discrete quantity, and to reactions a transition relation $\longrightarrow_S$ between discrete states, i.e. vectors $\mathbf{x}$ of $\mathbb{N}^s$. A transition is enabled in state $\mathbf{x}$ by a reaction $(R_i, M_i, P_i, f_i) \in \mathscr{R}$ if there are enough reactants, and the propensity is defined by evaluating the rate function f_i in $\mathbf{x}$:

$$\forall (R_i, M_i, P_i, f_i) \in \mathscr{R}, \mathbf{x} \longrightarrow_S \mathbf{x}' \text{ with propensity } f_i \text{ if } \mathbf{x} \geq R_i, \mathbf{x}' = \mathbf{x} - R_i + P_i$$

The transition probabilities between discrete states are obtained by normalization of the propensities of all the enabled reactions, and the time of the next reaction is given by the propensities with an exponential distribution (Gillespie 1977; Kurtz 1978). It is worth noting that in this interpretation like in the differential semantics, the inhibitors decrease the reaction propensity but do not prevent the reaction to proceed.

In Example 1, the stochastic interpretation can exhibit some noisy oscillations similar to the differential interpretation, but also, and almost surely, the extinction of the predator as shown in Fig. 1 right.

The *discrete* or *Petri Net semantics* defines a similar transition relation $\longrightarrow_D$ over discrete states, but ignoring the rate functions. It is thus a trivial abstraction of the stochastic semantics by a forgetful functor, we have

$$\forall (R_i, M_i, P_i, f_i), \mathbf{x} \longrightarrow_D \mathbf{x}' \text{ if } \mathbf{x} \geq R_i, \mathbf{x}' = \mathbf{x} - R_i + P_i$$

The *Boolean semantics* is similar to the discrete semantics but on Boolean vectors x of $\mathbb{B}^s$, obtained by the "zero, non-zero" abstraction of integers ($>0 : \mathbb{N} \to \mathbb{B}$. With

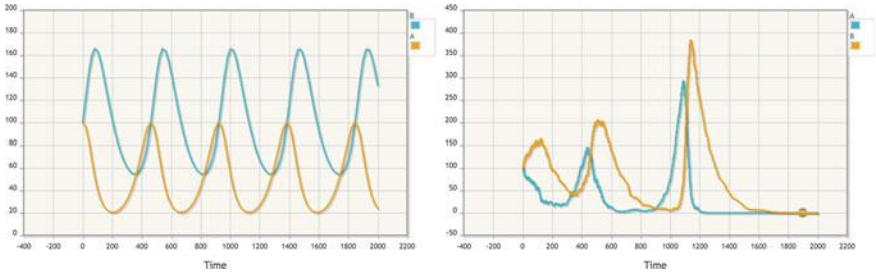


Fig. 1 ODE and stochastic simulation of Lotka Volterra prey-predator model

this abstraction, when the number of a molecule is decremented, it can still remain present, or become absent. It is thus necessary to take into account all the possible complete consumption or not of the reactants in order to obtain a correct Boolean abstraction of the discrete and stochastic semantics (Fages and Soliman 2008a). The *Boolean transition system* $\longrightarrow_B$ is thus defined by considering all subsets of the set of reactants $\mathrm{Set}(R_i)$:

$$\forall (R_i, M_i, P_i, f_i), \forall C \in \mathscr{P}(\mathrm{Set}(R_i)), \mathbf{x} \longrightarrow_B \mathbf{x}' \text{ if } \mathbf{x} \supseteq \mathrm{Set}(R_i), \mathbf{x}' = \mathbf{x} \setminus C \cup \mathrm{Set}(P_i)$$

Interestingly, with these definitions, the last three semantics are related by successive Galois connections (Fages and Soliman 2008a). The set of Boolean traces is thus a correct abstraction of the stochastic traces for any rate functions, in the sense that the Boolean abstraction of the stochastic traces is contained in the set of traces of the Boolean semantics. This means that *if a behaviour is not possible in the Boolean semantics, it is not possible in the stochastic semantics* whatever the reaction rate functions are, and justifies the use of Boolean reasoning tools for many questions.

On the other hand, the differential semantics does not constitute an abstraction of the stochastic semantics, but provides, under strong assumptions, an approximation of the mean stochastic behavior, for instance when the number of each molecule tends to the infinity (Gillespie 1977; Kurtz 1978).

Example 2 In the Lotka-Volterra example, one can show that the extinction of the predator is almost sure in the stochastic semantics, whereas the differential semantics exhibits sustained oscillations (the condition on large numbers of molecules is clearly not satisfied). The Boolean semantics exhibits a set of possible Boolean behaviors which over-approximates the set of stochastic traces. Under this Boolean interpretation, one can observe either the stable existence of the prey (in case of extinction of the predator), the unstable existence of the predator (which can always disappear), or the disappearance of both of them, but not the extinction of the prey without the preceding extinction of the predator, nor any Boolean oscillation in absence here of synthesis reaction (e.g. migration). These properties can be directly expressed by Temporal Logic formulae described in Sect. 4.1, and automatically generated by the model-checking techniques described in Sect. 5.1 as follows:

```
biocham: present({A,B}).
biocham: generate_ctl_not.
reachable(stable(A))
reachable(steady(B))
reachable(stable(not A))
reachable(stable(not B))
checkpoint(B,not(A))
```

In presence of synthesis reactions such as protein synthesis, the discrepancies between the differential and stochastic interpretations may be less extreme. For these reasons the differential semantics is widely used for quantitative biological modelling. The following example shows a typical case of biochemical reaction system for signalling, where the differential semantics approximates the mean stochastic behavior.

Example 3 The MAPK (Mitogen Activated Protein Kinase) biochemical reaction system is an extremely frequent signalling module that exists in several copies in eukaryote cells for different signalling tasks. This network is composed of three stages for a total of 30 reactions, where at each stage a protein gets phosphorylated once or twice, and under this phosphorylated form, catalyzes the phosphorylations of the next stage. The input `E1` of this signalling cascade, directly linked to the transmembrane receptor, phosphorylates the kinase `KKK` of the first stage which then phosphorylates the kinase `KK` which itself phosphorylates the protein `K` to produce the output of the cascade `PP_K` which can migrate to the nucleus and modify gene transcription. Figure 2 shows the three levels structure of the reaction system.

Figure 3 shows the ODE simulation and the dose-response diagram (i.e. `PP_K`, `PP_KK` and `P_K` at steady state versus `E1` varying in the range $[1e-6, 1e-4]$). This shows that MAPK acts as an analog-digital converter in the cell, with the stiffest response at the third level output (Huang and Ferrell 1996).

It is worth noticing that the reaction inhibitors have not been used for the definition of the hierarchy of semantics in this section. The reason is that in the differential

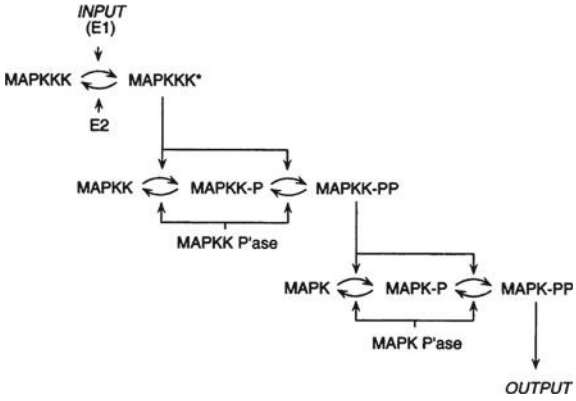


Fig. 2 MAPK signalling reaction network structure, with three levels of simple (at the first stage) and double (at the second and third stages) phosphorylations, with reverse dephosphorylation reactions catalyzed by phosphatases (Huang and Ferrell 1996)

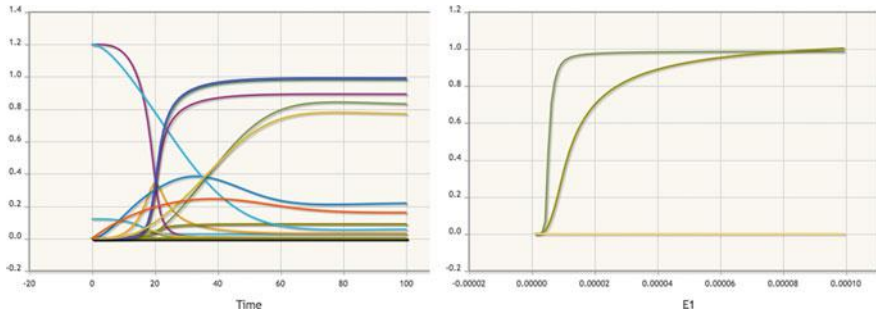


Fig. 3 ODE simulation of the MAPK signalling model, and dose-response diagram showing stiffer all-or-nothing response at lower levels of the cascade, revealing the analog-digital converter function of the MAPK circuit

semantics an inhibitor decreases the rate of a reaction without preventing it completely from proceeding. One can also define a *Boolean semantics with negation* where the inhibitors of a reaction are seen as a conjunction of negative conditions that must be satisfied for the reaction to proceed, by:

$$\forall(R_i, M_i, P_i, f_i)\forall C \in \mathscr{P}(\mathrm{Set}(R_i))\mathbf{x} \longrightarrow_{BN} \mathbf{x}'$$
$$\text{if } \mathbf{x} \supseteq \mathrm{Set}(R_i), \mathbf{x} \cap M_i = \emptyset, \mathbf{x}' = \mathbf{x} \setminus C \cup \mathrm{Set}(P_i)$$

This interpretation is used in many systems, including Boolean Petri Nets and Rewriting Logic (Eker et al. 2002) yet with no connection to the other semantics.

2.1.3 Hybrid Discrete-Continuous Models

In the perspective of applying engineering methods to the analysis and control of biological systems, the issue of building complex models by composition of elementary models is a central one. Reaction systems can be formally composed by the multiset union of the reactions and interpreted in one common semantics, but there is also a need to compose models with different semantics. For instance, it makes a lot of sense to combine a differential model of protein activation for high numbers of molecules, with a Boolean or stochastic model of gene expression, since genes are in single or double copies in a cell.

The hierarchy of semantics of reaction systems provides a clear picture for studying the combination of several reaction models with different semantics and designing hybrid discrete/continuous digital/analog models of cell processes. A *hybrid model* is a model obtained by composition of models with heterogeneous semantics (continuous, stochastic, Boolean, etc.), and *hybrid simulation* is the topic of simulating such hybrid models. In Chiang et al. (2015), it is shown that the combination of events with kinetic reactions, as already present in SBML, provides enough expressive power for combining the discrete and continuous semantics of reaction systems. Such hybrid reaction systems can also be visualized as hybrid automata (Henzinger

1996) in which there is a state with a particular ODE for each combination of the trigger values, and there is a transition from one state to another state when at least one trigger changes value from false to true in the source state.

Hybrid modelling is used in Systems Biology for reducing the complexity of modelling tasks (Alur et al. 2001; Berestovsky et al. 2013), e.g. in signalling (Ghosh and Tomlin 2001) cell cycle control (Singhania et al. 2011), gene regulation (Matsuno et al. 2000; Ahmad et al. 2006), and most notably, for achieving whole cell simulation (Karr et al. 2012).

2.2 *Influence Systems*

Influence systems are a somewhat simpler formalism which is also widely used by modellers to merely describe the positive and negative influences between molecular species, without fixing their implementation by biochemical reactions. In particular, Thomas's regulatory networks form a particular class of Boolean influence systems, implemented in modelling tools such as GINsim (Naldi et al. 2009), GNA Batt et al. (2012) or Griffin (Rosenblueth et al. 2014). It is also worth mentioning that influence systems with spatial information are developed in Chazelle (2012) as a formalism particularly suitable for describing natural algorithms in life sciences and social dynamics.

2.2.1 Syntax

In Thomas's framework, a regulatory network is defined by an influence graph given with a Boolean update function for each node. In order to define the other interpretations of an influence system, we shall distinguish here in the syntax the conjunctive conditions from the disjunctive conditions, with the convention that an influence on a target with several sources denotes a conjunctive condition, while different influences on a same target express a disjunction of conditions. Given a set $S = \{x_1, \ldots, x_s\}$ of molecular species, an *influence system* I is a set of quintuples (P, N, t, σ, f) called *influences*, where $P \subset S$ is called the *positive sources* of the influence, $N \subset S$ the *negative sources*, $t \in S$ is the *target*, *sign* $\sigma \in \{+, -\}$ is the sign of the influence, and f is a real-valued mathematical function of $\mathbb{R}^s$, called the *force* of the influence. The influences of sign $+$ are called *positive influences* and those of sign $-$, *negative influences*. They are noted in Biocham syntax (`f for R/M -> P`) and (`f for R/M -< P`) respectively.

Example 4 The prey-predator model of Lotka–Volterra of Example 1 can also be presented by the following system of four influences

```
k1*A*B for A,B -< A.
k1*A*B for A,B -> B.
k2*A for A -> A.
k3*B for B -< B.
```

The variant where a species C competes with A for nutrients gives an example of negative sources in the positive influences for proliferation:

```
k2*A/(k4+C) for A/C -> A.
k5*C/(k6+A) for C/A -> C.
```

The distinction between the positive and negative sources of an influence (either positive or negative) is similar to the distinction between the reactants and the inhibitors of a reaction. An influence (P, N, t, σ, f) is *well formed* if the following conditions hold:

1. $f(x_1, \ldots, x_s)$ is a partially differentiable function, non-negative on $\mathbb{R}_+^s$;
2. $x_i \in P$ if and only if $\sigma = +$ (resp. $-$) and $\partial f/\partial x_i(\mathbf{x}) > 0$ (resp. < 0) for some value $\mathbf{x} \in \mathbb{R}_+^s$;
3. $x_i \in N$ if and only if $\sigma = +$ (resp. $-$) and $\partial f/\partial x_i(\mathbf{x}) < 0$ (resp. > 0) for some value $\mathbf{x} \in \mathbb{R}_+^s$;
4. $t \in P$ if $\sigma = -$.

2.2.2 Semantics

Given a set of species $S = \{x_1, \ldots, x_s\}$ and an influence system $\mathscr{I}$ over S, the *differential* semantics associates the following ODE system:

$$\frac{dx_k}{dt} = \sum_{(P_i, N_i, x_k, +, f_i) \in \mathscr{I}} f_i - \sum_{(P_j, N_j, x_k, -, f_j) \in \mathscr{I}} f_j$$

Intuitively, it adds up all the forces of the positive influences on x_k and subtracts all the forces of the negative influences on x_k in the derivative of x_k over time. For instance, in Example 4, one can check that we get the same ODEs as in Example 1.

It is worth noticing that the negative sources in a well-formed influence decrease the force of the influence but do not disable it. Consequently, the *stochastic* semantics of an influence system with forces, can be defined similarly to reaction systems, by a transition system, noted $\longrightarrow_S$, between discrete states, i.e. vectors $\mathbf{x}$ of $\mathbb{N}^s$, with the condition that the positive sources are present in sufficient number, without any condition on the negative sources:

$$\forall (P_i, N_i, A_i, \sigma_i, f_i), \mathbf{x} \longrightarrow_S^{f_i} \mathbf{x}' \text{ with propensity } f_i \text{ if } \mathbf{x} \geq P_i, \mathbf{x}' = \mathbf{x}\ \sigma_i\ A_i$$

Transition probabilities between discrete states are obtained through normalization of the propensities of all the enabled transitions, with time of next reaction (Gillespie 1977). As before, the *discrete* (or *Petri Net*) semantics simply ignores the forces:

$$\forall (P_i, N_i, A_i, \sigma_i, f_i), \mathbf{x} \longrightarrow_D \mathbf{x}' \text{ if } \mathbf{x} \geq P_i, \mathbf{x}' = \mathbf{x}\ \sigma_i\ A_i$$

and the *Boolean semantics* is defined on Boolean vectors x of $\mathbb{B}^s$, by the "zero, non-zero" abstraction. It is worth noticing that in this view, and similarly to reaction systems, the Boolean semantics associates two transitions to a negative influence:

$$\forall (P_i, N_i, A_i, +, f_i), \mathbf{x} \longrightarrow_B \mathbf{x}' \text{ if } \mathbf{x} \geq P_i, \mathbf{x}' = \mathbf{x} + A_i$$

$$\forall (P_i, N_i, A_i, -, f_i), \mathbf{x} \longrightarrow_B \mathbf{x}' \text{ if } \mathbf{x} \geq P_i, \mathbf{x}' = \mathbf{x} - A_i \text{ or } \mathbf{x}' = \mathbf{x}$$

That Boolean semantics is positive in the sense that it ignores the negative sources of an influence and contains no negation in the influence enabling condition.

In Lotka-Volterra Examples 1 and 4, the Boolean transitions are the same in this particular case, since there is no reaction that can produce a simultaneous change of the Boolean values of both the prey and the predator. However in general, reaction systems can produce simultaneous Boolean updates which cannot be represented by an influence system.

2.2.3 Expressive Power Compared to Reaction Systems

One can show that any (well-formed) influence system with forces can be represented by a (well-formed) reaction system, with the same Boolean, discrete, stochastic and differential semantics (Fages et al. 2018), i.e. an influence system can always be simulated by a reaction system for the different semantics. The converse does not hold for the discrete semantics. For instance for the Boolean semantics, the decomplexation reaction $C \Rightarrow A + B$, has a transition from the state $(A, B, C) = (0, 0, 1)$ to $(1, 1, 0)$ which is obviously not possible in any influence system since only one variable can change in one transition. What is possible is to simulate a reaction system by an influence system which over-approximates its Boolean semantics.

However, the converse holds for the differential semantics, i.e. (well-formed) influence and reaction systems have the same expressive power (Fages et al. 2018). This means that as far as the differential semantics is concerned, the influence systems have the same expressive power as reaction systems and there is no theoretical reason to develop a reaction model. This does not mean that there is a canonical reaction system associated with an influence system. Generally, different implementations with reactions are possible without changing the differential semantics. They represent extra information that is irrelevant to the analysis or simulation of the differential equations, but can lead to different stochastic simulations for instance.

2.2.4 Functional Boolean Semantics with Negation *à la* Thomas

The formalism of Thomas and D'Ari (1990) is a Boolean variant of influence systems which considers negative conditions and deterministic functional updates instead of relational updates. The success of this formalism lies, on the one hand, in the beautiful theory of necessary conditions for oscillations and multistability (Remy et al. 2008; Ruet 2016) which explains for instance cell differentiation by the purely qualitative existence of a positive circuit in the influence graph of the system, and, on the other hand, for its widespread use for the logical modelling of a variety of cell processes

beyond gene networks, such as cell cycle (Fauré et al. 2009) cell signalling (Grieco et al. 2013) or morphogenesis (González et al. 2008; Sánchez et al. 2008).

In the *Boolean semantics with negation*, the negative sources are interpreted as negations in the enabling condition, as follows:

$$\forall (P_i, N_i, A_i, \sigma_i, f_i), \mathbf{x} \longrightarrow_{BN} \mathbf{x}' \text{ if } \mathbf{x} \geq P_i,\ \mathbf{x} \cap N_i = \emptyset,\ \mathbf{x}' = \mathbf{x}\ \sigma_i\ A_i$$

This interpretation makes it possible to represent any *Boolean unitary transition system*, i.e. any transition system that updates at most one variable of $\mathbf{x}$ in each transition (Fages et al. 2018). Furthermore, the Boolean semantics of Thomas's networks is *functional*, in the sense that the next Boolean state $\mathbf{x}'$ is defined by a Boolean function $\phi(\mathbf{x})$. The synchronous semantics is thus deterministic and the non-deterministic asynchronous semantics is obtained by interleaving, i.e. by considering all the possible transitions that change the Boolean value of one of the genes at a time.

For these reasons, a truly non-deterministic influence system such as

$$\{(A, \emptyset, B, +, f),\ (A, \emptyset, B, -, g)\}$$

(for which the transition relation is not a function) cannot be represented in Thomas's setting. This excludes self-loops in the state transition graph (on non-terminal states). This is even more striking in Thomas's multilevel setting, where the above system can (in the discrete semantics) have transitions from $(1, 1)$ both to $(1, 0)$ and to $(1, 2)$. That would necessitate the corresponding logical parameter for B to be at the same time <1 and >1. Conversely, any Thomas's gene regulatory network can be represented by an influence system with the Boolean semantics with negation.

2.3 Logic Programming

The transition systems defined in Sect. 2.2.2 can be straightforwardly represented by Logic Programs (LP), and Constraint Logic Programs (CLP) for the quantitative semantics, where the states and the transition relation are defined by atoms, and the transition enabling conditions are defined by Horn clauses (see chapter "Logic Programming" of Volume 2). This LP representation of reaction and influence systems suggests the use of a variety of LP tools for reasoning about them, such as deductive model-checking (Delzanno and Podelski 2001; Chabrier and Fages 2003), inductive logic programming (Muggleton 1995; Fages and Soliman 2008c), and probabilistic logic programming (Angelopoulos and Muggleton 2002a).

In Inoue (2011), it is shown how Thomas's Boolean networks can be directly represented by Normal Logic Programs (NLP), and how their trajectories and attractors can be computed with methods based on the similarity between the fixed points of Boolean networks and the immediate consequence operator T_P operator of NLPs. In particular, point attractors of both synchronous and asynchronous Boolean networks are characterized as the supported models of their associated logic programs so that SAT techniques can be applied to compute them.

Furthermore, NLPs provide a first-order representation which can be used to to describe the dynamics of influence systems on an infinite domains, such as the Petri Net semantics. In return for Logic Programming, this shows that logic programs that have cyclic attractors and are inconsistent under the supported or stable model semantics (Fages 1994) can have meanings under the "attractor semantics" for NLPs.

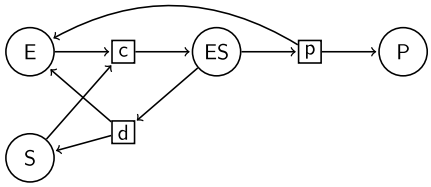
3 Automated Reasoning on Model Structures

3.1 Petri Net Invariants

Beyond being a useful interpretation of reaction and influence systems in its own right, the Petri Net semantics provides interesting information on the differential and stochastic semantics of reaction systems. Petri nets have been introduced historically as a simple chemically-inspired formalism for describing and analyzing concurrent, asynchronous, non-deterministic, and possibly distributed, information processing systems (Peterson 1981). The use of Petri nets for studying biochemical reaction systems, by mapping molecular species to places and reactions to transitions, was considered quite late in Reddy et al. (1993) for the analysis of metabolic networks. In this context, the traditional Petri net concepts of place-invariants (P-invariants), transition-invariants (T-invariants), siphons and traps have shown to have important applications especially in metabolism (von Kamp and Schuster 2006; Zevedei-Oancea and Schuster 2003; Varma and Palsson 1994; Rezola et al. 2011; Larhlimi and Bockmayr 2009; de Figueiredo et al. 2009; Herrgård et al. 2008). This motivated the search for efficient algorithms to scale-up to the size of biological models in model repositories, and revealed the astonishing performance of SAT and Constraint Logic Programming solvers which can outperform dedicated algorithms through a straightforward Boolean or Finite Domain constraint modelling (Soliman 2012; Nabli et al. 2016).

A P-invariant is a multiset of places V (i.e. molecular species) such that the sum of the markings (i.e. numbers of molecules) remains constant for any scheduling of the transitions, i.e. $V.I = 0$ where I is the incidence matrix of the Petri net $I = \sum_i P_i - R_i$ with the notations of Sect. 2.1.2, i.e. I_{ij} is the number of arcs from transition i to place j, minus the number of arcs from place j to transition i. Such a P-invariant represents a structural conservation law between molecular species, and corresponds to a linear invariant in the ODE semantics of the reactions, i.e. a multiset of differential functions having their sum equal to zero which corresponds to a multiset of molecules whose sum of concentrations remains constant.

Example 5 The Michaelis–Menten enzymatic reaction system is composed of three reactions: one of complexation and one of decomplexation of the enzyme with the substrate, and one of transformation of the product with release of the enzyme. This simple system shown in Fig. 4 has two minimal P-invariants which express the conservation of the enzyme in free and complexed form, and the conservation of the




```
k1*E*S for E+S => ES.
k2*ES for ES => E+S.
k3*ES for ES => E+P.

biocham: search_conservations.
E+ES
ES+P+S

biocham: list_ode.
d(P)/dt=k3*ES
d(E)/dt=(k2+k3)*ES-k1*E*S
d(ES)/dt=k1*E*S-(k2+k3)*ES
d(S)/dt=k2*ES-k1*E*S
```

Fig. 4 Michaelis–Menten system of three reactions representing the binding of an enzyme on its substrate and its transformation in a product, and computation of the two minimal P-invariants $\{E,\ ES\}$ and $\{ES,\ P,\ S\}$ corresponding to linear invariants of the differential semantics

substrate in free, complexed and product form. These structural conservation laws can also be seen in the ODE semantics of the model by summing the corresponding differential functions.

The MAPK model of Example 3 uses Michaelis–Menten reactions for each phosphorylation and dephosphorylation step. It has seven P-invariants, one for each kinase and phosphatase expressing its conservation among its different phosphorylated and complexed forms.

P-invariants can be computed either by standard Fourier–Motzkin elimination (Colom and Silva 1991), or by linear algebra methods such as QR-factorization, Mixed Integer Programming, or more simply, and in fact more efficiently, by Constraint Logic Programming methods over finite domains, CLP(FD). The idea here is to solve the equation $V.I = 0$ in $V \in \mathbb{N}^s$ as a Constraint Satisfaction Problem (CSP) over finite domains by posting

- $V.R_i = V.P_i$ for each reaction i,
- $V.1 > 0$,

and by enumerating the values of V from low to high for finding P-invariants that are then checked for minimality by subsumption check (Soliman 2012).

Beyond its efficiency, the beauty of the CSP approach is that it generalizes straightforwardly to the computation of other invariants. T-invariants are the dual notion of P-invariants. A T-invariant is a multiset V of transitions such that $I.V = 0$, i.e. a multiset of reaction firing that leave invariant any marking. T-invariants revealed to be equivalent to the notion of extremal fluxes in metabolic networks (von Kamp and Schuster 2006; Zevedei-Oancea and Schuster 2003; Varma and Palsson 1994), one of the main tools for analyzing and optimizing metabolic networks (Rezola et al. 2011; Larhlimi and Bockmayr 2009; de Figueiredo et al. 2009; Herrgård et al. 2008). Furthermore in CSP, just by replacing equality constraints by inequalities, for instance $V.I \leq 0$ or $I.V \geq 0$, one can compute static subinvariants of markings or fluxes which can only grow or decrease during simulation (Soliman 2012). To reduce

the combinatorial complexity, recent results using SAT modulo theory (SMT) solver have shown further improvements for the enumeration of extremal flux modes (Peres et al. 2014).

Siphons and traps are other interesting Petri Net concepts. They denote meaningful pools of places that display a specific behaviour in the Petri net dynamics, and that guarantee some persistence properties, independently of the rate functions. A siphon is a set of places that, once unmarked, remains unmarked. A trap is a set of places that, once marked, can never loose all its tokens. These structural properties provide sufficient conditions for reachability (whether the system can produce a given protein or reach a given state from a given initial state) and liveness (deadlock freedom from a given initial state) properties in ordinary Petri nets. It has been shown that the problems of existence of a minimal siphon of a given cardinality, or containing a given place, are NP-complete. In Nabli et al. (2016), a Boolean model is proposed to solve these minimal enumeration problems, either by calling a SAT solver iteratively, or by backtracking with a Constraint Logic Program (CLP) over Booleans. Interestingly, the SAT and CLP solvers both outperform by one or two orders of magnitude the state-of-the-art algorithms from the Petri net community described in Cordone et al. (2005) for computing minimal sets of siphons and traps, that have already been shown to outperform Mixed Integer Linear Programs. On a benchmark of 345 biological models from the curated part of the BioModels repository (le Novère et al. 2006), the Boolean method for enumerating the set of all minimal siphons takes a few seconds in MiniSAT. It also scales very well in the size of the net. The CLP(B) program also solves all but one instances of the benchmark, with a better performance than MiniSAT in average, but does not scale-up as well on the largest size Petri nets, such as for instance on Kohn's map with 509 species and 775 reactions. The efficiency of the MiniSAT and CLP(B) methods for enumerating in a few seconds the set of all solutions of an NP-complete problem for all, including large, instances of the BioModels benchmark is quite surprising. In Nabli et al. (2016), it is shown that the SAT phase transition threshold and complexity wall is traversed on those instances, but that the problem is tractable on graphs with bounded treewidth which seems to be the case of biochemical networks since most models in BioModels have a small treewidth less than 10. Still this does not explain why SAT and CLP solvers perform so well on this problem.

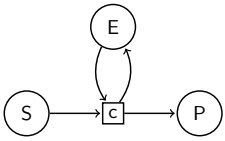
3.2 Graph Matching

Models in Systems Biology are built with two somewhat contradictory perspectives:

- Models for aggregating knowledge on particular cell processes, in this perspective the more detailed the better;
- Models for answering particular questions on cell processes, in this perspective the more abstract the better, for getting rid of useless details that are not necessary to the questions at hand.

One way to reconcile these two perspectives is to relate models by model reduction relationships, that is currently not the case in model repositories. Model reduction is a central topic in dynamical systems theory, for reducing the complexity of detailed models, finding important parameters, and developing multi-scale models for instance. While perturbation theory is a standard mathematical tool to analyze the different time scales of a dynamical system, and decompose the system accordingly, Systems Biology needs novel methods for comparing and reducing models on a very large scale.

Graph matching techniques can be used to detect model reduction relationships between models within large repositories like BioModels. However the standard notion of subgraph isomorphism (SISO) for finding graph motifs is not adequate. For instance, the very basic reduction of Michaelis–Menten which consists in reducing the system of three reactions of Example 5 to one single catalytic reaction `E+S => E+P`, produces the graph



which is not isomorphic to a subgraph of the graph of Example 5. In this example, the reduced graph can be obtained from the source graph by a sequence of *delete and merge operations on species and reaction* vertices. These transformations can typically be justified in chemistry by considering for instance: (i) reaction deletions for slow reverse reactions, (ii) reaction mergings for reaction chains with a limiting reaction, (iii) molecular species deletions for species in excess and (iv) molecular mergings for quasi-steady state approximations.

This operational view of graph reduction by graph transformation operations is equivalent to the existence of a subgraph (corresponding to delete operations) epimorphism (i.e. surjective homomorphism, corresponding to merge operations) from a source graph to a reduced graph (Gay et al. 2010). Formally, let G and G' denote graphs, with $G = (V, A)$ and $G' = (V', A')$, an *epimorphism* from G to G' is a surjective function $f : V \to V'$ such that

- for all $u, v \in V$, if $(u, v) \in A$, then $(f(u), f(v)) \in A'$ (graph homomorphism), and,
- for all $(u', v') \in A'$, there exists $(u, v) \in A$ such that $f(u) = u'$ and $f(v) = v'$ (surjectivity on arcs).

The subgraph of G induced by a subset of vertices $U \subseteq V$ of G, is $G_{\downarrow U} = (U, A \cap (U \times U))$. A *subgraph epimorphism* (SEPI) from G to G' is an epimorphism f from an induced subgraph G_0 of G to G'.

In Example 5, the two graphs of the Michaelis–Menten reduction, are related by a SEPI where the induced subgraph of the first graph is obtained by deleting the vertices ES and d, and where both reaction vertices c and p are mapped to the vertex c of the second graph.

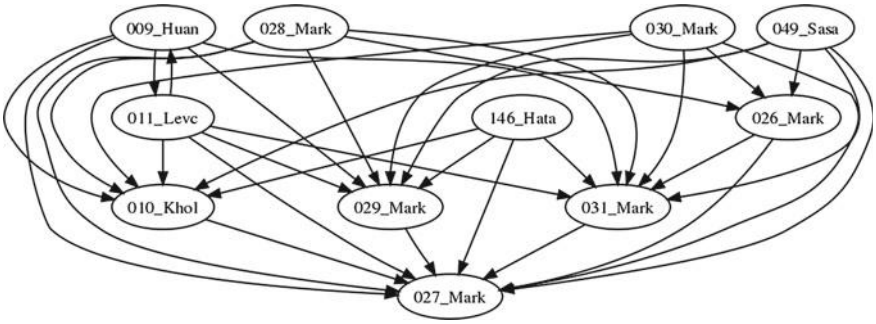


Fig. 5 Hierarchy of MAPK models in BioModels automatically constructed by SEPI matching (Gay et al. 2010). Schoeberl's model 14 and Levchenko's model 19 are not represented here, they do not map each other but map to the other models

Subgraph epimorphisms differ from subgraph isomorphisms by allowing merge operations in addition to delete operations. On undirected graphs, SEPIs differ from graph minors in several points: non adjacent vertices may be merged, merging adjacent vertices creates loops, and arcs cannot be deleted without deleting or merging vertices. Determining whether there exists a SEPI from a graph G to a graph G' is NP-complete (Gay et al. 2014). Nevertheless a simple CLP(FD) program or SAT solver can solve this problem on all pairs of reaction graphs in the repository BioModels with just a few timeouts for some pairs of models.

Graph morphisms can be modelled by introducing one variable per node of the source graph, with the set of nodes of the target graph as integer domain. A variable assignment then represents a mapping from the source nodes to the target nodes. The morphism condition itself is written with a tabular constraint of CLP(FD) which forces a tuple of variables to take its value in a list of tuples of integers. The surjectivity property can be enforced by creating variables for the target arcs with the set of source arcs as domain, and using the global constraint `all_different` of CLP(FD). Then, the enumeration on the target arc variables enforces surjectivity and the enumeration of node variables enforce the computation of a complete morphism (Gay et al. 2014).

Figure 5 shows the hierarchy of MAPK signalling models in BioModels that has been automatically reconstructed by graph matching, i.e. by computing SEPIs between all pairs of models. The arrows between models denote model reductions and double arrows denote reaction graph isomorphisms, e.g. between models 9 and 11 which differ just by molecule names and rate functions. These models have the same structure shown in Example 3. They reduce to model 10 which is also three level but without the reverse dephosphorylation reactions. It reduces also to models 29 and 27 which are one level models with ad without the dephosphorylation reactions.

It is remarkable that meaningful clusters of reaction models can be automatically reconstructed just by comparing the structure of the reaction graphs with the appropriate notion of SEPI morphism (Gay et al. 2010). This means that the function of

natural biochemical networks are largely determined by the structure of the networks and not that much by the rate functions and parameter values of each reaction. This recurrent remark can be made in other ways but is not well understood. We see it as a fundamental consequence of the necessary robustness of natural biochemical circuits to implement their function since they cannot realistically rely on precise rate functions.

4 Modelling Dynamical Behaviours

4.1 Propositional Temporal Logics

In the early days of computational Systems Biology, propositional temporal logic was soon proposed by computer scientists to formalize the Boolean properties of the behaviour of biochemical reaction systems (Eker et al. 2002; Chabrier and Fages 2003) and gene influence systems (Bernot et al. 2004; Batt et al. 2005). In this approach, it is possible to evaluate qualitatively, at a high level of abstraction, what may or must happen in interaction networks of large size (e.g. of one thousand reactions and species), and also to compute the initial conditions that exhibit a particular behaviours. This can be achieved by using the powerful symbolic model-checking tools designed over the last decades for circuit and program verification (Clarke et al. 1999; Cimatti et al. 2002) using SAT solvers.

The *Computation Tree Logic* CTL* (Clarke et al. 1999) is an extension of classical logic which allows reasoning on an infinite tree of Boolean state transitions from an initial state. It uses modal operators about branches (non-deterministic choices) and time (state transitions) to qualify where and when a proposition is true. Two path quantifiers A and E are thus introduced to handle non-determinism: $A\phi$ meaning that ϕ is true on all paths, and $E\phi$ that it is true on at least one path. Several time operators are introduced, $X\phi$ means that ϕ is true at the next state, $G\phi$ (globally) that ϕ is true in all future states, $F\phi$ (finally) that ϕ is true in some future state, $\phi\mathbf{U}\psi$ (until) that ϕ is always true before ψ becomes true, and $\phi\mathbf{R}\psi$ (release) that ψ is either globally true or always true up to the first occurrence of ψ included. Table 1 defines the truth value of a formula in a Kripke structure where the states are defined by Boolean variables. In this logic, $F\phi$ is equivalent to $\mathit{true}\mathbf{U}\phi$, $G\phi$ to $\phi\mathbf{R}\mathit{false}$, and we have the following duality properties: $\neg\mathbf{X}\phi = \mathbf{X}\neg\phi$, $\neg\mathbf{E}\phi = \mathbf{A}\neg\phi$, $\neg\mathbf{F}\phi = \mathbf{G}\neg\phi$, $\neg(\phi\mathbf{U}\psi) = \neg\phi\mathbf{R}\neg\psi$.

The LTL fragment of CTL* contains no path quantifier. An LTL formula is true if it is true on all paths. The CTL fragment of CTL* enforces that each temporal operator is preceded by a path operator, and each path operator is immediately followed by a temporal operator. In the context of computational Systems Biology, the following abbreviations for CTL formulae are particularly useful to analyze Boolean attractors (Chabrier and Fages 2003; Traynard et al. 2016a):

- `reachable(P)` stands for $\mathbf{EF}(P)$;
- `steady(P)` stands for $\mathbf{EG}(P)$;

Table 1 Inductive definition of the truth value of a CTL* formula in a given state s or path π, for a Kripke structure K

$s \models \alpha$	if α is a propositional formula true in the state s
$s \models E\phi$	if there exists a path π starting from s s.t. $\pi \models \phi$
$s \models A\phi$	if for all paths π starting from s, $\pi \models \phi$
$\pi \models \neg\phi$	if $\pi \not\models \phi$,
$\pi \models \phi \wedge \psi$	if $\pi \models \phi$ and $\pi \models \psi$
$\pi \models \phi \vee \psi$	if $\pi \models \phi$ or $\pi \models \psi$
$\pi \models \phi \Rightarrow \psi$	if $\pi \models \neg\phi$ or $\pi \models \psi$
$\pi \models \phi$	if $s \models \phi$ where s is the first state of π
$\pi \models \mathbf{X}\phi$	if $\pi^1 \models \phi$
$\pi \models \mathbf{F}\phi$	if $\exists k \geq 0$ s.t. $\pi^k \models \phi$
$\pi \models \mathbf{G}\phi$	if $\forall k \geq 0$, $\pi^k \models \phi$
$\pi \models \phi \mathbf{U} \psi$	if $\exists k \geq 0$ s.t. $\pi^k \models \psi$ and $\pi^j \models \phi \; \forall j \; 0 \leq j < k$
$\pi \models \phi \mathbf{R} \psi$	if $\forall k \geq 0 \; \pi^k \models \psi$ or $\exists j < k \; \pi^j \models \phi$

- `stable(P)` stands for $\mathbf{AG}(P)$;
- `checkpoint(Q,P)` stands for $\neg E(\neg Q \mathbf{U} P)$;
- `oscil(P)` stands for $\mathbf{AG}((\mathbf{EF}\ P) \wedge (\mathbf{EF}\ \neg P))$.

It is worth noting that that notion of checkpoint here is correlational but not necessarily causal. The last abbreviation is actually a necessary but not sufficient condition for oscillations. The correct formula for oscillations is indeed the CTL* formula $\mathbf{EG}(\mathbf{F}P \wedge \mathbf{F}\neg P)$ which cannot be expressed in CTL. The formula `reachable(stable(P))` which is not expressible in LTL, expresses that the state denoted by formula P is a reachable stable state. In Example 2, these formulae are used as patterns to enumerate the interesting properties of the Boolean semantics of the prey-predator system.

4.2 First-Order Quantitative Temporal Logics

Generalizing temporal logic techniques to quantitative models can be done in two ways: either by discretizing the different regimes of the dynamics in piece-wise linear or affine models (Batt et al. 2004, 2010; de Jong et al. 2004), or by taking a first-order version of temporal logic with constraints on concentrations, as query language for the numerical traces (Antoniotti et al. 2003; Fages and Rizk 2008; Donzé and Maler 2010). The first approach brings us back to symbolic propositional methods to analyze quantitative models (Batt et al. 2012). In this section, we present the second approach.

The idea is to lift it to a first-order setting with numerical (linear) constraints over the reals, in order to express threshold and timing constraints and more complex

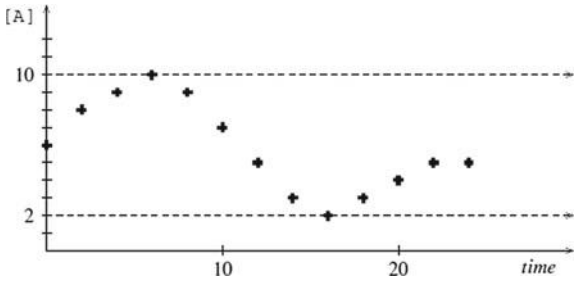


Fig. 6 Numerical trace depicting the time evolution of a protein concentration

Table 2 Grammar of FO-LTL($\mathbb{R}_{\text{lin}}$) formulae where c denotes linear constraints over molecular concentrations, free variables and the time variable

$\phi ::= c \mid \phi \Rightarrow \psi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathbf{X}\phi \mid \mathbf{F}\phi \mid \mathbf{G}\phi \mid \phi \mathbf{U} \phi \mid \phi \mathbf{R} \phi$

constraints on the concentrations of the molecular compounds. For instance, the reachability of a threshold concentration for a molecule A can be expressed with the formula $\mathbf{F}(A > v)$ for some value or free variable v. Such formulae can then be interpreted on a finite numerical trace (extended with a loop on the last state) obtained either from a biological experiment, or from the numerical simulation of an ODE model, giving the concentrations of the molecules at discrete time points, e.g. Fig. 6.

This is possible in the First-Order Linear Time Logic with linear constraints over the reals (FO-LTL($\mathbb{R}_{\text{lin}}$)) and in different variants like Signal Temporal Logic (Donzé and Maler 2010). Table 2 summarizes the grammar of FO-LTL($\mathbb{R}_{\text{lin}}$) formulae. Timing constraints can be expressed with the time variable and free variables to relate the time of differents events. For instance, the formula $\mathbf{G}(Time \leq t_1 \Rightarrow [A] < 1 \wedge Time \geq t_2 \Rightarrow [A] > 10) \wedge (t_2 - t_1 < 60)$ expresses that the concentration of molecule A is always less than 1 up to some time t_1, always greater than 10 after time t_2, and the switching time between t_1 and t_2 is less than 60 units of time. A local maximum for molecule concentration A can be defined with the formula $\mathbf{F}(A \leq x \wedge \mathbf{X}(A = x \wedge \mathbf{X}A \leq x))$. This formula can be used to define oscillation properties, with period constraints defined as time separation constraints between the local maxima of the molecule, as well as phase constraints between different molecules (Fages and Traynard 2014).

The *validity domain* $\mathscr{D}_{(s_0,\ldots,s_n),\phi}$ of the free variables of an FO-LTL($\mathbb{R}_{\text{lin}}$) formula ϕ on a finite trace $(s_0, \ldots, s_n)$, can be computed by finite unions and intersections of polyhedra, by a simple extension of the model-checking algorithm to a constraint solving algorithm (Fages and Rizk 2008, 2009), as follows:

- $\mathscr{D}_{(s_0,\ldots,s_n),\phi} = \mathscr{D}_{s_0,\phi}$,
- $\mathscr{D}_{s_i,c(\boldsymbol{x})} = \{\boldsymbol{v} \in \mathbb{R}^k \mid s_i \models c[\boldsymbol{v}/\boldsymbol{x}]\}$ for a constraint $c(\boldsymbol{x})$,
- $\mathscr{D}_{s_i,\phi\wedge\psi} = \mathscr{D}_{s_i,\phi} \cap \mathscr{D}_{s_i,\psi}$,
- $\mathscr{D}_{s_i,\phi\vee\psi} = \mathscr{D}_{s_i,\phi} \cup \mathscr{D}_{s_i,\psi}$,
- $\mathscr{D}_{s_i,\mathbf{X}\phi} = \mathscr{D}_{s_{i+1},\phi}$,
- $\mathscr{D}_{s_i,\mathbf{F}\phi} = \bigcup_{j=i}^{n} \mathscr{D}_{s_j,\phi}$,

- $\mathscr{D}_{s_i,\mathbf{G}\phi} = \bigcap_{j=i}^{n} \mathscr{D}_{s_j,\phi}$,
- $\mathscr{D}_{s_i,\phi\mathbf{U}\psi} = \bigcup_{j=i}^{n} (\mathscr{D}_{s_j,\psi} \cap \bigcap_{k=i}^{j-1} \mathscr{D}_{s_k,\phi})$.

For instance, on the numerical trace of Fig. 6, the validity domain, depicted in Fig. 7, of the formula $\mathbf{F}(A \geq y_1 \wedge \mathbf{F}(A \leq y_2))$, where y_1 and y_2 are free variables, is $y_1 \leq 10 \wedge y_2 \geq 2$. This can be used for analyzing experimental traces, and extracting logical formulae from data time series.

However, for some important applications such as parameter search, sensitivity and robustness measures, presented in Sect. 5.2 the classical true/false valuation of a logical formula is not well suited (see also chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1). State-of-the-art continuous optimization algorithms such as evolutionary algorithms require a fitness function to measure progress towards satisfiability, i.e. they require to valuate TL formulae with a continuous satisfaction degree in the interval [0, 1].

A method based on variable abstraction is described in Rizk et al. (2009, 2011) for computing the continuous satisfaction degree of an FO-LTL($\mathbb{R}_{\text{lin}}$) formula over a numerical trace. A closed formula, for instance

$$\phi_2 = \mathbf{F}(A \geq 7 \wedge \mathbf{F}(A \leq 0)),$$

is first abstracted in a formula with free variables by replacing constants with free variables, i.e.

$$\phi = \mathbf{F}(A \geq y_1 \wedge \mathbf{F}(A \leq y_2))$$

with the objective values 7 for y_1 and 0 for y_2. Then, the validity domain $\mathscr{D}_{T,\phi}$ of the formula ϕ on a trace T makes it possible to define the *violation degree* $vd(T, \phi, o)$ of the formula on T with objective o, simply as the distance between the validity domain and the objective point o, e.g. 2 in Fig. 7. A *continuous satisfaction degree* in the interval [0, 1] can then be defined by normalization as the inverse of the violation degree d plus one, i.e. 1/3 in Fig. 7:

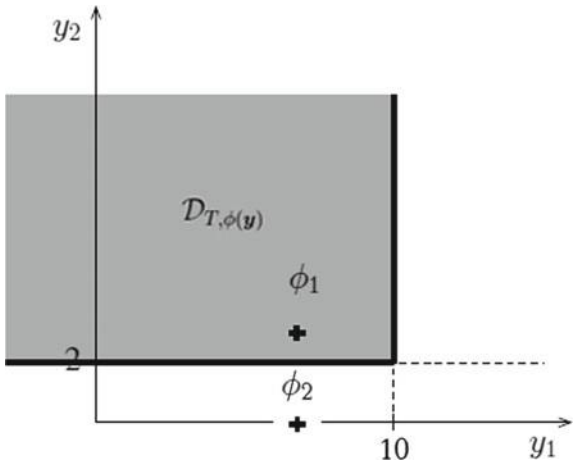


Fig. 7 Validity domain of the formula $\mathbf{F}(A \geq y_1 \wedge \mathbf{F}(A \leq y_2))$ on the trace of Fig. 6. The two points correspond to the formulae $\phi_1 = \mathbf{F}(A \geq 7 \wedge \mathbf{F}(A \leq 3))$ (true) and $\phi_2 = \mathbf{F}(A \geq 7 \wedge \mathbf{F}(A \leq 0))$ (false) respectively

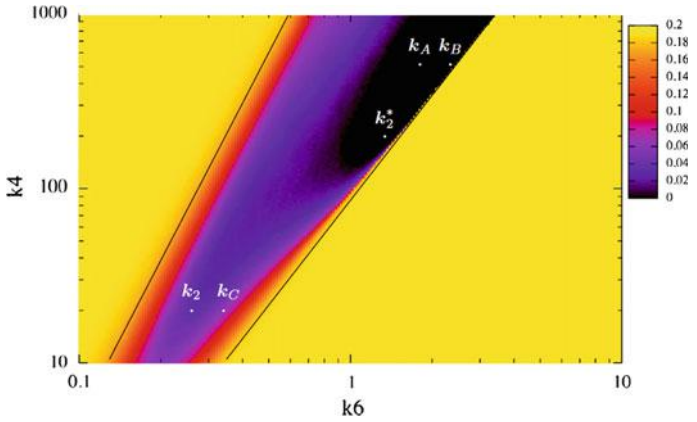


Fig. 8 Landscape of the continuous satisfaction degree of an oscillation property with amplitude constraint, on a color scale from yellow to black, as a function of two parameters in a quantitative model of the yeast cell cycle from Tyson (1991). The parameter sets $\boldsymbol{k}_A$, $\boldsymbol{k}_B$ and $\boldsymbol{k}_2^*$ satisfy the specification (Rizk et al. 2011). The parameter sets $\boldsymbol{k}_c$ and $\boldsymbol{k}_2$ violate the amplitude constraint. The non-yellow zone where there are oscillations is equivalently delimited by the bifurcation diagram considered in Tyson (1991)

$$sd(T, \phi, o) = \frac{1}{1 + vd(T, \phi, o)}$$

In a model of the yeast cell cycle by Tyson (1991), a FO-LTL($Rlin$) formula of oscillation with amplitude constraint produces the landscape of continuous satisfaction degree depicted in Fig. 8 obtained by varying two parameters of the model. Such a landscape is compatible with bifurcation diagrams but is not limited in dimension and can be used for robustness measures and parameter search as shown in Sects. 5.2 and 5.3.

5 Automated Reasoning on Model Dynamics

5.1 Symbolic Model-Checking of Biochemical Circuits

Regulatory, signalling and metabolic networks are very complex mechanisms which are far from being understood on a global scale. Data on the rate functions of the individual reactions are also rare and unreliable, making the building of quantitative models particularly challenging in many cases. In those situations, qualitative analyses can however be conducted in the Boolean semantics of the reactions, using the powerful model-checking tools developed for circuit and program verification (Clarke et al. 1999) in the last decades.

Figure 9 reproduces Kohn's map of the mammalian cell cycle (Kohn 1999) using some graphical conventions introduced by K. Kohn to represent the different types

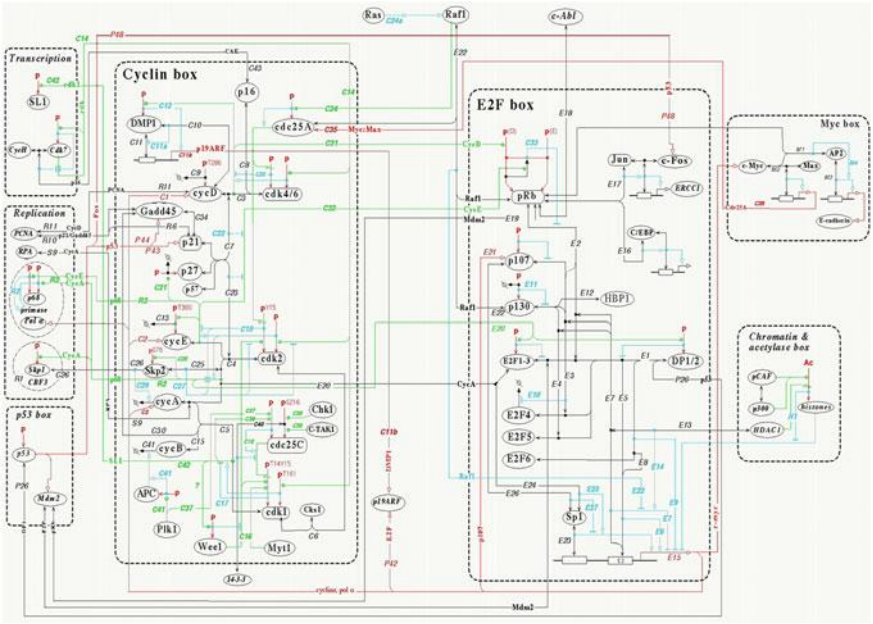


Fig. 9 Kohn's map of the mammalian cell cycle control (Kohn 1999)

of interactions (complexation, binding, phosphorylations, modifications, synthesis, etc.). This map has been transcribed in a reaction model of 732 reaction rules over 165 proteins and genes, and 532 variables taking into account the different forms of the molecular species (Chabrier-Rivier et al. 2004). The astronomical number of Boolean states in this system, 2^{532}, prevents the explicit representation of the state graph, however, a set of states in this space can be represented *symbolically* by a Boolean formula over 532 variables, and the transition relation by a Boolean formula over twice that number of variables. For instance the formula *false* represents the empty set, *true* the universe of all states, x the set of 2^{531} states where x is present, etc. The results reported in Chabrier-Rivier et al. (2004) showed the performance of the state-of-the-art symbolic model checker NuSMV (Cimatti et al. 2002) using the representation of Boolean formulae by ordered binary decision diagrams (OBDD), on this non standard transition system from biology. The compilation of the whole 732 reactions into Boolean formulae took 29 s, and simple reachability and oscillations properties could be checked in a few seconds. Furthermore in this example, the negative answer to the query concerning the oscillation of cyclin B revealed the omission of the synthesis of cyclin B in the map.

A symbolic model-checker can also compute the set of initial states, represented by a *boolean constraint*, for which a formula is true. This may suggest biological experiments to verify a CTL property predicted by the model, in particular conditions on the real biological object (Bernot et al. 2004). For instance, the checkpoints proved

in a model of the cell cycle, or of a signalling network, provide possible drug targets to block the cell cycle or a signalling cascade.

5.2 Parameter Sensitivity and Robustness Computation

In Kitano (2007), Kitano gives a general definition of the *robustness* of a property ϕ in a system with respect to a set P of perturbations given with their probability distribution, as the mean functionality of the system with respect to ϕ under the perturbations., In the FO-LTL($\mathbb{R}_{\text{lin}}$) Temporal Logic framework, this definition instanciates straightforwardly to a computable notion of robustness of a property of a system, simply by taking the continuous satisfaction degree as functionality measure (Rizk et al. 2009), i.e.

$$\mathscr{R}_{S,\phi,P} = \int_{p \in P} prob(p)\ sd(T_p, \phi)\ dp.$$

In a model, this mathematical definition of robustness can be evaluated by (i) sampling the perturbations according to their distribution, (ii) measuring the satisfaction degree of the property for each simulation of the perturbed model, and (iii) returning the average satisfaction degree.

This methodology has been used in Batt et al. (2007) to design a robust switch satisfying some timing constraints implemented in vivo by synthetic biology means with an artificial cascade of gene inhibitions. Moreover, continuous parameter *sensitivity indices* computed in this approach determined the most important parameters for improving the robustness of the design with respect to the timing constraints, that unexpectedly appeared to be the degradation rate parameters.

On the quantitative model of the yeast cell cycle (Tyson 1991) and the oscillation with amplitude constraint depicted in Fig. 8, the estimated degree of robustness for parameters $\boldsymbol{k}_A$, $\boldsymbol{k}_B$ and $\boldsymbol{k}_C$ are respectively 0.991, 0.917 and 0.932. This is consistent with the location of points $\boldsymbol{k}_A$, $\boldsymbol{k}_B$ and $\boldsymbol{k}_C$. Perturbations around point $\boldsymbol{k}_A$ have high probabilities of staying in the region satisfying the specification whereas perturbations around point $\boldsymbol{k}_B$ have high probabilities of moving the system to the region with no oscillation. $\boldsymbol{k}_C$ is more robust than $\boldsymbol{k}_B$ even though, as opposed to $\boldsymbol{k}_B$, its violation degree is non null. This is explained by the abrupt transition between oscillating and non oscillating regions near $\boldsymbol{k}_B$ compared to the smoother transition near $\boldsymbol{k}_C$.

5.3 Parameter Search with Temporal Logic Constraints

Probably the most central difficulty in quantitative systems biology, is that the kinetic parameter values of biochemical reactions are usually unknown, but are mandatory for building quantitative models. They must be estimated from the observation of the

system under various conditions: gene knock-outs, addition of inhibitors, control of the milieu, injection of drugs, etc.

This problem amounts to solve the inverse problem of finding the parameter values of a parametric model (ODE model or even harder CTMC model) in order to reproduce the experimental curves, or more precisely, the relevant properties of the experimental curves which are noisy. The formalization of those properties in quantitative temporal logic is particularly useful in biology where experimental data are imprecise and uncertain, with irregular oscillation periods and phases, and important cell-to-cell variability.

The continuous satisfaction degree of FO-LTL($\mathbb{R}_{\text{lin}}$) formulae provide the necessary objective or fitness function to apply black box optimization algorithms with the all bunch of meta-heuristics (Sun et al. 2011) (see chapter "Meta-heuristics and Artificial Intelligence" of Volume 2) such as Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Neural Networks and portfolio algorithms for parameter estimation (Banga 2008). Of particular relevance in this context, is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) of Hansen and Ostermeier (2001) which enjoy all desirable invariance properties with respect to scaling and symmetries. CMA-ES can be used with the satisfaction degree of an FO-LTL($\mathbb{R}_{\text{lin}}$) specification as fitness function, for searching kinetic parameter values, initial concentrations or control parameters (Rizk et al. 2011).

On the quantitative model of the cell cycle of Tyson (1991), Fig. 8 depicts the landscape of the satisfaction degree of an oscillation property with amplitude constraint, as a function of two parameters of the model. This landscape is iteratively sampled by CMA-ES meta-heuristics to find a path towards satisfaction, and optimize the model parameter values, for instance going from $\boldsymbol{k}_2$ to $\boldsymbol{k}_2^*$ in a few steps.

This strategy for optimizing parameters with respect to an FO-LTL($\mathbb{R}_{\text{lin}}$) specification makes it possible to solve a wide variety of problems in computational systems biology, for fitting models to experimental data in high dimension, up to 100 parameters. This methodology has been used in Heitzler et al. (2012) to elucidate the complex quantitative dynamics of GPCR cell signalling networks, by revisiting the structure of the known reactions following the failure of CMA-ES to fit the FO-LTL($\mathbb{R}_{\text{lin}}$) properties of some mutants, making new biological hypotheses based on sensitivity analyses, and verifying them by new biological experiments.

In Traynard et al. (2016b), it served to build a quantitative model of the cell cycle and the circadian clock explaining unexpected observations in embryonic fibroblasts, and make the prediction of an up-regulation of clock-gene Reverb-α during mitosis in those cells.

The same strategy for parameter optimization can also be used to compute control parameters in order to achieve a desired behaviour at the single cell or cell population levels. This has been shown for long-term model-based real-time control of gene expression in yeast cells using a microfluidic device in Uhlendorf et al. (2012), and in the context of cancer chronotherapies, at the whole body scale, to couple models of the cell cycle, circadian clock, DNA repair system and drug metabolism, to optimize anti-cancer drug administration laws in Ballesta et al. (2011), De Maria et al. (2011).

5.4 *Turing Completeness and Automated Synthesis of Reaction Networks*

"What I cannot create, I do not understand", Richard Feynman.

It was known since a long time that reaction systems under the differential semantics are universal circuits, in the sense that any computable real number can be computed by such a circuit (Helmfelt et al. 1991; Magnasco 1997; Cook et al. 2009). However it is only in Fages et al. (2017), that we showed that continuous chemical reaction networks are Turing complete, in the sense that for any computable real function (see chapters "Heuristically Ordered Search in State Graphs" and "Meta-heuristics and Artificial Intelligence" of Volume 2), there exists a finite reaction system on a finite set of molecular species that computes the result of the function on any arguments, all given by the concentrations (in arbitrary but finite precision) of a fixed set of input and output molecular species.

Interestingly, one can derive from the proof of this result a general method for automatically synthesizing a complete reaction system (i.e. its structure and rate constants for mass action law kinetics) for computing any computable real valued function specified as the solution of a polynomial ODE initial value problem (PIVP). This method applies to functions over the reals, either input/output functions for the specification of dose-response diagrams (e.g. for signalling circuits), or functions of time for the specification of transient behaviors (e.g. for cycling circuits). This approach to analog biochemical computation is currently under investigation in an ambitious project in biocomputing, on the one hand in synthetic biology for the automated synthesis of useful biochemical networks for instance for making diagnosis devices (Courbet et al. 2018), and on the other hand, for the understanding of natural networks with their comparison to synthesized networks having the same or similar input/output function, for instance in the case of Example 3 with the analog-digital converter sigmoid input/output function of the MAPK network (Fages et al. 2017).

6 Learning Mechanistic Models from Temporal Data

Biological modelling is still an art which is currently limited in its applications by the number of available modellers. Automating the process of model building is thus a very desirable goal to attack new applications, develop patient-tailored therapeutics, and also design experiments that can now be largely automated at both the single cell and cell population levels, with a gain in both the quantification and the reliability of the observations.

Machine learning is revolutionarizing the statistical methods in biological data analytics, data classification and clustering, and for making predictions from static measurements (see chapters "Statistical Computational Learning" and "Reinforcement Learning" of Volume 1). However, learning dynamical models from temporal data is more challenging, since it addresses hard issues for modeling time and causality (Pearl 2009) (see chapter "A Glance at Causality Theories for Artificial

Intelligence" of Volume 1). There has been early work on the use of machine learning techniques, such as inductive logic programming (Muggleton 1995) or heuristic breath-first tree search (Valdès-Pérè 1995) combined with active learning in the vision of the "robot scientist", to infer gene functions (Bryant et al. 2001), metabolic pathway descriptions (Angelopoulos and Muggleton 2002a, b) or gene influence systems (Bernot et al. 2004), or to revise a reaction model with respect to CTL properties (Calzone et al. 2006a). A recent survey on probabilistic programming (Gordon et al. 2014) highlighted the difficulties associated with modelling time, and concluded that existing frameworks are not sufficient in their treatment of dynamical systems. Since a few years, progress in those fields can be measured on public benchmarks of the "Dream Challenge" competition (Meyer et al. 2014). In this fastly moving field, we focus here on a general purpose framework for learning the structure of a mechanistic model.

6.1 Probably Approximatively Correct Learning

In his seminal paper on a theory of the learnable (Valiant 1984), Valiant questioned what can be learned from a computational viewpoint, and introduced the concept of probably approximate correct (PAC) learning, together with a general-purpose polynomial-time learning protocol. Beyond the learning algorithms that one can derive with this methodology, Valiant's theory of the learnable has profound implications on the nature of biological and cognitive processes, of collective and individual behaviors, and on the study of their evolution (Valiant 2013). In this section, we simply recall the general theory of PAC learning, and illustrate it with the learning of Boolean gene networks from gene expression data.

The learning protocol for Boolean functions considers a finite set of Boolean variables $x_1, \ldots, x_s$. A vector is an assignment of the s variables to $\{0, 1, *\}$ where the symbol $*$ denotes the undetermined. A vector is total if it contains no undetermined value. A Boolean function $F : \{0, 1\}^s \to \{0, 1\}$ assigns a Boolean value to each total vector. A Boolean concept $C : \{0, 1, *\}^s \to \{0, 1\}$ assigns similarly a Boolean value to non total vectors, with the following independence constraint: for any vector v and any total extension w of v (i.e. where the undetermined values in v are replaced by 0 or 1) we have $C(v) = C(w)$.

The PAC learning protocol considers a hidden Boolean function F, a class $\mathscr{M}$ of models to learn, $f(x_1, ..., x_s) \in \{0, 1, *\}$, a set of positive examples, i.e. a set of vectors v for which $F(v) = 1$, and an arbitrary probability distribution D over this set for representing the relative frequency of the positive examples. The restriction to positive examples is for the sake of simplicity. The PAC learning protocol then allows for

- calls for positive examples, i.e. vectors v such that $F(v) = 1$ given with probability $D(v)$,
- calls for oracle on some input v to know the value of $F(v)$

Example 6 This Boolean framework perfectly fits the Boolean semantics of Thomas's gene regulatory networks described in Sect. 2.2.4. Indeed in that formalism, each gene $x_1, \ldots, x_s$ is given with a Boolean function $F_{x_i} : \{0,1\}^s \rightarrow \{0,1\}$ which defines the activation update function of that gene according to the expression vector of the other genes in the different possible states. These Boolean functions are best represented by Boolean concepts in PAC terminology in order to make explicit the independent genes. Then, the problem of building such a Boolean model *à la* Thomas of gene activation is to give for each gene a Boolean transition function that is compatible with the observed temporal data of gene activation. It is worth noticing that the PAC learning protocol makes it possible to learn such Boolean models of gene regulation not only from a given finite set of positive gene activation observations, but also from new biological experiments designed by the PAC learning algorithm itself through the queries to the oracle.

A class $\mathscr{M}$ of models is learnable in a given learning protocol, if there exists an algorithm $\mathscr{A}$ such that:

- $\mathscr{A}$ runs in polynomial time in s and h, the size of the models to learn,
- For all models f in $\mathscr{M}$, all vector distributions D on which f outputs 1, $\mathscr{A}$ deduces with probability $\geq 1 - h^{-1}$ a model g in $\mathscr{M}$ such that
  - $g(v) = 1$ implies $f(v) = 1$ (no false positives)
  - $\sum_{v \ s.t. \ f(v)=1 \ g(v)\neg=1} D(v) < h^{-1}$ (low probability of false negatives)

Interestingly, Valiant showed the learnability of some important classes of functions in this framework, in particular for Boolean formulae in conjunctive normal forms with at most k literals (k-CNF) and for monotone (i.e. negation free) Boolean formulae in disjunctive normal form (DNF). The computational complexity of the PAC learning algorithms for these classes of functions is expressed in terms of the function $L(h, S)$ defined as the smallest integer i such that in i independent Bernoulli trials, each with probability at least $h - 1$ of success, the probability of having fewer than S successes is less than $h - 1$. Interestingly, this function is quasi-linear in h and S, i.e. for all integers $S \geq 1$ and reals $h > 1$, $L(h, S) \leq 2h(S + log_e h)$.

First, for any k, the class of k-CNF formulae is learnable with an algorithm that uses $L(h, (2s)^{k+1})$ examples and no oracle (Valiant 1984). The simple algorithm used in the proof proceeds as follows

1. initialize g to the conjunction of all possible $(2s)^k$ disjunctions of at most k literals,
2. do $L(h, (2s)^{k+1})$ times
   a. v :=Sample()
   b. delete all the disjunctions in g that do not contain a literal true in v
3. output g

Example 7 k-CNF formulae can be used to represent Thomas's gene regulatory network functions with some reasonable restrictions on their connectivity. In this case, the algorithm is repeated s times for learning each gene activation function

from temporal series of gene activation temporal series (Carcano et al. 2017). The initialization of the learned function g to the most constrained conjunction of all possible disjunctions leads to the learning of a minimal generalization of the positive examples in this representation. Interestingly, the bound on the errors provided by PAC learning according to the distributions of the transition samples is quite acurate. Carcano et al. (2017) shows that many short time series from a uniform distribution of initial states are more informative than long time series from few initial states, since they allow to distinguish causal from purely correlational relationships between activations and to recover hidden influence models in absence of noise in the data.

Second, the class of monotone DNF formulae is also learnable with an algorithm that uses $L(h, d)$ examples and ds calls to the oracle, where d is the largest number of prime implicants in an equivalent prime DNF formula (Valiant 1984). It is worth noting that the calls to the oracle make of this algorithm an active learning method, and in the context of biological modeling, an abstract method for designing experiments. The algorithm is the following:

1. initialize g with constant zero,
2. do $L(h, d)$ times
   a. $v := Sample()$
   b. If g is not implied by v then for $i := 1$ to s do
      i. if x_i is determined in v and ORACLE($v[x_i \leftarrow *]$)=1 then $v := v[x_i \leftarrow *]$
      ii. $g := g + m$ where m is the product of all litterals q implied by v
3. output g

Example 8 The (positive) Boolean semantics of biochemical influence systems described in Sect. 2.2.2 can be directly represented by the disjunction of the (positive) enabling conditions of each, either positive or negative, influence on a given target, i.e. by a monotone DNF formula for each activation or inhibition of each target. In the Lotka-Volterra influence system of Example 4, the algorithm above is thus expected to actively learn the structure of the influence system (without the stoichiometry of course), from the observation that the prey can disappear only in presence of the predator while the predator can always disappear in presence or absence of the prey.

Example 9 Learning reaction models from observed transitions is more tricky in this approach, since some reactions may change the Boolean value of several reactants or products in one single transition. Therefore, it is not only the activation and inhibition functions of each species which are to be learnt, but the update functions of pairs and triples of species if we restrict to elementary reactions with at most two reactants or products. In this case, the update functions can be represented by monotonic DNF formulae, since the (positive) Boolean semantics of a reaction system does not test the absence. Furthermore, one cannot expect to learn the structure of such a reaction network from the observation of the state transitions from one single initial state. The learning algorithms assumes that the positive examples of the state transition relation be distributed among the whole vector space. For instance, in the MAPK

Example 3, in addition to the initial state of the wild type organism where all the kinases and phosphatases are present, it is necessary to consider some mutated organisms, in which some kinases or phosphatases are absent, in order to gain information on the precise conditions of activation and deactivation of the different forms of the kinases. This strategy is essentially similar to what the biologists do to elucidate the structure of biological processes in a qualitative manner.

6.2 Answer Set Programming

Logic Programming, and especially *Answer Set Programming* (ASP, see chapter "Logic Programming" of Volume 2), provide particularly efficient tools such as CLASP (Gebser et al. 2007) to develop learning algorithms for Boolean models. They were applied in Gebser et al. (2008) to detect inconsistencies in large biological networks, and have been subsequentially applied to the inference of gene networks from gene expression data.

Interestingly, ASP has also been combined with CTL model-checking in Ostrowski et al. (2016) to learn mammalian signalling networks from time series data, and identify erroneous time-points in the data, a possibility not considered in the previous presentation of PAC learning.

6.3 Budgeted Learning

Budgeted learning extends active learning with a notion of cost for the calls to the oracle. The original motivation for the budgeted learning protocol came from medical applications in which the outcome of a treatment, drug trial, or control group is known, and the results of running medical tests are each available for a price (Deng et al. 2013). In this context, multi-armed bandit methods (Deng et al. 2007) provide the best strategies. In Llamosi et al. (2014), a bandit-based active learning algorithm is proposed for experiment design in dynamical system identification. These approaches are directly relevant to biological experiment design and modelling. They should gain importance in the forthcoming years with the increasing automation of biological experiments.

7 Conclusion

"The varied titles of Turing's published work disguise its unity of purpose. The central problem with which he started, and to which he constantly returned, is the extent and the limitations of mechanistic explanations of nature.", Max Newman.

Computer Science is born with the perspective of Artificial Intelligence, i.e. creating machines that reproduce human intelligence (Turing 1950). The application of Computer Science concepts and tools to the analysis of Biological Systems, beyond solving Bioinformatics combinatorial problems with AI techniques, provides a new perspective for Computation Science: Biology, i.e. understanding the living, how cells sense their environment and compute their decision, and beyond discribing natural biochemical interaction networks (Barabási 2016), understand their functions, evolution history and evolution capabilities (Valiant 2013).

Though one lesson of Computer Science was that analog computation does not scale up to large systems, while digital computation does, the biological perspective provides a new impetus to analog computation and mixed analog/digital parallel computation. The concept of biochemical computation can now be experimented, either in synthetic biology, through the modification and reprogramming of living cells (Nielsen et al. 2016; Courbet et al. 2015), or in synthetic biochemistry, through the creation and programming of non-living microfluidic vesicles (Courbet et al. 2018). The social behaviors of cells and tissue homeostasis add one more dimension to the problem of designing useful computational devices at the microscale. These research fields provide numerous challenges for AI, both conceptual and algorithmic.

References

Ahmad J, Bernot G, Comet JP, Lime D, Roux O (2006) Hybrid modelling and dynamical analysis of gene regulatory networks with delays. ComplexUs 3:231–251

Alur R, Belta C, Ivanicic F, Kumar V, Mintz M, Pappas GJ, Rubin H, Schug J (2001) Hybrid modeling and simulation of biomolecular networks. In: Proceedings of the 4th international workshop on hybrid systems: computation and control, HSCC'01, Springer-Verlag, Rome, Italy. Lecture notes in computer science, vol 2034, pp 19–32

Angelopoulos N, Muggleton SH (2002a) Machine learning metabolic pathway descriptions using a probabilistic relational representation. Electron Trans Artif Intell 7(9); also in Proceedings of Machine Intelligence 19

Angelopoulos N, Muggleton SH (2002b) SLPs for probabilistic pathways: modeling and parameter estimation. Technical Report TR 2002/12, Department of Computing, Imperial College, London, UK

Antoniotti M, Policriti A, Ugel N, Mishra B (2003) Model building and model checking for biochemical processes. Cell Biochem Biophys 38:271–286

Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G (2000) Gene ontology: tool for the unification of biology. Nat Genet 25:25–29

Ballesta A, Dulong S, Abbara C, Cohen B, Okyar A, Clairambault J, Levi F (2011) A combined experimental and mathematical approach for molecular-based optimization of irinotecan circadian delivery. PLOS Comput Biol 7(9). https://doi.org/10.1371/journal.pcbi.1002143

Banga JR (2008) Optimization in computational systems biology. BMC Syst Biol 2: https://doi.org/10.1186/1752-0509-2-47

Barabási AL (2016) Network science. Cambridge University Press, Cambridge

Batt G, Bergamini D, de Jong H, Garavel H, Mateescu R (2004) Model checkinggenetic regulatory networks using GNA and CADP. In: Proceedings of the 11th international SPIN workshop on model checking of software SPIN'2004, Barcelona, Spain

Batt G, Ropers D, de Jong H, Geiselmann J, Mateescu R, Page M, Schneider D (2005) Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in Escherichia coli. Bioinformatics 21(Suppl. 1):i19–i28

Batt G, Yordanov B, Weiss R, Belta C (2007) Robustness analysis and tuning of synthetic gene networks. Bioinformatics 23(18):2415–2422

Batt G, Page M, Cantone I, Goessler G, Monteiro P, de Jong H (2010) Efficient parameter search for qualitative models of regulatory networks using symbolic model checking. Bioinformatics 26(18):i603–i610

Batt G, Besson B, Ciron P, de Jong H, Dumas E, Geiselmann J, Monte R, Monteiro P, Page M, Rechenmann F, Ropers D (2012) Genetic network analyzer: a tool for the qualitative modeling and simulation of bacterial regulatory networks. Bacterial molecular networks. Springer, Berlin, pp 439–462

Berestovsky N, Zhou W, Nagrath D, Nakhleh L (2013) Modeling integrated cellular machinery using hybrid Petri-Boolean networks. PLoS Comput Biol 9(11):1003,306. https://doi.org/10.1371/journal.pcbi.1003306

Bernot G, Comet JP, Richard A, Guespin J (2004) A fruitful application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. J Theor Biol 229(3):339–347

Bryant CH, Muggleton SH, Oliver SG, Kell DB, Reiser PGK, King RD (2001) Combining inductive logic programming, active learning and robotics to discover the function of genes. Electron Trans Artif Intell 6(12)

Calzone L, Chabrier-Rivier N, Fages F, Soliman S (2006a) Machine learning biochemical networks from temporal logic properties. In: Plotkin G (ed) Transactions on computational systems biology VI. Lecture notes in bioinformatics, vol 4220. Springer, Berlin, pp 68–94. https://doi.org/10.1007/11880646_4; cMSB'05 Special Issue

Calzone L, Fages F, Soliman S (2006b) BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. Bioinformatics 22(14):1805–1807. https://doi.org/10.1093/bioinformatics/btl172

Carcano A, Fages F, Soliman S (2017) Probably approximately correct learning of regulatory networks from time-series data. In: CMSB'17: Proceedings of the fiveteen international conference on computational methods in systems biology, vol 10545, pp 74–90. DOI https://doi.org/10.1007/978-3-319-67471-1_5

Chabrier N, Fages F (2003) Symbolic model checking of biochemical networks. In: Priami C (ed) CMSB'03: proceedings of the first workshop on computational methods in systems biology, Springer-Verlag, Rovereto, Italy. Lecture notes in computer science, vol 2602, pp 149–162

Chabrier-Rivier N, Chiaverini M, Danos V, Fages F, Schächter V (2004) Modeling and querying biochemical interaction networks. Theor Comput Sci 325(1):25–44

Chazelle B (2012) Natural algorithms and influence systems. Commun ACM 55(12):101–110. https://doi.org/10.1145/2380656.2380679

Chiang HJ, Fages F, Jiang JH, Soliman S (2015) Hybrid simulations of heterogeneous biochemical models in SBML. ACM Trans Model Comput Simul (TOMACS) 25(2):14:1–14:22. https://doi.org/10.1145/2742545

Cimatti A, Clarke E, Enrico Giunchiglia FG, Pistore M, Roveri M, Sebastiani R, Tacchella A (2002) NuSMV 2: an opensource tool for symbolic model checking. In: Proceedings of the international conference on computer-aided verification, CAV'02, Copenhagen, Danmark

Clarke EM, Grumberg O, Peled DA (1999) Model checking. MIT Press, Cambridge

Colom JM, Silva M (1991) Convex geometry and semiflows in P/T nets. a comparative study of algorithms for computation of minimal p-semiflows. In: Rozenberg G (ed) Advances in Petri Nets 1990. Lecture notes in computer science, vol 483. Springer, London, UK, pp 79–112. https://doi.org/10.1007/3-540-53863-1_22

Cook M, Soloveichik D, Winfree E, Bruck J (2009) Programmability of chemical reaction networks. In: Condon A, Harel D, Kok JN, Salomaa A, Winfree E (eds) Algorithmic bioprocesses. Springer, Berlin, pp 543–584. https://doi.org/10.1007/978-3-540-88869-7_27

Cordone R, Ferrarini L, Piroddi L (2005) Enumeration algorithms for minimal siphons in petri nets based on place constraints. IEEE Trans Syst Man Cybern Part A Syst Hum 35(6):844–854

Courbet A, Endy D, Renard E, Molina F, Bonnet J (2015) Detection of pathological biomarkers in human clinical samples via amplifying genetic switches and logic gates. Sci Transl Med 352(6281):289ra83

Courbet A, Amar P, Fages F, Renard E, Molina F (2018) Computer-aided biochemical programming of synthetic microreactors as diagnostic devices. Mol Syst Biol 14(4):10.15252. arXiv:msb.20177845

Cousot P, Cousot R (1977) Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL'77: proceedings of the 6th ACM symposium on principles of programming languages, ACM Press, New York, pp 238–252 (Los Angeles)

De Maria E, Fages F, Rizk A, Soliman S (2011) Design, optimization, and predictions of a coupled model of the cell cycle, circadian clock, DNA repair system, irinotecan metabolism and exposure control under temporal logic constraints. Theor Comput Sci 412(21):2108–2127. https://doi.org/10.1016/j.tcs.2010.10.036

Delzanno G, Podelski A (2001) Constraint-based deductive model checking. STTT 3(3):250–270

Deng K, Bourke C, Scott SD, Sunderman J, Zheng Y (2007) Bandit-based algorithms for budgeted learning. In: ICDM

Deng K, Zheng Y, Bourke C, Scott S, Masciale J (2013) New algorithms for budgeted learning. Mach Learn 90. https://doi.org/10.1007/s10994-012-5299-2

Donzé A, Maler O (2010) Robust satisfaction of temporal logic over real-valued signals. In: FORMATS 2010, Springer-Verlag. Lecture notes in computer science, vol 6246, pp 92–106

Eisenberg M (1991) The kineticist's workbench: Combining symbolic and numerical methods in the simulation of chemical reaction mechanisms. Technical Report 1306, MIT Technical Report

Eker S, Knapp M, Laderoute K, Lincoln P, Meseguer J, Sönmez MK (2002) Pathway logic: symbolic analysis of biological signaling. In: Proceedings of the seventh pacific symposium on biocomputing, pp 400–412

Fages F (1994) Consistency of Clark's completion and existence of stable models. Methods Log Comput Sci 1:51–60

Fages F, Rizk A (2008) On temporal logic constraint solving for the analysis of numerical data time series. Theor Comput Sci 408(1):55–65. https://doi.org/10.1016/j.tcs.2008.07.004

Fages F, Rizk A (2009) From model-checking to temporal logic constraint solving. In: Proceedings of CP'2009, 15th international conference on principles and practice of constraint programming, vol 5732. Lecture notes in computer science. Springer, pp 319–334. https://doi.org/10.1007/978-3-642-04244-7_26

Fages F, Soliman S (2008a) Abstract interpretation and types for systems biology. Theoret Comput Sci 403(1):52–70. https://doi.org/10.1016/j.tcs.2008.04.024

Fages F, Soliman S (2008b) Formal cell biology in BIOCHAM. In: Bernardo M, Degano P, Zavattaro G (eds) 8th international school on formal methods for the design of computer, communication and software systems: computational systems biology SFM'08, Springer-Verlag, Bertinoro, Italy. Lecture notes in computer science, vol 5016, pp 54–80. https://doi.org/10.1007/978-3-540-68894-5_3

Fages F, Soliman S (2008c) Model revision from temporal logic properties in systems biology. In: de Raedt L, Frasconi P, Kersting K, Muggleton S (eds) Probabilistic inductive logic programming. Lecture notes in computer science, vol 4911. Springer, pp 287–304. https://doi.org/10.1007/978-3-540-78652-8_11

Fauré A, Thieffry D (2009) Logical modelling of cell cycle control in eukaryotes: a comparative study. Mol Biosyst 5(12):1569–1581

Fages F, Traynard P (2014) Temporal logic modeling of dynamical behaviors: first-order patterns and solvers. In: del Cerro LF, Inoue K (eds) Logical modeling of biological systems. Wiley, pp 291–323. https://doi.org/10.1002/9781119005223.ch8

Fages F, Gay S, Soliman S (2015) Inferring reaction systems from ordinary differential equations. Theoret Comput Sci 599:64–78. https://doi.org/10.1016/j.tcs.2014.07.032

Fages F, Le Guludec G, Bournez O, Pouly A (2017) Strong turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs (best paper award). In: CMSB'17: Proceedings of the fiveteen international conference on computational methods in systems biology, Springer-Verlag. Lecture notes in computer science, vol 10545, pp 108–127. https://doi.org/10.1007/978-3-319-67471-1_7

Fages F, Martinez T, Rosenblueth D, Soliman S (2018) Influence networks compared with reaction networks: semantics, expressivity and attractors. IEEE/ACM Trans Comput Biol Bioinform. https://doi.org/10.1109/TCBB.2018.2805686

Fauré A, Naldi A, Lopez F, Chaouiya C, Ciliberto A, Thieffry D (2009) Modular logical modelling of the budding yeast cell cycle. Mol Biosyst 5:1787–1796

Feinberg M (1977) Mathematical aspects of mass action kinetics. In: Lapidus L, Amundson NR (eds) Chemical reactor theory: a review. Prentice-Hall, pp 1–78

de Figueiredo LF, Schuster S, Kaleta C, Fell DA (2009) Can sugars be produced from fatty acids? A test case for pathway analysis tools. Bioinformatics 25(1):152–158. https://doi.org/10.1093/bioinformatics/btn621

Funahashi A, Matsuoka Y, Jouraku A, Morohashi M, Kikuchi N, Kitano H (2008) Celldesigner 3.5: a versatile modeling tool for biochemical networks. Proc IEEE 96(8):1254–1265. https://doi.org/10.1109/JPROC.2008.925458

Gay S, Soliman S, Fages F (2010) A graphical method for reducing and relating models in systems biology. Bioinformatics 26(18):i575–i581. https://doi.org/10.1093/bioinformatics/btq388, special issue ECCB'10

Gay S, Fages F, Martinez T, Soliman S, Solnon C (2014) On the subgraph epimorphism problem. Discret Appl Math 162:214–228. https://doi.org/10.1016/j.dam.2013.08.008

Gebser M, Kaufmann B, Neumann A, Schaub T (2007) Clasp: a conflict-driven answer set solver. In: Proceedings of the LPNMR'07. Springer, pp 260–265

Gebser M, Schaub T, Thiele S, Usadel B, Veber P (2008) Detecting inconsistencies in large biological networks with answer set programming. In: de la Banda MG, Pontelli E (eds) ICLP'08, Proceedings of the 24th international conference on logic programming, Springer-Verlag. Lecture notes in computer science, vol 5366, pp 130–144. https://doi.org/10.1007/978-3-540-89982-2_19

Ghosh R, Tomlin C (2001) Lateral inhibition through delta-notch signaling: a piecewise affine hybrid model. In: Springer-Verlag (ed) Proceedings of the 4th international workshop on hybrid systems: computation and control, HSCC'01, Rome, Italy. Lecture Notes in Computer Science, vol 2034, pp 232–246

Gillespie DT (1977) Exact stochastic simulation of coupled chemical reactions. J Phys Chem 81(25):2340–2361

Glass L, Kauffman SA (1973) The logical analysis of continuous, non-linear biochemical control networks. J Theor Biol 39(1):103–129

González AG, Chaouiya C, Thieffry D (2008) Qualitative dynamical modelling of the formation of the anterior-posterior compartment boundary in the drosophila wing imaginal disc. Bioinformatics 24:234–240

Gordon AD, Henzinger TA, Nori AV, Rajamani SK (2014) Probabilistic programming. Proceedings of the on future of software engineering, ACM, New York, NY, USA, FOSE 2014:167–181. https://doi.org/10.1145/2593882.2593900

Grieco L, Calzone L, Bernard-Pierrot I, Radvanyi F, Kahn-Perlès B, Thieffry D (2013) Integrative modelling of the influence of MAPK network on cancer cell fate decision. PLOS Comput Biol 9(10):e1003,286

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195

Heitzler D, Durand G, Gallay N, Rizk A, Ahn S, Kim J, Violin JD, Dupuy L, Gauthier C, Piketty V, Crépieux P, Poupon A, Clément F, Fages F, Lefkowitz RJ, Reiter E (2012) Competing G protein-coupled receptor kinases balance G protein and β-arrestin signaling. Mol Syst Biol 8(590). https://doi.org/10.1038/msb.2012.22

Helmfelt A, Weinberger ED, Ross J (1991) Chemical implementation of neural networks and turing machines. PNAS 88:10,983–10,987

Henzinger TA (1996) The theory of hybrid automata. In: Proceedings of the 11th annual symposium on logic in computer science (LICS). IEEE Computer Society Press, pp 278–292; an extended version appeared in Verification of Digital and Hybrid Systems

Herrgård MJ, Swainston N, Dobson P, Dunn WB, Arga KY et al (2008) A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. Nat Biotechnol 26(10):1155–1160. https://doi.org/10.1038/nbt1492

Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U (2006) Copasi - a complex pathway simulator. Bioinformatics 22(24):3067–3074

Huang CY, Ferrell JE (1996) Ultrasensitivity in the mitogen-activated protein kinase cascade. PNAS 93(19):10,078–10,083

Hucka M et al (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19(4):524–531

Hucka M, Hoops S, Keating SM, Nicolas LN, Sahle S, Wilkinson D (2008) Systems biology markup language (SBML) level 2: structures and facilities for model definitions. Nat Preced. https://doi.org/10.1038/npre.2008.2715.1

Ideker T, Galitski T, Hood L (2001) A new approach to decoding life: systems biology. Annu Rev Genomics Hum Genet 2:343–372

Inoue K (2011) Logic programming for boolean networks. In: Proceedings of the twenty-second international joint conference on artificial intelligence - volume two, AAAI Press, IJCAI'11, pp 924–930. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-160

de Jong H, Gouzé JL, Hernandez C, Page M, Sari T, Geiselmann J (2004) Qualitative simulation of genetic regulatory networks using piecewise-linear models. Bull Math Biol 66(2):301–340

von Kamp A, Schuster S (2006) Metatool 5.0: fast and flexible elementary modes analysis. Bioinformatics 22(15):1930–1931

Kanehisa M, Goto S (2000) KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Res 28(1):27–30

Karr JR, Sanghvi JC, Macklin DN, Gutschow MV, Jacobs JM, Bolival B, Assad-Garcia N, Glass JI, Covert MW (2012) A whole-cell computational model predicts phenotype from genotype. Cell 150(2):389,401

Kitano H (2007) Towards a theory of biological robustness. Mol Syst Biol 3:137

Kleene S (1956) Representation of events in nerve nets and finite automata. Princeton University Press, Princeton, pp 3–41

Kohn KW (1999) Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Mol Biol Cell 10(8):2703–2734

Kurtz TG (1978) Strong approximation theorems for density dependent Markov chains. Stoch Process Appl 6(3):223–240. https://doi.org/10.1016/0304-4149(78)90020-0

Larhlimi A, Bockmayr A (2009) A new constraint-based description of the steady-state flux cone of metabolic networks. Discret Appl Math 157(10):2257–2266. https://doi.org/10.1016/j.dam.2008.06.039; networks in Computational Biology

le Novère N, Bornstein B, Broicher A, Courtot M, Donizelli M, Dharuri H, Li L, Sauro H, Schilstra M, Shapiro B, Snoep JL, Hucka M (2006) BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acid Res 1(34):D689–D691

Llamosi A, Mezine A, d'Alché Buc F, Letort V, Sebag M (2014) Experimental design in dynamical system identification: a bandit-based active learning approach. In: Calders T, Esposito F,

Hüllermeier E, Meo R (eds) Machine learning and knowledge discovery in databases ECML PKKDD'14, Springer-Verlag. Lecture notes in artificial intelligence, vol 8724, pp 306–321

Magnasco MO (1997) Chemical kinetics is turing universal. Phys Rev Lett 78(6):1190–1193

Matsuno H, Doi A, Nagasaki M, Miyano S (2000) Hybrid petri net representation of gene regulatory network. In: Proceedings of the 5th pacific symposium on biocomputing, Stanford, Hawaii, USA, pp 338–349

McCulloch W, Pitts W (1943) A logical calulus of ideas immanent in nervous activity. Bull Math Biophys 5:115–133

Meyer P, Cokelaer T, Chandran D, Kim KH, Loh PR, Tucker G, Lipson M, Berger B, Kreutz C, Raue A, Steiert B, Timmer J, Bilal E, Sauro HM, Stolovitzky G, Saez-Rodriguez J (2014) Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach. BMC Syst Biol 8(1):1–18. https://doi.org/10.1186/1752-0509-8-13

Muggleton SH (1995) Inverse entailment and progol. New Gener Comput 13:245–286

Naldi A, Berenguier D, Fauré A, Lopez F, Thieffry D, Chaouiya C (2009) Logical modelling of regulatory networks with GINsim 2.3. Biosystems 97(2):134–139. https://doi.org/10.1016/j.biosystems.2009.04.008

Naldi A, Carneiro J, Chaouiya C, Thieffry D (2010) Diversity and plasticity of the cell types predicted from regulatory network modelling. PLoS Comput Biol 6(9):e1000,912. https://doi.org/10.1371/journal.pcbi.1000912

Nabli F, Martinez T, Fages F, Soliman S (2016) On enumerating minimal siphons in petri nets using CLP and SAT solvers: theoretical and practical complexity. Constraints 21(2):251–276. https://doi.org/10.1007/s10601-015-9190-1

Neumann JV (1966) Theory of self replicating automata. University of Illinois Press

Nielsen AAK, Der BS, Shin J, Vaidyanathan P, Paralanov V, Strychalski EA, Ross D, Densmore D, Voigt CA (2016) Genetic circuit design automation. Science 352(6281). https://doi.org/10.1126/science.aac7341

Ostrowski M, Paulevé L, Schaub T, Siegel A, Guziolowski C (2016) Boolean network identification from perturbation time series data combining dynamics abstraction and logic programming. Biosystems 149:139–153. https://doi.org/10.1016/j.biosystems.2016.07.009

Pearl J (2009) Causality: models, reasoning and inference, 2nd edn. Cambridge University Press, New York

Peres S, M M, Simon L, (2014) Sat-based metabolics pathways analysis without compilation. In: et al (Eds): CMSB PM. Lecture note in bioinformatics, vol 8859. Springer International Publishing, pp 20–31

Peterson JL (1981) Petri net theory and the modeling of systems. Prentice Hall, New Jersey

Reddy VN, Mavrovouniotis ML, Liebman MN (1993) Petri net representations in metabolic pathways. In: Hunter L, Searls DB, Shavlik JW (eds) Proceedings of the 1st international conference on intelligent systems for molecular biology (ISMB), AAAI Press, pp 328–336

Remy E, Ruet P, Thieffry D (2008) Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework. Adv Appl Math 41(3):335–350. https://doi.org/10.1016/j.aam.2007.11.003

Rezola A, de Figueiredo LF, Brock M, Pey J, Podhorski A, Wittmann C, Schuster S, Bockmayr A, Planes FJ (2011) Exploring metabolic pathways in genome-scale networks via generating flux modes. Bioinformatics 27(4):534–540. 10.1093/bioinformatics/btq681, http://bioinformatics.oxfordjournals.org/content/27/4/534.full.pdf+html

Rizk A, Batt G, Fages F, Soliman S (2009) A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics 12(25):il69-il78. https://doi.org/10.1093/bioinformatics/btp200

Rizk A, Batt G, Fages F, Soliman S (2011) Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. Theor Comput Sci 412(26):2827–2839. https://doi.org/10.1016/j.tcs.2010.05.008

Rosenblueth DA, Muñoz S, Carrillo M, Azpeitia E (2014) Inference of Boolean networks from gene interaction graphs using a SAT solver. In: AlCoB 2014: proceedings of the 1st international conference on algorithms for computational biology, Springer-Verlag. Lecture notes in bioinformatics, vol 8542, pp 235–246. https://doi.org/10.1007/978-3-319-07953-0_19
Ruet P (2016) Local cycles and dynamical properties of Boolean networks. Math Found Comput Sci 26(4):702–718
Sánchez L, Chaouiya C, Thieffry D (2008) Segmenting the fly embryo: logical analysis of the role of the segment polarity cross-regulatory module. Int J Dev Biol 52:1059–1075
Singhania R, Sramkoski RM, Jacobberger JW, Tyson JJ (2011) A hybrid model of mammalian cell cycle regulation. PLOS Comput Biol 7(2):e1001,077
Snoussi EH (1998) Necessary conditions for multistationarity and stable periodicity. J Biol Syst 6:3–9. https://doi.org/10.1142/S0218339098000042
Soliman S (2012) Invariants and other structural properties of biochemical models as a constraint satisfaction problem. Algorithms Mol Biol 7(15). https://doi.org/10.1186/1748-7188-7-15
Soliman S (2013) A stronger necessary condition for the multistationarity of chemical reaction networks. Bull Math Biol 75(11):2289–2303. https://doi.org/10.1007/s11538-013-9893-7
Soulé C (2003) Graphic requirements for multistationarity. ComplexUs 1:123–133
Sun J, Garibaldi JM, Hodgman C (2011) Parameter estimation using meta-heuristics in systems biology: a comprehensive review. IEEE/ACM Trans Comput Biol Bioinform (New Jersey) 9(1):185–202
Thomas R (1973) Boolean formalisation of genetic control circuits. J Theor Biol 42:565–583
Thomas R (1981) On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations. Springer Ser Synerg 9:180–193
Thomas R (1991) Regulatory networks seen as asynchronous automata: a logical description. J Theor Biol 153:1–23
Thomas R, D'Ari R (1990) Biological feedback. CRC Press, Boca Raton
Thomas R, Kaufman M (2001) Multistationarity, the basis of cell differentiation and memory. Chaos 11(1):170–195
Traynard P, Fauré A, Fages F, Thieffry D (2016a) Logical model specification aided by model- checking techniques: application to the mammalian cell cycle regulation. Bioinformatics 32(17):i772–i780. https://doi.org/10.1093/bioinformatics/btw457
Traynard P, Feillet C, Soliman S, Delaunay F, Fages F (2016b) Model-based investigation of the circadian clock and cell cycle coupling in mouse embryonic fibroblasts: prediction of reverb-alpha up-regulation during mitosis. Biosystems 149:59–69. https://doi.org/10.1016/j.biosystems.2016.07.003
Turing A (1950) Computing machinery and intelligence. Mind 49:433–460
Tyson JJ (1991) Modeling the cell division cycle: cdc2 and cyclin interactions. Proc Natl Acad Sci 88(16):7328–7332
Uhlendorf J, Miermont A, Delaveau T, Charvin G, Fages F, Bottani S, Batt G, Hersen P (2012) Long-term model predictive control of gene expression at the population and single-cell levels. Proc Natl Acad Sci USA 109(35):14,271–14,276. https://doi.org/10.1073/pnas.1206810109
Valdès-Pérè R (1995) Machine discovery in chemistry: new results. Artif Intell 74:191–201
Valiant L (1984) A theory of the learnable. Commun ACM 27(11):1134–1142
Valiant L (2013) Probably approximately correct. Basic Books, New York
Varma A, Palsson B (1994) Metabolic flux balancing: basic concepts, scientific and practical use. Nat Biotechnol 12(10):994–998
Zevedei-Oancea I, Schuster S (2003) Topological analysis of metabolic networks based on Petri net theory. Silico Biology 3(29)

When Artificial Intelligence and Computational Neuroscience Meet

Frédéric Alexandre, Peter F. Dominey, Philippe Gaussier, Benoît Girard, Mehdi Khamassi and Nicolas P. Rougier

Abstract Computational Intelligence and Artificial Intelligence are both aiming at building machines and softwares capable of intelligent behavior. They are consequently prone to interactions, even if the latter is not necessarily interested in understanding how cognition emerges from the brain substrate. In this chapter, we enumerate, describe and discuss the most important fields of interactions. Some are methodological and are concerned with information representation, processing and learning. At the functional level, the focus is set on major cognitive functions like perception, navigation, decision making and language. Among the salient characteristics of the critical contributions of Computational Neuroscience to the development of intelligent systems, its systemic view of the cerebral functioning is particularly precious to model highly multimodal cognitive functions like decision making and language and to design cognitive architectures for the autonomous behavior of robots.

F. Alexandre (✉) · N. P. Rougier
Inria Bordeaux Sud-Ouest, LaBRI UMR 5800, IMN UMR 5293, CNRS,
Université de Bordeaux, Bordeaux INP, Bordeaux, France
e-mail: Frederic.Alexandre@inria.fr

N. P. Rougier
e-mail: Nicolas.Rougier@inria.fr

P. F. Dominey
Inserm 1208 Stem Cell and Brain Research Institute, Human and Robot Cognitive Systems,
18, avenue du doyen Jean Lépine, 69675 Bron Cedex, France
e-mail: peter.dominey@inserm.fr

P. Gaussier
ETIS (ENSEA, University of Cergy Pontoise CNRS UMR 8051), 2,
avenue Adolphe-Chauvin, B.P. 222, 95302 Cergy-Pontoise Cedex, France
e-mail: gaussier@ensea.fr

B. Girard · M. Khamassi
Sorbonne Universités, CNRS, Institute of Intelligent Systems
and Robotics (ISIR), 75005 Paris, France
e-mail: benoit.girard@isir.upmc.fr

M. Khamassi
e-mail: mehdi.khamassi@upmc.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_9

1 Introduction

The goal of Computational Neuroscience (CN) is to study relations between brain structures and functions by the means of information processing techniques (Marr 1982; Schwartz 1990; Churchland and Sejnowski 1992; Dayan and Abbott 2001). This scientific domain has to deal more specifically with three major topics, neuronal processing, learning and brain functions and aims at establishing its achievements by the design of hardware and software systems to be compared with brain performances. It has consequently large overlaps with connectionism, machine learning and cognitive science but it is also different because it is specifically interested in understanding how these topics are implemented in real brains, with possible effects in neuroscience and in psychology.

Artificial Intelligence (AI) is another scientific domain overlapping CN. Though AI is not directly interested in understanding the brain, it is also a computational science aiming at building machines and softwares capable of intelligent behavior. There are consequently many fields where these two domains could cross-fertilize but others are more confrontational because each of the two domains relies on specific bases not to say dogmas.

In this chapter, we propose to visit several fields of interest to better understand the rich relations between AI and CN. Some fields are directly related to some cognitive functions that have been deeply studied in AI and where CN has investigated and modeled the cerebral structures generally reported to be mainly involved in this cognitive function. This is the case with decision making and related processes of action selection and reinforcement learning involving basal ganglia (Sect. 5) and also with language and the central role of the prefrontal cortex (Sect. 6). Other fields are more methodological since CN can be also useful to propose to AI original, efficient and robust mechanisms for information representation. We begin to evoke here the problem of the level of description in computational models, from neurons to symbols (Sect. 2), and turn to the representation of sensory information in the cortex (Sect. 3) and to their multimodal integration in the hippocampus (Sect. 4). Having mentioned these points of influence in computational mechanisms and in cognitive functions will be a good basis for discussion proposed in Sect. 7.

2 From Neurons to Symbols

Initial modeling approaches of neuronal functioning (McCulloch and Pitts 1943) and learning (Hebb 1949) in the middle of the XXth century are often considered to be at the root of the development of both CN and connectionism (artificial neural networks without biological inspiration, presented in chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2), even if oldest efforts exist (as discussed by Brunel and van Rossum 2007), particularly related to neuron excitability, in the context of CN.

At more microscopic levels of description, CN is also interested in understanding the behavior of single neurons through the dynamic evolution of their membrane potential, as it is the case for example in another seminal model by Hodgkin and Huxley (1995) or in other models dedicated to lower levels of description like dendrites, axons or even ion channels. Building such biophysical models is important to understand how neurons process information - how they compute - and some bottom up approaches propose to build on them up to the brain level and high level cognitive functions. This is specifically the case with huge simulation projects like the Blue Brain Project (Markram et al. 2015) and more recently the Human Brain Project, but many researchers wonder if such ascending projects are constrained enough to drive directly from sub-neuronal levels to cognition (Frégnac and Laurent 2014; Chi 2016). In some way, the reciprocal criticism is sometimes given to cognitive psychology, stating that a descending approach, purely driven with functional consideration, is too vague to anchor in biological reality.

This is reminiscent of the duality in AI between the difficulty of making symbols emerge from numerical computation and the reciprocal Symbol Grounding Problem (Harnard 1990). This duality has been nicely addressed by D. Marr considering the visual brain as an information processing system, with the Tri-Level Hypothesis (Marr 1982). He proposes to define the computational level, describing the (visual) functions of the brain, the implementational level describing the underlying neuronal circuitry and, in-between, argues for an algorithmic level including the representations and processes employed by the implementational level to create the computational level.

Another property of this intermediate level is that it can be partly disconnected from the other two levels. At some moment, it can be interesting to wonder how cognitive functions are implemented by lower level mechanisms or to wonder how some mechanisms can result from a precise neuronal circuitry without having all the three levels of analysis in mind. Beyond the domain of vision, this fruitful analysis is often used by researchers exploiting a certain formalism of computation, like bayesian statisticians describing the brain (Friston 2012) or theoreticians defining neuronal operations at the level of the population of neurons (Coombes 2005) in reference to biological data about arrangements of neurons in repetitive circuits (Hubel et al. 1978). Apart from the huge bottom-up projects mentioned above, most of the research in CN aiming at studying cognitive functions exploit such intermediate algorithmic levels with intermediate processing units (representing circuits or populations of neurons) and intermediate mechanisms (representing connectivity or learning rules).

Connectionism (artificial neural networks without biological inspiration) has also been used to encode, combine and manipulate symbols (Connectionist Symbol Processing (Sun and Alexandre 1997)) and has been confronted to similar problems as AI and particularly to the Frame Problem, related to the difficulty of adequate knowledge representation. This is also illustrated in the Searle's Chinese Room (Searle 1980), where an agent can appear as intelligent by manipulating syntactic rules but turns out to have no knowledge about the meaning of its responses. This has been studied in Embodied AI (Pfeifer et al. 2007) by creating loops between the agent

and its environment, through sensors and actuators, to create circular low-level sensorimotor relations, instead of elaborating complex high level formal rules (Brooks 1990). This approach is also frequently used in CN, often concerned by perception and action, with the brain seen as a way to associate them, as we are going to describe in the next sections.

3 Sensory Perception, Cortex and Unsupervised Learning

In the brain, the representation of information begins at the lower level by the representation of perceptual information received by sensors and has been primarily studied in the somatosensory system. The primary somatosensory cortex (S1 or SI) is a part of the cerebral cortex that is situated in the lateral postcentral gyrus, posterior to the central sulcus. It is, together with the primary motor cortex (anterior to the central sulcus), one of the first cortex to develop and certainly one of the most important. SI is innervated by sensory receptors (thermoreceptors, mechanoreceptors, chemoreceptors and nociceptors) that originate from both the surface of the body (e.g. skin) and from inside the body (e.g. bones and joints).

Even though the somatotopic organization of the cortex has been hypothesized more than a century ago by John Hughlings Jackson (1886) while studying epileptic patients, its existence was really demonstrated in the forties (Penfield and Boldrey 1937; Marshall et al. 1937; Adrian 1941). This has been since then illustrated with the so-called sensory homunculus representation based on the work of Wilder Penfield using electrical stimulations on epileptic patients (even though this homunculus does not make justice to the amazing work of Wilder Penfield). These studies highlighted the somatotopic organization of the cortex and demonstrated a point-to-point correspondence between the surface of the body and the somatosensory cortex. However, such orderly representations are not restricted to the somatosensory cortex and, using similar approaches in the auditory, visual and motor domain (Diamond and Neff 1957; Hubel and Wiesel 1969; Merzenich and Kaas 1980; Gould et al. 1986; Kaas 1994), a large number of topographic maps have been described all over the cortex for different modalities (tonotopic maps, retinotopic maps, motor maps).

But they have different properties. Early observations of Leyton and Sherrington (1917) (as reported in Lemon 2008) on the adult anthropoid apes demonstrated the ability of the motor cortex to recover from extensive cortical lesions. The authors hypothesized consequently the existence of a neural substrate and/or a mechanism for such extensive recovery. However, about forty years later, Hubel and Wiesel published a very influential paper (Hubel and Wiesel 1959) that promoted the idea of fixed cortical representations following the post-natal developmental period (the so-called critical period). This hypothesis has prevailed for a long time until the studies of Merzenich and Kaas (1982), Kaas et al. (1983), Kaas (1991) provided experimental evidence for the somatosensory cortex reorganization after a peripheral nerve injury or amputation in the adult monkey.

The initial formation of these maps depends on a number of mechanisms occuring at the different stages in the development of the brain and the body and relies essen-

tially on a complex molecular axon guidance (Tessier-Lavigne and Goodman 1996) and a local selection of synapses and connections, based on temporal correlations (Katz and Shatz 1996). However, the exact mechanisms behind such organization has puzzled researchers for quite a long time. How can you preserve neighborhood relationships during the course of development such that ultimately, two neighbor skin patches tend to activate two neighbor cortical patches? Either you have to consider a genetic encoding where each cell "*knows*" where to connect or you have to consider an autonomous process that results in such orderly organization.

This latter hypothesis has been proposed in the seventies by Willshaw and von der Malsburg (Willshaw and von der Malsburg 1976) with the idea of a self-organization. They proposed a model of the retina considering a set of cells that have short-range excitation (cooperation) and long range inhibition (competition). The distance between cells is provided by their actual position onto the cortical sheet, hence providing an explicit topology. They used this model to explain the formation of representation in V1 using a model of the retina that already possessed a topography. This model had a great influence and provided a very elegant explanation to the aforementionned question. Some years later, Kohonen (1982) proposed an alternative model where he got rid of the lateral connectivity in favor of a winner takes all algorithm as well as an explicit lateral connectivity function. This model has become popular far beyond the computational neuroscience domain since it also provided an elegant solution to any vector quantization problem (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2).

Vector quantization (VQ) refers to the modelling of a probability density function into a discrete set (codebook) of prototype vectors (a.k.a. centroids) such that any point drawn from the associated distribution can be associated to a prototype vector. Most VQ algorithms try to match the density through the density of their codebook: high density regions of the distribution tend to have more associated prototypes than low density region. This generally allows to minimize the distortion as measured by the mean quadratic error. For a more complete picture, it is to be noted that there also exist some cases where only a partition of the space occupied by the data (regardless of their density) is necessary. In this case, one wants to achieve a regular quantification a priori of the probability density function. For example, in some classification problems, one wants to achieve a discrimination of data in terms of classes and thus needs only to draw frontiers between data regardless of their respective density. Such vector quantization can be achieved using several methods such as variations of the k-means method (Macqueen 1967), Linde–Buzo–Gray (LBG) algorithm (Linde et al. 1980) or neural network models such as the self-organizing map (SOM, a priori topology) (Kohonen 1982), neural gas (NG, no topology) (Martinetz et al. 1993) and growing neural gas (GNG, a posteriori topology) (Fritzke 1995). Nonetheless, the SOM algorithm remains the most popular in the field of computational neurosciences since it gives a plausible account on the organization of receptive fields in sensory areas where adjacent neurons share similar representations as explained previously.

However, the stability and the quality of this self-organization depends heavily on a decreasing learning rate as well as a decreasing neighbourhood function. This is a major drawback of most neural map algorithms because it is thus necessary to

have a finite set of observations to perform adaptive learning starting from a set of initial parameters (learning rate, neighbourhood or temperature) at time t_i down to a set of final parameters at time t_f. In the framework of signal processing or data analysis, this may be acceptable as long as we can generate a finite set of samples in order to learn it off-line. However, from a more general point of view, it is not always possible to have access to a finite set and we must face on-line learning as for example during a robotic task. The question is thus how to achieve both stability and plasticity.

To answer this question, variants of the original SOM learning algorithm have been proposed where the time dependency has been removed (see Rougier and Boniface 2011 for example). Based on several experiments in both two-dimensional, high-dimensional and dynamic cases, these variants allow for on-line and continuous learning ensuring a tight coupling with the environment. Following up on these ideas, (Detorakis et al. 2012; Detorakis and Rougier 2014) investigated the formation and maintenance of ordered topographic maps in the primary somatosensory cortex and the reorganization of representations after sensory deprivation or cortical lesion. Their model is based on neural field theory using plastic feed-forward thalamocortical connections while cortico-cortical connections drive the competition mechanism.

Beyond these limitations, the original self-organizing map by Kohonen has become ubiquitous in the artificial intelligence landscape for learning and representing simple sensory information. SOM has been and is still used in a huge number of works in both image and signal processing, pattern recognition, speech processing, artificial intelligence, etc. Hundreds of variants of the original algorithm exist today (Kaski et al. 1998; Oja et al. 2003) such that it is literally impossible to review all of them here. Being both simple to implement and fast to compute, it is used in a wide variety of tasks ranging from navigation, compression, encoding, feature selection, and many others.

However, for the representation of more complex multimodal information (of "objects"), some other cerebral structures are necessary, namely, the hippocampus. This structure plays a central and critical role in the integration of various sources of information as well as in the encoding of multimodal regions such as the associative cortex. We now evoke the main ingredients of these complex processes.

4 The Hippocampus for Multimodal Binding

4.1 Functional Organization of the Hippocampus: Implication for Learning

The hippocampal system (HS) seems to participate in a lot of cognitive functions. In humans and animals, the HS plays an important role in spatial cognition but also in more general memorization processes. One important fact is that the hippocampus (Hip), a part of the HS with a seahorse shape, is connected through the entorhinal cortex (EC) to all the cortical associative areas. This makes the HS a very special

convergence point to integrate multimodal information (McClelland et al. 1995). The bilateral ressection of the hippocampi in human causes a severe anterograde amnesia while the ability to learn new skills remains intact (Scoville and Milner 1957). Moreover, Alzheimer disease targets first the hippocampus and induces the same kind of memory issues. All these researches suggest the HS plays a major role in the formation of new memories and more specifically in episodic and autobiographic memories. Most of these memories would next be recoded at the cortical level by mechanisms which are still not well understood.

The HS can be seen as a generic tool to index or to build hash codes of the cortical activity (Teyler and DiScenna 1986) in order to detect and learn in a fast way new events that cannot be easily detected by cortical neurons (Cohen and Eichenbaum 1993; Eichenbaum et al. 1994; Bunsey and Eichenbaum 1996; Buzsáki 2013; Buzsáki and Moser 2013) because of the practical limitation of the neurons connectivity. As a matter of fact, "typical" neurons are connected to around 10 000 other neurons limiting the capability of one neuron to detect complex events related to the co-occurence of signals present in different sensory cortical areas for instance.[1] Using the "small world" connectivity found in the brain allows any neuron in the cortex to contact any other neuron with a quite limited number of intermediate neurons. Yet, building such a network cannot be done in one shot. The neocortex is characterized by a slow learning rate and overlapping distributed representations allowing the extraction of the general statistical structure of the environment, while the hippocampus learns rapidly, using separated representations to encode the details of specific events while suffering minimal interference (O'Reilly and Rudy 2000) providing both structures a quite complementary role in learning. This dual system for learning and memorization could be used in AI to limit the learning to the statistic of "important" events.

This fast learning capability of the Hip is supported by the properties of the Long Term Potentiation (LTP) found in the granular cells of the dentate gyrus (DG) and in the pyramidal cell of the Cornu Ammonia areas (CA). The specific organization of the recurrent connections in the CA3 region has been exploited in numerous models of auto associative memories (Marr et al. 1971; Hopfield 1982; McNaughton and Morris 1987) in order to explain pattern completion or pattern retrieval and even for the learning of sequences of events. In these models, the dentate gyrus (DG), one of the major inputs to CA3 is supposed to perform pattern separation or pattern orthogonalization of the activities coming from and through EC (Marr et al. 1971; McNaughton and Morris 1987; Treves and Rolls 1994). More recent theories propose that the hippocampus is a predictive auto-encoder (Gluck and Myers 1993) playing a major role in detecting new complex events and allowing their learning thanks to its reciprocal connections with the septal nuclei which controls the acetylcholine (ACh) neuromodulation (Hasselmo et al. 1995, 1996). The ACh provided by the medial septum seems to mediate the learning and memory capabilities in the rest of the

[1] We can imagine this limitation is due to wiring issues: if some neurons were connected to all the neurons in other cortical areas, the size and weight of the brain would increase dramatically due to the space need for the dendritic trees of these neurons.

brain. For instance, it has been shown that lesions of the HS disrupt the acquisition of long-latency conditioned responses (Berger and Thompson 1978) such as the eyeblink conditioning learned in the cerebellum (Thompson 1986; Kim et al. 1995).

Hence, the neurobiological results teach us the brain uses at least two memory systems: the cortex learns on the long term but slowly the statistical structures of our interactions with the environment while the HS learns quickly (but for a maximum of few weeks) some compressed codes allowing to detect novelty and to control cortical learning.

4.2 The Hippocampus in Navigation Tasks: An Example of Multimodal Integration

The discovery of place cells in the hippocampus (O'Keefe and Dostrovsky 1971; Morris et al. 1982) and later the discovery of grid cells (Fyhn et al. 2004) in the dorso median entorhinal cortex (dMEC) has emphasized the role of the hippocampus in spatial cognition. In this framework, the HS is supposed to play a specific role in spatial cognition and even to constitute a "cognitive map" (O'Keefe and Nadel 1978). A lot of works have focused on explaining how neurons in Hip can become specific to a given place. Some of the models start from visual information and show that a code built from the concatenation of several visual inputs is sufficient to recognize one place (Arleo and Gerstner 2000; Milford et al. 2004; Krichmar et al. 2005). The use of more specific codes to recognize one place as a constellation of landmark $\times$ azimuth couples (using conjunctive cells) is also used to improve generalization capabilities (Zipser 1985; Bachelder and Waxman 1994; Gaussier and Zrehen 1995) when the animal is moving. Other models are using the distance instead of the azimuth (Burgess et al. 1997) or can use both since the elevation can be seen as a distance measure (Giovannangeli et al. 2006). Yet, most of these models suppose that some place-action associations can be learned presumably thanks to the output of the Hip in the direction of the basal ganglia to control the direction of the action to be performed in a given place (see next section for more details on the basal ganglia).

The nature of the inputs and their coding can have very important impact on how place recognition can be used. For instance, if the azimuth information is coded thanks to a 1 dimension neural field (a bubble of activity centered on the neuron associated to a given azimuth) then the landmark-azimuth conjunctions are sensitive to the azimuth variation between the learned place and the actual place. The resulting place cells exhibit very large and reliable place fields (Gaussier et al. 2000; Giovannangeli et al. 2006). In a room of 10 $\times$ 10 m, the place field can easily be 2 $\times$ 2 m and can scale with the size of the environment. It is really an interesting property since after the learning of few place-action associations in the vicinity of a "goal" location, the competition between actions allows reaching the goal place from never visited places and from locations far away from the learned places.[2] In this case, the capability of

[2]To the extend that these places still belong to the same visual environment i.e. that they are in the area surrounded by the visual landmarks used for learning.

one neuron to recognize one place is no more important. It is only the rank of the place cells activity level (their firing rate) that matters: the action is chosen according to the winning place (competitive process). This is an important change from classical place cells models since the issue is no more to recognize one place but to be able to reach that place. In real world conditions, being sure of recognizing one place can be quite difficult when landmarks are moved or occluded (the effect of these changes on the cells activities can be higher than when moving away from the learned place). Yet, the rank of the place cells will remain the same: all the neurons are usually impacted in the same way if landmarks are randomly hidden or displaced. For instance, using 40 visual landmarks while learning places allows maintaining a good homing behavior even if more than half of the landmarks are hidden (basically 2–3 well recognized landmarks are sufficient). This generalization capability can also be very interesting to find a shortcut or to perform a detour according to the experimental situation. Moreover, if the landmark $\times$ azimuths conjunctions are selected in a 180° image instead of a 270° panorama then the cell activities are no more place cells but view cells and could explain the recording of "view cells" in the monkey hippocampus (Rolls and O'Mara 1995) as opposed to the place cells in the rat. Yet, the place fields found in the CA region of the Hip look like really small (size about 20 cm) as compared to the large place fields described here. One hypothesis could be that those place cells would be located before the Hip in the ventromedial part of EC and that they would not be considered as place cells because of their broad receptive field (Quirk et al. 1992). The small place fields in the Hip would be explained by the result of a competition layer allowing to obtain place field with a size equal to the mean distance between the learned places. In a new environment these cells will continue to react and provide some activities allowing to recognize the new places as a compound code. However, these visual place cells cannot explain how place cells can be maintained in the dark.

This approach is sometimes opposed to models supposing place cells are primarily built from path integration information (McNaughton et al. 1996; Touretzky and Redish 1996; Redish and Touretzky 1997) and that the HS is dedicated to spatial cognition. These models suppose path integration is computed in a discrete way using a pre-existing two dimensional grid of neurons to store successive positions (Wan et al. 1994; Touretzky and Redish 1996; Samsonovich and McNaughton 1997; Redish and Touretzky 1997; Clark and Taube 2012). Starting from a given place, the integration of velocity signal and direction of movement (supported by head direction cells) allows predicting a new position and then an update from place to place.

Hence, event if these different models differs in the way they suppose the place cells are built, all of them emphasize the role of the hippocampus in merging multimodal information: i.e exteroceptive information (vision, tactile, odor...) and proprioceptive information (speed, orientation,...). Robotics experiments provide an interesting way to test the limitations and the emergent properties of these different models.

4.3 Grid Cells in the Entorhinal Cortex: Information Compression and Coding

The seminal finding of grid cells[3] in EC (Hafting et al. 2005) has reinforced the idea of the HS as a cartesian map. The models based on a direct computation of place recognition from the association of the dynamical memory of the departure place and a speed vector have been transformed to use the grid cells instead of the place cells (McNaughton et al. 2006; Fuhs and Touretzky 2006; Burak and Fiete 2009). In these attractor models, grid activity is explained by the folding of a 2-dimensional Cartesian map representing the physical environment (a torus which has the advantage that it avoids the side effects related to the borders of the map). As pointed out by Burak and Fiete (2006), the first continuous attractor models (Fuhs and Touretzky 2006; McNaughton et al. 2006; Samsonovich and McNaughton 1997) work correctly only if the activity bubble moves exactly in register with the rat position. In other words, the network must precisely integrate rat velocity (which, in turn, must fit the environment discretization used in the simulation). If speed or movement direction does not correspond exactly to the parameters used for the discretization of the grid (in terms of angle and distance[4]), there is an accumulation of errors inducing a rapid blurring of the grid activity (Burak and Fiete 2006). Burak and Fiete (2009) proposed a solution for this issue. Yet, this model has still a lot of constraints on the network connectivity to work correctly. But above all, these models suppose that the entorhinal cortex and/or the hippocampus are devoted to navigation (see "the hippocampus as a cognitive map" (O'Keefe and Nadel 1978)) and still need visual information for the (re)calibration of the path integration system.

However, if the hippocampus is not dedicated to spatial computation how to explain grid cell activities in the EC (Hafting et al. 2005)? Gaussier et al. (2007), Jauffret et al. (2015) propose that grid cell activity results from a special case of the compression of the cortical activity projected onto the EC in order to build a hash code usable by the HS to recover information and detect complex novel states. Hence, grid cell activity would result from the simple projection and merging of a long-distance path integration onto the dorso medial entorhinal cortex (dMEC). Using a kind of modulo projection such that the same cell is activated from neurons associated to different distances allows the building of a very compact code with grid activity able to differentiate correctly different locations if the modulo factors are prime.

Recent studies on the lateral entorhinal cortex (LEC) and the way it can merge visual information for instance could reconcile both approaches. However, it is a matter of debate whether the hippocampus performs the path integration by itself or is just a generic structure to build a compact code of some path integration performed outside the hippocampus (Gaussier et al. 2007). Anyhow, these works have led to efficient bio-inspired architectures (Gaussier and Zrehen 1995; Gaussier et al. 1998;

[3]Grid cells: cells with a spatial firing frequency related to the wandered distance and direction of the animal movements.

[4]Distance = speed ⋆ time_constant of the computation time in the hippocampal loop.

Krichmar et al. 2005; Milford and Wyeth 2008) merging path integration and visual information in order to build robust place cells and to recalibrate path integration (Arleo and Gerstner 2000; Gaussier et al. 2007; Jauffret et al. 2015).

4.4 Implication of the Hippocampus in Planning and Transition Recognition

Using direct place-action associations either with a strict recognition of place (need of learning a large number of places to pave regularly the environment) or with a competition mechanism (allowing to build a Voronoi tessellation of the environment and playing with the generalization properties of place cells) allows obtaining good performances in repetitive tasks (learning an habitual behavior).

When the goals can change, the place-action associations need to be relearnt (long procedure) or need to be duplicated for each potential context or goal (with as many motor mappings as the number of goals). Reinforcement learning lacks some plasticity to explain specific learning capabilities such as latent learning (Tolman 1948). In this case, it is useful to learn independently the graph or the cognitive map connecting the known places so as to use them for planning the route in the direction of any known place without the need for more learning (Mataric 1991; Schmajuk and Thieme 1992; Guazzelli et al. 1998). If we suppose a fronto-parietal cognitive map is ultimately built from the known places using Hebbian learning (neighbor places are connected to each other), several update rules allow to see the shortest path to reach a given goal location. For instance, if the connection weights are constant (or inversely proportional to the distance or to the difficulty to reach a place) and if the neuron activity for one node on the cognitive map is the maximum of the incoming activities then after the diffusion of the goal activity onto the map, following the maximum gradient of activity allows taking the shortest path to reach that goal location.

Yet, there is a need for introducing an algorithm to read this gradient information since at a given place, the rat has not access to the activity in future places (until it reaches one of them). This problem is usually solved by using an ad hoc algorithm having a global view of the cognitive map to select the next place to be reached[5] but as far as we suppose the neurons are only performing local computations there is a clear homunculus issue. One solution is to use a vicarious trial and error approach (Schmajuk and Thieme 1992), where the animal is checking the different alternative directions before choosing the most active one. This solution is correct if important changes can be perceived from the decision point (i.e. the jonction where the choice has to be made).

Unfortunately, in a real size environment, it is more likely that corridors or paths will look like quite identical for over long distances making this approach inefficient. One solution is to suppose the hippocampus role might be to predict transitions of multimodal events (Schmajuk 1991; Grossberg and Merrill 1996; Banquet et al.

[5] Yet selecting the correct action can be tricky when the place fields are not regular enough.

1997). Working with "transition cells" instead of classical steady state place cells solves the issue of the cognitive map readout by building a graph of transition cells and allowing a given transition cell to be connected to a single action (Revel et al. 1998; Gaussier et al. 2002; Banquet et al. 2005; Hirel et al. 2013). As a matter of fact, building a cognitive map is no longer learning to connect for instance place "A" to place "B" then next to place "C" but is learning instead that transition "AB" is connected to transition "BC". If the current recognized place is "A" and "A" is connected to "B" and "D" for instance then the transitions "AB" and "AD" will have the same activity when being in A (both transition are predicted at the same level). If a diffusion activity from the goal to the different transitions on the map adds a little more activity to "AB" than "AD" then the action associated to "AB" will win and the agent can move in the correct direction since one transition is always associated to one unique direction of movement.

Coming back to the hippocampus, learning temporal transitions implies to have access both to the previous and the current places and in parallel to be able to integrate the movement from A to B (i.e. the heading direction of the animal (Sharp 1999) when going from "A" to "B"). In our case, we consider two kinds of short term memories inside the HS. First, the recurrent connections in the dentate gyrus (DG) between the granular cells and the mossy cells could allow building a temporal trace that the CA3 pyramidal cells can use to learn when a new place will be reached (let's say when "B" will be reached from the memory trace of the previous place "A"). Next, CA1 neurons could use a more rustic short term memory relying on EC3 pyramidal cells (and built from EC2 activities) to build transition cells in CA1. Then, CA1 activity could be propagated to the fronto-parietal network to build long term cognitive maps and also to the nucleus accumbens (ACC) to learn and propose the different possible transitions. Finally, the neurons in the ACC receiving the activities from the fronto-parietal network and the activity from EC1 could decide about the transition to be selected. Interestingly, recent results on "time cells" in the hippocampus (Naya and Suzuki 2011; Kraus et al. 2013; Pfeiffer and Foster 2013; Eichenbaum 2014) are coherent with this view of the hippocampus as a system to predict temporal transitions (Hirel et al. 2013).

In conclusion, the HS is mainly known and studied in rodents for its implication in navigation tasks while in primates and humans it is known to participate also in higher cognitive functions such as the building of autobiographical memories. The specific architecture and connectivity of the hippocampal system makes it important for the building of declarative (or explicit) memories as opposed to the procedural (or implicit) memories directly stored in the cortical and subcortical structures. The HS capability to build a spatio-temporal code and to perform fast or even one shot learning is a crucial element to build place codes for navigation tasks, to predict transitions between multi modal states and even to implement some timing properties for sequences learning and recognition. The complementary role of the HS and cortex in the building of different kind of memories is certainly a key element for the understanding of the human intelligence and our capability to filter the huge amount of data our brain faces during real life interactions. It is clear that mimicking the way the hippocampus is interacting with the cortical areas and the basas ganglia could

be helpful in AI when facing the management of big data in real time conditions. Moreover, neurobiological data and robotics experiments show that at least for navigation tasks, recognizing perfectly a place is not necessary to reach that place in a robust way. Hence recognizing a place or an object is much more being able to come back to that place or to grasp and act on that object. As opposed to a purely passive recognition scheme, a sensory-motor approach of cognition has strong impact on the way information has to be coded and emphasizes the importance of taking into account the action selection in all the stages of the cognitive processes (i.e. even in the design of a pattern recognition system).

5 Action Selection, Reinforcement Learning and the Basal Ganglia

5.1 The Basal Ganglia as a Central Action Selection Device in the Brain

The basal ganglia are a group of inter-connected subcortical nuclei, which receive massive convergent input from most regions of cortex, hippocampus and amygdala and output to targets in the thalamus and brainstem. It is thus considered as a privileged region in the brain having access to diverse information (from sensorimotor, emotional, motivational and reward information to associative memory and episodic memory) and directly influencing motor regions (Alexander et al. 1990; Voorn et al. 2004). It has even been hypothesized to implement some sort of dimensionality reduction – formalized as a similar process to a Principal Component Analysis – in order to sort out the most important and relevant features among the large amount of information to which an individual is confronted in order to decide which motor response should be performed at a given moment (Bar-Gad et al. 2000).

One of the main theories of the role of the basal ganglia is that it constitutes a neural substrate of a central action selection device within the brain (Mink 1996; Redgrave et al. 1999; Gurney et al. 2001a, b). The theory explicitly makes interesting links with the problem of module selection within distributed, modular control architectures in the field of Engineering. It argues that a central selection device minimizes the number of connections as well as human prior knowledge. While a distributed selection architecture requires inhibitory interconnections between all modules, a central selection device only requires connections between each module and itself. While a subsumption architecture (Brooks 1986) requires a preprogrammed, fixed priority scheme, a central selection device can compare modules with a common currency (thus enabling learning mechanisms as described below).

These features have attracted the attention of several researchers in Artificial Intelligence and Robotics, who wanted to study the potential benefits of neuro-inspired basal ganglia action selection models for artefacts compared to Engineering methods (Prescott et al. 1999; Girard et al. 2003, 2005, 2008; Khamassi et al. 2005,

2006). In particular, Girard and colleagues have shown that such basal ganglia models have some persistence properties which enable a robot to save more energy than a classical winner-takes-all mechanism (Girard et al. 2003). The anatomical loops that the basal ganglia form with the cortex and the thalamus provide a feedback mechanism to the action selection mechanism, enabling a selected channel (or action) to have a slight bonus over competing channels during the competition at the next time step. Hence the persistance in action selection. Such a feedback is particularly relevant to avoid "hesitations" in the system when two channels have the same saliency and are thus oscillatorily selected one after the other, resulting in the absence of any displacement by the robot. In other words, such a persistance mechanism can help solving the Buridan donkey paradox, which in extension to a discussion raised by Aristotle suggests that a donkey having to choose between food and water, and being as hungry as thirsty, would remain unable to decide, immobile and would die from starvation.

Another interesting property of action selection mechanisms in the basal ganglia is that they operate through disinhibition rather than excitation of the selected action (Chevalier and Deniau 1990). Basal ganglia output nuclei are indeed tonically inhibiting their motor targets, so that the selection of an action results in the suppression of the inhibition of the corresponding channel (hence a disinhibition). This results in a faster action initiation compared to alternative mechanisms where action selection would result in the initiation of a motor command. Moreover, it has been shown in the oculomotor domain that such inhibition mechanisms enable to prevent the blocking of voluntary sight orientation movements by compensatory eye movements due to the vestibulo-occular reflex (Berthoz 2002).

The basal ganglia is not only fed with cortical input information (i.e. neurotransmission) but is also strongly modulated by neuromodulators such as dopamine, noradrenaline, serotonin and acetylcholine. These neuromodulators have been hypothesized to perform a meta-control or meta-learning process on top of basal ganglia action section mechanisms (Doya 2002). They have been shown to both affect neural plasticity and instantaneous gain modulations of information transmission (Servan-Schreiber et al. 1990; Reynolds et al. 2001). One of the advantages of such neuromodulation mechanisms is that they enable to reduce the combinatorial explosion in the number of required channels to represent different variations of the same action. For instance, rather than representing a different channel for the same action performed with different response vigors, with different speeds of movements, or in relation to different contexts and goals, the neuromodulatory system can learn to perform different levels of modulations on the same action channel (Niv et al. 2007; Humphries et al. 2012).

Finally, the basal ganglia are anatomically organized into different territories which form different parallel loops with different cortical and thalamic territories (Alexander et al. 1990; Voorn et al. 2004). Such organization is well conserved through evolution (Redgrave et al. 1999; Prescott et al. 1999; Stephenson-Jones et al. 2011). Since each basal ganglia territory and group of nuclei is seen as roughly organized in the same manner as other territories (Gerfen and Wilson 1996), such a parallelism permits a reuse of the same selection mechanisms applied to different

action domains (locomotor, oculomotor, etc.), different levels of selection (movement selection, action selection, action plan selection, strategy selection, goal selection). These parallel loops are thus considered as engaged in different cognitive functions such as motor, associative, limbic and oculomotor (Alexander et al. 1990; Haber et al. 2000; Uylings et al. 2003). An architecture with two simulated basal ganglia loops has for example been successfully used in a robotic task to select among both appetitive actions (directions of movement) and consummatory ones (stops at different reloading stations) and to coordinate them (Girard et al. 2005). Furthermore, the limbic loop having a privileged access to reward as well as other emotional information from the amygdala, hypothalamus and brainstem, and being in a position of influence over other loops through neuromodulatory projections, this suggests that the limbic loop of the basal ganglia may play a central role in learning (Graybiel 1998; Bornstein and Daw 2011; Ito and Doya 2011; Khamassi and Humphries 2012; van der Meer et al. 2012).

5.2 The Basal Ganglia as a Center for Reinforcement Learning

Animals' ability to learn from their own experience and errors, in particular in the context of sparse reward and punishment signals, is considered to rely on reinforcement learning processes (Doya 2000; Foster et al. 2000; Balleine and O'Doherty 2010; Khamassi and Humphries 2012; van der Meer et al. 2012; Palminteri et al. 2015). The most central theory, developed in the field of Artificial Intelligence, currently considers that such learning relies on: (1) the competition between actions, resulting in action selection as a function of the actions' relative probabilities; (2) the anticipation of the value of rewards and punishments that could follow the execution of the action; (3) the computation of a *reward prediction error* comparing what was expected with what is actually obtained; (4) the use of such a reward prediction error as a feedback (i.e. positive, negative or null reinforcement signal) to update either the probability of the performed action or the predictive value associated to the action and to the stimuli present in this context (Sutton and Barto 1998).

This formalism can be seen as an extension of the Rescorla-Wagner model (Rescorla and Wagner 1972) developed in Psychology, in which learning requires prediction errors to explain various properties of associative learning during animals classical conditioning. Prediction errors can indeed explain the *blocking* phenomenon – when a stimulus B cannot be associated with a reward if it is presented together with a stimulus A which is already fully predictive of the reward –, and cases of *over-expectation* – when the concomitant presentation of two reward predictive stimuli influences behavior as if they were adding up, to form a stronger prediction.

A particular subgroup of RL algorithms implementing what is called Temporal-Difference (TD) learning extends the Rescorla-Wagner model in that prediction error signals contain three terms rather than two. The Rescorla-Wagner indeed compares

past expectation with present outcome (e.g. reward). The TD learning rule adds to this comparison a term representing future expectations of reward. As a consequence, a reinforcement signal can be computed even before the reward is attained by comparing temporally consecutive expectations of reward – hence the term *Temporal-Difference*: e.g. when an action leads to a situation or state where reward expectations are higher than previous ones, this action should be reinforced.

Since nearly twenty years, this theory has provided Neuroscientists with formal tools which contributed to important breakthroughs in the understanding of neural correlates of learning. Reinforcement Learning models turned out to be able to explain a wide range of adaptive behaviors experimentally observed both in humans (e.g. Frank et al. 2009; Balleine and O'Doherty 2010) and in non-human animals (e.g. Yin and Knowlton 2006; Khamassi and Humphries 2012). This formalism also enabled to explain a variety of neural correlates of learning (Schultz et al. 1997; Khamassi et al. 2008). The most striking example and probably the most central in the field is the observation that phasic responses of dopaminergic neurons (which send massive neuromodulatory projections to the basal ganglia Haber et al. 2000) follow the profile of reward prediction errors as they are formalized by the RL theory: an increase in activity when the outcome of action is better than expected; a decrease in activity when it is worse than expected; an absence of response when it meets the expectations (Schultz et al. 1997). In addition, the third term of the learning rule mentioned above enables TDRL models to account for reward anticipation signals in the rat ventral striatum (the main nucleus in the limbic part of the basal ganglia) (Khamassi et al. 2008) as well as in some dopaminergic neurons (Bellot et al. 2012).

The accumulation of neurophysiological results corroborated by this computational theory has also enabled to establish that the learning of reward values and action values depends on plasticity in projections from the cortex to the basal ganglia (in particular to the striatum, the main input structure of the basal ganglia), and that these adjustments depend on dopaminergic signals sent from the substantia nigra pars compacta and the ventral tegmental area (Barto 1995; Houk et al. 1995; Schultz et al. 1997; Reynolds et al. 2001). Numerous computational models of the basal ganglia were derived from these experimental results (Houk et al. 1995; Schultz et al. 1997; Doya 2000; Joel et al. 2002; Khamassi et al. 2005; Frank 2005; Guthrie et al. 2013; N'Guyen et al. 2014), and were built on the central assumption mentioned above that the basal ganglia play a critical role in action selection (Redgrave et al. 1999; Gurney et al. 2001a, b).

Strikingly, this field of investigation has entertained an important dialog between AI and Neuroscience. One of the main recent examples is the interest that neuroscientists have gained for the distinction between different types of TDRL algorithms in the field of AI, namely: Actor-Critic, Q-learning, SARSA. These three algorithms use different information in their learning rule, which is respectively based on state value (independent from the action), the maximal action value in the current state, the value of the action chosen to be performed in the current state at the next timestep. These three variations of TDRL thus lead to different profiles of reward prediction error signals. As a consequence, several neuroscience groups have designed experiments to specifically investigate whether reinforcement signals in the brain are consistent

with either Actor-Critic, Q-learning or SARSA. Contradictory results have been obtained by different groups so far (Morris et al. 2006; Roesch et al. 2007; Bellot et al. 2012). There could exist differences between species (monkeys and rats) or between experimental configurations (presentation of single versus pairs of reward predicting stimuli). Future dialogs between Neuroscience and AI thus promise fertile exchanges along this line of research.

Finally, in the last decade computational neuroscientists have discovered that the classical distinction between learning strategies considered in AI and Machine Learning, namely model-based and model-free RL, also applies to Neuroscience by capturing different experimentally observed behavioral strategies in mammals, and related brain activities in different networks involving different basal ganglia territories (Daw et al. 2005; Khamassi and Humphries 2012; Dollé et al. 2018). More precisely, it turns out that mammals often start learning a task by trying to build an internal model of the task states, actions and transitions between them. This enables initial flexible behavior – which in particular enables to quickly adapt to task changes – in parallel to the slower acquisition of model-free local action values in the background. Once the latter learning process has converged – if the stability and familiarity of the task permit this long convergence – it starts expressing its learned behavioral sequences. This permits to free the parts of the brain which are responsible for model-based decisions (including the prefrontal cortex), thus enabling quicker but at the same time more rigid action selection. If the task changes after a long period of stability, it is more difficult for mammals to break their habits and adapt to the new task contingencies.

Nevertheless, this line of Neuroscience research promises interesting future inspiration for AI by investigating how the brain efficiently coordinates these different types of learning. For instance, Daw and colleagues have suggested that the relative uncertainty within each learning system could help the brain decide which system should control behavior at any given moment (Daw et al. 2005). In addition, more recent models can explain a variety of animal behavior in different tasks by employing a meta-controller which meta-learns which learning system was the most efficient in each state of each task (Dollé et al. 2018). This suggests principles for the coordination of multiple learning systems which could inspire Artificial Intelligence in return. Recent applications of these principles to Robotics suggest that this can work on real robots in a variety of tasks (Caluwaerts et al. 2012; Renaudo et al. 2014). While classical AI-based robots usually try to solve a given problem with a single algorithm (either planning or reinforcement learning), trying to make this algorithm the best possible on the considered solution, it turns out that different problems require different algorithms (Kober et al. 2013). Here the neuro-inspired solution suggests that a system or cognitive architecture coordinating multiple learning processes (as it is the case in the basal ganglia) may benefit from the advantages of each process/algorithm, and may learn by itself which one is the most appropriate in each given situation (Caluwaerts et al. 2012; Renaudo et al. 2015; Khamassi et al. 2016). While this may lead to good but suboptimal performance in a given problem (at is the case for mammals), this may enable the agent to adapt to many different situations, which could be of great potential interest for Artificial Intelligence research.

In return, more recent developments in AI and Robot learning permitting to efficiently and dynamically tune the exploration-exploitation trade-off in reinforcement learning (Wang et al. 2016; Khamassi et al. 2018) as well as to bootstrap learning with prioritized experience replay (Schaul et al. 2015) could greatly inspire future improvements of reinforcement learning models in Neuroscience (Caze et al. 2018).

6 Language and the Prefrontal Cortex

As the most recent arrival in the long evolution of the primate cortex, the prefrontal cortex (PFC) is considered to one of the pillars of higher cognitive function (Fuster 1991; Goldman-Rakic 1987; Miller and Cohen 2001; Wang et al. 2015). Generally speaking, there is a transition in the posterior to anterior extent of the cortex from sensory-motor functions posteriorly to progressively more integrated and abstract functions as we move more anterior in cortex, culminating in prefrontal cortex (Fuster 1991). Thus, PFC has been a candidate for numerous computational models (Bastos et al. 2012; Dominey et al. 1995; Duncan 2001; O'Reilly and Frank 2006). Complimentary to its place as a highly associative area, one of the principal neurophysiological characteristics of prefrontal cortex is the density of local recurrent connections (Goldman-Rakic 1987). A second neurophysiological characteristic of the prefrontal cortex is its privileged relation with the basal ganglia and thalamus. We will see how these characteristics can lead to impressive computational capabilities.

The computational power of recurrent networks has traditionally been demonstrated in a number of domains (Douglas et al. 1995; Hermans and Schrauwen 2012; Pearlmutter 1995). One of the great challenges in modeling recurrent networks concerns how to adapt the recurrent connection weights, as it is difficult to assign credit to recurrent connections (Pearlmutter 1995). When an input excites the network, activation can circulate through the recurrent connections numerous times before the output is generated. If the output is incorrect, and one wants to modify a recurrent connection in order to reduce the error, one must keep in memory the different roles of this connection throughout the numerous cycles of activation. A number of technical solutions that can involve cutting of the recurrent history to simplify the credit assignment, or unrolling the recurrent cycles, have been employed (Elman 1990; Jordan 1986; Pearlmutter 1995). The resulting recurrent neural network (RNN) models have a long and rich history in cognitive science (Cleeremans and McClelland 1991; Elman 1991).

An alternative approach is to retain the rich temporal dynamics within recurrent network, with no cutoff, by maintaining the recurrent connections fixed, and modifying connections between the recurrent units and the output units. This approach was first invented by Dominey and colleagues (Dominey 1995; Dominey et al. 1995). Later it was again independently developed by Maass as the liquid state machine (Maass et al. 2002), and by Jaeger as the echo state network (Jaeger 2001; Jaeger and Haas 2004). These three approaches have been integrated under the title of reservoir computing (Lukosevicius and Jaeger 2009).

The initial motivation for these recurrent networks was to understand how the prefrontal cortex encodes sequential structure. Barone and Joseph (1989) studied neural activity in the prefrontal cortex of monkeys that had been trained to perform a sequence learning task that involved watching the presentation of a visual sequence on a response button board, and then after a short delay, reproducing the sequence by touching the buttons on the board in the same order that they were presented. They observed that neurons in the dorsolateral prefrontal cortex (DLPFC) displayed two characteristic responses to stimuli in the sequence task. First, as had previously been observed, the neurons were spatially selective, with preferences for stimuli in particular locations in the retinal image. The second characteristic was new, and revolutionary: many of these neurons also displayed a "sequence rank" effect, that is, they had preferences for stimuli that had appeared first, second or last in the input sequence. Thus, the spatial selectivity in many neurons was modulated by the rank or order of the element in the sequence. This indicated that DLPFC embodies a mechanism for discriminating the order of items in a perceptual sequence.

These observations motivated us to develop a recurrent network that received retinotopic (spatially organized) inputs, and combined these inputs with mixed excitatory and inhibitory connections. We reasoned that this would lead to a form of mixed selectivity combining spatial location and sequence rank, as observed in the primate. This is indeed what we observed, thus demonstrating that such recurrent networks have significant sequence learning capability, and that their coding corresponds to that seen in the prefrontal cortex (Dominey et al. 1995).

More recently we have used this same type of reservoir model to solve a rather complex task where the system should search for the one rewarded target amongst four possibilities, then repeat that response for several rewards, before starting anew with the search for the new rewarded target. Model neurons showed a remarkable similarity to neurons recorded in the primate anterior cingulate cortex, including the non-linear mixture of task relevant parameters (Enel et al. 2016). It is now considered that these recurrent networks with feedback have universal computing properties (Maass et al. 2007, 2002). Intuitively, the recurrent connections project the inputs into an infinitely high dimensional space that incorporates past history. Modifiable readout neurons can then be trained to select the appropriate representation for the task at hand.

Such universal computing should be appropriate for language learning. Language learning can be characterized as learning from paired sentence-meaning examples how to generate the corresponding meaning for a new sentence. In order to have a simplified version of such a task, we looked to neurolinguistic tasks used to measure human language comprehension. Caplan and colleagues developed a task that exploited nine different sentence types, and measured patients ability to perform thematic role assignment, or to determine who did what to whom (Caplan et al. 1985). Typical sentences tested included

1. Dative passive: The elephant was given to the monkey by the rabbit.
2. Subject-Object relative: The elephant that the monkey hit hugged the rabbit.

Patients were asked to read such sentences, and then by pointing to pictures, indicate in order the agent, the object and the recipient of the described actions. Patients with lesions in the left hemisphere, in the region surrounding the sylvian fissure, displayed deficits in using these grammatical cues to determine who did what to whom. Instead, they relied on the so called canonical order and indicating that the first mentioned noun was the agent, second object, third recipient (Caplan et al. 1985).

We reasoned that this thematic role assignment problem could be considered as a sequence processing problem, where the system should take the input sentence and reorder the nouns (if necessary) into the agent, object, recipient order, based on the grammatical function words like "was", "to" and "by". In this context, meaning can be represented in a predicate-argument form such as "predicate(agent, object, recipient)", and sentences are represented as sequences of words. As illustrated in Fig. 1, these grammatical words are processing the recurrent network (that we posit to be in BA47 PFC region), and the open class words (nouns and verbs) are held in a working memory (that we posit to be in prefrontal cortex BA44). Through learning, the system associates different states of activity in the recurrent network with selection of different elements in the working memory, thus linking different sentence forms with different re-ordering of the open class elements into the agent, object, recipient order, as required for Caplan's task (Dominey et al. 2003).

For example, "It was the cat[1] that the dog[2] chased[3]" becomes "The dog[2] chased[3] the cat[1]". From an abstract perspective, this corresponds to an abstract structure ABC-BAC which is a sequence of six elements where the second triplet is a systematically transformed version of the first (Dominey et al. 1998). Interestingly this model made a strong prediction about the equivalent processing of linguistic sequences

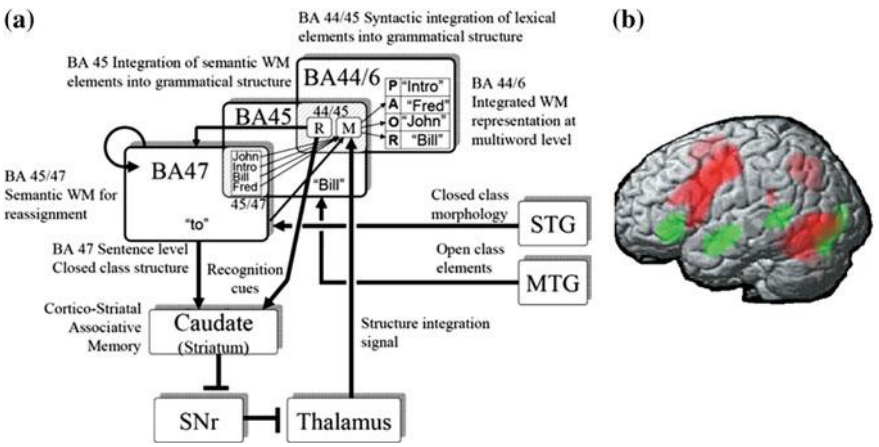


Fig. 1 A. Neural network model of fronto-striatal system for sentence and non-linguistic abstract sequence processing (from Dominey et al. 2009). B. Neural activity for sentence and abstract sequence processing (from Hoen et al. 2006). Common activity for sentences and sequences in red (corresponding to recurrent network and structure mapping), and language specific activity in green (corresponding to integration of semantic content into grammatical structure via ventral pathway and BA45)

(i.e. sentences) and non-linguistic abstract sequences in the brain: both sentences, and non-linguistic abstract sequences that required a systematic re-ordering of certain elements should recruit a common brain network for structure processing, while language should require additional processing to integrate semantic contents.

One of the remarkable features of language processing, and thus a key property for any model of language processing is the ability to learn grammatical structure from a limited number of examples, and to generalize this learning to new grammatical sentences. By using more optimized learning techniques to learn the associations between activity in the recurrent network and the corresponding responses, we have demonstrated such generalization. Figure 2 illustrates the updated reservoir model for language comprehension, and generalization performance on untrained sentence types in a parameter exploration where network parameters are systematically varied.

An interesting property of the behavior of the network can be seen in Fig. 3. The two panels illustrate activity in the readout neurons that code for the semantic role of the first noun in two different grammatical types. In the second sentence, this noun is the agent or subject of the main (second) verb and the object of the relative (first) verb. The two sentences are identical up to the arrival of the fourth word. In the subject-subject sentence depicted in A, the model accurately predicts what happens and there is little change in the output indicated at the arrow. In the less frequent subject-object sentence in B, the model's prediction must be updated at the arrival of the fourth word causing a large visible change in activity. This is precisely the kind of activity change that is seen in the human brain in response to lower frequency grammatical structures, referred to as the syntactic positive shift or P600 (a positivity that comes 600 ms after the offending word) (Frisch et al. 2002; Hagoort and Brown 2000).

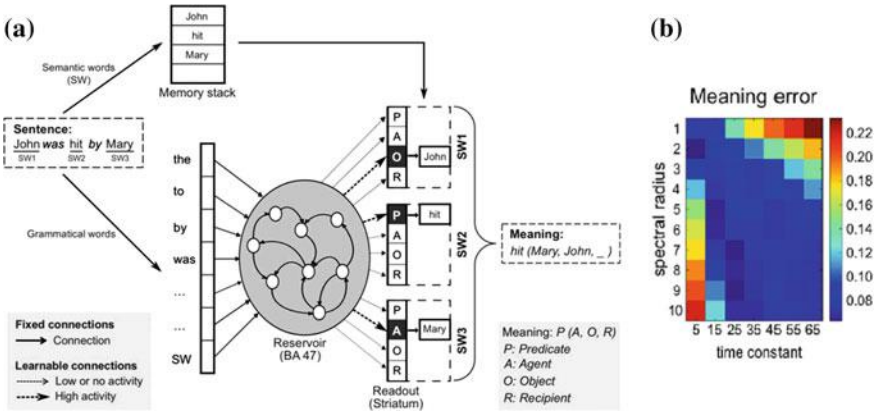


Fig. 2 Simplified reservoir model for sentence comprehension. A. Grammatical words are input to the reservoir. Through learning, connections from reservoir to the readout can associate patterns of activity in the reservoir with activation of neurons that represent the semantic role (predicate, agent, object or recipient) for each semantic word in the sentence. B. Performance error in comprehension in a test of generalization to new untrained constructions. Note an extended parameter space with good generalization

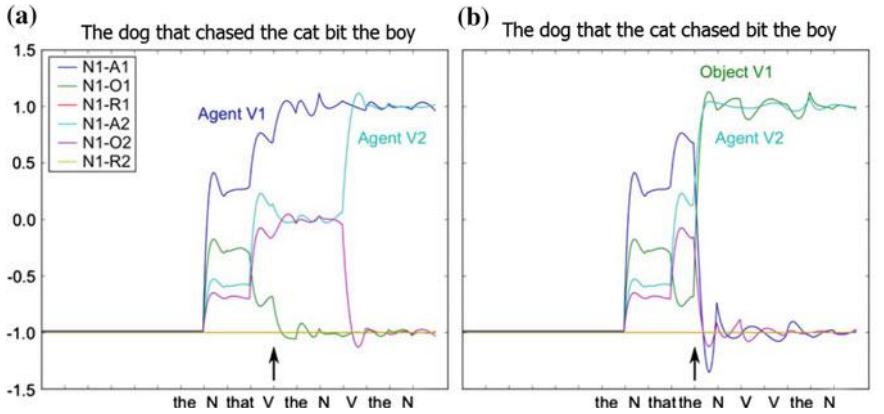


Fig. 3 Processing of Subject-Relative and Object-Relative sentences in corpus where subject-relatives are more frequent than object-relatives. A. Subject-relative. For the word "V" following "that", there is relatively small change in the readout neurons, indicating that the predictions of the model were essentially confirmed. B. Object-relative. For the "the" following "that", there is a significant shift in activity, corresponding to a reassignment of the most probable coded meaning. From Hinaut and Dominey 2013

We have seen that recurrent network models of PFC can perform complex tasks like language comprehension. Recent work at the forefront between computational neuroscience and machine learning has contributed to the concept that the prefrontal cortex is a recurrent network that has universal coding properties. Rigotti and colleagues have shown that recurrent reservoir networks inherently display mixed selectivity in their single units, and that this mixed selectivity is precisely the high dimensional coding required for solving complex cognitive tasks (Fusi et al. 2016; Rigotti et al. 2013). Interestingly this is the same kind of mixed selectivity that we modeled (mixing spatial location and sequence rank) in our first instantiation of the reservoir concept (Dominey et al. 1995), where a network of neurons connected by fixed recurrent connections provides a universal high dimensional encoding, and modifiable connections allow the system to learn to associate these representations with the desired output. Future research should attempt to further understand these neurocomputational systems, and address questions concerning how the encoding of task related context can render these systems even more powerful.

7 Conclusion

AI and CN are two scientific domains developing their own formalisms but they are both interested in understanding how cognitive functions can emerge from computational mechanisms. Specifically, CN takes a strong inspiration from the brain circuitry, which is undoubtedly a choice source of information to study the emergence of cognitive functions. Accordingly, CN can provide AI with original and first-

hand mechanisms related to cognition. In addition, Neuroscience is a very dynamical domain extracting more and more information from the mysterious brain, and CN is perpetually renewed by following these progresses and sometimes contributing to them. In this chapter, we have proposed some illustrations on important questions on which the domains can interact.

The problem of representation of information is a central issue in AI, from the Physical Symbol System hypothesis to more recent questions evoked in most chapters of Volume 1, about the representation of complex objects or events (Russell and Norvig 2003). Whereas the existence in the cortex of topographic maps of neurons responding each to preferential stimuli (cf. Sect. 3) can give arguments for localized representations at the symbolic and subsymbolic levels, the dynamics of recurrent networks in the PFC (cf. Sect. 6) and the hippocampus (cf. Sect. 4) is rather reminiscent of distributed encoding. It is also important to mention that the corresponding neuronal models are associated to well explored learning rules that allow to build both kinds of representation from sampling in the environment. Concerning the elaboration of information representation, a related topic is about the opposition between top down and button up approaches in AI (cf. Sect. 2). Pieces of evidence have been proposed here that Marr's algorithmic level could be fed with sensory information in an ascending way and controled in the opposite way by structures like the PFC (cf. Sect. 6).

Whereas learning is a central mechanism in cognition, it has a particular status in AI. Sometimes it is not directly addressed and it is believed that information can be efficiently injected in a cognitive system as formalized knowledge. Sometimes it is the central topic of an algorithm in Machine Learning that is intended to solve the considered cognitive task by processing iteratively data received in experimental cases. From this duality between adaptation by integration of knowledge or data (Sun and Alexandre 1997), CN argues that knowledge injected into a cognitive system can correspond to the very slow learning, at the scale of species evolution, of the structures and characteristics of the neuronal circuitry and proposes, for adaptation at shorter time constants, a large variety of biologically inspired learning algorithms that we have evoked throughout this chapter, particularly mentioning their strong links to classical algorithms in Machine Learning presented in chapter "Reinforcement Learning" of Volume 1 and chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2, and their cross-fertilization. It is thus notable that the recently developed Deep Learning approach evoked in chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2, which offers among the most efficient artificial learning systems, makes strong references to neuroscience and in particular to the visual system (Yamins and DiCarlo 2016), although we must also relativize this type of analogy. Specifically, the effectiveness of these systems is based largely on their training from very large corpus of examples, whereas natural learning generally has fewer trials to adapt.

Furthermore, as it can be observed in behaving animals (or robots !), an intelligent behavior often corresponds to the capacity to adapt to a variety of tasks and not to be a specialist for only one task. This is clearly one domain where AI is only starting out (but consider the domains of Lifelong Machine Learning and of Artificial General

Intelligence) and where CN is more mature, allowing to coordinate multiple ways of learning in modular networks, as it has been evoked several times above.

A related characteristic of central interest in intelligent behavior is autonomy, also too poorly dealt with in AI. Here also, CN is some steps beyond because its systemic view is more adapted to implement a versatile agent able to adapt by itself to changing and unknown conditions and also because recent models of the loops between the basal ganglia and the PFC (Koechlin et al. 2003; O'Reilly et al. 2010) are beginning to address the functions of self-evaluation of performances and cognitive control, not to mention consciousness, which are also fundamental ingredients to autonomous behavior. Nevertheless, these approaches are still far from proposing an architecture of control making the agent fully autonomous and able to exploit previously elaborated knowledge in new circumstances, and to identify these circumstances, particularly in the case of unstationary environment and a lot of work remains to be done in that direction.

Our capacity to adapt to completely unexpected situations is certainly a central explanation to the fact that our brain today spends most of its time making symbolic manipulations (with natural language, with mathematics, but also with digital devices) it was not necessarily designed for at the origin. Understanding how the systemic arrangement of cerebral structures presented here produces this general purpose information processing system specially interested in symbolic analysis is certainly another rich domain of interaction between AI and CN. Particularly, inspiration from natural sciences can bring to AI an original understanding of this problem, focusing for example on developmental issues (how skills can be installed one after the other to produce an increasingly mature system) and also on the importance of social interactions (imitation, teaching) and their impact on the development of the cerebral system.

Whereas general intelligence is clearly an important goal of CN-AI interactions but remains on a long-term perspective, we have also shown in this chapter that very concrete interactions are already existing for cognitive functions corresponding to more classical goals of AI, like perception, navigation, decision making (including action selection, reasoning, planning) and language. We have explained here that CN can provide AI with useful mechanisms and principles of information representation, by deciphering the brain circuitry involved in these functions. In addition, CN can offer another decisive contribution for helping computational models to master more widely these functions. Their development is certainly linked to the better integration of multimodal sensorimotor flows and to their interconnections, one with the others. The systemic approach of CN in clearly an asset in that direction.

In conclusion, CN has already demonstrated interesting contributions to AI at the methodological level, for information representation, processing and learning and also at the functional level for the implementation of a variety of cognitive functions (Hassabis et al. 2017). Other types of contribution are particularly precious because they exploit the unique capability of CN to develop an integrated approach of brain modeling, in a systemic view. They should be encouraged for the development of AI in domains like autonomous robotics, for multimodal cognitive functions like decision making and language and also for general intelligence.

References

Adrian ED (1941) Afferent discharges to the cerebral cortex from peripheral sense organs. J Physiol Lond 100:159–191

Alexander G, Crutcher M, DeLong M (1990) Basal ganglia-thalamocortical circuits: parallel substrates for motor, oculomotor, "prefrontal" and "limbic" functions. Prog Brain Res 85:119–146

Arleo A, Gerstner W (2000) Spatial cognition and neuro-mimetic navigation: a model of hippocampal place cell activity. Biol Cybern 83(3):287–299

Bachelder IA, Waxman AM (1994) Mobile robot visual mapping and localization: a view-based neurocomputational architecture that emulates hippocampal place learning. Neural Netw 7(6):1083–1099

Balleine BW, O'Doherty JP (2010) Human and rodent homologies in action control: corticostriatal determinants of goal-directed and habitual action. Neuropsychopharmacology 35(1):48–69

Banquet J, Gaussier P, Dreher JC, Joulain C, Revel A, Günther W (1997) Space-time, order, and hierarchy in fronto-hippocampal system: a neural basis of personality. In: Matthews G (ed) Cognitive science perspectives on personality and emotion, vol 124. Elsevier, Amsterdam, North Holland, pp 123–189

Banquet J-P, Gaussier P, Quoy M, Revel A, Burnod Y (2005) A hierarchy of associations in hippocampo-cortical systems: cognitive maps and navigation strategies. Neural Comput 17(6):1339–1384

Barone P, Joseph JP (1989) Prefrontal cortex and spatial sequencing in macaque monkey. Exp Brain Res 78(3):447–464

Barto A (1995) Adaptive critics and the basal ganglia. In: Houk J, Davis J, Beiser D (eds) Models of information processing in the basal ganglia. MIT Press, Cambridge, pp 215–232

Bar-Gad I, Havazelet-Heimer G, Goldberg JA, Ruppin E, Bergman H (2000) Reinforcement-driven dimensionality reduction - a model for information processing in the basal ganglia. J Basic Clin Physiol Pharmacol 11:30520

Bastos AM, Usrey WM, Adams RA, Mangun GR, Fries P, Friston KJ (2012) Canonical microcircuits for predictive coding. Neuron 76(4):695–711

Bellot J, Sigaud O, Khamassi M (2012) Which temporal difference learning algorithm best reproduces dopamine activity in a multi-choice task? In: Ziemke T, Balkenius C, Hallam J (eds) From animals to animats 12, vol 7426. Lecture notes in computer science. Berlin, Springer, pp 289–298

Berger TW, Thompson RF (1978) Neuronal plasticity in the limbic system during classical conditioning of the rabbit nictitating membrane response. I. The hippocampus. Brain Res 145(2):323–346

Berthoz A (2002) Le Mouvement. Odile Jacob, Paris, France

Bornstein AM, Daw ND (2011) Multiplicity of control in the basal ganglia: computational roles of striatal subregions. Curr Opin Neurobiol 21(3):374–380

Brooks RA (1986) A robust layered control system for a mobile robot. IEEE J Robot Autom R.A-2:14–23

Brooks RA (1990) Elephants don't play chess. Robot Auton Syst 6(1):3–15

Brunel N, van Rossum MCW (2007) Lapicque's 1907 paper: from frogs to integrate-and-fire. Biol Cybern 97(5):337–339

Bunsey M, Eichenbaum H (1996) Conservation of hippocampal memory function in rats and humans. Nature 379:255–257

Burak Y, Fiete I (2006) Do we understand the emergent dynamics of grid cell activity? J Neurosci 26(37):9352–4; discussion 9354

Burak Y, Fiete IR (2009) Accurate path integration in continuous attractor network models of grid cells. PLoS Comput Biol 5(2):e1000291

Burgess N, Donnett JG, Jeffery KJ, John O et al (1997) Robotic and neuronal simulation of the hippocampus and rat navigation. Philos Trans R Soc Lond B: Biol Sci 352(1360):1535–1543

Buzsáki G (2013) Cognitive neuroscience: time, space and memory. Nature 497(7451):568–569

Buzsáki G, Moser EI (2013) Memory, navigation and theta rhythm in the hippocampal-entorhinal system. Nat Neurosci 16(2):130–138
Caluwaerts K, Staffa M, N'Guyen S, Grand C, Dollé L, Favre-Félix A, Girard B, Khamassi M (2012) A biologically inspired meta-control navigation system for the Psikharpax rat robot. Bioinspiration Biomimetics 7(2):025009
Caplan D, Baker C, Dehaut F (1985) Syntactic determinants of sentence comprehension in aphasia. Cognition 21(2):117–175
Caze R, Khamassi M, Aubin L, Girard B (2018) Hippocampal replays under the scrutiny of reinforcement learning models. J Neurophysiol
Chevalier G, Deniau J (1990) Disinhibition as a basic process in the expression of striatal functions. Trends Neurosci 13(7):277–280
Chi KR (2016) Neural modelling: abstractions of the mind. Nature 531:S16–S17
Churchland P, Sejnowski TJ (1992) The computational brain. MIT Press, Cambridge
Clark BJ, Taube JS (2012) Vestibular and attractor network basis of the head direction cell signal in subcortical circuits. Front Neural Circuits 6(7):10–3389
Cleeremans A, McClelland JL (1991) Learning the structure of event sequences. J Exp Psychol Gen 120(3):235–253
Cohen N, Eichenbaum H (1993) Memory, amnesia, and the hippocampal system. MIT Press, Cambridge
Coombes S (2005) Waves, bumps and patterns in neural field theories. Biol Cybern 93:91–108
Daw N, Niv Y, Dayan P (2005) Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. Nat Neurosci 8(12):1704–1711
Dayan P, Abbott LF (2001) Theoretical neuroscience: computational and mathematical modeling of neural systems. MIT Press, Cambridge
Detorakis G, Rougier Nicolas P (2012) A neural field model of the somatosensory cortex: formation, maintenance and reorganization of ordered topographic maps. PLoS ONE 7(7):e40257
Detorakis GI, Rougier NP (2014) Structure of receptive fields in a computational model of area 3b of primary sensory cortex. Front Comput Neurosci 8:26
Diamond IT, Neff WD (1957) Ablation of temporal cortex and discrimination of auditory patterns. J Neurophysiol 20:300–315
Dollé L, Chavarriaga R, Guillot A, Khamassi M (2018) Interactions of spatial strategies producing generalization gradient and blocking: a computational approach. PLoS Comput Biol 14(4):e1006092
Dominey PF (1995) Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biol Cybern 73(3):265–74
Dominey PF, Arbib MA, Joseph J-P (1995) A model of cortico-striatal plasticity for learning oculomotor associations and sequences. J Cogn Neurosci 7(3):311–336
Dominey PF, Lelekov T, Ventre-Dominey J, Jeannerod M (1998) Dissociable processes for learning the surface structure and abstract structure of sensorimotor sequences. J Cogn Neurosci 10(6):734–51
Dominey PF, Hoen M, Blanc J-M, Lelekov-Boissard T (2003) Neurological basis of language and sequential cognition: evidence from simulation, aphasia, and ERP studies. Brain Lang 86(2):207–225
Dominey PF, Inui T, Hoen M (2009) Neural network processing of natural language: II. Towards a unified model of corticostriatal function in learning sentence comprehension and non-linguistic sequencing. Brain Lang 109(2–3):80–92
Douglas R, Koch C, Mahowald M, Martin K, Suarez H (1995) Recurrent excitation in neocortical circuits. Science 269(5226):981–985
Doya K (2000) Complementary roles of basal ganglia and cerebellum in learning and motor control. Curr Opin Neurobiol 10(6):732–739
Doya K (2002) Metalearning and neuromodulation. Neural Netw 15(4–6):495–506
Duncan J (2001) An adaptive coding model of neural function in prefrontal cortex. Nat Rev Neurosci 2, 2(11):820–829

Eichenbaum H (2014) Time cells in the hippocampus: a new dimension for mapping memories. Nat Rev Neurosci 15(11):732–744
Eichenbaum H, Otto T, Cohen N (1994) Two functional components of the hippocampal memory system. Behav, Brain Sci 17(3):449–517
Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
Elman JL (1991) Distributed representations, simple recurrent networks, and grammatical structure. Mach Learn 7:195–225
Enel P, Procyk E, Quilodran R, Dominey PF (2016) Reservoir computing properties of neural dynamics in prefrontal cortex. PLoS Comput Biol 12(6):1–35
Foster D, Morris R, Dayan P (2000) A model of hippocampally dependent navigation, using the temporal difference learning rule. Hippocampus 10(1):1–16
Frank MJ (2005) Dynamic dopamine modulation in the basal ganglia: a neurocomputational account of cognitive deficits in medicated and nonmedicated parkinsonism. J Cogn Neurosci 17(1):51–72
Frank MJ, Doll BB, Oas-Terpstra J, Moreno F (2009) Prefrontal and striatal dopaminergic genes predict individual differences in exploration and exploitation. Nat Neurosci 12(8):1062–1068
Frégnac Y, Laurent G (2014) Where is the brain in the human brain project? Nature 513:27–29
Frisch S, Schlesewsky M, Saddy D, Alpermann A (2002) The P600 as an indicator of syntactic ambiguity. Cognition 85(3):B83–B92
Friston K (2012) A free energy principle for biological systems. Entropy (Basel, Switzerland) 14(11):2100–2121
Fritzke B (1995) A growing neural gas network learns topologies. In: Tesauro G, Touretzky D, Leen T (eds) Advances in neural information processing systems 7. MIT Press, Cambridge, pp 625–632
Fuhs MC, Touretzky DS (2006) A spin glass model of path integration in rat medial entorhinal cortex. J Neurosci 26(16):4266–4276
Fusi S, Miller EK, Rigotti M (2016) Why neurons mix: high dimensionality for higher cognition. Curr Opin Neurobiol 37:66–74
Fuster JM (1991) Chapter 10 the prefrontal cortex and its relation to behavior. In: Holstege G (ed) Role of the forebrain in sensation and behavior, vol 87. Elsevier, Progress in brain research, pp 201–211
Fyhn M, Molden S, Witter MP, Moser EI, Moser M-B (2004) Spatial representation in the entorhinal cortex. Science 305(5688):1258–1264
Gaussier P, Zrehen S (1995) Perac: a neural architecture to control artificial animals. Robot Auton Syst 16(2):291–320
Gaussier P, Moga S, Quoy M, Banquet J-P (1998) From perception-action loops to imitation processes: a bottom-up approach of learning by imitation. Appl Artif Intell 12(7–8):701–727
Gaussier P, Joulain C, Banquet J-P, Leprêtre S, Revel A (2000) The visual homing problem: an example of robotics/biology cross fertilization. Robot Auton Syst 30(1):155–180
Gaussier P, Revel A, Banquet J-P, Babeau V (2002) From view cells and place cells to cognitive map learning: processing stages of the hippocampal system. Biol Cybern 86:15–28
Gaussier P, Banquet J, Sargolini F, Giovannangeli C, Save E, Poucet B (2007) A model of grid cells involving extra hippocampal path integration, and the hippocampal loop. J Integr Neurosci 6(03):447–476
Gerfen C, Wilson C (1996) The basal ganglia. In: Swanson L, Björklund A, Hökfelt T (eds) Integrated systems of the CNS, Part III, chapter II. Handbook of chemical neuroanatomy, vol 12. Elsevier Science B.V, pp 371–468
Giovannangeli C, Gaussier P, Banquet J (2006) Robustness of visual place cells in dynamic indoor and outdoor environment. Int J Adv Robot Syst 3(2):115–124
Girard B, Cuzin V, Guillot A, Gurney K, Prescott T (2003) A basal ganglia inspired model of action selection evaluated in a robotic survival task. J Integr Neurosci 2(2):179–200
Girard B, Filliat D, Meyer J-A, Berthoz A, Guillot A (2005) Integration of navigation and action selection functionalities in a computational model of cortico-basal ganglia-thalamo-cortical loops. Adapt Behav 13(2):115–130

Girard B, Tabareau N, Pham Q, Berthoz A, Slotine J-J (2008) Where neuroscience and dynamic system theory meet autonomous robotics: a contracting basal ganglia model for action selection. Neural Netw 21(4):628–641

Gluck MA, Myers CE (1993) Hippocampal mediation of stimulus representation: a computational theory. Hippocampus 3(4):491–516

Goldman-Rakic PS (1987) Circuitry of primate prefrontal cortex and regulation of behavior by representational memory. Comprehensive physiology, Wiley, Hoboken

Gould JHI, Cusick CG, Pons TP, Kaas JH (1986) The relationship of corpus callosum connections to electrical stimulation maps of motor, supplementary motor, and frontal eye fields in owl monkeys. J Comp Neurol 247:297–325

Graybiel A (1998) The basal ganglia and chunking of action repertoires. Neurobiol Learn Mem 70(1–2):119–136

Grossberg S, Merrill J (1996) The hippocampus and cerebellum in adaptively timed learning, recognition, and movement. J Cogn Neurosci 8:257–277

Guazzelli A, Bota M, Corbacho FJ, Arbib MA (1998) Affordances. motivations, and the world graph theory. Adapt Behav 6(3–4):435–471

Gurney K, Prescott T, Redgrave P (2001a) A computational model of action selection in the basal ganglia. I. A new functional anatomy. Biol Cybern 84(6):401–410

Gurney K, Prescott T, Redgrave P (2001b) A computational model of action selection in the basal ganglia. II. Analysis and simulation of behaviour. Biol Cybern 84(6):411–423

Guthrie M, Leblois A, Garenne A, Boraud T (2013) Interaction between cognitive and motor cortico-basal ganglia loops during decision making: a computational study. J Neurophysiol 109(12):3025–3040

Haber S, Fudge J, McFarland N (2000) Striatonigrostriatal pathways in primates form an ascending spiral from the shell to the dorsolateral striatum. J Neurosci 20(6):2369–2382

Hafting T, Fyhn M, Molden S, Moser M-B, Moser E (2005) Microstructure of a spatial map in the entorhinal cortex. Nature 436:801–806

Hagoort P, Brown CM (2000) ERP effects of listening to speech compared to reading: the p600/sps to syntactic violations in spoken sentences and rapid serial visual presentation. Neuropsychologia 38(11):1531–1549

Harnard S (1990) The symbol grounding problem. Phys D: Nonlinear Phenom 42:335–346

Hassabis D, Kumaran D, Summerfield C, Botvinick M (2017) Neuroscience-inspired artificial intelligence. Neuron 95(2):245–258

Hasselmo ME, Schnell E, Barkai E (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. J Neurosci 15(7):5249–5262

Hasselmo ME, Wyble BP, Wallenstein GV (1996) Encoding and retrieval of episodic memories: role of cholinergic and gabaergic modulation in the hippocampus. Hippocampus 6(6):693–708

Hebb D (1949) The organization of behavior: a neuropsychological theory. Wiley, New York

Hermans M, Schrauwen B (2012) Recurrent kernel machines: computing with infinite echo state networks. Neural Comput 24(1):104–133

Hinaut X, Dominey PF (2013) Real-time parallel processing of grammatical structure in the fronto-striatal system: a recurrent network simulation study using reservoir computing. PLoS ONE 8(2):e52946

Hirel J, Gaussier P, Quoy M, Banquet J-P, Save E, Poucet B (2013) The hippocampo-cortical loop: spatio-temporal learning and goal-oriented planning in navigation. Neural Netw 43:8–21

Hoen M, Pachot-Clouard M, Segebarth C, Dominey PF (2006) When Broca experiences the Janus syndrome: an ER-fMRI study comparing sentence comprehension and cognitive sequence processing. Cortex 42(4):605–623

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci 79(8):2554–2558

Houk J, Adams J, Barto A (1995) A model of how the basal ganglia generate and use neural signals that predict reinforcement. In: Houk J, Davis J, Beiser D (eds) Models of information processing in the basal ganglia. MIT Press, Cambridge, MA, pp 249–270

Hubel D, Wiesel T (1959) Receptive fields of single neurones in the cat's striate cortex. J Physiol 148(3):574–591

Hubel D, Wiesel T (1969) Anatomical demonstration of columns in the monkey striate cortex. Nature 221:747–750

Hubel DH, Wiesel TN, Stryker MP (1978) Anatomical demonstration of orientation columns in macaque monkey. J Comp Neurol 177:361–380

Humphries M, Khamassi M, Gurney K (2012) Dopaminergic control of the exploration-exploitation trade-off via the basal ganglia. Front Neurosci 6:9

Ito M, Doya K (2011) Multiple representations and algorithms for reinforcement learning in the cortico-basal ganglia circuit. Curr Opin Neurobiol 21(3):368–373

Jaeger H (2001) The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Bonn Ger: Ger Natl Res Cent Inf Technol GMD Tech Rep 148:34

Jaeger H, Haas H (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science 304(5667):78–80

Jauffret A, Cuperlier N, Gaussier P (2015) From grid cells and visual place cells to multimodal place cell: a new robotic architecture. Front Neurorobotics 9:1

Joel D, Niv Y, Ruppin E (2002) Actor-critic models of the basal ganglia: new anatomical and computational perspectives. Neural Netw 15(4–6):535–547

Jordan MI (1986) Serial order: a parallel, distributed processing approach. Technical Report 8604. Institute for Cognitive Science, University of California, San Diego

Kaas J (1991) Plasticity of sensory and motor maps in adult mammals. Annu Rev Neurosci 14(1):137–167

Kaas JH (1994) Baer JF, Weller RE, Kakoma, I (eds) The organization of sensory and motor cortex in owl monkeys. Aotus: the owl monkey. Academic Press, Orlando

Kaas J, Merzenich M, Killackey H (1983) The reorganization of somatosensory cortex following peripheral nerve damage in adult and developing mammals. Annu Rev Neurosci 6:325–356

Kaski S, Jangas J, Kohonen T (1998) Bibliography of self-organizing map papers: 1981–1997. Neural Comput Surv 1(3&4):1–176

Katz LC, Shatz CJ (1996) Synaptic activity and the construction of cortical circuits. Science 274:1133–1138

Khamassi M, Humphries M (2012) Integrating cortico-limbic-basal ganglia architectures for learning model-based and model-free navigation strategies. Front Behav Neurosci 6:79

Khamassi M, Lacheze L, Girard B, Berthoz A, Guillot A (2005) Actor-critic models of reinforcement learning in the basal ganglia: from natural to arificial rats. Adapt Behav 13:131–148

Khamassi M, Martinet L-E, Guillot A (2006) Combining self-organizing maps with mixtures of experts: application to an actor-critic model of reinforcement learning in the basal ganglia. From animals to animats, vol 9. LNAI 4095. Springer, Berlin, pp 394–405

Khamassi M, Mulder A, Tabuchi E, Douchamps V, Wiener S (2008) Anticipatory reward signals in ventral striatal neurons of behaving rats. Eur J Neurosci 28:1849–1866

Khamassi M, Girard B, Clodic A, Devin S, Renaudo E, Pacherie E, Alami R, Chatila R (2016) Integraton of action, joint action and learning in robot cognitive architectures. Intellectica 1:169–203

Khamassi M, Velentzas G, Tsitsimis T, Tzafestas C (2018) Robot fast adaptation to changes in human engagement during simulated dynamic social interaction with active exploration in parameterized reinforcement learning. IEEE Trans Cogn Dev Syst 10:881–893

Kim JJ, Clark RE, Thompson RF (1995) Hippocampectomy impairs the memory of recently, but not remotely, acquired trace eyeblink conditioned responses. Behav Neurosci 109(2):195

Kober J, Bagnell JA, Peters J (2013) Reinforcement learning in robotics: a survey. Int J Robot Res 32:1238–1274

Koechlin E, Ody C, Kouneiher F (2003) The architecture of cognitive control in the human prefrontal cortex. Science 302(5648):1181–1185

Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69

Kraus BJ, Robinson RJ, White JA, Eichenbaum H, Hasselmo ME (2013) Hippocampal "time cells": time versus path integration. Neuron 78(6):1090–1101

Krichmar JL, Seth AK, Nitz DA, Fleischer JG, Edelman GM (2005) Spatial navigation and causal analysis in a brain-based device modeling cortical-hippocampal interactions. Neuroinformatics 3(3):197–221

Lemon R (2008) An enduring map of the motor cortex. Exp Physiol 93:798–802

Leyton S, Sherrington C (1917) Observations on the excitable cortex of the chimpanzee, orang-utan and gorilla. Quaterly J Experimntal Physiol 11:135–222

Linde Y, Buzo A, Gray R (1980) An algorithm for vector quantization design. IEEE Trans Commun 28:84–95

Lukosevicius M, Jaeger H (2009) Survey: reservoir computing approaches to recurrent neural network training. Comput Sci Rev 3(3):127–149

Maass W, Natschläger T, Markram H (2002) Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput 14(11):2531–2560

Maass W, Joshi P, Sontag ED (2007) Computational aspects of feedback in neural circuits. PLoS Comput Biol 3(1):1–20

Macqueen JB (1967) Some methods of classification and analysis of multivariate observations. In: Proceedings of the fifth berkeley symposium on mathematical statistics and probability, pp 281–297

Markram H, Muller E, Ramaswamy S, Reimann MW, Abdellah M, Sanchez CA, Ailamaki A, Alonso-Nanclares L, Antille N, Arsever S, Kahou GA, Berger TK, Bilgili A, Buncic N, Chalimourda A, Chindemi G, Courcol J-D, Delalondre F, Delattre V, Druckmann S, Dumusc R, Dynes J, Eilemann S, Gal E, Gevaert ME, Ghobril J-P, Gidon A, Graham JW, Gupta A, Haenel V, Hay E, Heinis T, Hernando JB, Hines M, Kanari L, Keller D, Kenyon J, Khazen G, Kim Y, King JG, Kisvarday Z, Kumbhar P, Lasserre S, Le Bé J-V, Magalhães BRC, Merchán-Pérez A, Meystre J, Morrice BR, Muller J, Muñoz Céspedes A, Muralidhar S, Muthurasa K, Nachbaur D, Newton TH, Nolte M, Ovcharenko A, Palacios J, Pastor L, Perin R, Ranjan R, Riachi I, Rodríguez J-R, Riquelme JL, Rössert C, Sfyrakis K, Shi Y, Shillcock JC, Silberberg G, Silva R, Tauheed F, Telefont M, Toledo-Rodriguez M, Tränkler T, Van Geit W, Díaz JV, Walker R, Wang Y, Zaninetta SM, DeFelipe J, Hill SL, Segev I, Schürmann F (2015) Reconstruction and simulation of neocortical microcircuitry. Cell 163(2):456–492

Marr D (1982) Vision: a computational investigation into the human representation and processing of visual information. Henry Holt and Co., Inc, New York

Marr D, Willshaw D, McNaughton B (1971) Simple memory: a theory for archicortex. Philos Trans R Soc Lond Ser B Biol Sci 262(841):23–81

Marshall WH, Woolsey CN, Bard R (1937) Cortical representation of tactile sensibility as indicated by cortical potentials. Science 85:388–390

Martinetz TM, Berkovich SG, Schulten KJ (1993) Neural-gas network for vector quantization and its application to time-series prediction. IEEE Trans Neural Netw 4(4):558–569

Mataric MJ (1991) Navigating with a rat brain: a neurobiologically inspired model. From animals to animats; proceedings of the first international conference on simulation of adaptive behavior. MIT Press, Cambridge

McClelland JL, McNaughton BL, O'Reilly RC (1995) Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychol Rev 102(3):419

McCulloch W, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133

McNaughton B, Barnes C, Gerrard J, Gothard K, Jung M, Knierim J, Kudrimoti H, Qin Y, Skaggs W, Suster M et al (1996) Deciphering the hippocampal polyglot: the hippocampus as a path integration system. J Exp Biol 199(1):173–185

McNaughton B, Battaglia FP, Jensen O, Moser EI, Moser M-B (2006) Path integration and the neural basis of the 'cognitive map. Nat Rev Neurosci 7:663–678

McNaughton BL, Morris RG (1987) Hippocampal synaptic enhancement and information storage within a distributed memory system. Trends Neurosci 10(10):408–415

Merzenich M, Kaas J (1982) Reorganization of mammalian somatosensory cortex following peripheral nerve injury. Trends Neurosci 5:434–436

Merzenich MM, Kaas JH (1980) Principles of organization of sensory-perceptual systems in mammals. In: Sprague JM, Epstein AN (ed) Progress in psychobiology and physiological psychology, vol 9, pp 1–42

Milford MJ, Wyeth GF (2008) Mapping a suburb with a single camera using a biologically inspired slam system. IEEE Trans Robot 24(5):1038–1053

Milford MJ, Wyeth GF, Rasser D (2004) Ratslam: a hippocampal model for simultaneous localization and mapping. In: Proceedings. ICRA'04. 2004 IEEE international conference on robotics and automation, 2004, vol 1, pp 403–408. IEEE

Miller EK, Cohen JD (2001) An integrative theory of prefrontal cortex function. Annu Rev Neurosci 24:167–202

Mink JW (1996) The basal ganglia: focused selection and inhibition of competing motor programs. Prog Neurobiol 50(4):381–425

Morris R, Garrud P, Rawlins J, O'Keefe J (1982) Place navigation impaired in rats with hippocampal lesions. Nature 297:24

Morris G, Nevet A, Arkadir D, Vaadia E, Bergman H (2006) Midbrain dopamine neurons encode decisions for future action. Nat Neurosci 9(8):1057–1063

Naya Y, Suzuki WA (2011) Integrating what and when across the primate medial temporal lobe. Science 333(6043):773–776

Nelson M, Rinzel J (1995) The Hodgkin-Huxley model. Bower and Beeman. The book of Genesis. Springer, New York, pp 27–51

N'Guyen S, Thurat C, Girard B (2014) Saccade learning with concurrent cortical and subcortical basal ganglia loops. Front Comput Neurosci 8:48

Niv Y, Daw N, Joel D, Dayan P (2007) Tonic dopamine: opportunity costs and the control of response vigor. Psychopharmacology (Berl) 191(3):507–520

Oja M, Kaski S, Kohonen T (2003) Bibliography of self-organizing map papers: 1998–2001 addendum. Neural Comput Surv 3:1–156

O'Keefe J, Dostrovsky J (1971) The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. Brain Res 34(1):171–175

O'Keefe J, Nadel N (1978) The hippocampus as a cognitive map. Clarendon Press, Oxford

O'Reilly RC, Frank MJ (2006) Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. Neural Comput 18(2):283–328

O'Reilly RC, Rudy JW (2000) Computational principles of learning in the neocortex and hippocampus. Hippocampus 10(4):389–397

O'Reilly RC, Herd SA, Pauli WM (2010) Computational models of cognitive control. Curr Opin Neurobiol 20(2):257–261

Palminteri S, Khamassi M, Joffily M, Coricelli G (2015) Medial prefrontal cortex and the adaptive regulation of reinforcement learning parameters. Nat Commun 6:8096

Pearlmutter BA (1995) Gradient calculations for dynamic recurrent neural networks: a survey. IEEE Trans Neural Netw 6(5):1212–1228

Penfield W, Boldrey E (1937) Somatic motor and sensory representation in the cerebral cortex as studied by electrical stimulation. Brain 60:389–443

Pfeiffer BE, Foster DJ (2013) Hippocampal place-cell sequences depict future paths to remembered goals. Nature 497(7447):74–79

Pfeifer R, Bongard J, Grand S (2007) How the body shapes the way we think: a new view of intelligence. Bradford Books, MIT Press
Prescott T, Redgrave P, Gurney K (1999) Layered control architectures in robots and vertebrates. Adapt Behav 7:99–127
Quirk GJ, Muller RU, Kubie JL, Ranck J (1992) The positional firing properties of medial entorhinal neurons: description and comparison with hippocampal place cells. J Neurosci 12(5):1945–1963
Redgrave P, Prescott T, Gurney K (1999) The basal ganglia: a vertebrate solution to the selection problem? Neuroscience 89(4):1009–1023
Redish AD, Touretzky DS (1997) Cognitive maps beyond the hippocampus. Hippocampus 7(1):15–35
Renaudo E, Girard B, Chatila R, Khamassi M (2014) Design of a control architecture for habit learning in robots. In: Duff A, Lepora N, Mura A, Prescott T, Verschure P (eds) Biomimetic and biohybrid systems, third international conference, living machines 2014, pp 249–260
Renaudo E, Girard B, Chatila R, Khamassi M (2015) Respective advantages and disadvantages of model-based and model-free reinforcement learning in a robotics neuro-inspired cognitive architecture. Procedia Comput Sci 71:178–184
Rescorla R, Wagner A (1972) A theory of pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement. In: Black A, Prokasy W (eds) Classical conditioning II: current research and theory. Appleton-Century-Crofts, New-York, pp 64–99
Revel A, Gaussier P, Lepretre S, Banquet J (1998) Planification versus sensory-motor conditioning: what are the issues ? In SAB' 98: From animals to animats 5, pp 129–138
Reynolds J, Hyland B, Wickens J (2001) A cellular mechanism of reward-related learning. Nature 413(6851):67–70
Rigotti M, Barak O, Warden MR, Wang XJ, Daw ND, Miller EK, Fusi S (2013) The importance of mixed selectivity in complex cognitive tasks. Nature 497(7451):585–590
Roesch M, Calu D, Schoenbaum G (2007) Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. Nat Neurosci 10(12):1615–1624
Rolls ET, O'Mara SM (1995) View-responsive neurons in the primate hippocampal complex. Hippocampus 5(5):409–424
Rougier NP, Boniface Y (2011) Dynamic self-organising map. Neurocomputing 74(11):1840–1847
Russell S, Norvig P (2003) Artificial intelligence - a modern approach. Prentice-Hall, Upper Saddle River, New Jersey
Samsonovich A, McNaughton BL (1997) Path integration and cognitive mapping in a continuous attractor neural network model. J Neurosci 17:5900–5920
Schaul T, Quan J, Antonoglou I, Silver D (2015) Prioritized experience replay. arXiv:1511.05952
Schmajuk N (1991) A neural network approach to hippocampal function in classical conditioning. Behav Neurosci 105(1):82–110
Schmajuk N, Thieme A (1992) Purposive behavior and cognitive mapping: a neural network model. Biol Cybern 67:165–174
Schultz W, Dayan P, Montague P (1997) A neural substrate of prediction and reward. Science 275(5306):1593–1599
Schwartz EL (1990) Computational neuroscience. MIT Press, Cambridge
Scoville WB, Milner B (1957) Loss of recent memory after bilateral hippocampal lesions. J Neurol Neurosurg Psychiatry 20(1):11
Searle JR (1980) Minds, brains, and programs. Behav Brain Sci 3:417–457
Servan-Schreiber D, Printz H, Cohen J (1990) A network model of catecholamine effects: gain, signal-to-noise ratio, and behavior. Science 249(4971):892–895
Sharp PE (1999) Comparison of the timing of hippocampal and subicular spatial signals: implications for path integration. Hippocampus 9(2):158–172
Stephenson-Jones M, Samuelsson E, Ericsson J, Robertson B, Grillner S (2011) Evolutionary conservation of the basal ganglia as a common vertebrate mechanism for action selection. Curr Biol 21(13):1081–1091

Sun R, Alexandre F (1997) Connectionist-symbolic integration: from unified to hybrid approaches. Erlbaum Associates, Lawrence
Sutton R, Barto A (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
Tessier-Lavigne M, Goodman CS (1996) The molecular biology of axon guidance. Science 274:1123–1133
Teyler TJ, DiScenna P (1986) The hippocampal memory indexing theory. Behav Neurosci 100(2):147
Thompson RF (1986) The neurobiology of learning and memory. Science 233(4767):941–947
Tolman E (1948) Cognitive maps in rats and men. Psychol Rev 55(4):189–208
Touretzky DS, Redish AD (1996) Theory of rodent navigation based on interacting representations of space. Hippocampus 6(3):247–270
Treves A, Rolls E (1994) Computational analysis of the role of the hippocampus in memory. Hippocampus 4(3):374–391
Uylings H, Groenewegen H, Kolb B (2003) Do rats have a prefrontal cortex? Behav Brain Res 146(1–2):3–17
van der Meer M, Kurth-Nelson Z, Redish AD (2012) Information processing in decision-making systems. Neurosci 18(4):342–359
Voorn P, Vanderschuren L, Groenewegen H, Robbins T, Pennartz C (2004) Putting a spin on the dorsal-ventral divide of the striatum. Trends Neurosci 27(8):468–474
Wan H, Touretzky D, Redish A (1994) Towards a computational theory of rat navigation. In: Mozer M, Smolensky P, Touretzky D, Elman J, Weigend A (eds) Proceedings of the 1993 connectionist models summer school, pp 11–19. Lawrence Erlbaum Associates
Wang L, Li X, Hsiao SS, Lenz FA, Bodner M, Zhou YD, Fuster JM (2015) Differential roles of delay-period neural activity in the monkey dorsolateral prefrontal cortex in visual-haptic crossmodal working memory. Proc Natl Acad Sci USA 112(2):E214–219
Wang JX, Kurth-Nelson Z, Tirumala D, Soyer H, Leibo JZ, Munos R, Blundell C, Kumaran D, Botvinick M (2016) Learning to reinforcement learn. arXiv:1611.05763
Willshaw DJ, von der Malsburg C (1976) How patterned neural connections can be set up by self-organization. Proc R Soc Lond B 194:431–445
Yamins DLK, DiCarlo JJ (2016) Using goal-driven deep learning models to understand sensory cortex. Nat Neurosci 19(3):356–365
Yin H, Knowlton B (2006) The role of the basal ganglia in habit formation. Nat Rev Neurosci 7(6):464–476
Zipser D (1985) A computational model of hippocampal place fields. Behav Neurosci 99(5):1006–1018

Artificial Intelligence and Pattern Recognition, Vision, Learning

Isabelle Bloch, Régis Clouard, Marinette Revenu and Olivier Sigaud

Abstract This chapter describes a few problems and methods combining artificial intelligence, pattern recognition, computer vision and learning. The intersection between these domains is growing and gaining importance, as illustrated in this chapter with a few examples. The first one deals with knowledge guided image understanding and structural recognition of shapes and objects in images. The second example deals with code supervision, which allows designing specific applications by exploiting existing algorithms in image processing, focusing on the formulation of processing objectives. Finally, the third example shows how different theoretical frameworks and methods for learning can be associated with the constraints inherent to the domain of robotics.

1 Introduction

The intersection between the domains of artificial intelligence (AI), and of pattern recognition, computer vision and robotics is getting more and more important and visible. The overlap between these domains was significantly enlarged during the last years. The objective of this chapter is to show a few aspects of this overlap, in particular for high level visual scene understanding and for integrating knowledge in processing and interpretation methods.

I. Bloch (✉)
LTCI, Télécom ParisTech, Université Paris Saclay, Paris, France
e-mail: isabelle.bloch@telecom-paristech.fr

R. Clouard · M. Revenu
GREYC, Normandie Univ., ENSICAEN, CNRS, Caen, France
e-mail: regis.clouard@ensicaen.fr

M. Revenu
e-mail: marinette.revenu@ensicaen.fr

O. Sigaud
ISIR, CNRS, Sorbonne Université, Paris, France
e-mail: sigaud@isir.upmc.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_10

Several topics addressed in other chapters and several of the therein described methods can also be associated with problems in pattern recognition, artificial vision or image understanding, and robotics. For instance, uncertainty theories are widely used for modelling imperfections of data, of objectives and of reasoning procedures, as for image fusion. Learning is at the core of many recent developments, such as for image mining or for robotics. Multi-agents systems have been exploited for developing cooperation between methods in image processing, as well as for developing interactions between or with robots. Finally, as a last example, structural representations (graphs, hypergraphs, Bayesian networks, ontologies, knowledge based systems…) are naturally used for modelling and interpreting image or video content. They allow associating low level information with higher level one and with knowledge, to guide the interpretation of the observed scene. This is for instance the case in spatial reasoning (see also chapter "Qualitative Reasoning About Time and Space" of Volume 1).

In this chapter, we describe a few examples of these multiple interactions. In Sect. 2, an overview of interactions between artificial intelligence and computer vision is proposed, in particular for recognizing objects in images, focusing on knowledge based systems. While ontologies are more and more developed to guide scene understanding, by describing and formalizing concepts related the scene contents, they are also exploited to describe the objective of image processing. In this perspective, Sect. 3 presents code supervision methods for automatically generating applications in image processing. Finally, in Sect. 4, the domain of robotics is presented under the light of related learning aspects.

2 AI for Computer Vision and Pattern or Object Recognition

In this section, an overview of interactions between AI and computer vision is proposed, focusing on knowledge based systems for image and visual scene understanding, pattern or shape recognition in images. The general objective of these approaches is to add semantics to the images, by associating visual information and features extracted from the images on the one hand, and knowledge or models on the other hand (Crevier and Lepage 1997; Le Ber and Napoli 2002).

One of the main difficulties, beyond knowledge representation and reasoning issues, is to establish a matching between perceptual and conceptual levels. The perceptual level includes features extracted from images, hence close to pixel (in 2D) or voxel (in 3D) information. The conceptual level is often given in textual form. This problem of matching is known as semantic gap, defined by Smeulders et al. (2000) as: "*the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation*". This problem is close to other problems in AI and robotics, such as symbol grounding or anchoring (Harnad 1990; Coradeschi and Saffiotti 1999).

2.1 Knowledge

The type of knowledge modelled in knowledge based systems is related to the scene and anything that can be useful for its interpretation. According to the classical categorization of Matsuyama and Hwang (1990), the following types are distinguished:

- generic knowledge on the type of scene, describing the objects it contains or may contain, the relationships between these objects, or the type of image;
- specific knowledge about the image, including the observation of the scene and its processing, which is required to extract useful information from images;
- knowledge bridging the semantic gap between a real scene and its observations as images.

2.2 Spatial Relations

Knowledge about space, in particular about spatial relations, is very important for image understanding (Bloch 2005; Kuipers and Levitt 1988). Indeed, human beings use intensively spatial relations for describing, detecting and recognizing objects. They allow solving ambiguities between objects of similar shape or appearance, based on their spatial arrangement, and are often more stable than characteristics of objects themselves. This is for instance the case of anatomical structures, as illustrated later in this chapter.

Spatial reasoning has raised a lot of attention in computer vision and pattern recognition, in artificial intelligence, in cognitive sciences, in mobile robotics, or in geographical information systems. According to the semantic hierarchy proposed by Kuipers and Levitt (1988), important spatial relations can be grouped into topological and metrical relations. Among the metrical relations, directional and distance relations can be distinguished, as well as more complex relations such as "between" or "along".

In the domain of qualitative spatial reasoning, most representation models are symbolic, often relying on logical formalisms, and mostly deal with topological (Vieu 1997) or cardinal (directional) (Ligozat 1998) relations (see chapter "Qualitative Reasoning about Time and Space" of Volume 1). To reason on real data such as images, quantitative or semi-quantitative formalisms are more expressive. For instance, fuzzy models of numerous spatial relations have been proposed (Bloch 2005). They are appropriate to address the issue of the semantic gap, for instance using the concept of linguistic variable, the semantic of each linguistic value being given by a fuzzy set in the concrete domain of the variable. As an example, the fuzzy representation of a concept such as "close to" allows representing the imprecision inherent to this concept, and instantiating its semantics according to the considered application domain (Hudelot et al. 2008). It also allows answering two main questions in structural image understanding:

- to which degree is a spatial relation satisfied between two given objects?
- what is the area of space in which a spatial relation to a reference object is satisfied (up to some degree)?

Among such fuzzy models of spatial relations, those relying on mathematical morphology offer a unified representation framework, able to handle purely quantitative, purely qualitative, as well as semi-quantitative or fuzzy representations (Bloch 2006).

2.3 Knowledge Representation and Organization

As in other domains, in vision and pattern recognition one may characterize knowledge representation by:

- the definition of a representation as a set of syntactic and semantic conventions for describing a knowledge element;
- logical representations, with a level of expressivity depending on the logic;
- compact representations, where only relevant properties and characteristics are explicitly represented;
- easy manipulation;
- explicit representation of what is useful for reasoning.

Since most data in the domain of computer vision and pattern recognition are numerical, using logical representations (which are often more compact than numerical ones) requires to convert such data in a symbolic form.

Requirements for symbolic representations are ontological, epistemic and computational. The first two levels impose constraints on the representation language, and the third level on the inference mechanisms.

Recent knowledge based systems can be seen as extensions of classical expert systems, by providing different ways for knowledge representation and reasoning. A few classical examples include:

- production rules, which are easy to adapt or extend, and their results can be explained; however expressivity highly depends on the involved logics;
- frames (Minsky 1974), which are declarative systems well adapted to describe objects classes based on their attributes and properties; hierarchical links allow handling different levels of granularity, with inheritance, specialization or generalization mechanisms; an example in image processing can be found in the methods proposed by Clément and Thonnat (1993);
- semantic networks (Quillian 1967), which rely on a graphical representation of a knowledge base, in which vertices represent concepts and objects, and edges represent relations; inference rules exploit inheritance from a class of objects to a more specific class; their representation as attributed relational graphs is often used to model spatial information;

- conceptual graphs (Sowa 1984; Chein and Mugnier 2008), which represent concepts and relations as vertices, linked by edges; again graphical representations are computationally efficient;
- ontologies and description logics, which provide a shared, consistent conceptual formalization of knowledge in a given domain (Gruber 1993) (see also chapter "Reasoning with Ontologies" of Volume 1).

In computer vision and image processing, where the environment is only partially known, early applications of knowledge based systems have been developed for program supervision (Clément and Thonnat 1993; Nazif and Levine 1984) and for image understanding (Desachy 1990; Hanson and Rieseman 1978; Matsuyama 1986; McKeown et al. 1985). Specific problems related to focalization of attention, adaptation of procedures to revise, repair or maintain consistency, cooperation and fusion, coordination could also be added to knowledge based systems (Garbay 2001). A renewed interest led recently to several works in these areas.

For instance, recent works use ontologies to add a semantic level and to solve the semantic gap problem. For instance in the methods proposed by Town (2006), the terms of a query language are anchored in the image domain using supervised learning, for application to keyword based image mining. A similar approach was used by Mezaris and Kompatsiaris (2004) and Hudelot (2005), who defined an ontology of visual concepts, anchored to descriptors extracted from the images. This type of approach allows both performing queries in a qualitative way based on the ontology concepts, and filtering or selecting relevant results according to their visual features.

Reasoning procedures associated with these different types of representations depend on the involved logic. One of the difficult problems to be solved is the matching between a knowledge model and information extracted from images, because of the semantic gap. This problem is simplified when information is directly linked to object representations (Saathoff and Staab 2008; Benz et al. 2004). Otherwise, for instance when only an over-segmentation of the image is available (i.e. several regions should be merged to be interpreted as an object), methods such as inexact graph matching, constraint satisfaction or spatial reasoning have to be developed (Perchant and Bloch 2002; Bengoetxea et al. 2002; Deruyver and Hodé 1997, 2009; Colliot et al. 2006; Fouquier et al. 2012; Nempont et al. 2013; Atif et al. 2013).

2.4 Uncertainty

In image understanding and computer vision, one has to deal with imperfect information. These imperfections are of different natures, and include ambiguity, bias, noise, incompleteness, imprecision, uncertainty, inconsistency, conflict…Additionally, when dealing with dynamic scenes, the information can be variable and evolves during time. These imperfections, found similarly in different problems in general information processing (Dubois and Prade 2001), may be due to the observed phenomenon itself, limitations of sensors, image reconstruction and processing methods

and algorithms, noise, lack of fiability, representation models, knowledge and concepts that are handled.

It is of high importance to account for these imperfections in representation models and in reasoning methods.

The main numerical models used in image processing and understanding to model uncertainty rely on probability theory and statistics, belief functions, fuzzy sets and possibility theory. They were developed in particular in the domain of information fusion (Bloch 2008), where the combination of several sources of information aims at making better decision while coping with imperfections of information, but also to represent structural information such as spatial relations (Bloch 2005).

In probabilistic representations, the language is constituted by probability distributions on a given reference domain. They account rigorously for random and stochastic uncertainty, but not easily for other types of imperfections, from both semantic and formal point of view. Bayesian inference is often used in this framework.

Belief functions (or Dempster–Shafer theory (Shafer 1976)) rely on a language defining several functions (belief function, plausibility…) on the power set of the decision space. Such representations cope with both imprecision and uncertainty (including of subjective nature), with ignorance and incompleteness, and allow computing a degree of conflict between data or information sources. The well known Dempster orthogonal rule performs a conjunctive combination, while other rules propose different types of behaviour in the combination (Denœux 2008).

In fuzzy sets and possibility theory (Dubois and Prade 1980, 1988; Zadeh 1965), the language includes fuzzy sets defined on a domain, or possibility distributions. Qualitative, imprecise and vague information can be suitably represented. Inference relies on logical rules, and qualitative reasoning is available. The usefulness of fuzzy sets for information processing in image and vision can be found at several levels (Bloch 2003, 2006):

- the ability of fuzzy sets to represent spatial information in images along with its imprecision, at different levels (local, regional, global), and under different forms (ranging from purely quantitative to purely qualitative) and different levels of granularity;
- the possibility to represent heterogeneous information, either extracted from the images or derived from external knowledge (such as expert or generic knowledge about a domain or an applicative problem);
- the possibility to generalize to fuzzy sets many operations to handle spatial information;
- the flexibility of combination operators, useful to combine information of different natures in various situations.

More details about uncertainty representations can be found in chapters “Representations of Uncertainty in Artificial Intelligence: Probability and Possibility” and “Representations of Uncertainty in Artificial Intelligence: Beyond Probability and Possibility” of Volume 1.

These models have been integrated in the knowledge representation methods described above, including ontologies (Hudelot et al. 2008, 2010), for successful applications in image understanding.

2.5 *Example: Recognition of Brain Structures in 3D MRI*

The automatic interpretation of complex scenes such as the brain requires a model representing knowledge on the structures present in the scene. In the easiest situations, each object has a different appearance, and prior knowledge on it may be sufficient to detect and recognize the objects. However, this is not the case in magnetic resonance images (MRI) of the brain, since the appearance is not discriminative enough. Other properties such as the spatial arrangement of the structures is then very important and helpful.[1]

Brain anatomy is commonly described in a hierarchical fashion and can be formalized using ontologies, such as the Foundational Model of Anatomy (FMA) (Rosse and Mejino 2003). In addition, the spatial organization of the anatomical structures is a major component of linguistic descriptions of the brain anatomy (Hasboun 2005; Waxman 2000). The overall structure of the brain is quite stable, while the shapes and sizes of the individual structures are prone to substantial variability, and therefore it is relevant to include spatial relations in a model of the brain anatomy. This allows coping with anatomical variability and offering good generalization properties.

Graphs are often used to represent the structural information in image interpretation, where the vertices represent objects or image regions (and they may carry attributes such as their shapes, sizes, and colours or grey levels), and the edges carry the structural information, such as the spatial relations among objects, or radiometric contrasts between regions. Although this type of representation has become popular in the last 30 years (Conte et al. 2004), a number of open problems remain in its efficient implementation. In one type of approach, the graph is derived from the image itself, based on a preliminary segmentation into homogeneous regions, and the recognition problem is expressed as a graph matching problem between the image and model graphs, which is an annotation problem. However this scheme often requires solving complex combinatorial problems (Conte et al. 2004). These approaches assume a correct initial segmentation of the image. However, the segmentation problem is a known challenge in image processing, to which no universal solution exists. The segmentation is usually imperfect, and no isomorphism exists between the graphs being matched. An inexact matching must then be found, for instance by allowing several image regions to be assigned to one model vertex or by relaxing the notion of morphism to that of fuzzy morphism (Perchant and Bloch 2002; Cesar et al. 2005). For example, previous studies (Deruyver and Hodé 1997, 2009) employ an over-segmentation of the image, which is easier to obtain. A model

[1] This section is to a large part adapted from Nempont et al. (2013).

structure (i.e. a graph vertex) is then explicitly associated with a set of regions, and the recognition problem is expressed as a constraint satisfaction problem.

To deal with the difficulty of obtaining a relevant segmentation, the segmentation and recognition can also be performed simultaneously. For instance, in the methods proposed by Bloch et al. (2003), Colliot et al. (2006), the structures of interest are segmented and recognized sequentially, in a pre-calculated order (Fouquier et al. 2008, 2012). The structures that are easier to segment are considered first and adopted as reference objects. The spatial relations to these structures are encoded in the structural model and are used as constraints to guide the segmentation and recognition of other structures. This approach benefits from an ontological representation of anatomical knowledge and of fuzzy models of spatial relations, which establish the links between concepts and image space, thus addressing the semantic gap issue (Hudelot et al. 2008). Due to the sequential nature of the process, the errors are potentially propagated. Backtracking may then be needed, as proposed by Fouquier et al. (2012).

To overcome the problems raised by sequential approaches while avoiding the need for an initial segmentation, an original method was proposed by Nempont et al. (2013). It still employs a structural model, but solves the problem in a global fashion. A solution is the assignment of a spatial region to a model object, in a way that satisfies the constraints expressed in the model. A progressive reduction of the solution domain for all objects is achieved by excluding assignments that are inconsistent with the structural model. Constraint networks constitute an appropriate framework for both the formalization of the problem and the optimization (see chapter "Constraint Reasoning" of Volume 2 for constraint reasoning methods). An original feature of this approach is that the regions are not predetermined, but are instead constructed during the reduction process. The image segmentation and recognition algorithm therefore differs from an annotation procedure, and no prior segmentation of the image into meaningful or homogeneous regions is required. More precisely, a constraint network is constructed from the structural model, and a propagation algorithm is then designed to reduce the search space. Finally, an approximate solution is extracted from the reduced search space. This procedure is illustrated in Fig. 1, using the interpretation of a brain MRI as an example. The solution space for the left caudate nucleus CNl is derived from the constraint "CNl is exterior to the left lateral ventricle LVl". Once the propagation process terminates, the solution space is typically reduced substantially for all of the model structures. The final segmentation and recognition results can then be obtained using any segmentation method that is constrained by this solution space. An example of result in a pathological case is illustrated on one slice in Fig. 2.

This approach has been extended by Vanegas et al. (2016) to deal with complex relations, involving groups of objects, unknown numbers of instances of concepts in the images and fuzzy constraints, for applications in remote sensing image understanding.

A concluding message is that model based understanding is a growing research topic, at the cross-road of image processing, computer vision and pattern or object recognition on the one hand, and of artificial intelligence on the other hand. The association between generic structural models and specific information related to the

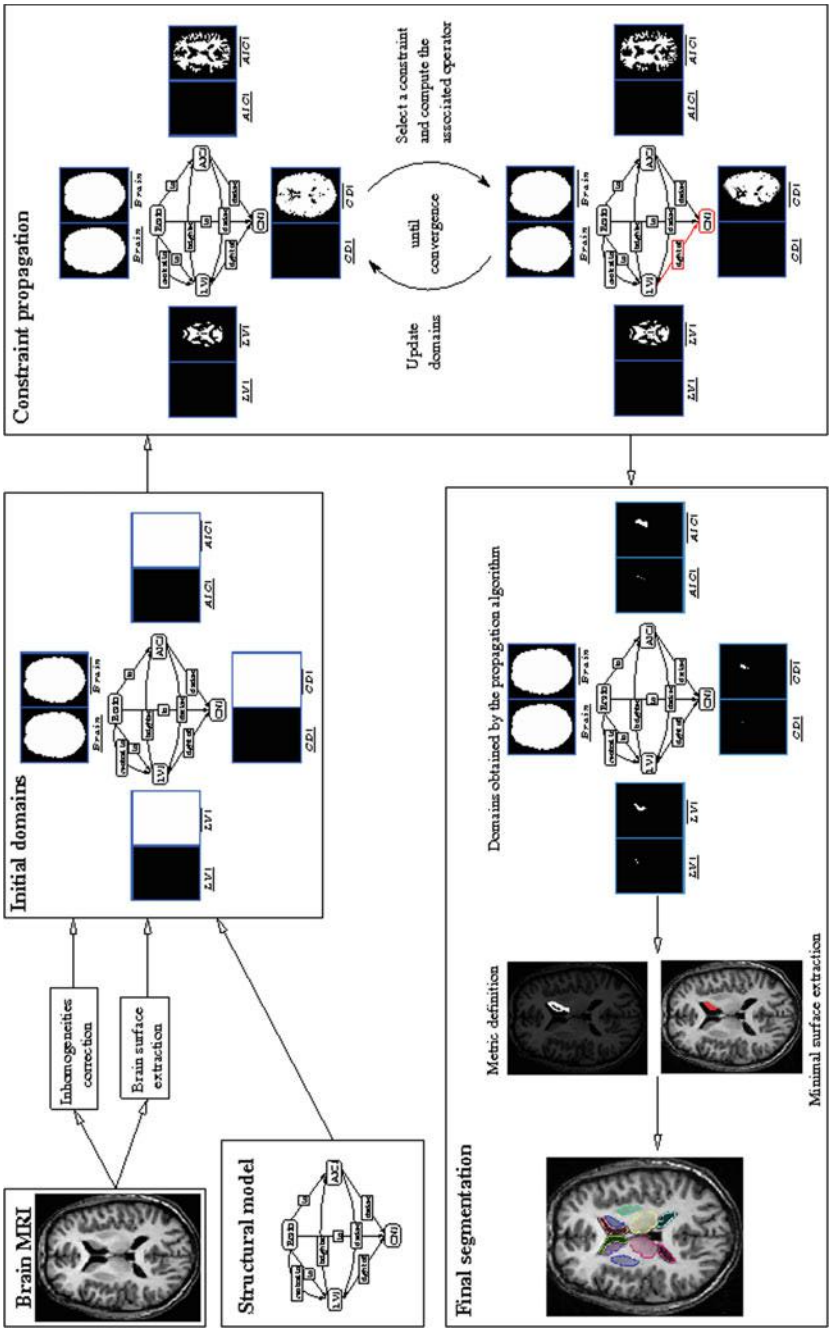


Fig. 1 Overview of the constraint propagation method for segmenting and recognizing brain structures in MRI (Nempont et al. 2013)

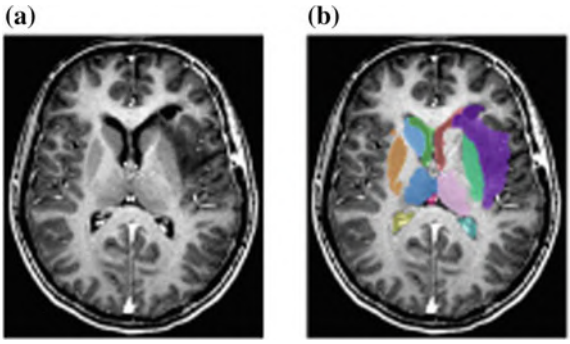


Fig. 2 **a** Axial slice of a 3D MRI of a patient with a brain tumour. **b** Segmentation and recognition results for several internal brain structures (Nempont et al. 2013)

context, accounting for uncertainty and variability, allows one to cope with the semantic gap problem and to propose computationally efficient methods to solve it. These approaches are currently further developed for image and video annotation, segmentation and recognition of structures, spatial reasoning for image exploration, or the derivation of high level descriptions of the content of images or image sequences.

3 Code Supervision for Automatic Image Processing

The need for automatic image analysis software is becoming increasingly pressing as digital image emerges as a privileged source of information. Acquisition devices now provide access to previously unknown or inaccessible data that are of strategic importance in many fields such as medicine, security, quality control, astronomy, environmental protection. However, the multiplicity of these devices leads to the production of an ever-expanding volume of data that is impossible to exploit manually.

Image processing is a preliminary stage that aims to prepare the images for subsequent analysis by humans or interpretation systems. It covers all objectives of image-to-image transformation that are intended to reduce, refine or organize the initial data. Five image processing objectives are usually distinguished: data *compression*, *enhancement* of visual rendering, *restoration* of missing information, *reconstruction* of spatio-temporal information (3D or motion), *segmentation* into more abstract primitives (regions or contours) and *detection* of known objects. Image processing has no decision-making power, but its role is crucial since it must ensure that changes on images are made without loss or alteration of the relevant information.

Image processing research traditionally provides its expertise in the form of image processing algorithms. Many algorithms covering a wide range of operations have been developed. Each algorithm is developed on a presupposed model of information to be processed, which determines its domain of applicability and effectiveness. Therefore, there is no universal algorithm. A concrete application should combine several of these algorithms according to a top-down, bottom-up or mixed processing

strategy. Thus, the development consists in selecting, tuning and linking appropriate algorithms. However, appropriate use of image processing algorithm libraries requires highly specialized expertise to know when and how to utilize the algorithms.

Code supervision systems are designed to provide users with a tool to build their own applications by exploiting a library of precoded algorithms. Users no longer need to be experts in image processing. Their role is focused on the formulation of application objectives. It is the system responsibility to control the code library for building programs suited to the application objectives.

3.1 Formulation of Application Objectives

The formulation of application objectives is of paramount importance because it is used by the system to guide selection, tuning and chaining of codes. Two categories of information should be given by users for an exhaustive formulation:

1. The *definition of the image class* is required to bridge the sensory and semantic gaps (Smeulders et al. 2000) (see Fig. 3). Part of the definition should describe the image acquisition process in order to restore information about the observed scene that were lost, altered or hidden during the image production. Another part should assign a semantics to the scene content in order to specify information that has to be considered as relevant for that precise application.
2. The *specification of the processing goals* is required to clarify the role of the application in the complete analysis system.

Image Class Definition

Various models of image class definition have been proposed in the literature whether the definition is done by extension or by intension.

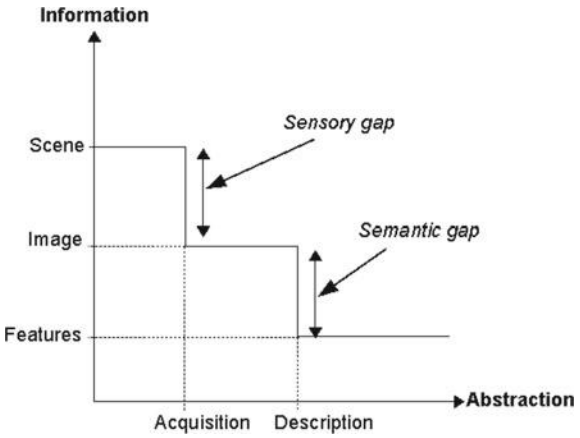


Fig. 3 The sensory gap results from the loss of information between the reality of a scene and its representation as an image. The semantic gap separates the interpretation of a scene that anyone can make from an image-based representation and from a feature-based description (Smeulders et al. 2000)

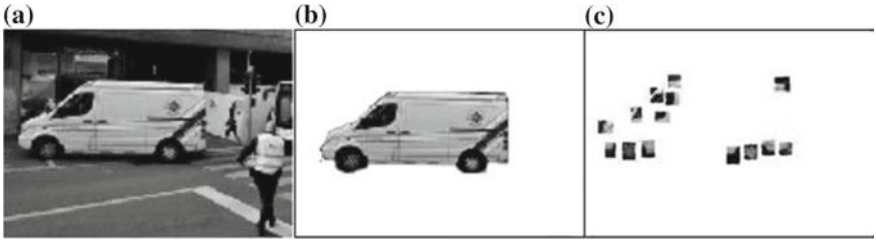


Fig. 4 Two ways to extensionally describe the vehicle in figure **a**: **b** by a mask that specifies the object pixels, **c** by a list of patches around points of interest

An extensional definition represents information using an iconic dictionary built with image parts. These parts can be specified either by *masks* or by *patches*. A mask delineates an object of interest or a specific image area (see example in Fig. 4b.). They are used by the system to automatically extract a set of feature values of the specified object (colour, shape, size, etc.) or a set of image characteristics of the specified area (type of acquisition noise, illumination distribution, etc.). A patch is a thumbnail extracted from a sample image that isolates one salient part of an object of interest (often localized around a point of interest as shown in Fig. 4c). They are used by the system to detect instances of these objects in images from their characteristic parts (Agarwal et al. 2004; Leibe et al. 2008). The benefit of extensional definition is to limit the cognitive load of users since no representation language is required. The drawback is that the same feature extraction or patch selection algorithms are used for all applications. Thus, a part of the application definition is assigned by the system and cannot be adapted to each application.

An intensional definition represents information about images using a linguistic description. It provides a language to represent the acquisition effect and the scene content semantics. Ontologies are widely used for this purpose (Hunter 2001; Bloehdorn et al. 2005; Town 2006; Renouf et al. 2007; Anouncia and Saravanan 2007; Maillot and Thonnat 2008; Neumann and Möller 2008; Gurevich et al. 2009). The description language is usually constructed from an *ontology domain* that provides the language primitives. The description of a particular image class is an *application ontology* that is obtained by selection and reification of domain ontology primitives (Cãmara 2001). For example, Maillot and Thonnat (2008) propose the "Ontology of Visual Concepts" which defines the concepts of texture, colour, geometry and topological relations. Figure 5 gives a textual representation of the definition of a pollen grain with this ontology. To better reflect the variability of the visual manifestations of the objects in the scene, the language accepts qualitative values for the features such as ("*pink*", "*very circular*", "*strongly oblong*") and for the spatial relations such as ("*in front of*", "*close to*"). The advantage of this definition is to take greater advantage of the user's expertise about scene content and thus better capture application variability. However, the construction of the solution requires quantitative values. Therefore, intensional definition must address the problem of symbol grounding in order to connect linguistic symbols to image data values. Sym-

Fig. 5 Textual representation of the definition of a pollen grain of type "poaceae" from the "Ontology of Visual Concepts" proposed by Maillot and Thonnat (2008)

bol grounding can be based on dictionaries such as the "Colour Naming System" (Berk et al. 1982) where the HSL space is divided into 627 distinct colours, each of them labelled with a name, or the "Texture Naming System dictionary" (Rao and Lohse 1993). However, most often symbol grounding is seen as a learning problem from a set of masks. Therefore, usually mixed approaches are preferred. Intensional definition is completed with extensional definition that allows anchoring ontology concepts into data (Maillot and Thonnat 2008; Hudelot et al. 2008; Clouard et al. 2010).

Goal Specification

The specification of application goals can be made either by examples of the expected results or by tasks to perform.

According to specification by example, a goal is formulated through reference images containing the representation of the results to be obtained on test images. Three different representations of the expected results have been proposed in the literature:

- *Sketches* are lines drawn by the user on test images that give examples of the expected contours or regions boundaries (Draper et al. 1999), as in Fig. 6a.
- *Manual segmentations* give the region areas to be obtained on test images (Martin et al. 2006), as in the example in Fig. 6b.
- *Scribbles* are markers that indicate regions of interest without completely delineate them (Protire and Sapiro 2007). Generally, scribbles are lines drawn directly inside the regions of interest and inside the background region, as in Fig. 6c.

The advantage of the specification by example paradigm is its quantitative nature since it takes values directly into the image data. In addition, it reduces the cognitive load of users because no specialized vocabulary is required. The drawback is that a reference image is not sufficient to formulate all kinds of goals. Only segmentation, detection and possibly enhancement goals are really addressed. Compression, restoration and reconstruction goals are not straightforward. Moreover, it does not cover all image classes. In particular, it is tedious to implement for 3D images and image sequences. Finally, there is no means for varying constraints attached to goals, such as "*prefer false detection to misdetection*" or "*prefer no result to imperfect result*".

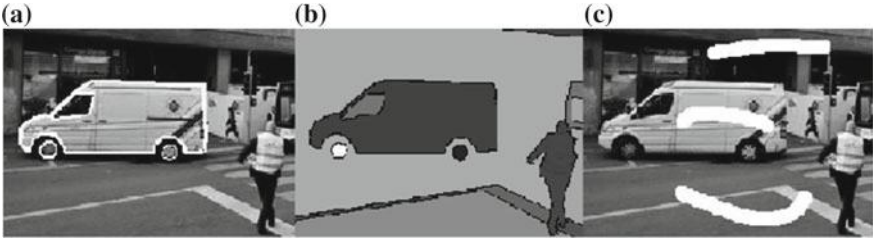


Fig. 6 Three different approaches to specify a goal by example: **a** by sketch, **b** by manual segmentation, **c** by scribbles

The specification by task paradigm requires a language. A task describes a system functionality by means of a sentence, such as "detect object vehicle" or "segment the image". The advantage of this approach is that it is possible to associate constraints to the task in order to restrict its scope. Moreover, all image processing objectives can be covered: it is sufficient to name a task and related constraints. The drawback is that the formulation is qualitative with no real link to the image data. This has two important consequences: first, specification by task is not strongly grounded into data, and secondly, there is only a finite number of possible objective formulations. That is why recent approaches use mixed approaches that combine specification by task and specification by example paradigms. Figure 7 presents an ontology (Clouard et al. 2010) that covers the definition of the image class by mixing intensional and extensional approaches and specifying goals by mixing approaches by task and by example.

3.2 Code Supervision

The formulation of application objectives is the prerequisite for the development of a solution as a processing chain. In the paradigm of code supervision (Thonnat and Moisan 2000), image processing techniques are implemented as independent executable codes and stored in a library. An image processing program is represented in canonical form as a directed graph of codes. Links between codes describe network of images and parameter values exchanged between codes. For example, Fig. 8 shows a processing chain that performs edge detection by difference of two Gaussians.

The problem of code supervision was addressed in several ways in the literature of which the most advanced are:

- competitive strategy;
- plan skeleton instantiation;
- case-based reasoning;
- chain planning;
- incremental result construction.

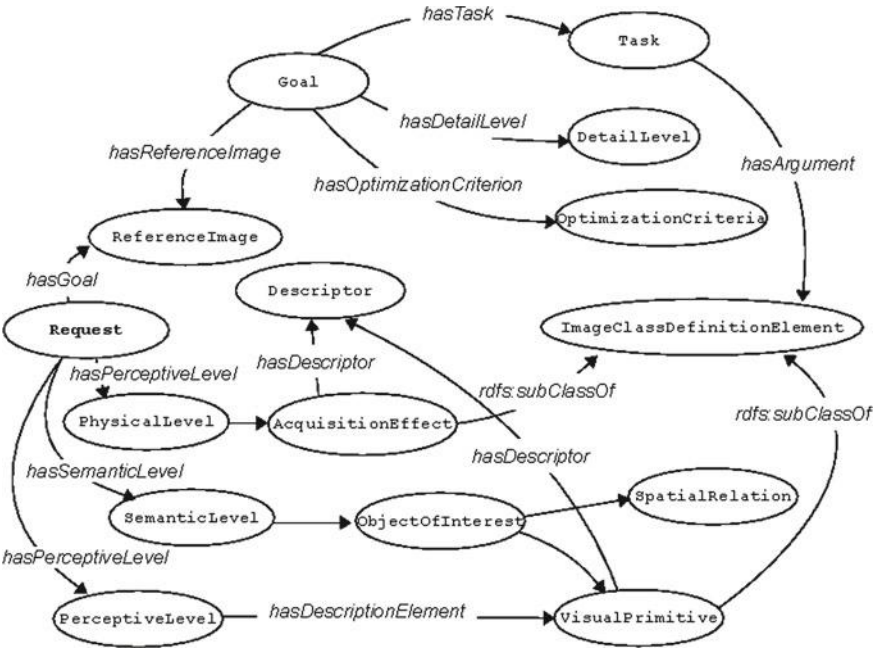


Fig. 7 The concepts of an ontology for formulating image processing goals (Clouard et al. 2010)


```
pexponentialfiltering 0.2 image1 image2
pexponentialfiltering 0.8 image1 image3
psub image2 image3 image4
pzerocross 8 image4 result
```

Fig. 8 A program is a graph of parametrized executable codes. On the left is given the representation of an edge detection algorithm using the DOG (Difference of Gaussian) in the form of a code graph. On the right, the same algorithm is represented as a script of executable codes

Competitive Strategy

The main idea behind this approach is to exploit the competition between several predefined processing strategies. For example, Charroux and Philipp (1995) execute several image segmentation chains in parallel, and then build the final result with the best segmented regions yielded by each of these chains. The quality of a region is measured by its degree of membership to domain object classes, calculated by a classifier trained to recognize the domain object classes from masks made on sample images.

Martin et al. (2006) create competition between multiple image segmentation chains and then select the best chain with the best settings. The selection is made off-line through supervised learning where a set of sample images with related handmade reference segmentation is used to train the classifier. The resulting chain, with its setting, is the one that minimizes the distance between the segmentation obtained on test images and the reference segmentation made for these images.

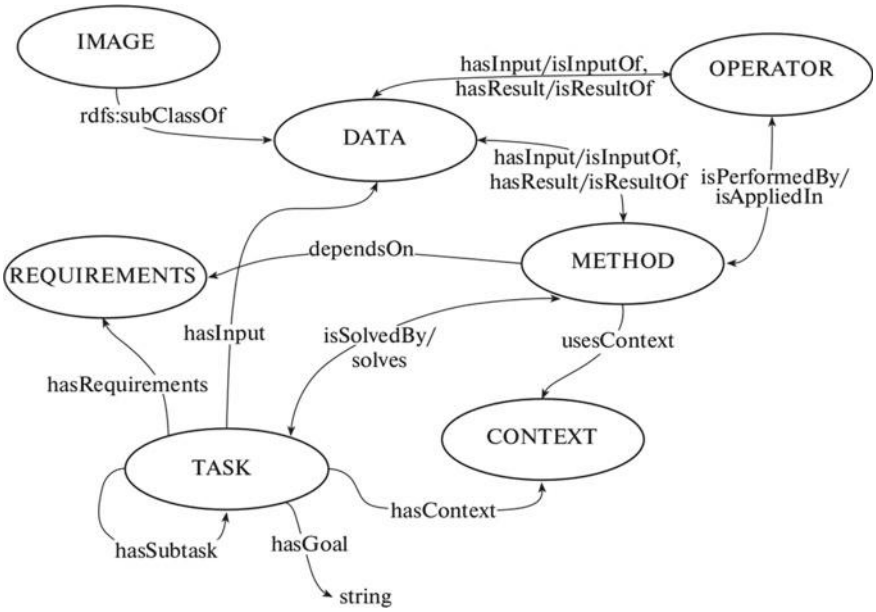


Fig. 9 Concepts and basic elements of an image processing ontology which specifies how to solve a task using operators with regard to a specific context (Gurevich et al. 2009)

The advantage of this approach is that it requires no explicit expertise. Only reference object masks or reference images must be provided. The drawback is that it relies on processing chains that are fixed and in finite number. Parameter tuning is the only possible adaptation.

Plan Skeleton Instantiation

This is certainly the approach that has generated the higher number of systems, with pioneering work such as: OCAPI (Clément and Thonnat 1993), VSDE (Bodington 1995), CONNY (Liedtke and Blömer 1992), COLLAGE (Lansky et al. 1995) or MVP (Chien and Mortensen 1996).

The processing expertise is encoded in hierarchical plan skeletons that combine along several decomposition levels a task corresponding to a problem with a set of codes that constitute elements of a possible chain of processing. Plan skeletons are encoded as AND/OR trees that indicate how a task can be decomposed into subtasks. Production rules are attached to each node. They are used to select the most appropriate branch of the decomposition and parameter values with regard to formulation elements.

Figure 9 presents an ontology that models the way to solve a task with a sequence of operators with regard to a specific context.

Compared to competitive strategy, this approach allows chain adjustment to the specifications given in the formulation of objectives. However, it requires knowing how to identify and represent the expertise for each possible problem type.

Case-Based Reasoning

Case-based reasoning exploits processing chains built successfully for past applications to process a new "similar" one.

In image processing, this approach has been used to build processing plans (Charlebois 1997; Ficet-Cauchard et al. 1999) or to find out convenient set of parameters to configure a general processing chain (Perner et al. 2005; Frucci et al. 2008). The reasoning is based on the analysis of the problem formulation to try to find a similar case. The retrieved case is then adapted to the context of the current problem. If there is no similar case, then a new case has to be learned and stored in the database.

Case-based reasoning does not require explicit representation of processing expertise. However, the critical point of this approach lies in the adaptation of cases to the particular context of the application that is of considerable importance in image processing regarding the high variability of images in a class.

Chain Planning

Unlike previous approaches which explicitly encode a set of processing chains, in chain planning the processing chains are built dynamically.

Systems using linear planning are based on modelling a type of expression that can be propagated along the processing chains. The reasoning is focused on the operations to be applied to the initial expression to build the expected final expression. The initial expression is the formulation provided by users in intensional or extensional form. In the latter form, expression is constructed by automatic extraction of features in sample images. The generation of chains can be combinatorial. In this case, each operator in the chain is modelled by a list of preconditions and a list of effects on the expression, as in the system EXTI (Dejean and Dalle 1996). But, the generation of chains can also be achieved by production rules attached to nodes that select the next operators according to the current expression, as in systems LLVE (Matsuyama 1989) and SOLUTION (Rost and Mnkel 1998).

The planning approach creates chains from scratch for each application. However, it faces the difficulty to model the problem as an expression that can be propagated along processing chains and especially the difficulty of having to a priori estimate the impact of operations on the expression. To improve planning efficiency, this problem has also been addressed using a hierarchical planning. The BORG system (Clouard et al. 1999) used a blackboard to build plans using multiple levels of abstraction. The initial goal formulated by the user is gradually divided into more and more precise subtasks until they correspond to executable codes. Knowledge sources encode various decomposition alternatives of a task to lower level subtasks. Figure 10 presents an example of construction of such a plan.

In all cases, the final application is the processing chain built operator by operator, which produces a standalone program. To limit the impact of choices made during the construction of chains, Draper et al. (1999), with the ADORE system, propose to keep all the alternative chains in the program. This system then uses a Markov decision process to dynamically choose the best path in these chains during the execution of the solution, from features automatically extracted from the processed image.

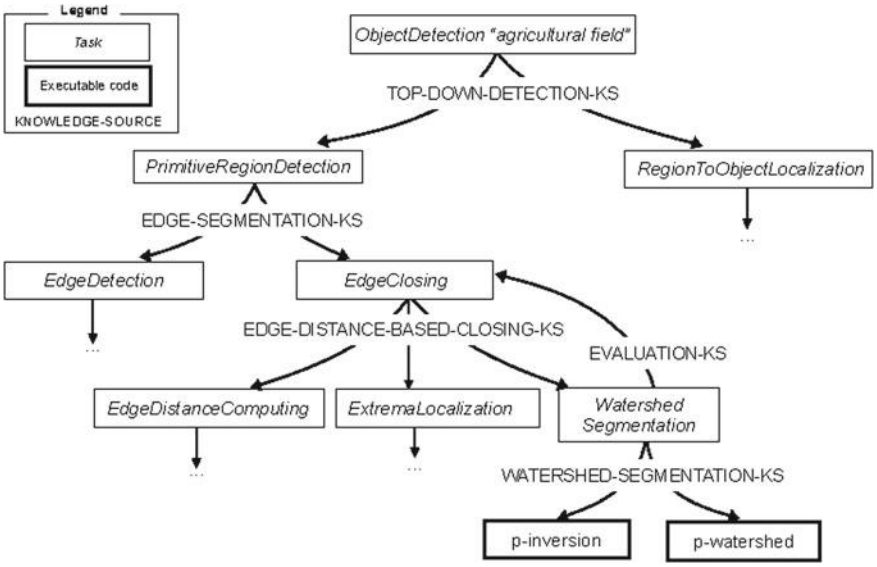


Fig. 10 Excerpt from a hierarchical processing plan for the detection of agricultural fields in aerial images (Clouard et al. 2010)

Incremental Result Construction

Incremental construction of results proceeds by gradual and controlled evolution of the input image to the desired output image. This approach can be seen as dual of the previous approaches in the sense that the reasoning is focused on the analysis of data produced after application of processing. The image processing algorithms are completely split into a set of production rules (Nazif and Levine 1984) or independent rational agents (Boucher et al. 1998; Bovemkamp et al. 2004). In such an approach, there is no explicit strategy of generation of processing chains. The reasoning remains focused on the analysis of the current state of the image after application of the first processing in order to determine the next processing to be applied in the case of production rules or resolve data access conflicts in the case of multi-agent.

The design of such systems requires a knowledge acquisition phase. Nevertheless, the decentralized control makes the acquisition of such knowledge easier, since it is not necessary for the knowledge engineer to explain the resolution strategies. However, the overall resolution process remains complex to master because convergence towards a solution is only guaranteed by the action of rules or agents that have only a local vision of their effects. Each rule or agent is responsible for estimating the value of its contribution compared to the current state of the resolution. This limit often requires adding abstraction levels in the hierarchy of rules or rational agents to have a more global vision of the resolution.

3.3 Conclusion

The challenge of the research on code supervision for automatic image processing and image analysis is to develop solutions that allow image consumers unskilled in image processing (e.g., geographers, biologists, librarians, special effect technicians) to design their own software alone. In shorter term, the goal is to build configurable systems that help vision engineers rapidly deploy dedicated applications without any programming activity.

Today, the results of these works are exploited in recent research in semantic image indexing, content-based image search and video analysis. These problems are also addressed using statistical methods with spectacular results for face detection or object recognition for example. They operate from the extensional definition of image classes using comparison with learned sample images. But these statistical methods are insufficient in cases of complex scenes or problems other than detection. In such situations, artificial intelligence methods and techniques are an undeniable asset. They cover a wider variety of applications and moreover they better take into account of the user needs. In this context, statistical methods are integrated as regular codes that can be used in specific cases.

However, the design of systems covering a wide range of applications with high efficiency remains a challenge. With this in mind, some current research work is directed towards the development of solutions based on human machine interaction, which emphasize collaboration to jointly converge towards building a suitable processing chain, each bringing its skills, the user's knowledge of the problem and the system knowledge of image processing.

4 Machine Learning for Robotics

Most industrial robots of the last century were used in highly structured and controlled environments such as assembly lines. All day long, they were realizing highly repetitive and specialized tasks without any room for uncertainty and away from human workers, mostly for security issues.

In the early 21st century, a new generation of robots is now emerging, whose employment context is fundamentally different (see also chapter "Robotics and Artificial Intelligence" in this volume). These so-called "personal" robots, whether they be food processors, playful companions or patient support, will have to perform extremely varied tasks in unknown changing environments, where uncertainty is omnipresent, and in direct contact with their users, who will not be experts in robotics. In this context, specifying in advance the behaviour of robots for any possible situation and for any possible task is no longer possible. The only reasonable alternative is to equip these versatile robots with abilities to learn and adapt to their environment.

While machine learning methods have been extensively developed in the last two decades, the robotic framework confronts these methods to specific constraints such

as the limited duration of the experiments, the often prohibitive cost of failures, the need to operate in real time or the large number of high-dimensional problems to be solved.

Therefore, no unifying theoretical framework has yet imposed itself to formalize the corresponding robot learning problems, and there are many attempts of varied natures to equip robots with learning abilities.

Part of the work is based on different theoretical machine learning frameworks (see chapters "Statistical Computational Learning" and "Reinforcement Learning" of Volume 1, and "Designing Algorithms for Machine Learning and Data Mining" of Volume 2): supervised learning, reinforcement learning, inductive learning, etc. to build tools specifically adapted to the robotic constraints.

Another part, which intersects significantly with the former, relies on understanding learning processes in biological systems to develop new methods inspired from these processes. This is the case of imitation learning, developmental robotics, evolutionary robotics, or various neuro-mimetic approaches to learning, for example. The intersection arises because these methods will eventually use machine learning tools designed within the first approach.

4.1 Machine Learning Methods and Robotics

Of all the approaches mentioned above, the one that provides the most obvious alternative for replacing direct programming of behaviour is imitation learning, also called learning by demonstration. This approach is relatively well developed and produced many significant results in recent years, through quite different methodological approaches. Some researchers use motion capture tools to record the movement of humans trying to perform a task in a particular context, then make sure that the robot performs the same movement in the same context. This last point requires to solve a problem known as the "correspondence problem" when the geometry, kinematics and dynamics of the human and the robot are significantly different, which is usually the case, except for a few humanoid robots. To avoid solving this correspondence problem, another approach consists in driving the robot through a remote operation system to make it realize the required movement once, and then to build on the recorded movement to perform it again and again. However, those two approaches pose a widespread problem: the circumstances being never exactly the same, the recorded movement is never perfectly adequate and the robot must adapt to these variations. For various syntheses or particularly outstanding work in the context of learning by imitation, we refer the reader to the work by Atkeson et al. (1997), Schaal (1999), Ijspeert et al. (2002), Calinon (2009), Coates et al. (2008), Ratliff et al. (2009).

Another approach that directly takes into consideration the need to generalize is to solve an "inverse reinforcement learning" (or inverse optimal control) problem. The idea is to consider a set of trajectories made by experts as optimal and extract the cost function that experts seem to have followed. Given the cost function, an optimization

algorithm can be used to generate the new robot movements that optimize the same cost function (Abbeel 2008).

Learning by imitation is not enough to solve all the problems posed by the need for robots that adapt to their environment. Indeed, in the general context of use described above, it is not possible to show the robot what should be its behaviour in all situations it would be likely to encounter. To go further, it is necessary that the robot is able to adapt its behaviour to unexpected situations. For this, one must still provide the robot with a capacity to assess the quality of its behaviour in a given situation, which can be done through a cost function. Learning how to improve one's behaviour by seeking to minimize a cost function (or maximize a performance function) is a problem that is formalized within the framework of reinforcement learning (Sutton and Barto 1998). The difficulty encountered in robotics to use reinforcement learning methods arises because these methods were originally developed in the problem solving context in which situations and actions are finite and limited, while in robotics problems are often continuous or very large. However, many recent algorithmic advances helped obtain increasingly significant results in this area (Stulp and Sigaud 2012).

Moreover, the command used for complex robots often uses kinematics, velocity kinematics and dynamics models of these robots, mainly for planning by determining the immediate response of the robot to a particular command. Identification is the activity of determining these models using a set of simple experiments that extract all relevant variables. The supervised learning methods that approximate functions from elementary data provide an interesting alternative to traditional parametric identification, to the extent that a robotic model is a function that can be estimated from the sensors of the robot. On the one hand, these methods require no a priori assumption on the shape of the models (Stulp and Sigaud 2015). Moreover, model learning can be performed during the robot operation, thus avoiding a tedious preliminary phase and, above all, allowing to immediately adapt the model in case of alteration of the robot or variation of the mechanical conditions of use. Though these supervised learning methods are still largely confined to learning robot models themselves (D'Souza et al. 2001; Salaun et al. 2010), they begin to tackle more original questions related to the interaction with a priori unknown objects (Vijayakumar et al. 2005), which falls within the more ambitious context of use that we described in the introduction to this section.

Robot learning finds its most compelling application context in the interaction between a robot and a human (Najar et al. 2015). Indeed, this context prominently requires rapid adaptation to a changing context from the robot and provides the framework within which imitation learning comes most naturally. Imitation is also a kind of human-robot interaction, allowing to consider the latter area as more general than the former. There are also research works that do not fit in previous frameworks, such as research on the social acceptability of behaviour of robots (Kruse 2010) or human-robot verbal interaction in a cooperation framework (Dominey 2007).

The human-robot interaction can be physical, when either of the protagonists exerts a force on the other. This is the case for example in the context of robotic assistance and rehabilitation, when it comes to helping patients with motor disorders (Saint-Bauzel et al. 2009). The implementation of learning technologies in this

context is a new trend (Pasqui et al. 2010). The interaction may also be simply communicative, whether through the spoken word or through other nonverbal methods (Dominey and Warneken 2009). The interaction may finally be fully implicit, when the human and the robot adapt their behaviour to each other without any communication, just by adjusting their behaviour to the behaviour observed in the other.

4.2 Bio-inspired Learning and Robotics

A second approach to learning in robotics is to attempt to replicate the learning mechanisms found in living beings. The goal is to endow robots with adaptive properties similar to those of animals or humans, which is far from the case today. Such an approach is likely to improve the development of adaptive mechanisms for robots. Furthermore, and vice versa, this approach is likely to contribute to progress in understanding the adaptation mechanisms of living beings, through validation or invalidation by robotics experiments (Guillot and Meyer 2008).

These bio-inspired approaches can take very different forms depending on the level at which the adaptation mechanisms are integrated. Indeed, living systems are characterized by a complex hierarchy of physiological and psychological processes at different scales, and adaptive mechanisms can be found at most of these levels, if not all.

Broadly speaking, there are two main research lines:

- the first finds its inspiration in psychological research about child development and is called "developmental robotics". It is mainly concerned with works modelling the cognitive learning abilities of babies and young children (Lungarella et al. 2003; Oudeyer et al. 2007; Quinton et al. 2008) and is particularly interested in solving the so-called "symbol grounding problem" that any artificial intelligence system is facing (Harnad 1990);
- the second is rather inspired from neuroscience research and proposes "neuro-mimetic" approaches, which can be clustered into two main families. The first is interested in decomposing the brain into distinct functional areas and proposes models whose components mimic the functions of these different areas. For example, one model the learning capabilities of rodents by building a neuro-mimetic model of the rat basal ganglia, which are deep nuclei of the brain which are believed to play a role in the evaluation of our behaviour (Doya 2000; Lesaint et al. 2014). The second focuses instead on the elementary computational properties of neurons, again at different levels, depending on whether one looks at the average activity of the neuron over time or at its propensity to issue elementary pulses according to a specific dynamics.

The central challenge that faces this general bio-inspired approach is due to the complex stack of integration levels. For a given adaptive phenomenon, it is sometimes difficult to determine whether a unique level of integration can account for the phenomenon, or whether the mechanisms from several levels should systematically

be combined. In this context, the robot proves an invaluable tool for the advance of knowledge in living sciences by providing a demanding experimental validation framework in which different theories can be analysed or compared.

4.3 Current Challenges

The desire to endow robots with learning ability is doubtlessly not new, but the corresponding research has substantially grown in recent years, with the emergence of many workshops dedicated to this topic in the main robotics conferences, the publication of numerous special issues in journals, or the growing number of dedicated summer schools. The result of this rapid growth is a burgeoning development in which many approaches are being developed in parallel in sometimes very different directions, often attacking very different problems. It seems that in the more or less close future, all of these searches should be structured and that new models combining different mechanisms should emerge from this abundance. A very recent and major evolution in robot learning results from the emergence of deep learning techniques (LeCun et al. 2015). The outstanding pattern recognition capabilities of these techniques and their focus on learning flexible representations from data opens new perspective on solving the symbol grounding problem in a developmental robotics perspective. But the methodological constraints of developmental robotics differ from those of standard pattern recognition challenges, thus the emergence of dedicated deep learning techniques is required with a potentially huge impact on robot learning (Sigaud and Droniou 2016).

5 Conclusion

In this chapter, far from being exhaustive, illustrations have shown convergence areas between artificial intelligence, computer vision, pattern recognition, learning and robotics. These convergences can be found in other domains, such as speech recognition and automatic natural language processing. Associating theories and methods from different domains is an ever growing approach, and leads to important developments and original research works. In image understanding, high level approaches use more and more intensively knowledge representation methods and reasoning services. For instance, abduction and revision, integrating learning and uncertainty models, can be used for image or video understanding (Atif et al. 2013) (see also chapters “Knowledge Representation: Modalities, Conditionals and Nonmonotonic Reasoning”, “Reasoning with Ontologies”, “Belief Revision, Belief Merging and Information Fusion”, “Multicriteria Decision Making” and “Decision Under Uncertainty” of Volume 1). In parallel to these model and knowledge based methods, a large field of research is now based on learning (and in particular deep learning) methods, with impressive results based on large training data sets (and without exploiting

knowledge) (LeCun et al. 2015; Vinyals et al. 2015) (see also chapters "Statistical Computational Learning" and "Reinforcement Learning" of Volume 1, and "Designing Algorithms for Machine Learning and Data Mining" of Volume 2). Man-machine interactions can also support new solutions, as mentioned for code supervision, but also for other domains, such as robotics. Finally, the multiplication of methods and models incite researchers to combine their advantages.

References

Abbeel P (2008) Apprenticeship learning and reinforcement learning with application to robotic control. PhD thesis, Stanford University, Computer Science

Agarwal S, Awan A, Roth D (2004) Learning to detect objects in images via a sparse, part-based representation. IEEE Trans Pattern Anal Mach Intell 26(11):1475–1490

Anouncia SM, Saravanan R (2007) Ontology-based process plan generation for image processing. Int J Metadata Semant Ontol 2(3):211–222

Atif J, Hudelot C, Bloch I (2013) Explanatory reasoning for image understanding using formal concept analysis and description logics. IEEE Trans Syst Man Cybern 44:552–570

Atkeson CG, Moore AW, Schaal S (1997) Locally weighted learning. Artif Intell Rev 11(1):11–73

Bengoetxea E, Larranaga P, Bloch I, Perchant A, Boeres C (2002) Inexact graph matching by means of estimation of distribution algorithms. Pattern Recognit 35(12):2867–2880

Benz U, Hofmann P, Willhauck G, Lingenfelder I, Heynen M (2004) Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information. ISPRS J Photogramm Remote Sens 58(3–4):239–258

Berk T, Brownston L, Kaufman A (1982) A new color-naming system for graphics languages. IEEE Comput Graph Appl 2(3):37–44

Bloch I (2003) Traitement d'images. In: Bouchon-Meunier B, Marsala C (eds) Traitement de données complexes et commande en logique floue (Chap. 3). Hermes, Paris, pp 95–152

Bloch I (2005) Fuzzy spatial relationships for image processing and interpretation: a review. Image Vis Comput 23(2):89–110

Bloch I (2006) Spatial reasoning under imprecision using fuzzy set theory, formal logics and mathematical morphology. Int J Approx Reason 41:77–95

Bloch I (2008) Information fusion in signal and image processing. ISTE-Wiley, London

Bloch I, Géraud T, Maître H (2003) Representation and fusion of heterogeneous fuzzy information in the 3D space for model-based structural recognition - application to 3D brain imaging. Artif Intell 148(1–2):141–175

Bloehdorn S, Petridis K, Saathoff C, Simou N, Tzouvaras V, Avrithis Y, Handschuh S, Kompatsiaris Y, Staab S, Strintzis M (2005) Semantic annotation of images and videos for multimedia analysis. Second European Semantic Web Conference (ESWC). Crete, Greece, pp 592–607

Bodington R (1995) A software environment for the automatic configuration of inspection systems. International Workshop on Knowledge Based Systems for the reUse of Program Libraries (KBUP). Sophia Antipolis, France, pp 100–108

Boucher A, Doisy A, Ronot X, Garbay C (1998) A society of goal-oriented agents for the analysis of living cells. Artif Intell Med 14(1–2):183–199

Bovemkamp E, Dijkstra J, Bosch J, Reiber J (2004) Multi-agent segmentation of IVUS images. Pattern Recognit 37:647–63

Calinon S (2009) Robot programming by demonstration: a probabilistic approach. EPFL/CRC Press, Lausanne

Cãmara G, Engenhofer M, Fonseca F, Monteiro A (2001) What's in an image? International Conference on Spatial Information Theory: Foundations of Geographic Information Science, Morro Bay, CA 2205:474–488
Cesar R, Bengoetxea E, Bloch I, Larranaga P (2005) Inexact graph matching for model-based recognition: evaluation and comparison of optimization algorithms. Pattern Recognit 38(11):2099–2113
Charlebois D (1997) A planning system based on plan re-use and its application to geographical information systems and remote sensing. PhD thesis, University of Ottawa, Canada
Charroux B, Philipp S (1995) Interpretation of aerial images based on potential functions. 9th Scandinavian Conference on Image Analysis. Uppsala, Sweden, pp 671–678
Chien S, Mortensen H (1996) Automating image processing for scientific data analysis of a large image database. IEEE Trans Pattern Anal Mach Intell 18(8):854–859
Chein M, Mugnier M (2008) Graph-based knowledge representation: computational foundations of conceptual graphs. Springer, New York
Clément V, Thonnat M (1993) A knowledge-based approach to integration of image procedures processing. CVGIP Image Underst 57(2):166–184
Clouard R, Porquet C, Elmoataz A, Revenu M (1999) Borg: a knowledge-based system for automatic generation of image processing programs. IEEE Trans Pattern Anal Mach Intell 21(2):128–144
Clouard R, Renouf A, Revenu M (2010) An ontology-based model for representing image processing objectives. Int J Pattern Recognit Artif Intell 24(8):1181–1208
Coates A, Abbeel P, Ng AY (2008) Learning for control from multiple demonstrations. In: 25th International Conference on Machine Learning, pp 144–151
Colliot O, Camara O, Bloch I (2006) Integration of fuzzy spatial relations in deformable models - application to brain MRI segmentation. Pattern Recognit 39:1401–1414
Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. Int J Pattern Recognit Artif Intell 18(3):265–298
Coradeschi S, Saffiotti A (1999) Anchoring symbols to vision data by fuzzy logic. In: Hunter A, Parsons S (eds) ECSQARU'99. LNCS, vol 1638. Springer, London, pp 104–115
Crevier D, Lepage R (1997) Knowledge-based image understanding systems: a survey. Comput Vis Image Underst 67(2):161–185
Dejean P, Dalle P (1996) Image analysis operators as concept constructors. IEEE Southwest Symposium on Image Analysis and Interpretation. San Antonio, USA, pp 66–70
Denœux T (2008) Conjunctive and disjunctive combination of belief functions induced by nondistinct bodies of evidence. Artif Intell 172(2–3):234–264
Deruyver A, Hodé Y (1997) Constraint satisfaction problem with bilevel constraint: application to interpretation of over-segmented images. Artif Intell 93(1–2):321–335
Deruyver A, Hodé Y (2009) Qualitative spatial relationships for image interpretation by using a conceptual graph. Image Vis Comput 27(7):876–886
Desachy J (1990) ICARE: an expert system for automatic mapping from satellite imagery. In: Pau LF (ed) Mapping and spatial modelling for navigation, vol F65. NATO-ASI. Springer, Berlin
Dominey PF (2007) Sharing intentional plans for imitation and cooperation: integrating clues from child developments and neurophysiology into robotics. In: Proceedings of the AISB 2007 Workshop on Imitation
Dominey PF, Warneken F (2009) The basis of shared intentions in human and robot cognition. New Ideas Psychol
Doya K (2000) Complementary roles of basal ganglia and cerebellum in learning and motor control. Curr Opin Neurobiol 10:732–739
Draper B, Bins J, Baek K (1999) ADORE: adaptive object recognition. International Conference on Vision Systems (ICVS). Las Palmas de Gran Canaria, Spain, pp 522–537
D'Souza A, Vijayakumar S, Schaal S (2001) Learning inverse kinematics. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 1:298–303. https://doi.org/10.1109/IROS.2001.973374
Dubois D, Prade H (1980) Fuzzy sets and systems: theory and applications. Academic, New-York

Dubois D, Prade H (1988) Possibility theory. Plenum, New-York
Dubois D, Prade H (2001) La problématique scientifique du traitement de l'information. Inf-Interact-Intell 1(2):79–98
Ficet-Cauchard V, Porquet C, Revenu M (1999) CBR for the management and reuse of image-processing expertise: a conversational system. Eng Appl Artif Intell 12(6):733–747
Fouquier G, Atif J, Bloch I (2008) Sequential spatial reasoning in images based on pre-attention mechanisms and fuzzy attribute graphs. European Conference on Artificial Intelligence, ECAI, Patras, Greece 178:611–615
Fouquier G, Atif J, Bloch I (2012) Sequential model-based segmentation and recognition of image structures driven by visual features and spatial relations. Comput Vis Image Underst 116(1):146–165
Frucci M, Perner P, Sanniti di Baja G (2008) Case-based-reasoning for image segmentation. Int J Pattern Recognit Artif Intell 22(5):829–842
Garbay C (2001) Architectures logicielles et contrôle dans les systèmes de vision. In: Jolion JM (ed) Les systèmes de Vision (Chap. 7). Hermès, Paris, pp 197–252
Gruber TR (1993) Towards principles for the design of ontologies used for knowledge sharing. In: Guarino N, Poli R (eds) Formal ontology in conceptual analysis and knowledge representation. Kluwer Academic Publishers, Deventer. http://citeseer.ist.psu.edu/gruber93toward.html
Guillot A, Meyer JA (2008) La Bionique: quand la science s'inspire de la nature. Dunod, collection UniverSciences
Gurevich IB, Salvetti O, Trusova YO (2009) Fundamental concepts and elements of image analysis ontology. Pattern Recognit Image Anal 19(4):603–611
Hanson AR, Rieseman EM (1978) Visions: a computer system for interpreting scenes. Academic, New York, pp 303–333
Harnad S (1990) The symbol grounding problem. Physica 42:335–346
Hasboun D (2005) Neuranat. http://www.chups.jussieu.fr/ext/neuranat/index.html
Hudelot C (2005) Towards a cognitive vision platform for semantic image interpretation; application to the recognition of biological organisms. PhD in computer science, Université de Nice Sophia Antipolis
Hudelot C, Atif J, Bloch I (2008) Fuzzy spatial relation ontology for image interpretation. Fuzzy Sets Syst 159(15):1929–1951
Hudelot C, Atif J, Bloch I (2010) Integrating bipolar fuzzy mathematical morphology in description logics for spatial reasoning. In: ECAI 2010, Lisbon, Portugal, pp 497–502
Hunter J (2001) Adding multimedia to the semantic web - building an MPEG-7 ontology. International Semantic Web Working Symposium (SWWS), Stanford, CA, pp 261–281
Ijspeert JA, Nakanishi J, Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: International Conference on Robotics and Automation (ICRA)
Kruse T, Kirsch A, Sisbot EA, Alami R (2010) Exploiting human cooperation in human-centered robot navigation. In: International Symposium in Robot and Human Interactive Communication (IEEE ROMAN), pp 212–217
Kuipers BJ, Levitt TS (1988) Navigation and mapping in large-scale space. AI Mag 9(2):25–43
Lansky A, Friedman M, Getoor L, Schmidler S, Short N Jr (1995) The collage/khoros links: planning for image processing tasks. AAAI Spring Symposium: Integrated Planning Applications. Menlo Park, CA, pp 67–76
Le Ber F, Napoli A (2002) The design of an object-based system for representing and classifying spatial structures and relations. J Univers Comput Sci 8(8):751–773
LeCun Y, Bengio J, Hinton GE (2015) Deep learning. Nature 521(7553):436–444
Leibe B, Ettlin A, Schiele B (2008) Learning semantic object parts for object categorization. Image Vis Comput 26(1):15–26
Lesaint F, Sigaud O, Flagel SB, Robinson TE, Khamassi M (2014) Modelling individual differences in the form of pavlovian conditioned approach responses: a dual learning systems approach with factored representations. PLoS Comput Biol 10(2):e1003466

Liedtke C, Blömer A (1992) Architecture of the knowledge based configuration system for image analysis "Conny". In: IEEE International Conference on Pattern Recognition (ICPR), The Hague, Netherlands, pp 375–378
Ligozat G (1998) Reasoning about cardinal directions. J Vis Lang Comput 9:23–44
Lungarella M, Metta G, Pfeifer R, Sandini G (2003) Developmental robotics: a survey. Connect Sci 15:151–190
Maillot N, Thonnat M (2008) Ontology-based complex object recognition. Image Vis Comput 26(1):102–113
Martin V, Maillot N, Thonnat M (2006) A Learning approach for adaptive image segmentation. In: 4th IEEE International Conference on Computer Vision Systems (ICVS), New York, pp 40–47
Matsuyama T (1986) Knowledge-based aerial image understanding systems and expert systems for image processing. In: International Geoscience and Remote Sensing Symposium (Zurich), pp 1026–1038
Matsuyama T (1989) Expert systems for image processing: knowledge-based composition of image analysis processes. Comput Vis Graph Image Process 48(1):22–49
Matsuyama T, Hwang VSS (1990) SIGMA: a knowledge-based aerial image understanding system. Plenum, New York
McKeown DM, Harvey WA, McDermott J (1985) Rule-based interpretation of aerial imagery. IEEE Trans Pattern Anal Mach Intell PAMI 7(5):570–585
Mezaris V, Kompatsiaris I (2004) Strintzis MG (2004) Region-based image retrieval using an object ontology and relevance feedback. Eurasip J Appl Signal Process 6:886–901
Minsky M (1974) A framework for representing knowledge. In: Winston P (ed) The psychology of computer vision. McGraw Hill, New York
Najar A, Sigaud O, Chetouani M (2015) Social-task learning for HRI. Social robotics. Springer, Cham, pp 472–481
Nazif A, Levine M (1984) Low level image segmentation: an expert system. IEEE Trans Pattern Anal Mach Intell 6(5):555–577
Nempont O, Atif J, Bloch I (2013) A constraint propagation approach to structural model based image segmentation and recognition. Inf Sci 246:1–27
Neumann B, Möller R (2008) On scene interpretation with description logics. Image Vis Comput 26(1):82–110
Oudeyer PY, Kaplan F, Hafner V (2007) Intrinsic motivation systems for autonomous mental development. IEEE Trans Evol Comput 11(2):265–286
Pasqui V, Saint-Bauzel L, Sigaud O (2010) Characterization of a least effort user-centered trajectory for sit-to-stand assistance user-centered trajectory for sit-to-stand assistance. In: IUTAM Symposium on Dynamics Modeling and Interaction Control in Virtual and Real Environments
Perchant A, Bloch I (2002) Fuzzy morphisms between graphs. Fuzzy Sets Syst 128(2):149–168
Perner P, Holt A, Richter M (2005) Image processing in case-based reasoning. Knowl Eng Rev 20(3):311–314
Protire A, Sapiro G (2007) Interactive image segmentation via adaptive weighted distances. IEEE Trans Image Process 16(4):1046–1057
Quillian M (1967) Word concepts: a theory and simulation of some basic semantic capabilities. Behav Sci 12(5):410–430
Quinton JC, Buisson JC, Perotto F (2008) Anticipative coordinated cognitive processes for interactivist and Piagetian theories. In: Wang P, Goertzel B, Franklin S (eds) 1st Conference on Artificial General Intelligence. Frontiers in Artificial Intelligence and Applications vol 171. IOS Press, Memphis, pp 287–298
Rao A, Lohse G (1993) Towards a texture naming system: identifying relevant dimensions of texture. 4th IEEE Conference of Visualization, San Jose, CA, pp 220–227
Ratliff N, Silver D, Bagnell J (2009) Learning to search: functional gradient techniques for imitation learning. Auton Robot 27(1):25–53
Renouf A, Clouard R, Revenu M (2007) How to formulate image processing applications? International Conference on Computer Vision Systems (ICVS). Bielefeld, Germany, pp 1–10

Rosse C, Mejino J (2003) A reference ontology for bioinformatics: the foundational model of anatomy. J Biomed Inform 36(6):478–500
Rost U, Mnkel H (1998) Knowledge-based configuration of image processing algorithms. International Conference on Computational Intelligence and Multimedia Applications (ICCIMA). Monash, Australia, pp 9–11
Saathoff C, Staab S (2008) Exploiting spatial context in image region labelling using fuzzy constraint reasoning. In: Ninth International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS'08, pp 16–19
Saint-Bauzel L, Pasqui V, Monteil I (2009) A reactive robotized interface for lower limb rehabilitation: clinical results. IEEE Trans Robot Spec Issue Rehabil Robot 25:583–592
Salaun C, Padois V, Sigaud O (2010) Learning forward models for the operational space control of redundant robots. In: Peters J, Sigaud O (eds) From Motor Learning to Interaction Learning in Robots, vol 264. Springer, Berlin, pp 169–192. https://doi.org/10.1007/978-3-642-05181-4_8
Schaal S (1999) Is imitation learning the route to humanoid robots? Trends Cogn Sci 6:233–242
Shafer G (1976) A mathematical theory of evidence. Princeton University Press, Princeton
Sigaud O, Droniou A (2016) Towards deep developmental learning. IEEE Trans Cogn Dev Syst 8(2):99–114. https://doi.org/10.1109/TAMD.2015.2496248
Smeulders A, Worring M, Santini S, Gupta A, Jain R (2000) Content-based image retrieval at the end of the early years. IEEE Trans Pattern Anal Mach Intell 22(12):1349–1380
Sowa J (1984) Conceptual graphs: information processing in mind and machine. Addison Wesley, Reading, p 234
Stulp F, Sigaud O (2012) Path integral policy improvement with covariance matrix adaptation. 29th International Conference on Machine Learning (ICML). Edinburgh, Scotland, pp 1–8
Stulp F, Sigaud O (2015) Many regression algorithms, one unified model: a review. Neural Netw 69:60–79
Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
Thonnat M, Moisan S (2000) What can program supervision do for program reuse? IEEE Proc Softw 147(5):179–185
Town C (2006) Ontological inference for image and video analysis. Mach Vis Appl 17(2):94–115, www.cl.cam.ac.uk/~cpt23/papers/TownMVA2006.pdf
Vanegas MC, Bloch I, Inglada J (2016) Fuzzy constraint satisfaction problem for model-based image interpretation. Fuzzy Sets Syst 286:1–29
Vieu L (1997) Spatial representation and reasoning in artificial intelligence. In: Stock O (ed) Spatial and temporal reasoning. Kluwer, Dordrecht, pp 5–41
Vijayakumar S, D'Souza A, Schaal S (2005) Incremental online learning in high dimensions. Neural Comput 12:2602–2634
Vinyals O, Toshev A, Bengio S, Erhan D (2015) Show and tell: a neural image caption generator. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 3156–3164
Waxman S (2000) Correlative neuroanatomy, 24th edn. McGraw-Hill, New York
Zadeh LA (1965) Fuzzy sets. Inf Control 8:338–353

Cross-Fertilisation Between Human-Computer Interaction and Artificial Intelligence

Christophe Kolski, Guy André Boy, Guy Melançon, Magalie Ochs and Jean Vanderdonckt

Abstract Human-Computer Interaction (HCI) and Artificial Intelligence (AI) are two disciplines that followed parallel trajectories for about four decades. They also both complement each other and overlap in various problem-rich domains. This chapter is far from being exhaustive, but provides a representative story of how HCI and AI cross-fertilise each other since their inception. It reviews the following domains: intelligent user interfaces and more specifically conversational animated affective agents; capitalisation, formulation and use of ergonomic knowledge for the design and evaluation of interactive systems; synergy between visualisation and data mining.

C. Kolski (✉)
LAMIH, Univ. Polytechnique Hauts-de-France, Valenciennes, France
e-mail: Christophe.Kolski@uphf.fr

G. A. Boy
CentraleSupélec (Paris Saclay University) & ESTIA Institute of Technology, Gif-sur-Yvette, France & Bidart, France
e-mail: guy-andre.boy@centralesupelec.fr; g.boy@estia.fr

G. Melançon
LaBRI, Université de Bordeaux, Bordeaux, France
e-mail: Guy.Melancon@u-bordeaux.fr

M. Ochs
Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
e-mail: magalie.ochs@lis-lab.fr

J. Vanderdonckt
LiLab, UCL, Ottignies-Louvain-la-Neuve, Belgium
e-mail: jean.vanderdonckt@uclouvain.be

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_11

1 Introduction

Goals of this chapter are to study the various interfaces between Human-Computer Interaction[1] (HCI) and Artificial Intelligence (AI), and to advocate their necessary association. AI purpose is to create and develop intelligent systems (i.e., systems that are capable to perceive, reason, act and learn by themselves). This fundamental objective led to the development of methods, techniques and tools that enabled systems to effectively perceive, reason, act and learn, within limited narrow environments. Even if first results were limited, AI ambition and framework clearly lead to a long-term endeavour. For example, robotics made impressive progress. Most recent example is "Curiosity", the NASA robot that is currently exploring planet Mars. HCI is more focused on usability of new technology, promoting creativity and innovation. HCI always had shorter-term goals. We should note that remote Curiosity operations from the Earth cannot be performed without user interfaces based on solid HCI concepts and principles, more specifically regarding planning of its activities. In other words, NASA ground operators must have the best situation awareness of how Curiosity's instruments can be used and what are resources required for the various scheduled activities.[2]

It is useful to realise that both AI and HCI have to take into account people who will be involved in the use of technology being developed. On one side, AI attempts to mimic human beings and rationalise their behaviours to build various types of intelligent systems, including robots. On the other side, HCI attempts to understand the human being to better adapt machines that improve safety, efficiency and comfort experience. AI focuses on internal mechanisms of a rational intelligence. Instead, HCI focuses on fundamental phenomena of interaction among people and tools, which they created and use.

It is now obvious that HCI specialists use more AI techniques to improve interaction. They use machine learning to contextualise Web search for example. AI specialists need user interfaces appropriate to the use of intelligent systems. Human-robot interaction is an excellent example of HCI-AI fusion. The writer, Isaac Asimov, provided his famous Three Laws of Robotics in 1941: "(Law 1) a robot may not injure a human being or, through inaction, allow a human being to come to harm; (Law 2) a robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law; and (Law 3) a robot must protect its own existence as long as such protection does not conflict with the First or Second Law." (Asimov 2008). Here we find again the concepts of safety, efficiency and comfort fit for human-computer interaction (Boy 2013).

This combination of HCI and AI is gradually emerging from increasing needs of meaning, knowledge, skills, experience… and finally common sense. The success of a new technology is often the product of an intelligent and meaningful integration of

[1] The term "Human-Computer Interaction", still very much used, is shifting toward "Human-Systems Interaction" since we are developing "systems" that include both computing and physical things.

[2] http://hci.arc.nasa.gov/mslice.html.

various methods, techniques and systems. AI provides tools for externally automating human behaviour, as well as creating new cognitive prostheses (Ford et al. 1997). HCI provides tools for safer, more efficient and more comfortable human interaction with resulting technology. It is certainly more interesting to head toward a more comprehensive approach to systems design which incorporates both intelligence and interaction, taking into account the use of computer science and social sciences in concert to go to human-centred design (Boy 2013).

The second part of the chapter presents a design history and genesis of interfaces between HCI and AI.[3] Intelligent user interfaces are presented in the third part. Among recent advances in the field, emotional embodied conversational agents will be presented in Part Four. Part Five is devoted to capitalisation, formalisation and use of ergonomic principles for the design and evaluation of interactive systems. Part Six is devoted to visualisation and data mining. The last part of the chapter concludes on cross-fertilisation between HCI and AI.

2 History of Interfaces Between HCI and AI: A Genesis

AI and HCI are two branches of computer science. They are complementary disciplines that have different objectives and natures. AI is studying intelligent agents (i.e., any entity capable of perceiving, inferring and acting on its environment using its own knowledge). In 1955, John McCarthy named and defined AI as a scientific and technical discipline that supports the making of intelligent machines. HCI studies devices to be used for the control of and communication with a machine, or with other people through a machine. ACM SIGCHI provides the following definition[4]: "Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." (Hewett et al. 1992).

It is certainly interesting to start talking about automatic control before talking about artificial intelligence. Although these two disciplines do not belong to the same scientific fields, they are continuous in the field of automation, this is to say in the industry. We automated transport aircraft since the nineteen-thirties. This type of automation enabled the implementation of analogue and then digital techniques to develop and use symbolic techniques and tools. Automation increasingly became a software engineering issue. For example, we observed the evolution of aircraft cockpits electronic equipment to computing equipment. In the nineteen-eighties, the Flight Management System (FMS) was introduced as an onboard database management system for commercial aircraft. It enabled freeing the aircrew from flight planning and navigation tasks. FMS includes necessary flight management intelligence that was previously the only ownership of pilots. Underlying methods were

[3] See also Grudin (2009).

[4] ACM (Association for Computing Machinery)-SIGCHI (Special Interest Group on Computer-Human Interaction).

based on rule-based systems and trajectory optimisation. In the beginning, the FMS, as a computing system, was often more difficult to handle and manage than paper maps and documents. Intelligence was transferred to the machine with very little care about human-computer interaction, generating new types of human errors. The interface was complicated due technology limitations (it would not be done the same today). Its use was difficult to learn and remember. In short, if automation was good, pilot-FMS interaction required a lot more work. We had to wait for more mature technology to get the type of user interface that we know today in the cockpit of the A380, for example. It is interesting to notice that this type of cockpit is now called "interactive cockpit" not because of the interaction with systems' mechanical parts of the aircraft but with the onboard user interface that involves a pointing device and screens.

The term HCI includes the term "human". The human entity is difficult to identify, define and model. For a long time, the Human Factors and Ergonomics (HF&E) community has addressed this issue (Woodson and Conover 1964). The human being can be characterised by various properties. Firstly, the human has a physical and physiological body (e.g., an issue that HF&E experts have questioned since the end of World War II with the development of digital manikins). The biomechanical aspects, fatigue, age and other human factors have been and are still widely and deeply considered. Physical ergonomics was that we experienced in the early nineteen-eighties when we had to certify transport aircraft cockpits designed for two crewmen cockpits. We very quickly realised that physical and physiological ergonomics was not enough and was very limited with respect to the evaluation of new aircraft systems. We had to go to cognitive ergonomics in order to properly support analysis of information processing (Norman 1986). In addition, conventional automation techniques were not sufficient. HCI techniques addressing highly automated systems became the central focus of problems to be solved. HCI was entering the cockpit, as it made its entrance in areas of critical systems in the years that followed. At the time, we focused on information processing and that is why cognitive science has taken a considerable extent in the engineering community. Cognitive ergonomics and cognitive engineering have become key disciplines for the analysis, understanding, design and evaluation of modern human(s)-machine(s) systems. Cognition has become a central must (Card et al. 1983; Hutchins 1995; Boy 1998).

The cognitive approach then gave birth to an organisational point of view with the introduction of groupware and collaborative work (CSCW: Computer-Supported Cooperative Work) (Grudin 1994). For these HCI specialists, the human became a worker in an organisation. Computer science suddenly became a strong support to management and business. In the background, Internet was becoming more persistent, since Douglas Engelbart's work in the sixties until the advent of the World Wide Web in 1992 by Tim Burners Lee and his team at CERN in Geneva. The Web has really established a drastic practice change in modern societies. The human has gradually become "informavore" (term introduced by George Miller in 1983). Information sciences took this new object and topic. The main issue then became information management using the Web. HCI cooperated with AI to optimize and adapt information search to users (Bellot 2011; Boy 1991a). The Web has become the

Semantic Web (Berners-Lee et al. 2001; Shadbolt et al. 2006), adding the intelligence of an universal librarian.

HCI evolution led us then to a new stage, that of social networks. The human becomes a social being for IT professionals in the field. The introduction of systems such as Google, Facebook, LinkedIn and Twitter led to the emergence of new practices. No need for structure, knowledge could be completely distributed, we could find it anywhere anytime. Research scientists and practitioners in sociology and anthropology came on stage to study this new type of environments. Interactions among people and systems are now inherently social but also emotional; human emotions can be both positive and negative facing an interactive system. Artificial intelligence of a system then also lies in its ability to manage the emotional aspects of interaction through interfaces capable of adapting to the social context and user's emotions. This research problem has created an IT research trend lying on the border between HCI and AI: affective computing (Picard 1997).

Where are we today? Information technology development brought us a lot of techniques and tools. Certainly too many for us to be able to reasonably integrate them easily, with respect to our problems to solve. It is time to ask the question of meaning. What is making sense? We are going from the problem of knowing to the problem of being. That is why design (in the sense of integrated design and aesthetics) is increasingly becoming a necessity in engineering sciences, and computing in particular. The design approach to problem solving uses creativity rather than already-made, blocking and ultimately sterile procedures. What characterises our 21st century society is certainly complexity. On this point, AI and HCI must get together to solve problems whose complexity is at the centre. We must all converge towards an ontological approach of problem stating and problem solving. Indeed, before solving a problem, it must be stated well! This is the art of abduction (i.e., knowing how to imagine assumptions and goals, look for opportunities and ultimately seek meaning) (Boy 2013).

3 Intelligent User Interfaces

Intelligent User Interface (IUI) design and evaluation is a vast and rich research and development area, which is at the intersection of human-system interaction and artificial intelligence, but also of cognitive sciences. Research on such user interfaces appeared in the early 80s (see for instance Edmonds 1981); in fact the first concepts date from late 70s in terms of so-called adaptive approaches. Many definitions have been proposed in the literature. So (Hancock and Chignell 1989) defined them as interfaces which provide tools to help minimise the cognitive distance between the mental model which the user has of the task, and the way in which the task is presented to the user by the computer when the task is performed. It even seems possible to go beyond this definition because, starting from the model of the theory of action of Norman (1986), while lying in an adaptive approach (many others were studied in

literature, see below), it is possible to locate different cognitive stages, against which one or more adaptations are especially important, Fig. 1.

Among the many other definitions include that of Maybury (1999): "*Intelligent User Interfaces (IUI) are human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-computer interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture). IUI are multifaceted, in purpose and nature, and include capabilities for multimedia input analysis, multimedia presentation generation, model-based interfaces, agent-based interfaces, and the use of user, discourse and task models to personalize and enhance interaction.* [. . .]".

Consequently, the field of intelligent interfaces covers a disciplinary field representing the intersection of human-system interaction, software ergonomics, cognitive science and artificial intelligence, including their respective sub-disciplines such as computer vision, automatic language processing, knowledge representation and reasoning, machine learning, knowledge discovery, planning, modeling of software and human agents, modeling of speech. It is also commonly accepted that the field of intelligent interfaces represents the intersection of human-system interaction and artificial intelligence, as well as Engineering of Interactive Computing Systems (EICS) represents the intersection of the human-system interaction and software engineering.

The definition of Maybury is interesting because it highlights that such user interfaces are designed to provide solutions in relation to different underlying criteria for human-system interaction, and they need to take into account explicitly several models to cover a set of steps in relation to Perception, Cognition and Action (if one refers to the PCA model well known in Artificial Intelligence). Included are problems and recurring locks associated with the recognition of user (or user group) intention, modeling knowledge and preferences, communication with the user, and

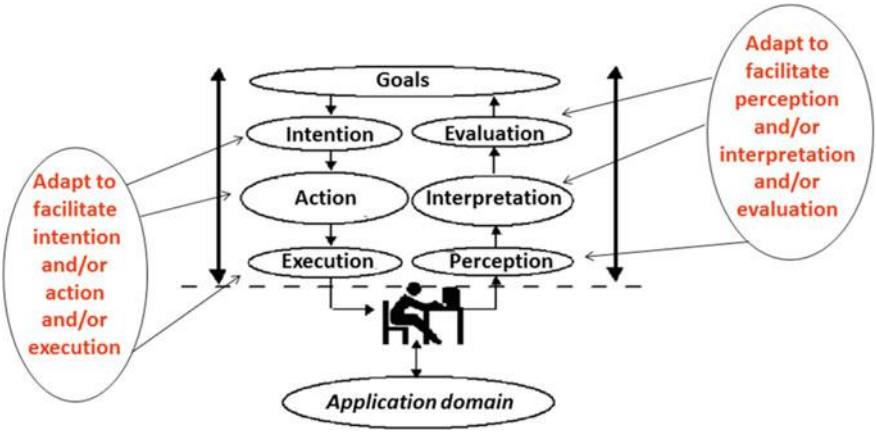


Fig. 1 Adaptation(s) in the light of the Norman's Theory of Action (adapted from Norman 1986)

also the reasoning and the decision making (what, when, where, to whom, how, why to present the information).

Intelligent user interfaces are now a field in its own right. It is important to emphasise that famous conference of the Association of Computing Machinery (ACM) is dedicated to them. This is called ACM IUI (Intelligent User Interfaces). From a historical perspective (see also http://www.iuiconf.org/IUI/History), the idea started with a workshop in March 1988 entitled "Architectures for Intelligent Interfaces", resulting in 1991 in the book "Intelligent User Interfaces" published by J.W. Sullivan and S.W. Tyler, workshop organisers). After a first workshop sponsored by the ACM in 1993, IUI has become an annual conference in 1997. In the call of the 2018 edition, note that the topics highlighted were the following: "Health and intelligent health technologies, Information retrieval and search, Intelligent assistants for complex tasks, Intelligent wearable and mobile interfaces, Intelligent ubiquitous user interfaces, Intelligent visualization tools, Interactive machine learning, Knowledge-based approaches to user interface design and generation, Modeling and prediction of user behavior, Multi-modal interfaces (speech, gestures, eye gaze, face, physiological information etc.), Natural language and speech processing, Persuasive and assistive technologies in IUI, Planning and plan recognition for IUI, Proactive and agent-based user interaction, Recommender systems, Smart environments and tangible computing, Social media analysis, User Modelling for Intelligent Interfaces, User-Adaptive interaction and personalization".

In addition to the regular publication of many books and articles on intelligent user interfaces in various journals and conferences, both in human-system interaction and in Artificial Intelligence, we can point out that both journals are dedicated to them: User Modeling and User-Adapted Interaction (Springer); ACM Transactions on Interactive Intelligent Systems.

Many approaches contributing to intelligent user interfaces can be found in the literature. It is not possible to mention them all here; the interested reader can find in various books or review papers further details on this topic (Chignell and Hancock 1988; Kolski and Le Strugeon 1998; Hook 2000; Jameson 2007; Akiki et al. 2014):

- The so-called adaptive approaches are those that led to most proposals (cf. Kolski et al. 1992; Schneider-Hufschmidt et al. 1993; Jameson and Gajos 2012), the adaptation being usually relatively different user and/or task characteristics or task.
- In continuation of work on adaptation, there are approaches on the current trend of providing more personalised services to users of information systems at large. It is in this case to adapt to the goals (needs or reasons that led the user to query the system), habits/preferences (all the criteria that distinguish a solution of another for the same query), the user capabilities (including both hardware and software capabilities available to the user, as its own physical or cognitive abilities, possibly related to disabilities). It is important that the system learns during the interaction with the user, as well as with users with close profile, to further customise, exploiting for example collaborative filtering algorithms (Su and Khoshgoftaar 2009). The works are numerous in this area (Abed et al. 2011; Brusilovsky et al.

2007; Peintner et al. 2008; Findlater and Gajos 2009; de Oliveira et al. 2013; Germanakos and Belk 2016).

- Certain intelligent user interfaces are called tolerant to human error (Rouse and Morris 1987; Beka Be Nguema et al. 2000). In this case the adaptation operates a classification of possible human errors and their consequences. Their goal of such IUI is to correct human error or alert the user in case of potential problems.
- Another category includes assistants, in the broadest sense of the term. They are at the service of the users, listening to them, to assist them in case of problems (Lieberman 1995) (see also Shneiderman and Maes 1997). A typical example is the assistant, called Clippy, integrated in several previous versions of the Microsoft Office environment. This assistant was made available to the user and advising (requested or not) on appropriate procedures (like how repaginate a document). Such approach was originally proposed in the aviation field, under the name of intelligent assistant systems, acting as co-pilot in a cockpit and reasoning in parallel to the user considered as the final decision maker (Boy 1991a). To improve human-system interaction, to make it more natural, design choices may include a human physiognomy of the assistant, see about the "Affective Embodied Conversational Agents" section of this chapter.
- Many researches focus on context in general and how to take it into account in UI design. The definitions of Abowd et al. (1999), Dey (2001) on this subject are the most cited. Different types of context-aware systems are also studied. It is in fact possible to find in the literature the following qualifiers: Context-aware and Context-sensitive, which denote the fact of using the context or the one to adapt to it. The works are numerous in this area (Boy 1991b, 1992; Winograd 2001; Coutaz and Rey 2002; Limbourg et al. 2004; van den Bergh 2006; Brossard et al. 2011; Moussa et al. 2015; Bauer et al. 2014; Bauer and Dey 2016).
- The objective of the so-called plastic user interfaces is to adapt to their context of use in respect of their usability (Thevenin and Coutaz 1999). The context of use must consider the characteristics of the user, platform of interaction, and environment (Calvary et al. 2004; Sottet et al. 2009; Coutaz and Calvary 2012).
- A more prospective approach considers the intelligent user interface and its socio-technical environment as a multi-agent system. It corresponds to a distributed approach to human-system interaction. It is important to note that, more generally, distributed user interfaces are subject of much research currently (Gallud et al. 2011; Lozano et al. 2013; Gallud et al. 2014). Such an interface, suggested by Mandiau et al. (1991); Kolski and Le Strugeon (1998), is composed of cognitive and reactive agents, working in parallel and/or cooperating in order to solve different problems relating to the tasks. The result of their treatment is provided to users through acts of communication, but we can imagine a variety of other actions directly on the system, for example. This principle was implemented as part of a road traffic simulation on interactive tabletop with RFID technology, virtual agents representing vehicles responding to the activation of tangible objects manipulated by users (Kubicki et al. 2013; Lebrun et al. 2015).

Areas of application of intelligent user interfaces are multiple, since they can be a contribution when a semi-automatic or automatic aid should be implemented by the interactive system, while taking into account different criteria or user preferences. Research and development focused as well on tasks considered as relatively simple (office, information retrieval, e-commerce, etc.) than on complex tasks, even critical (transport, supervision, health).

4 Affective Embodied Conversational Agents

Computers are increasingly used in roles that are typically fulfilled by humans, such as virtual tutors in a learning class or virtual assistants for task realisation. When computers are used in these roles they are often embodied by animated cartoon or human like virtual characters, called *Embodied Conversational Agents* (ECA) (Cassell 2000). An example of an ECA is illustrated Fig. 2. This enables a more natural style of communication for the human and allows the computer to avail of both verbal and non-verbal behaviour channels of communication. Several studies have demonstrated the acceptance and the efficiency of such agents (Hoffmann et al. 2009); indeed, the *persona effect* reveals that the presence of an ECA improves the subjective experience of an interaction for the user. Moreover, when people interact with such virtual agents, they tend to react naturally and socially as they would do with another person (Hoffmann et al. 2009).

The ECAs are not only simple humanoid graphical representations but embody (1) cognitive agents able to reason on complex internal semantic representation, (2) interactive agents able to interact in a multimodal way with a user, and (3) expressive agents able to show, through their verbal and non-verbal behavior, a particular cognitive and affective mental state. These three dimensions—cognitive, interactive and expressive—are essential to design an ECA to improve the interaction both regarding satisfaction of the user and user's performance in task execution. For instance, several research efforts have shown that agents expressing emotions improve interaction (Beale and Creed 2009). However, the expressive capacity of the agent is not sufficient: an emotion expressed in an inappropriate situation during an interaction may have a negative impact on the user's perception (Ochs et al. 2012b). That is the three dimensions—*cognitive*, *interactive* and *expressive*—that enable the agent to adopt an affective behavior relevant and efficient during the interaction with the user.

The cognitive dimension of an ECA implies knowledge representation and its reasoning capacity.[5] Knowledge refers to information related to both application domain and task: but corresponds also to the knowledge related to the social and emotional dimension of the interaction. Indeed, an ECA has to be able to represent and reason on its own emotions and those of its interlocutor given the social context. The

[5]The reader can refer to several papers in Volume 1 for an overview of the methods in Artificial Intelligence for the representation of knowledge and the reasoning.

Fig. 2 Example of an embodied conversational agent: the ECA Greta (Pelachaud 2009). Expressions of joy, sadness, and anger (from left to right)

cognitive representation of emotions has to include a representation of the conditions of emotions elicitation, i.e. when an individual may feel which emotion(s) given a situation. This information may be used by the ECA to identify when the ECA itself may express which emotion(s) but also which emotion(s) the user may potentially feel during the interaction. The elicitation of emotion being tightly related to the achievement or failure of a goal (Scherer 2000), the BDI representation (Belief, Desire and Intention) of emotions—through syntactic abbreviations of combinations of mental attitudes—is particularly adapted to this problematic.[6] Such a formalization has for instance been used to develop en ECA that is capable to express empathy

[6] An example of a BDI formalisation of emotions is proposed in chapter "Formalization of Cognitive-Agent Systems, Trust and Emotions" of Volume 1.

toward the user during a dialogue (Ochs et al. 2012b) or a virtual tutor that adapts its pedagogical strategy to the inferred emotions of the user (Jaques and Viccari 2004). Other methods have been proposed to represent emotions. For instance, to illustrate the dynamical aspect and the non-determinism of emotions, a representation based on Bayesian network (de Melo et al. 2012) and Dynamic Belief Network (deRosis et al. 2003) has been developed.

Moreover, the emotions of an agent may be used to determine the appropriate behavior of the agent in a virtual environment. The emotions are then integrated in the decision making process for the actions selection (Canamero 2003). Inspired by the coping theory claiming that people used specific cognitive strategy to cope with their emotions (Lazarus 1991), the impact of emotions on an ECA's behavior may be modeled by a modification of its mental state, i.e. a modification of its beliefs, desires and intentions (Gratch and Marsella 2004). For instance, if the ECA has a negative emotion, it can adopt an acceptation strategy implying the fact that the agent gives up the intention that has elicited the negative emotion by its failure. By consequence, the negative emotion of the agent will disappear.

During the interaction with the user, the emotional knowledge of the agent should be enriched given the progress of the interaction. Indeed, even if a formalisation of emotions may enable the agent to infer the user's emotions, a system to recognise in real-time the emotions expressed by the user may validate, refute or refine the agent's emotional knowledge, in particular concerning the effects of agent's actions on user's emotions. During the interaction, the user expresses her/his emotions through her/his non-verbal behavior (e.g. facial expressions and voice), her/his verbal behavior (e.g. emotional words) and through physiological signals (e.g. skin conductance). The system to automatically recognised emotions are generally constructed on an offline learning of the non-verbal, verbal or physiological characteristics of emotions based on real or acted corpus of data of individual feeling or expressing emotions. For instance, audio-visual corpora are collected to analyse the facial, gestural and acoustic characteristics of emotions expressed by individuals. The corpora of emotion expressions are generally manually annotated with different types of emotion and sometimes with associated intensity values. A method to extract emotional knowledge from these corpora consists in exploiting machine learning methods,[7] and more precisely supervised algorithms to correlate expressive characteristics (such as muscles activation of the face or acoustic parameters of the voice) to emotion types and intensity (Caplier 2011; Clavel and Richard 2011).

Machine learning methods are also used for the generation of expressive behaviors of ECAs (head movements, gaze direction, posture, etc.). The analysis of interpersonal interactions in corpora can be exploited to identify how an affective or cognitive state is expressed through facial expressions gestures or postures; but also how individuals coordinate their non-verbal behavior (e.g. mimicry). In this learning perspective of ECA's non-verbal behavior, two approaches may be distinguished. Al-

[7]Machine learning methods are presented in details in chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2.

gorithms said "black box"[8] can be used to model ECA's non-verbal behavior that is reflex or slightly correlated to another modality. For instance, a learnt Hidden Markov Model is used to predict the head movements of an ECA during an interaction with a user or for its lips synchronisation with its speech (Hofer and Richmond 2010). Conversely, algorithms said "white box" are used to extract knowledge that then are explicitly represented in the ECA. For instance, a classification method based on decision tree is used to identify the morphological and dynamic characteristics of different types of smile for ECA (amused, polite and embarrassed smiles) (Ochs et al. 2012a); and sequence mining algorithms are exploited to extract knowledge on the multimodal signals that the ECA may use to convey social attitudes (Chollet et al. 2017; Porhet et al. 2017). Models have been proposed to convey emotion expression variability, which integrate uncertainty of activation of human face muscles and some gestures based on fuzzy logic rules (as for instance in Niewiadomski and Pelachaud 2007).

Finally, ECA design implies various kinds of AI problems: knowledge representation, decision making, planning and learning. These problems, applied to model this complex phenomena of emotions, aim at integrating emotional intelligence into interactive systems (Salovey et al. 2000), need to be solved for optimizing human-machine interaction.

5 Consolidating, Formalizing and Exploiting Usability Knowledge for Designing and Evaluating Interactive Systems

Designing and evaluating interactive systems have been very active research domains since more than three decades primarily to formalise usability knowledge that could be useful in a computational framework. The framework should support high level abstraction, access, testing and fixing potential usability problems. The goal is to be released from the empirical assessment traditionally induced by various approaches using this usability knowledge. For instance, Ivory and Hearst (2001) observed that different software for evaluating accessibility guidelines on the same web site may produce inconsistent results. Several reasons explain this: accessibility guidelines are not formalised, if they are, the formalisation is varying depending on the interpretation used by different methods. Jambon et al. (2001) listed and classified several dozens of methods and techniques for specifying an interactive system that largely vary depending on their perspective: psycho-ergonomic, Engineering of Interactive Computing Systems (EICS), software engineering. Vanderdonckt and Coyette (2007), as well as Beaudouin-Lafon and Mackay (2003) also reported on several techniques and software for prototyping user interfaces in order to assess them as early as possible. Since the eighties, research and development mainly fo-

[8]The results of such algorithms are difficult to explain and to interpret, for instance System Vector Machine (SVM) or Neural Networks.

cused on model-based methods for designing user interfaces: the designer creates one or many models (e.g., a domain model, a task model, a user model, etc.) which are subsequently used for semi-automatic generation of user interface code and/or evaluating it. If this generation is not made possible, at least a structured development life cycle is induced by the models. Literature is significant in this area: (Szekely 1996; Vanderdonckt and Puerta 1999; Paterno 1999; Kolski and Vanderdonckt 2002), . . . Later on, research and development progressively shifted its focus of attention to structured methods based on Model-Driven Engineering (MDE) in line with the initiative launched by the *Object Management Group* (www.omg.org). Most approaches adopt a top-down life cycle in which models are progressively transformed until the code of a final user interface is obtained. The Cameleon Reference Framework (CRF) (Calvary et al. 2003) now reached a consensus in the community that four levels of abstraction could structure this development life cycle: task and domain, abstract user interface, concrete user interface, and final user interface. This framework is now recommended by the W3C Group on Model-Based User Interface (MBUI) (Gonzalez Calleros et al. 2010), https://www.w3.org/TR/mbui-intro/. Several works are compliant with this W3C recommendation, such as, but not limited to: (Jacob et al. 2004; Calvary et al. 2008; Seffah et al. 2009; Hussmann et al. 2011), etc. Regarding evaluation of interactive systems, especially with respect to usability and accessibility (Nielsen 1993; Bastien and Scapin 2001), but also with respect to acceptability (Stephanidis 2009), many methods, techniques, and software also exist that offer ample possibilities depending on the availability of the user interface (vs a prototype of it) and real users (vs a representation of these users). For instance, when real users are not available, a model of these users is used instead. Further reading can be found in Nielsen (1993), Baccino (2005), Huart et al. (2008), Ezzedine et al. (2012), Jacko (2012). A problem subsumed by designing and evaluating interactive systems consists in how to create and manipulate the knowledge bases containing the usability knowledge required for the knowledge-based approaches for design and evaluation. Vanderdonckt (1999) discusses five major milestones required to create a usability knowledge-based software: guidelines collecting, guidelines organisation, guidelines incorporation into approach, guidelines operationalisation, and guidelines usage. Probably the most critical step is the operationalisation since a guidelines as initially stated in a source could be significantly reduced or constrained when incorporated in a knowledge-based approach.

Any guideline could be related to important ergonomic criteria (Bastien and Scapin 1993) for guidance, dialogue control, error management, consistency, workload, adaptability, compatibility, and code significance.

Usability and accessibility guidelines are considered to be a valuable source for supporting the (semi-)automated detection of potential usability/accessibility problems for designing as well as for evaluating (Grammenos et al. 2000; Tran et al. 2013). A usability guideline (Vanderdonckt 1994) is hereby referred to as any design and/or evaluation principle that could be used to produce and/or guarantee the usability quality of a final user interface. Five categories of usability guidelines could be distinguished: design standards, usability principles, usability guides, style guides, and algorithms for knowledge-based generation of user interfaces. For instance, a

guideline could be expressed depending on a particular domain of application (Scapin 1986; Smith and Mosier 1986) or independently, as in standards. A guideline could be specifically made applicable for a style guide that is related to an operating system. Many standards contains a significant section of guidelines (Stewart and Travis 2002), such as international norms (e.g., ISO 9241, ISO/IEC 9126), national ones (e.g., HFES, AFNOR, BSI . . .). Figure 3 sorts these five types of guidelines according to their level of applicability: standards are considered as the most general sources of usability knowledge, therefore widely applicable in principle, but also requiring some interpretation to be correctly applied depending on the context of use and all its dimensions (user, platform, and environment). On the other side of this continuum, design rules only require a minimal interpretation before applying them since their applicability is specific. The more general a guideline is, the more interpretation it may require in order to properly apply it.

Figure 4 locates guidelines depending on two axes: the need for interpretation and the level of precision required to implement them in a software. Algorithms are straightforward to be applied since they require almost no interpretation, but are totally driven by their specifications, which make them notably hard to modify. On the other side, principles require a high level of interpretation, thus making them cautious to apply. A particular condition imposed by a context of use may validate or invalidate the application of such principles.

Several research and development problems related to intelligent user interfaces, which is often considered as the overlapping of human-computer interaction and artificial intelligence, are still open, such as:

- Consolidation of knowledge-based approaches for user interface design and evaluation: there is a lack of systematic method, a lack of consensus in which approach could be considered valuable, and a lack of properly integrating techniques from knowledge engineering (see chapter "Knowledge Engineering" of Volume 1).
- Uncertainty of usability knowledge: many guidelines used in knowledge-based approaches are intrinsically uncertain: they are fuzzy, incomplete, redundant, hard to access and to modify by a non-expert person, and their notable independence

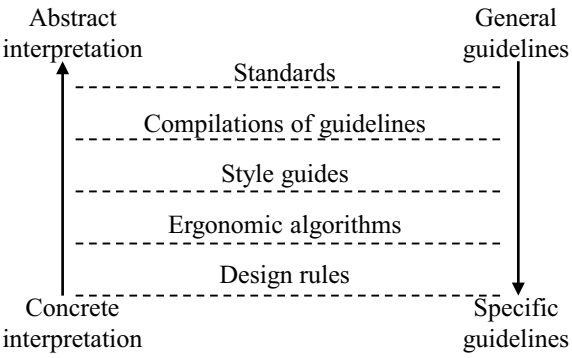


Fig. 3 Five types of guidelines sorted by level of applicability

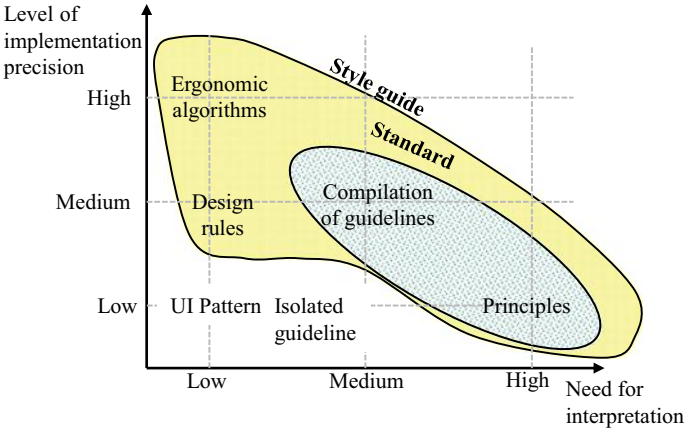


Fig. 4 Guidelines sorted by their level of interpretation and their implementation precision

with respect to the context of use unless such a context model is used (Vanderdonckt 1999).

- Lack of knowledge evolution: knowledge contained in knowledge bases used for user interface design, generation, evaluation is often constant and hard to modify, which may not satisfy the need for making it evolving depending the context of use, but also depending on the dynamic evolution of users. Machine learning techniques are particularly welcome for this purpose, which explain why knowledge could be improved by actual data of its usage.

6 Visualisation and Data Mining

Data processing and mining are nowadays a priority both in research and industry, small or large. Organisations now produce loads of data on their internal processes as well as on their client activities. Technological or strategic watch requires to collect and forage all available data in order to gain a competitive edge over competing organisations, or to better position themselves in the socio-economical context (Provost and Fawcett 2013).

Data can be massive—as in "big". Data is also complex, just as the phenomenon they emerge from, but also because it often is non-structured. Seeking for the information encapsulated within the data is difficult, discovering new insights from data is challenging (Zhang et al. 2012).

Data mining develops algorithms capable of identifying structural patterns, expressed as association rules for instance. A classical example is that of a supermarket seeking a better understanding of consumers' habits, typically through the use of fidelity cards or mobile application account, in order to refine its marketing strategy. A

telecom company will foresee the possibility to better understand how the teenager market differs from adult consumers. Data mining results can be complex just as the data that needs to be processed; they often need to be sorted, organised or classified before decision can be made.

Data mining is all about discovering structural patterns. Information visualisation relies on a crucial observation, that around 40% of our cortex activity is devoted to processing visual signals (Ware 2000). The challenge is thus to propose users with a graphical representation from which patterns in data now turned into visual patterns can be inferred. Being able to interactively manipulate and query the map additionally allows users to gain insight on the visual display of information. The literature gathers tons of approaches to graphically display data on the screen, often targeting specific types of data: temporal data (Silva and Catarci 2000; Daassi et al. 2006), geospatial data (Andrienko and Andrienko 2006), networks (Herman et al. 2000; von Landesberger et al. 2011) or multi-dimensional data (Hoffman and Grinstein 2002).

Visualisation aims at solving a problem which, although it can easily be formulated, can turn out to be quite complex. Computing cartesian coordinates of data points is but the first step. Visualisation also requires to use relevant visual variables (color, shapes, saliences, etc.) to highlight attribute properties (statistical distribution, correlation, proximity distorsion, etc.). Computing screen positions most of the times is a combinatorial optimisation problem while visual encodings involve graphics semiotics (Bertin 1998; Ware 2000) and escape purely computational approaches.

Visualisation is part of data and information processing. This process traditionally is represented as a pipeline chaining data processing steps, starting from data curation and organisation, statistical/combinatorial analysis up to computing a representation in (usually 2D) Euclidean space and its rendering on a screen (Card et al. 1999; Chi 2000; dos Santos and Brodlie 2004). This process, although depicted as a pipeline, does not necessarily deploy itself in sequence, but rather as iterations allowing structure to emerge and hypothesis to be formed: early iterations help focus on relevant subsets of data that need further investigation, leading to hypothesis that can be tested; visualisation will additionally be useful to disseminate results and support decision making. This process echoes Shneiderman's mantra ("Overview first, zoom and filter, then details on demand"; (see Shneiderman 1996)) and has been termed the "Sense-making loop" (cf. Fig. 5) by the Visual Analytics founders Thomas and Cook (Thomas and Cook 2006).

Because the data is complex, uncertain and changing, human intelligence resides at the heart of the foraging process. Echoing Shneiderman's mantra, Visual Analytics promoted by Thomas and Cook has established as a research area since a few decades. More than data analysis or pattern recognition, Visual Analytics aims at knowledge discovery.

The visualisation pipeline and the Sense-Making loop places users at the centre of the knowledge discovery process (van Wijk 2005). While the pipeline maps to a high level data processing architecture, it is human-computer interaction that enables users to drive data foraging. As a consequence, identifying why and how a given visualisation technique is more "efficient" in supporting human driven knowledge discovery is crucial. Controlled user experiments, often exercised in the area of

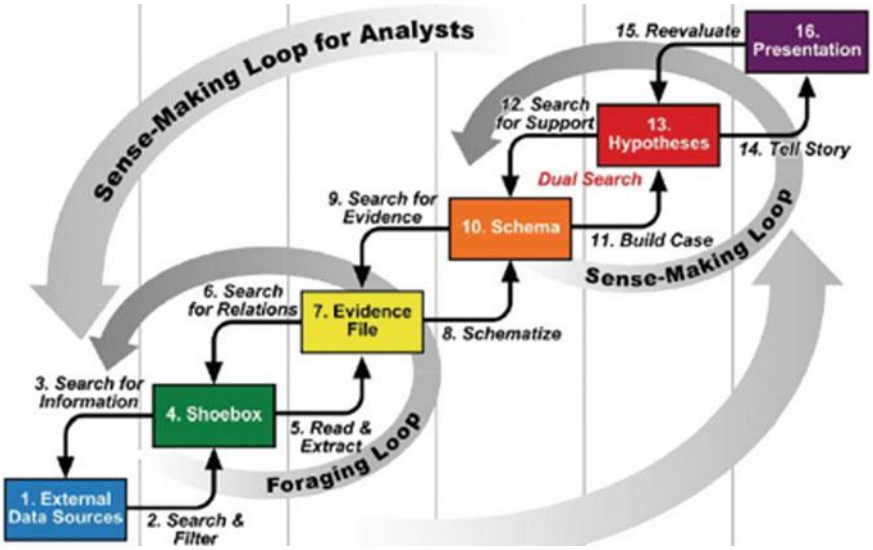


Fig. 5 The "Sense-Making Loop" illustrates the iterative process relying on visualisation as a main driver of the analytical process (Thomas and Cook 2006)

human-computer interaction studies, has now become common practice to validate visualization techniques (Purchase 2012; Sedlmair et al. 2012; Isenberg et al. 2013).

These controlled experiments indeed can help compare techniques supporting low-level, fine grained tasks (Amar et al. 2005; Lee et al. 2006). Knowledge discovery however is a high-level cognitive process. Controlled experiments can hardly validate that a technique (or more often a combination of techniques) favors knowledge discovery. Evaluation high-level cognitive processes does not boil down to error rates and time to task completion. They are more conveniently described in terms of design principles and best practices. Munzner (2009) (see also Meyer et al. 2012, Brehmer and Munzner 2013) proposes a nested model for the design of visualisation systems where data abstractions and user interactions are derived from domain questions. User tasks derived from these questions then specify requirements on visual encodings, including interaction, relevant visual representations and visual variables. Validating of a visualisation then unfolds in the opposite direction: algorithms must run under proper time complexity to insure fluidity of user action and are measured either theoretically or tested against benchmark datasets. Controlled experiments may be used to assess the efficiency of combined visual encodings and interaction techniques. User interviews can assess usability of the overall system (as an aid to decision making, for instance). User adoption of the system is the ultimate demonstration of the value of a visualisation technique (van Wijk 2005).

Visualisation can thus reveal structures hidden in data that is uncovered by analysis and mining approaches. It becomes a natural companion to artificial intelligence precisely because it engages human cognitive capabilities and intelligence. And be-

cause it addresses complex phenomenon emerging from real world situations, often explored and examined in an uncertain and changing environment.

7 Conclusion

The main goal of this chapter was to provide an overview of cross-fertilisation between HCI and AI focused on several key representative research areas without exhaustivity constraints. Significant work at the intersection of these two fields started forty years ago in the aeronautical field, closely combining intelligence and interactivity. This work is now continuing in different areas (simulation, semantic web, e-commerce, social networks, complex dynamic systems, ambient intelligence, and so on). Smart user interfaces quickly took advantage of the complementarity of HCI and AI to become a multifaceted domain. Approaches to user interfaces and affective embodied conversational agents, put forward in this chapter, have particularly promising prospects. Capitalization, formalization and operation of ergonomic knowledge for the design and evaluation of interactive systems have been the subject of many studies since the early 1980s. This work will be expanded, since the potential of HCI and AI cross-fertilization will continue to be a huge endeavor.

Finally, visualization and data mining, which have close ties, have been reviewed in the last section of this chapter, are also particularly representative of areas where HCI and AI naturally merge.

References

Abed M, Anli A, Kolski C, Grislin-Le Strugeon E (2011) A generic method for personalizing interactive systems: application to traveler information. In: Kolski C (ed) Human-computer interactions in transport. ISTE Ltd and Wiley, London, pp 51–91

Abowd GD, Dey AK, Brown PJ, Davies N, Smith M, Steggles P (1999) Towards a better understanding of context and context-awareness. In: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, HUC '99, pp 304–307

Akiki PA, Bandara AK, Yu Y (2014) Adaptive model-driven user interface development systems. ACM Comput Surv 47(1):9:1–9:33

Amar R, Eagan J, Stasko J (2005) Low-level components of analytic activity in information visualization. In: IEEE symposium on information visualization. IEEE Computer Society, pp 111–117

Andrienko N, Andrienko G (2006) Exploratory analysis of spatial and temporal data—a systematic approach. Springer, Berlin

Asimov I (2008) I robot. Random House Publishing Group

Baccino T, Bellino C, Colombi T (2005) Mesure de l'utilisabilité des interfaces. TIC et Sciences Cognitive, Hermes

Bastien JMC, Scapin DL (1993) Ergonomic criteria for the evaluation of human-computer interfaces. Technical report 156, INRIA, Rocquencourt

Bastien JMC, Scapin DL (2001) Evaluation des systèmes d'information et critères ergonomiques. In: Kolski C (ed) Environnement évolués et évaluation de l'IHM, Interaction Homme-Machine pour les SI, vol 2. Hermes, Paris, pp 53–80

Bauer C, Dey AK (2016) Considering context in the design of intelligent systems: current practices and suggestions for improvement. J Syst Softw 112:26–47
Bauer JS, Newman MW, Kientz JA (2014) What designers talk about when they talk about context. Hum Comput Interact 29(5–6):420–450
Beale R, Creed C (2009) Affective interaction: how emotional agents affect users. Int J Hum Comput Stud 67(9):755–776
Beaudouin-Lafon M, Mackay W (2003) Prototyping tools and techniques. In: Jacko JA, Sears A (eds) The human computer handbook: fundamentals, evolving technologies, and emerging applications. L. Erlbaum Associates Inc, Hillsdale, pp 1006–1031
Beka Be Nguema M, Kolski C, Malvache N, Waroux D (2000) Design of a human-error-tolerant interface using fuzzy logic. Eng Appl Artif Intell 13(3):279–292
Bellot P (2011) Recherche d'information contextuelle, assistée et personnalisée. Hermes Science Publications/Lavoisier, Collection recherche d'information et web
Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am Mag 284:34–43
Bertin J (1998) Sémiologie Graphique: Les Diagrammes, Les Réseaux, Les Cartes. Les réimpressions, Ecole des Hautes Etudes en Sciences Sociales
Boy G (1991a) Intelligent assistant systems. Academic Press, New York
Boy GA (1991b) Indexing hypertext documents in context. In: Proceedings of hypertext '91. ACM Press, New York, pp 51–61
Boy GA (1992) Semantic correlation in context: application in document comparison and group knowledge design. In: Boose JH, Clancey W, Gaines B, Rappaport A (eds) AAAI spring symposium on cognitive aspects of knowledge acquisition. Stanford University, CA, USA
Boy GA (1998) Cognitive function analysis. Ablex/Greenwood, Westport
Boy GA (2013) Orchestrating human-centered design. Springer, U.K
Brehmer M, Munzner T (2013) A multi-level typology of abstract visualization tasks. IEEE Trans Vis Comput Graph 19(12):2376–2385
Brossard A, Abed M, Kolski C (2011) Taking context into account in conceptual models using a model driven engineering approach. Inf Softw Technol 53(12):1349–1369
Brusilovsky P, Kobsa A, Nejdl W (eds) (2007) The adaptive web: methods and strategies of web personalization. Springer, Heidelberg
Calvary G, Coutaz J, Thevenin D, Limbourg Q, Bouillon L, Vanderdonckt J (2003) A unifying reference framework for multi-target user interfaces. Interact Comput 15(3):289–308
Calvary G, Coutaz J, Dâassi O, Balme L, Demeure A (2004) Towards a new generation of widgets for supporting software plasticity: The "comet". In: Engineering human computer interaction and interactive systems, joint working conferences EHCI-DSVIS 2004, Hamburg, Germany, 11–13 July 2004, Revised selected papers, pp 306–324
Calvary G, Pribeanu C, Santucci G, Vanderdonckt J (eds) (2008) Computer-aided design of user interfaces V—sixth international conference of computer-aided design of user interfaces 2006. Springer, Bucharest
Canamero L (2003) Designing emotions for activity selection in autonomous agents. In: Petta P, Payr S, Trappl R (eds) Emotions in humans and artifacts. MIT Press, Cambridge, pp 115–148
Caplier A (2011) Visual emotion recognition: status and key issues. In: Pelachaud C (ed) Emotion-oriented systems. Wiley, New York
Card S, Mackinlay J, Shneiderman B (1999) Readings in information visualization. Morgan Kaufmann Publishers, San Francisco
Card SK, Moran TP, Newell A (1983) The psychology of human-computer interaction. Lawrence Erlbaum Associates, Hillsdale
Cassell J (2000) More than just another pretty face: embodied conversational interface agents. Commun ACM 43:70–78
Chi EH (2000) A taxonomy of visualization techniques using the data state reference model. In: IEEE symposium on information vizualization. IEEE Computer Society, pp 69–75
Chignell MH, Hancock PA (1988) Intelligent interface design. In: Helander M (ed) Handbook of human-computer interaction. Elsevier, Amsterdam, pp 969–995

Chollet M, Ochs M, Pelachaud C (2017) A methodology for the automatic extraction and generation of non-verbal signals sequences conveying interpersonal attitudes. IEEE Trans Affect Comput
Clavel C, Richard G (2011) Recognition of acoustic emotion. In: Pelachaud C (ed) Emotion-oriented systems. Wiley, Hoboken
Coutaz J, Calvary G (2012) HCI and software engineering for user interface plasticity. CRC Press, Taylor and Francis Group, Boca Raton, pp 1195–1220
Coutaz J, Rey G (2002) Foundations for a theory of contextors. In: Kolski C, Vanderdonckt J (eds) Computer-aided design of user interfaces III In: Fourth international conference of computer-aided design of user interfaces, vol 3. Kluwer Academic, Dordrecht, pp 13–34
Daassi C, Nigay L, Fauvet MC (2006) A taxonomy of temporal data visualization techniques. Rev Inf Interact Intell 5(2):41–63
deRosis F, Pelachaud C, Poggi I, Carofiglio V, Carolis BD (2003) From Greta's mind to her face: modelling the dynamics of affective states in a conversational embodied agent. Int J Hum-Comput Stud 59(1–2):81–118
Dey AK (2001) Understanding and using context. Pers Ubiquitous Comput 5(1):4–7
dos Santos S, Brodlie K (2004) Gaining understanding of multivariate and multidimensional data through visualization. Comput Graph 28(3):311–325
Edmonds EA (1981) Adaptive man-computer interfaces. In: Coombs MJ, Alty JL (eds) Computing skills and the user interface. Academic, London, pp 389–426
Ezzedine H, Trabelsi A, Tran C, Kolski C (2012) Criteria and methods for interactive system evaluation: application to a regulation post in the transport domain. In: Hammadi S, Ksouri M (eds) Advanced mobility and transport engineering. ISTE Wiley, London, pp 183–230
Findlater L, Gajos KZ (2009) Design space and evaluation challenges of adaptive graphical user interfaces. AI Mag 30(4):68–73
Ford KM, Glymour C, Hayes PJ (1997) Cognitive prostheses. AI Mag 18(3):104–105
Gallud JA, Tesoriero R, Penichet VMR (eds) (2011) Distributed user interfaces - designing interfaces for the distributed ecosystem. Springer, London
Gallud JA, Lozano MD, Vanderdonckt J (2014) Distributed user interfaces: usability and collaboration. Int J Hum Comput Stud 72(1):44. https://doi.org/10.1016/j.ijhcs.2013.10.006
Germanakos P, Belk M (2016) Human-centred web adaptation and personalization - from theory to practice. Human-computer interaction series. Springer, Berlin
Gonzalez Calleros J, Meixner G, Paterno F, Pullmann J, Raggett DS, Vanderdonckt J (2010) Model-based ui xg final report. In: Cantera Fonseca J (ed) W3C incubator group report
Grammenos D, Akoumianakis D, Stephanidis C (2000) Integrated support for working with guidelines: the sherlock guideline management system. Interact Comput 12(3):281–311
Gratch J, Marsella S (2004) A domain-independent framework for modeling emotion. J Cogn Syst Res 5(4):269–306
Grudin J (1994) Computer-supported cooperative work: history and focus. Computer 27(5):19–26
Grudin J (2009) AI and HCI: two fields divided by a common focus. AI Mag 30(4):48–57
Hancock PA, Chignell MH (eds) (1989) Intelligent interfaces: theory, research and design. North-Holland, Amsterdam
Herman I, Marshall MS, Melançon G (2000) Graph visualisation and navigation in information visualisation: a survey. IEEE Trans Vis Comput Graph 6(1):24–43
Hewett T, Baecker R, Card S, Carey T, Gasen J, Mantei M, Perlman G, Strong G, Verplank W (1992) ACM SIGCHI curricula for human-computer interaction. ACM, New York
Hofer G, Richmond K (2010) Comparison of HMM and TMD methods for lip synchronisation. In: Kobayashi T, Hirose K, Nakamura S (eds) INTERSPEECH 2010, 11th annual conference of the international speech communication association, Makuhari, Chiba, Japan, 26–30 September 2010. ISCA, pp 454–457
Hoffman PE, Grinstein GG (2002) A survey of visualizations for high-dimensional data mining. In: Fayyad U, Grinstein G, Wierse A (eds) Information visualization in data mining and knowledge discovery. Morgan Kaufmann Publishers, San Francisco, pp 47–82

Hoffmann L, Krämer NC, Lam-Chi A, Kopp S (2009) Media equation revisited: do users show polite reactions towards an embodied agent? International workshop on intelligent virtual agents. Springer, Berlin, pp 159–165

Hook K (2000) Steps to take before intelligent user interfaces become real. Interact Comput 12:409–426

Huart J, Kolski C, Bastien C (2008) L'évaluation de documents multimédias, état de l'art. In: Merviel S (ed) Objectiver l'humain? Qualification, quantification, vol 1. Hermes, Paris, pp 211–250

Hussmann H, Meixner G, Zuehlke D (eds) (2011) Model-driven development of advanced user interfaces. Studies in computational intelligence, vol 340. Springer, Berlin

Hutchins E (1995) Cognition in the wild. MIT Press, Cambridge

Isenberg T, Isenberg P, Chen J, Sedlmair M, Möller T (2013) A systematic review on the practice of evaluating visualization. IEEE Trans Vis Comput Graph 19(12):2818–2827

Ivory MY, Hearst MA (2001) The state of the art in automating usability evaluation of user interfaces. ACM Comput Surv 33(4):470–516

Jacko JA (ed) (2012) The human computer handbook: fundamentals, evolving technologies, and emerging applications, 3rd edn. CRC Press, Boca Raton

Jacob RJK, Limbourg Q, Vanderdonckt J (eds) (2004) Computer-Aided Design of User Interfaces IV - Fifth International Conference of Computer-Aided Design of User Interfaces. Kluwer, Funchal

Jambon F, Brun P, Ait-Ameur Y (2001) Spécification des systèmes interactifs. In: Kolski C (ed) Analyse et Conception de l'IHM. Interaction Homme-Machine pour les SI, vol 1. Editions Hermes, Paris, pp 175–206

Jameson A (2007) Adaptive interfaces and agents. In: Sears A, Jacko JA (eds) The human computer handbook: fundamentals, evolving technologies, and emerging applications. CRC Press, Boca Raton, pp 433–458

Jameson A, Gajos KZ (2012) Systems that adapt to their users. In: Jacko JA (ed) The human computer handbook: fundamentals, evolving technologies, and emerging applications. CRC Press, Boca Raton, pp 431–456

Jaques PA, Viccari RM (2004) A BDI approach to infer student's emotions. In: The proceedings of the Ibero-American conference on artificial intelligence (IBERAMIA). Springer, Puebla, pp 901–911

Kolski C, Le Strugeon E (1998) A review of intelligent human-machine interfaces in the light of the arch model. Int J Hum-Comput Interact 10(3):193–231

Kolski C, Vanderdonckt J (eds) (2002) Computer-aided design of user interfaces III - fourth international conference of computer-aided design of user interfaces. Kluwer, Valenciennes

Kolski C, Tendjaoui M, Millot P (1992) A process method for the design of "intelligent" man-machine interfaces: case study: "the decisional module of imagery". Int J Hum Factors Manuf 2(2):155–175

Kubicki S, Lebrun Y, Lepreux S, Adam E, Kolski C, Mandiau R (2013) Simulation in contexts involving an interactive table and tangible objects. Simul Model Pract Theory 31:116–131

Lazarus RS (1991) Emotion and adaptation. Oxford University Press, Oxford

Lebrun Y, Adam E, Mandiau R, Kolski C (2015) A model for managing interactions between tangible and virtual agents on an RFID interactive tabletop: case study in traffic simulation. J Comput Syst Sci 81(3):585–598

Lee B, Plaisant C, Parr CS, Fekete J, Henry N (2006) Task taxonomy for graph visualization. In: AVI workshop on beyond time and errors: novel evaluation methods for information visualization BELIV '06. ACM, New York, pp 1–5

Lieberman H (1995) Letizia: an agent that assists web browsing. In: Fourteenth international joint conference on artificial intelligence, IJCAI 95, pp 924–929

Limbourg Q, Vanderdonckt J, Michotte B, Bouillon L, Florins M, Trevisan D (2004) Usixml: a user interface description language for context-sensitive user interfaces. In: ACM AVI'2004 workshop "developing user interfaces with xml: advances on user interface description languages", pp 55–62

Lozano MD, Mashat AS, Fardoun HM, Gallud JA, Penichet VMR, Tesoriero R, Vanderdonckt J (eds) (2013) Distributed user interfaces: models, methods and tools, DUI 2013. In: Conjunction with ACM EICS 2013 conference, London, UK, 24 June 2013

Mandiau R, Kolski C, Millot P, Chaib-Draa B (1991) A new approach for the cooperation between human(s) and assistance system(s): a system based on intentional states. In: Liebowitz J (ed) Expert systems world congress proceedings, vol 3. Pergamon Press, New-York, pp 1672–1679

Maybury MT (1999) Intelligent user interfaces: an introduction (tutorial notes)

de Melo CM, Carnevale P, Read S, Antos D, Gratch J (2012) Bayesian model of the social effects of emotion in decision-making in multiagent systems. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems - volume 1, international foundation for autonomous agents and multiagent systems, Richland, SC, AAMAS '12, pp 55–62

Meyer M, Sedlmair M, Munzner T (2012) The four-level nested model revisited: Blocks and guidelines. In: Workshop on beyond time and errors: novel evaluation methods for information visualization (BELIV)

Moussa F, Ismail I, Jarraya M (2015) Towards a runtime evolutionary model of user-adapted interaction in a ubiquitous environment: the RADEM formal model. Cogn Technol Work 17(3):391–415

Munzner T (2009) A nested process model for visualization design and validation. IEEE Trans Vis Comput Graph 15:921–928

Nielsen J (1993) Usability engineering. Academic, New York

Niewiadomski R, Pelachaud C (2007) Fuzzy similarity of facial expressions of embodied agents. In: Proceedings of the 7th international conference on intelligent virtual agents. Springer, Berlin, pp 86–98

Norman DA (1986) Cognitive engineering. In: Norman DA, Draper SW (eds) User centered system design: new perspectives on human-computer interaction. Lawrence Erlbaum Associates, Hillsdale, pp 31–61

Ochs M, Niewiadomski R, Brunet P, Pelachaud C (2012a) Smiling virtual agent in social context. Cogn Process 13(2):519–532

Ochs M, Sadek D, Pelachaud C (2012b) A formal model of emotions for an empathic rational dialog agent. Auton Agents Multi-Agent Syst 24(3):410–440

de Oliveira KM, Bacha F, Houda M, Abed M (2013) Transportation ontology definition and application for the content personalization of user interfaces. Expert Syst Appl 40(8):3145–3159

Paterno F (1999) Model-based design and evaluation of interactive applications, 1st edn. Springer, London

Peintner B, Viappiani P, Yorke-Smith N (2008) Preferences in interactive systems: technical challenges and case studies. AI Mag 29(4):13–24

Pelachaud C (2009) Modeling multimodal expression of emotion in a virtual agent. Philos Trans R Soc B Biol Sci 364:3539–3548

Picard R (1997) Affective computing. MIT Press, Cambridge

Porhet C, Ochs M, Saubesty J, Montcheuil G, Bertrand R (2017) Mining a multimodal corpus of doctor's training for virtual patient's feedbacks. In: Proceedings of 19th ACM international conference on multimodal interaction (ICMI), Glasgow, UK

Provost F, Fawcett T (2013) Data science and its relationship to big data and data-driven decision making. Big Data 1(1):51–59

Purchase HC (2012) Experimental human-computer interaction: a practical guide with visual examples. Cambridge University Press, Cambridge

Rouse WB, Morris NM (1987) Conceptual design of a human error tolerant interface for complex engineering systems. Automatica 23(2):231–235

Salovey P, Bedell B, Detweiler J, Mayer J (2000) Current directions in emotional intelligence research. In: Lewis M, Haviland-Jones J (eds) Handbook of emotions. Guilford Press, New York, pp 504–520

Scapin DL (1986) Guide ergonomique de conception des interfaces homme-ordinateur. Technical report 77, INRIA, Le Chesnay

Scherer K (2000) Emotion. In: Hewstone M, Stroebe W (eds) Introduction to social psychology: a European perspective. Oxford Blackwell Publishers, Oxford, pp 151–191
Schneider-Hufschmidt M, Khme T, Malinkowski U (eds) (1993) Adaptive user interfaces. North Holland, Amsterdam
Sedlmair M, Meyer M, Munzner T (2012) Design study methodology: reflections from the trenches and the stacks. IEEE Trans Vis Comput Graph (Proc InfoVis 2012) 18(12):2431–2440
Seffah A, Vanderdonckt J, Desmarais M (eds) (2009) Human-centered software engineering, software engineering models, patterns and architectures for HCI, vol 2. Springer, London
Shadbolt N, Hall W, Berners-Lee T (2006) The semantic web revisited. IEEE Intell Syst 21(3):96–101
Shneiderman B (1996) The eyes have it: a task by data type taxonomy for information visualization. In: IEEE conference on visual languages (VL'96). IEEE CS Press, pp 336–343
Shneiderman B, Maes P (1997) Direct manipulation versus interface agents. Interactions 4(6):42–61
Silva SF, Catarci T (2000) Visualization of linear time-oriented data: a survey. In: Proceedings of the first international conference on web information systems engineering, vol 1, pp 310–319
Smith SL, Mosier JN (1986) Guidelines for designing user interface software. Technical report MTR-10090, ESD-TR-86-278, MITRE Corp., Bedford, MA
Sottet J, Calvary G, Favre J, Coutaz J (2009) Megamodeling and metamodel-driven engineering for plastic user interfaces: MEGA-UI. In: Human-centered software engineering - software engineering models, patterns and architectures for HCI, pp 173–200
Stephanidis C (2009) The universal access handbook. CRC Press, Boca Raton
Stewart T, Travis D (2002) Guidelines, standards, and style guides. In: Jacko JA, Sears A (eds) The human computer handbook: fundamentals, evolving technologies, and emerging applications. L. Erlbaum Associates Inc, Hillsdale, pp 991–1005
Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. Adv Artif Intell 2009:1–19. Article ID 421,425
Szekely P (1996) Retrospective and challenges for model-based interface development. Design, specification and verification of interactive systems, vol 96. Springer, Berlin, pp 1–27
Thevenin D, Coutaz J (1999) Plasticity of user interfaces: framework and research agenda. In: Sasse A, Johnson C (eds) IFIP TC.13—Human-computer interaction, INTERACT 99. IOS Press
Thomas JJ, Cook KA (2006) Illuminating the path: the research and development agenda for visual analytics. IEEE Computer Society
Tran CD, Ezzedine H, Kolski C (2013) Eiseval, a generic reconfigurable environment for evaluating agent-based interactive systems. Int J Hum Comput Stud 71(6):725–761
van den Bergh J (2006) High-level user interface models for model-driven design of context-sensitive user interfaces. PhD thesis, Hasselt University, Belgium
Vanderdonckt J (1994) Guide ergonomique des interfaces homme-machine. Presses Universitaires de Namur, Namur, Facultés universitaires Notre-Dame de la Paix, Namur, Belgium
Vanderdonckt J (1999) Development milestones towards a tool for working with guidelines. Interact Comput 12(2):81–118
Vanderdonckt J, Coyette A (2007) Modèles, méthodes et outils de support au prototypage multi-fidélité des interfaces graphiques. Revue d'Interaction Homme-Machine 8(1):91–123
Vanderdonckt J, Puerta AR (eds) (1999) Computer-aided design of user interfaces II - third international conference of computer-aided design of user interfaces. Kluwer, Louvain-la-Neuve
van Wijk J (2005) The value of visualization. In: Silva C, Groeller E, Rushmeier H (eds) IEEE visualization. IEEE Computer Society, pp 79–86
von Landesberger T, Kuijper A, Schreck T, Kohlhammer J, van Wijk JJ, Fekete JD, Fellner DW (2011) Visual analysis of large graphs: state-of-the-art and future research challenges. Comput Graph Forum 30(6):1719–1749
Ware C (2000) Information visualization: perception for design. Morgan Kaufmann Publishers, San Francisco
Winograd T (2001) Architectures for context. Hum-Comput Interact 16(2):401–419

Woodson W, Conover D (1964) Human engineering guide for equipment designers. University of California Press, Berkeley

Zhang L, Stoffel A, Behrisch M, Mittelstadt S, Schreck T, Pompl R, Weber S, Last H, Keim D (2012) Visual analytics for the big data era?a comparative review of state-of-the-art commercial systems. In: 2012 IEEE conference on visual analytics science and technology (VAST). IEEE, pp 173–182

Robotics and Artificial Intelligence

Malik Ghallab and Félix Ingrand

Abstract Robotics is an interdisciplinary research field leveraging on control theory, mechanical engineering, electronic engineering and computer science. It aims at designing machines able to perceive, move around and interact with their environment in order to perform useful tasks. Artificial Intelligence (AI) is an area of computer science, overlapping with, but significantly distinct from robotics. Its purpose includes the development of computational models of intelligence, as well as the design and experiment with systems which implement these models. There is a significant convergence between Robotics and AI. Their intersection is critical for both areas. Robots implement a "*perception - decision - action*" loop. The decision making part is central in that loop for tackling variable environments and tasks. On the other hand, AI is broadening an initial focus on abstract tasks, as in mathematics and board games, to addressing embodied intelligence. This chapter covers some of the research topics and approaches in the intersection of robotics and AI. It surveys the state of the art in key issues such as planning and acting deliberately on the basis of tasks and world models, learning these models, and organizing the sensory-motor and cognitive functions of a robot into resilient and scalable architectures.

1 Introduction

Robotics and *Artificial Intelligence* are two overlapping but quite distinct research fields. This chapter surveys the state of the art at their intersection. Its purpose is to introduce the reader to the synergies between Robotics and Artificial Intelligence and to demonstrate that their overlap is a very rich and fruitful in scientific problems.

Robotics aims at designing machines which are able to perceive, move around and interact with their environment in order to perform some specified useful tasks. It is an interdisciplinary research field, which covers several disciplines, primarily control

M. Ghallab (✉) · F. Ingrand
LAAS-CNRS, University of Toulouse, Toulouse, France
e-mail: malik@laas.fr

F. Ingrand
e-mail: felix@laas.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_12

theory, mechanical engineering, electronic engineering and computer science. Its recent links with life sciences or materials sciences have opened new and exciting perspectives. It entertains growing synergies with neuroscience for the development of cognitive models and functions (e.g., Wolpert and Flanagan 2016, 2010; Wolpert and Ghahramani 2000). Robotics, as an enabling technology, provides a significant technical and conceptual support for the development of several other research fields such as medicine (e.g., surgery, biomechanics), or environment and space sciences (e.g., oceanography or planetology). It addresses a wide spectrum of applications.

Artificial Intelligence (AI) is a research area of computer science, mostly independent from robotics. Its purpose is to understand intelligence through effective computational models, design systems which implement them, and experiment with these systems in order to scientifically evaluate and qualify the proposed models of intelligence. AI entertains interdisciplinary links with mathematical logics, psychology, neuroscience, linguistics, philosophy and other cognitive sciences. It already brought a wealth of mature technologies, such as machine learning techniques, that are now seamlessly integrated in many computerized devices such as smartphones, cameras, web browsers, search engines and semantic web applications.

Robotics is quite often referred to in AI research. It is a natural reference for work on embodied intelligence and for experimental validation. The early beginnings of AI are rich in pioneering projects of autonomous robots, such as Shakey at SRI (Rosen and Nilsson 1966) or the Stanford Cart in the late 60s, and a few years later, Hilare at LAAS (Giralt et al. 1979) or the CMU Rover (Moravec 1983). These, and many other projects since that early period, clearly lie at the intersection of Robotics and AI, seeking to understand, model and design machines that combine autonomous perception, decision and action.

AI has been less frequently referred to in robotics publications. This is due to the breadth of the robotics field. This is also due to the early challenges on which the robotics community has focused. Early robots had reduced autonomy and limited sensing, locomotion and manipulation capabilities. This naturally set the initial challenges more about sensory-motor functions than about deliberation and cognitive functions. Significant progress during the last two decades on the sensory-motor level has, fortunately, put robotics deliberation problems on the limelight.

We are witnessing a growing convergence between Robotics and AI. Their intersection is critical for both areas (Rajan and Saffiotti 2017). Robots have been defined as a *"perception - decision - action"* control loop. The decision part is central in that loop. On the other hand, AI is moving from abstract intelligence, such as playing chess, to addressing *embodied intelligence*. The intersection of Robotics and AI covers in particular the following issues:

- Perception, semantic interpretation of sensory data, environment modeling;
- Acting deliberately: planning and achieving autonomously complex tasks, including navigation in open unknown environments;
- Learning to perceive, to act and behave with improved performance;
- Organizing sensory-motor and deliberation functions in a robot.

The first item is covered in chapter "Artificial Intelligence and Pattern Recognition, Vision, Learning" of this volume, to the exception of a brief mention of some aspects of perception that are specific to robotics. The survey is primarily devoted to the last three items, addressed successively in:

- Sections 3, 4 and 5, which are devoted respectively to motion planning and execution, tasks planning and acting, and interaction with humans or robots;
- Section 6 on learning (which complements the chapters "Statistical Computational Learning" and "Reinforcement Learning" of Volume 1, chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2, and chapter "Artificial Intelligence and Pattern Recognition, Vision, Learning" of this volume); and
- Section 7 on organization and architecture issues.

For a good understanding of the problems discussed here, the chapter starts with a general introduction to robotics and its applications (Sect. 2). It concludes with a short perspective on future research. In each section we have chosen to illustrate with enough technical details some basic techniques, and to refer the reader to the relevant publications for further deepening. A wide coverage of robotics can be found in the handbook of Siciliano and Khatib (2008). A similar coverage for AI is given in the textbook of Russell and Norvig (2002). A good illustration of recent research papers at the intersection of AI and Robotics is given in Special Issue on AI and Robotics (2017).

2 Overview of Robotics

A robot can be defined as a machine able to perform a set of *tasks* in a class of *environments* with some degree of *autonomy* and robustness. As for any natural being, the autonomous capabilities of a robot need to be qualified with respect to the *diversity* of the tasks and environments it can cope with. A robot integrates several components - actuators, sensors, computers, radio transmitters - which ensure in particular the following functions:

- *motion*, with wheels, legs, wings, propellers, caterpillars, fins;
- *manipulation*, with mechanical arms, clamps, hand, cups, specialized tools;
- *perception* by *proprioceptive* sensors which estimate the internal state of the machine: odometer and angular encoders, inclinometer, magnetometer, accelerometer, inertial measurement unit, GPS, and *exteroceptive* sensors, which estimate the environment: camera, laser, radar, spectrometer, IR or ultrasound range finder;
- *communication*, and
- *decision making*.

There are several classes of generic robotics applications corresponding to different classes of environments and tasks. Each such a class emphasizes specific problems depending on the level of autonomy desired for a robot. Well known examples are the following:

- *Manufacturing robots*: robot arms with adapted sensors at fixed positions for tasks such as painting, welding, assembly, loading and unloading a press or machine tools (Hägele et al. 2008);
- *Exploration robots*: mobile robots in outdoor environments (Feron and Johnson 2008) performing terrain mapping, soil analysis, mining (Corke et al. 2008), intervention in a contaminated site, deployment of equipments at the bottom of the ocean (Antonelli et al. 2008), in Antartica or on Mars (Yoshida and Wilcox 2008);
- *Service robots*: mobile robots in indoor environments for cleaning, surveillance, transportation in a shop, a workshop, a clean room or an hospital (Gini et al. 2010);
- *Personal robots*: mobile robots assisting people in professional environments or at home (Prassler and Kosuge 2008);
- *Medical robots*: robots specialized in assisting surgeons, in particular in "noninvasive surgery" (Täubig et al. 2008);
- *Robot carried by human*: exoskeleton allowing the extension of the sensory-motor skills of their carrier (Kazerooni 2008).

This list is not exhaustive. Other classes of robotics applications, such as agriculture, ecology, construction, de-mining or military operations give rise to active research. Specific environments in one of the above application classes, e.g., aerial exploration robotics, lead to particular problems. Finally, cooperation and interaction when the tasks are carried out by several robots or by human - robot teams bring additional challenges.

A key notion in robotics is the *diversity of environments and tasks* a robot must face. The technology is relatively mature when there is no diversity, that is for robots specialized in a single environment, well modeled and instrumented, and on just one well specified task. If one considers manufacturing robots, millions robot arms are operating in the industry (Fig. 1a). In service robotics, numerous autonomous ground vehicles are used in warehouses for logistic services (Guizzo 2008) (Fig. 1b) and in the electronic or pharmaceutical industry. In both cases, the well-modeled stable environment of the robot is the result of a significant engineering effort. The same remark applies to single-task robots, e.g., vacuum cleaner (more than 5 million sold) or lawn mower, which are a large commercial success.

When the environment or tasks are highly variable, the degree of autonomy of the robot becomes an important factor. We may distinguish three levels:

- no autonomy: the robot applies to its actuators pre-recorded or operator specified commands;
- tasks autonomy: the robot performs tasks precisely defined by the operator, e.g., goto point A then pick-up object O;
- autonomy to achieve missions specified in abstract terms, e.g., find and rescue injured persons in the area.

When there is no need for autonomy, many robotics technologies are already mature. This is due in particular to the highly simplified perception and deliberation problems. Robots tele-operated at the task level have been demonstrated in impressive experiments, e.g., in the exploration of planets (Fig. 2a). They are also used in

(a) Baxter, a robot manipulator for manufacturing (Rethink Robotics)

(b) Autonomous vehicles for logistics applications (Kiva Systems)

Fig. 1 Robots **a** for a fixed environment, and **b** for a single task

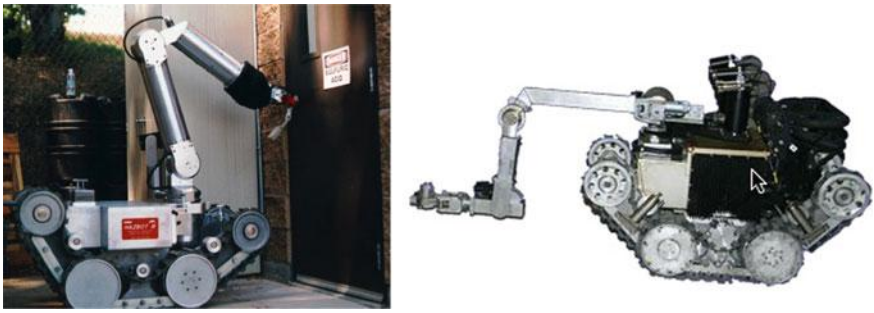
(a) Mars Rover Curiosity (NASA/ JPL)

(b) Surgical robotics assistance (DaVinci Intuitive Surgical)

Fig. 2 Robots tele-operated at the task level or at the motor level

successful applications, e.g., robotics surgery systems have been deployed at several thousands sites, despite their high cost and complexity (Fig. 2b). Remote manipulation has to address other technical challenges, such as how to provide good sensory feedback to a human operator to enable her to properly understand the state of the environment and the task, or how to reliably translate human commands to the robot actuators (*e.g.*, to filter the signal from the movements of the surgeon's hand to obtain a precise and safe trajectory of the scalpel and to control its motion with respect to the movement of the operated organ).

Limited autonomy simplifies perception and deliberation but it also constrains the tasks that can be performed by a tele-operated robot. Thus, Mars rovers of the previous generation, *Spirit* and *Opportunity*, were tele-operated at the motor control level. The communication delay (up to 40 min depending on the Mars-Earth configuration)



(a) NASA/JPL (b) INTRA Group

Fig. 3 Robots for hazardous environments

limited their remote operation to a few meters per day. At a later stage of their mission, the introduction of autonomous motion has allowed these robots to traverse up to 140 m per day. Later, Curiosity performed up to 1.5 km per day of autonomous navigation, but it is still tele-operated at the task level for its exploration activities. In some application, autonomy is not desired: the human operator wants to remain in full control of every command. However, it can be preferable to tele-operate a robot at the task level, e.g., tell it to make a precise line of surgical sutures, or to close an underwater valve, leaving it up to the robot to translate the task into controlled commands, under the supervision of the operator. Here also, the state of the art has reached some maturity, illustrated for example by robots used in hazardous environments (Fig. 3). Another illustration of the autonomy at the task level can be given by telepresence robots. These are mobile platforms carrying away the image and voice of the user, giving a visual and audible feedback, capable of simple tasks, e.g., find a person, asking her to lend an object and bringing it back to the robot's user (Fig. 4).

One may try to use these and similar platforms to achieve more autonomous and varied missions. But the state of the art faces many open problems, in particular for the interpretation of the environment, for planning and acting with incomplete and uncertain models and noisy sensory data.

Autonomy at the mission level already achieves good experimental success when the tasks are well structured and constrained, even when the environment is highly variable. Driverless cars provide a good illustration. The first success goes back to the 2005 "DARPA Grand Challenge": autonomous traversal of 320 km in the Mojave Desert in less than 7 h (Fig. 5a; Thrun 2006), which was followed in 2006 by the "DARPA Urban Challenge". Since then, several companies reported millions of kilometers of autonomous driving on roads and highways (Fig. 5b). Autonomous underwater vehicles (AUV) are another excellent example. Experimental AUVs are launched for up to 24 h in a mission of mapping, water sampling, oceanographic and biological measurement; in case of a problem, the AUV surfaces and indicates its position to be retrieved by its operators (Fig. 5c; McGann et al. 2008).

(a) Double Robotics (b) PadBot

Fig. 4 Telepresence robots

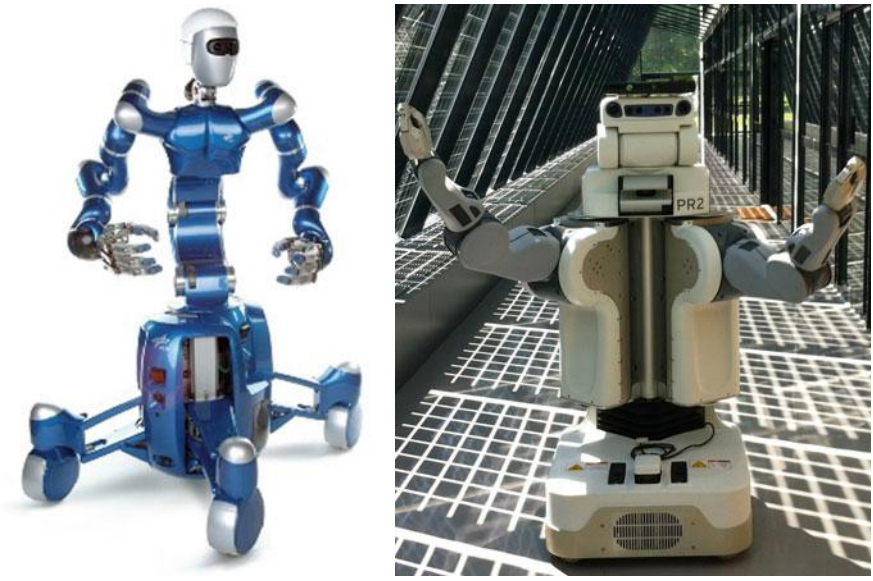
(a) Stanley, DARPA challenge 2005 (Stanford U.) (b) Autonomous Driving (Google) (c) Dorado, AUV (MBARI)

Fig. 5 Autonomous vehicles

Robotics research relies significantly on experiments. The advance of the field has been conditioned by the availability of inexpensive reliable platforms with broad functionalities that are easily deployable and programmable. Significant progress has been witnessed in the last decade. A good illustration is provided by humanoid robots: many research groups have now access to biped robotic platforms of human size (Fig. 6a, b). These robots demonstrate good motor skills as well as impressive mechatronics. Platforms on wheels with two arms, sometimes with an articulated trunk, also illustrate rich sensory-motor capabilities. These platforms are able for example to catch simultaneously two thrown balls (Justin in Fig. 7a), to fold laundry or to play billiards (PR2 in Fig. 7b).

(a) Pyrène (PAL Robotics) (b) Atlas (Boston Dynamics)

Fig. 6 Humanoid robots



(a) Justin (DLR) (b) PR2 at LAAS (Willow Garage)

Fig. 7 Mobile robots with two arms

Several research competitions stimulated the progress of the field. In addition to autonomous driverless cars, there are several other competitions, e.g., in robotics assembly, aerial robotics or humanoid robotics. The robotics soccer competition "*RoboCup*" is very popular. One can be critical for the oversimplifications often introduced in these competitions (artificial or "micro-worlds" problems). However, their effects in terms of attractiveness, visibility and team commitment, especially among students, remain largely beneficial to the progress of robotics.

3 Motion Planning, Mapping and Navigation

Mobility is a critical and widely studied function for autonomous robots (Latombe 1991; Choset et al. 2005; LaValle 2006). When the environment is well modeled, the movements of a robot can be planned and controlled in a robust manner. Otherwise, the robot has to explore its environment to acquire the needed geometrical and topological models. Let us discuss here these two problems of motion planning and environment modeling.

3.1 Motion Planning with Probabilistic Road Maps

We assume that the environment is described by a geometric model (such as a *Computer-Aided Design* model), which specifies the geometry of the obstacles and the free space. The robot is modeled by its kinematics, i.e., the set of degrees of freedom and the constraints of its moving limbs, as well as its dynamics, i.e., masses and inertia of its components, and the forces and torques of its actuators.

Motion planning consist in finding a trajectory for connecting an initial position to a goal position. This trajectory should be feasible in space and time. The problem is usually decomposed into two steps: *(i)* find a feasible path that satisfies the kinematics constraints of the robot and the geometric constraints of the environment, and *(ii)* find a control law along that path that satisfies the dynamic constraints of the robot. In simple cases these two problems *(i)* and *(ii)* can be solved independently. When there are no moving obstacles and the robot dynamic constraints are weak (e.g., slow motion), it is generally easy to map a feasible path into a feasible trajectory with simple control laws. Motion planning in robotics reduces mainly to a path planning problem, which we detail below.

A free rigid object in Euclidean space without kinematic constraint is characterized by six configuration parameters: (x, y, z) for the position of a reference point and three angles for the orientation of the solid in space. But a robot has kinematic constraints that restrict its movements. For example, a car in the plan has three configuration parameters (x, y and orientation θ), which generally are not independent (a car cannot move laterally). The PR-2 robot (Fig. 7b) has 20 configuration parameters (3 for the base, one for the trunk, 2 for the head, and 7 per arm). The humanoid robot HRP-4 (Fig. 6a) has 32 configuration parameters plus five for each hand.

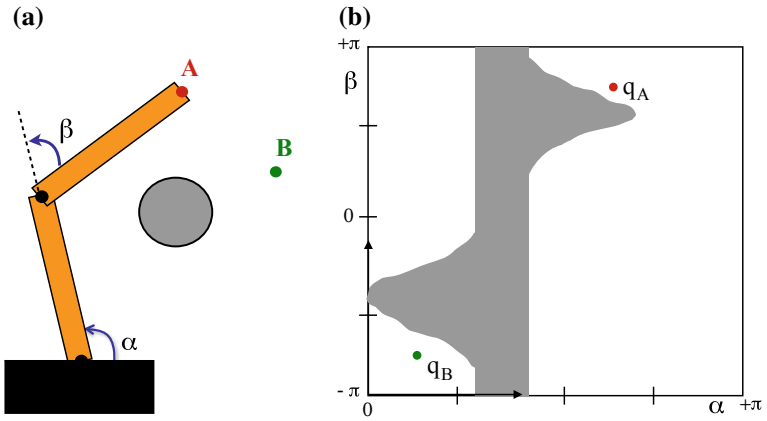


Fig. 8 **a** A planar robot with two angular joints, α and β facing a circular obstacle. **b** Corresponding configuration space: the projection of the obstacle in $\mathscr{C}$ shows that the two configuration q_A and q_B are not connected: no motion of the robot can move it from points A to B

For a robot with n configuration parameters in a given environment let us define:

- $q \in \Re^n$, the configuration of the robot, a vector of n real values that specifies the n parameters characterizing the position of the robot in a reference frame;
- $\mathscr{C}$, the configuration space of the robot, which describes all possible values of q in $\Re^n$ given the kinematic constraints, such as the max and min angular positions that each joint can have, and the dependencies between configuration parameters;
- $\mathscr{C}_f \subseteq \mathscr{C}$, the free configuration space which gives all possible values of $q \in \mathscr{C}$ given the constraints of the environment, i.e., the set of configurations for which the robot does not collide with obstacles.

These concepts are illustrated in Fig. 8 for a robot with two degrees of freedom.[1]

Planning a motion between an origin configuration q_o and a goal configuration q_g, both in $\mathscr{C}_f$, consists in finding a path between q_o and q_g in this n dimensional continuous space. The major difficulty here, as for any other planning problem, is that the search space $\mathscr{C}_f$ is not known explicitly. The explicit definition of $\mathscr{C}_f$ from the geometric model of the environment and the kinematic constraints of robot is an extremely complex problem, difficult to solve even for very simple robots and environments. In the trivial 2D case of the previous example, this problem corresponds to finding the analytical definition of the grey area in Fig. 8b. Significant research in computational geometry addressed this representation problem, see e.g., Schwartz et al. (1987). It opened the way to sampling-base approaches that helped to circumvent the problem, in particular with the following method.

The Probabilistic Roadmap algorithm of Kavraki et al. (1996) relies on two easily computable operations:

[1] Figure adapted from http://www.cs.cmu.edu/motionplanning/.

Probabilistic Roadmap ($\mathcal{G}$)
  iterate until *Termination*
    sample a random configuration $q \in \mathcal{C}$
    if $q \in \mathcal{C}_f$ then do
      if $\forall q' \in \mathcal{G}$: $\mathcal{L}(q,q') \notin \mathcal{C}_f$ then add q in $\mathcal{G}$ ▷ *(i)*
      else if $\exists q_1, q_2$ in $\mathcal{G}$ such that q_1 and q_2 are not connected
          and $\mathcal{L}(q,q_1) \subset \mathcal{C}_f$ and $\mathcal{L}(q,q_2) \subset \mathcal{C}_f$
          then add q and the edges (q,q_1) and (q,q_2) to $\mathcal{G}$ ▷ *(ii)*
  Return($\mathcal{G}$)

Fig. 9 Probabilistic roadmap algorithm for path planning

- *kinematic guidance*: find a direct kinematic path $\mathcal{L}(q, q')$ between two configurations q and $q' \in \mathcal{C}$ without worrying about environment constraints, i.e., $\mathcal{L}(q, q')$ satisfies the kinematic constraints but not necessarily the constraints of non-collision with obstacles. The techniques used for that are specific to the type of the robot kinematic constraints, e.g. composition of straight lines and curves;
- *collision test*: check whether a configuration q does or does not collide with obstacles, i.e., if $q \in \mathcal{C}_f$; check whether a path $\mathcal{L}(q, q')$ between two configurations is collision-free, i.e., if it passes entirely in $\mathcal{C}_f$. This relies on basic techniques of computational geometry.

A roadmap $\mathcal{G}$ in $\mathcal{C}_f$ is a graph whose vertices are configurations in $\mathcal{C}_f$; two vertices q and q' are adjacent in $\mathcal{G}$ iff there exists a path without collision $\mathcal{L}(q, q')$ in $\mathcal{C}_f$.

If a roadmap $\mathcal{G}$ in $\mathcal{C}_f$ is known, then planning a path between an origin configuration q_o and a goal configuration q_g can be solved with the three following steps:

- find a vertex q in $\mathcal{G}$ such that q is accessible from q_o i.e., $\mathcal{L}(q_o, q) \in \mathcal{C}_f$;
- find a vertex q' in $\mathcal{G}$ such that q_g is accessible from q', i.e., $\mathcal{L}(q', q_g) \in \mathcal{C}_f$;
- find a sequence of adjacent vertices in $\mathcal{G}$ between q and q'.

Path planning is then reduced to a simpler problem of finding a path in graph. If such a sequence of configurations is found, efficient algorithms allow to smooth and optimize locally this sequence of configurations in $\mathcal{G}$ into a kinematic path. It remains therefore to find a map $\mathcal{G}$ covering adequately $\mathcal{C}_f$, i.e., if there is a path in $\mathcal{C}_f$ then there is also a path in the roadmap $\mathcal{G}$ using the previous three steps.

The algorithm in Fig. 9 (Siméon et al. 2000) provides a graph $\mathcal{G}$ which *probabilistically* satisfies this coverage property. This algorithm incrementally generates $\mathcal{G}$ starting with an empty roadmap. It adds to the map under construction a randomly drawn configuration q in the following two cases:

- if q belongs to the free space and extends the coverage of $\mathcal{G}$, allowing to reach parts of $\mathcal{C}_f$ not yet covered (step ▷*(i)*), or
- if q belongs to the free space and extends the connectivity of $\mathcal{G}$, allowing to connect two components not currently connected in the roadmap (step ▷*(ii)*).

The *Termination* condition is based on the number of consecutive samples of unsuccessful random free configurations that do not add anything to the map. If k_{max} is such a number, then the probability that the resulting graph covers $\mathcal{C}_f$ is estimated by

$(1 - 1/k_{max})$. In practice, this algorithm is very efficient. The probabilistic roadmap technique and its incremental variants (called RRT for " Rapidly exploring Random Trees", LaValle 2006) are now widely used in robotics. They are also used in other application areas such as mechanical design, video animation, or computational biology for molecular docking problems to find whether a ligand can bind to a protein. They have been extended to take into account dynamic environments.

These techniques have advanced significantly the state of the art but they do not solve all motion planning problems in robotics. Many open problems remain, in particular for handling the robot dynamics. Further, one needs to synthesize plans that are robust to the uncertainty of the models and to the sensory-motor noise in the robot localization and motion control. For example, we may want a path that relies on known landmarks to maintain the localization uncertainty below an acceptable threshold. In addition, we need to restate the problem for concrete tasks. The previous formulation refers to a completely specified motion problem, i.e., from a configuration q_o to a configuration q_g. In practice, the problem arises with respect to a task, e.g., grasp an object. This leads to several open problems (Siméon et al. 2004). A grasp allows to infer the configuration of the end effector (hand and fingers) from the position of the object to be grasped. But the configuration of the end effector gives only a part of q_g. It is possible to decompose the problem into: *(i)* plan the movement of the base of the robot to a configuration "close" to the object, then *(ii)* plan a movement of the arm to a grasp position. However, the manipulation of an object can require intermediate poses at different moment with respect to the object, or the manipulation of other interfering objects. It is then necessary to change the structure of the search space according to the grasps and poses of objects handled. In addition, the above decomposition is not always feasible. For example, a humanoid robot requires a coordinated movement of its body and all limbs (Kanoun et al. 2011) (Fig. 10). Further, sensing and visibility issues bring additional constraints,

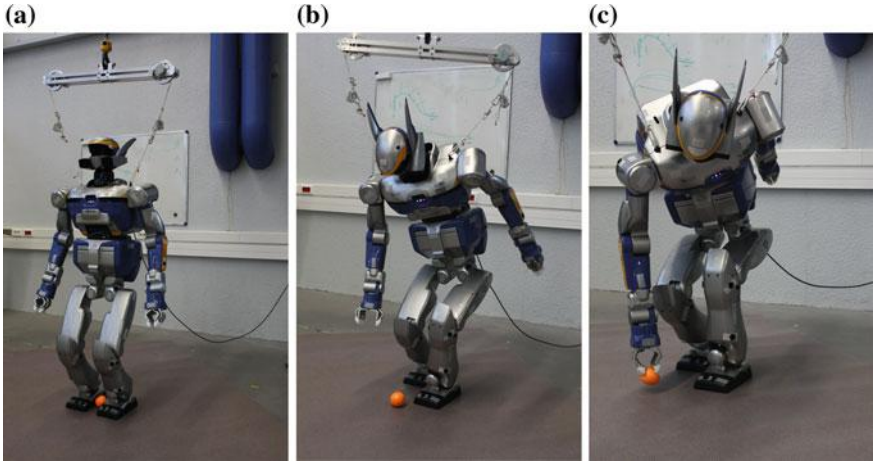


Fig. 10 Picking up a ball requires a coordinated whole body motion planning; here the synthesized plan led the robot to step back, bend and extend opposite arm to maintain its balance (LAAS)

e.g., planning a motion that avoids occultation between a camera carried by the robot's head and its hand, to allow for visual servoing (Chaumette and Hutchinson 2008).

3.2 Simultaneous Localization and Mapping

The execution of a planned motion requires the control of the actuators for achieving a trajectory, possibly with avoidance of unexpected obstacles. The synthesis of this control is done with models and methods from control theory. Robotics raises very interesting problems in automatic control, e.g., in the control of non-holonomic systems. These issues are not within the scope of this chapter. We refer the reader for example to the book of LaValle (2006) or the synthesis of Minguez et al. (2008).

The execution of a planned motion requires also to maintain a good estimate of the state of the robot throughout the execution of the command. In particular, the robot must always know where it is in the environment. Sometimes, one may use absolute localisation, as given by a GPS or a radio-positioning system if the environment provides the adequate infrastructure. However, to operate autonomously in a diversity of environments, a robot must be able to locate itself directly from the perceived natural elements of its environment and a map of this environment. Further, this map is generally partially known, or even unknown. In general a robot is faced with a problem called *simultaneous localization and mapping* (SLAM). This problem has been identified quite early (Chatilla and Laumond 1985; Smith et al. 1986), and has been since a very active research topic in robotics.[2]

To define the problem, let us discuss its two subproblems:

- *Localization*: the robot is localized in a fully known environment, modeled by k landmarks that are easily recognizable and perfectly positioned in space (2D or 3D). At time t, the robot is in a position estimated by $\tilde{x}_t$. It moves with the command u_t (giving the movement speed and orientation between t and t'). This allows to estimate the new position $\tilde{x}'$. The robot observes k landmarks where it expects to find them (from the estimated $\tilde{x}'$). It updates its position in relation to each recognized landmark. The observed positions of the landmarks are combined into a new estimated position of the robot $\tilde{x}_{t+1}$. The process is repeated at each time step as long as the robot remains within a fully known environment. The intermediate estimate $\tilde{x}'$ serves only to find landmarks. The localization error takes into account the sensing errors in the landmark observed positions, but it does not increase with time as long as the landmark locations in the map are error free. The error associated with the motor command u_t does not affect the localization.
- *Mapping*: The robot builds a map of its environment assuming it knows precisely its successive positions. The jth landmark is estimated at time t as $\tilde{x}_{j_t}$. The robot moves between t and $t+1$ to a new known position, from which it observes again the position of the jth landmark as $\tilde{x}'_j$ with sensing error. $\tilde{x}'_j$ and $\tilde{x}_{j_t}$ are combined into a more reliable estimate $\tilde{x}_{j_{t+1}}$. The map quality improves with time.

[2] See, e.g., the software repository: http://www.openslam.org/.

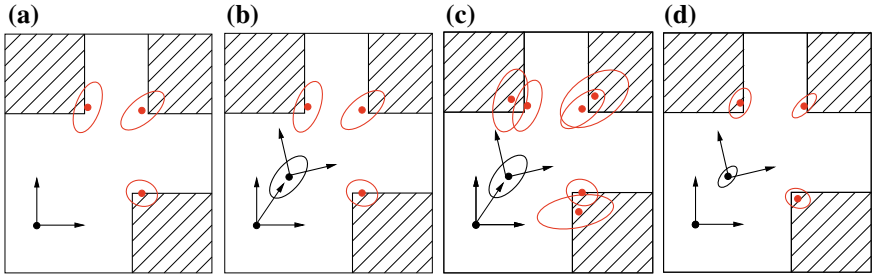


Fig. 11 SLAM procedure for a simple 2D robot: **a** Three landmarks (corners of obstacles) are detected and positioned with inaccuracy due to sensing noise. **b** The robot moves and estimates its position with a motion error. **c** The landmarks are observed and associated with the corresponding ones previously perceived. **d** Data fusion reduces the errors on the current position of the robot and the positions of the landmarks. The process is iterated for each new robot motion and sensing

In practice, the two problems have to be addressed simultaneously. The initial map, if there is one, is never error free. Errors in the map entail localization errors. Symmetrically, the robot localization is noisy, which entails errors in its updates of the map. However, the two sources of error, from sensing and motion, are not correlated (see Fig. 11). It is possible to combine the two subproblems into the simultaneous estimate of the positions of the robot and the landmarks.

One approach initially explored for solving the SLAM relies on *extended Kalman filters*. The technical details may seem complicated but a step by step presentation shows that the principle is simple. It is assumed that the environment is static and the sensors of the robot are properly calibrated and do not introduce systematic bias. Sensing errors are modeled as a Gaussian noise with zero mean and a standard deviation specific to each sensor. Let us assume two sensors, characterized respectively by σ_1 and σ_2, which both measure the distance to the same landmark. They return two values μ_1 and μ_2. We can estimate the true distance by averaging the returned values while giving more confidence to the most accurate sensor, i.e., the one with the smaller σ_i. Hence μ_i is weighted by $1/\sigma_i$. The estimated distance μ is associated with a standard deviation σ defined below (Eq. 1). This estimates has good properties: it minimizes the mean squared error. The error resulting from the combination of the two measures decreases, since $\sigma < \min\{\sigma_1, \sigma_2\}$.

$$\begin{aligned} \mu &= \alpha(\mu_1/\sigma_1 + \mu_2/\sigma_2), \text{ with } \alpha = \sigma_1\sigma_2/(\sigma_1 + \sigma_2) \\ 1/\sigma &= 1/\sigma_1 + 1/\sigma_2 \end{aligned} \tag{1}$$

This process is applied incrementally. We combine the current estimate (μ', σ') to the new measure (μ_z, σ_z). The new estimate at time t (μ_t, σ_t) integrating the new measure is given by the same equation, rearranged easily into the following form (Eq. 2):

$$\begin{aligned} \mu_t &= \mu' + K(\mu_z - \mu') \\ \sigma_t &= \sigma' - K\sigma' \\ K &= \sigma'/(\sigma_z + \sigma') \end{aligned} \tag{2}$$

Let us now introduce the robot's motion. At time $t-1$ the robot was in a position with respect to the landmark of interest estimated by $(\mu_{t-1}, \sigma_{t-1})$. Between $t-1$ and t the robot moves according to a command known with an uncertainty similarly modeled. Let (u_t, σ_u) be the estimate of this motion along the robot - landmark line. This estimate is given by the command sent to actuators and/or by the odometer. The relative distance to the landmark after the motion is estimated by (μ', σ'), noting that the error increases due to the motion:

$$\begin{aligned} \mu' &= \mu_{t-1} + u_t \\ \sigma' &= \sigma_{t-1} + \sigma_u \end{aligned} \tag{3}$$

We now can combine the two previous steps into a SLAM approach based on Kalman filtering. The estimate of the relative position robot - landmark is updated between $t-1$ and t in two steps:

(i) update due to motion (with Eq. 3): $(\mu_{t-1}, \sigma_{t-1}) \rightarrow (\mu', \sigma')$
(ii) update due to sensing (with Eq. 2): $(\mu', \sigma') \quad \rightarrow (\mu_t, \sigma_t)$

In the general case, these updates are applied to vectors instead of simple scalar values. We run the above process to the update of the positions of the robot and the landmarks in the Euclidean space, 2D or 3D. The position of the robot does not necessarily include all its configuration parameters, but only the portion of q necessary for the localization of a reference point and for the positioning of its sensors. The map is characterized by many landmarks positioned in space. A vector μ_t, whose components are the robot configuration parameters and the positions of the landmarks, is updated at each step. The error is no longer a scalar σ_t but a covariance matrix Σ whose element σ_{ij} is the covariance components i and j of the parameters of μ. The error on the position of the robot is coupled to the errors of the map and symmetrically. Furthermore, the above approach applies only to linear relations. But the relationship between the command and the motion is not linear. We approximate a solution to this problem by linearizing around small motions. This leads finally to the extended Kalman filter formulation of SLAM:

$$\begin{aligned} \mu' &= A\mu_{t-1} + Bu_t \\ \mu_t &= \mu' + K_t(\mu_z - C\mu') \\ \Sigma' &= \sigma_{t-1} + \Sigma_u \\ \Sigma_t &= \Sigma' - K_t C \Sigma' \\ K_t &= \Sigma' C^T (C \Sigma' C^T + \Sigma_z)^{-1} \end{aligned} \tag{4}$$

Two update steps are easily identified:

(i) $(\mu_{t-1}, \sigma_{t-1}) \rightarrow (\mu', \Sigma')$: vector u_t, matrices A and B for the motion,
(ii) $(\mu', \Sigma') \rightarrow (\mu_t, \Sigma_t)$: vector μ_z, matrix C for the new measurements.

One also takes into account the covariance associated with the motion and the measurements (Σ_u and Σ_z). It should be noted that the first step uses the motion to update the position of the robot as well as those of the landmarks. Similarly, the second step integrates the new measurements for both, the localization and mapping.

This approach has been successfully implemented and frequently used (Thrun 2002). It has many advantages. In particular, it maintains the robot localization and the corresponding bounds on the error. These bounds are very important in navigation: if the error grows beyond some threshold, specific action has to be taken. The method converges asymptotically to the true map, with a residual error due to initial inaccuracies. Finally, the estimate is computed incrementally. In practice, the number of landmarks increases dynamically. The robot maintains a list of landmark candidates which are not integrated into the map (nor in the vector μ) until a sufficient number of observations of these landmarks have been made. If n is the dimension of the vector μ (i.e., the number of landmarks), the complexity of the update by Eq. 4 is $O(n^2)$. The computations can be done online and on board of the robot for n in the order of 10^3, which means a sparse map.

Particle filtering offers another approach to SLAM with additional advantages. Instead of estimating the Gaussian parameters (μ, Σ), the corresponding probability distributions are estimated through random sampling. Let $P(X_t|z_{1:t}, u_{1:t}) = \mathcal{N}(\mu_t \Sigma_t)$, where X_t is the state vector of the robot and landmark positions at the time t, $z_{1:t}$ and $u_{1:t}$ are the sequences of measures and commands from 1 to t. Similarly $P(z_t|X_{t-1}) = \mathcal{N}(\mu_z \Sigma_z)$.

Let us decompose the state vector X_t into two components related to the robot and the landmarks: $X_t = (r_t, \phi_1, \ldots, \phi_n)^T$, where r_t is the position of the robot at time t, and $\phi = (\phi_1, \ldots, \phi_n)^T$ the position of landmarks, which do not depend on time because the environment is assumed static.[3] The usual rules of joint probabilities entail the following:

$$\begin{aligned} P(X_t|z_{1:t}, u_{1:t}) &= P(r_t|z_{1:t}, u_{1:t}) P(\phi_1, \ldots, \phi_n|z_{1:t}, u_{1:t}, r_t) \\ &= P(r_t|z_{1:t}, u_{1:t}) \prod_{i=1,n} P(\phi_i|z_{1:t}, r_t) \end{aligned} \tag{5}$$

The second line results from the fact that, given the position r_t of the robot, the positions of the landmarks do not depend on u and are conditionally independent. The robot does not known precisely r_t but it assumes that $r_t \in R_t = \{r_t^{(1)}, \ldots, r_t^{(m)}\}$, a set of m position hypotheses (or particles). Each hypothesis $r_t^{(j)}$ is associated with a weight $w_t^{(j)}$. R_t and the corresponding weights are computed in each transition from $t-1$ to t by the following three steps:

[3] Note that in μ_t the estimate ϕ evolves with t, but not the position of the landmarks.

- *Propagation*: for m' positions in R_{t-1} randomly sampled according to the weights $w_{t-1}^{(j)}$, we compute $r_t^{(j)}$ the position at time t of the resulting control u_t, with $m' > m$,
- *Weighting*: the weight $w_t^{(j)}$ of particle $r_t^{(j)}$ is computed taking into account the observation z_t from the product $P(z_t|\phi, r_t^{(j)})P(\phi|z_{1:t-1}, r_{t-1}^{(j)})$.
- *Sampling*: the m most likely assumptions according to the new weights $w_t^{(j)}$ are kept in R_t.

For each of the m particles, the probability $P(\phi_i|z_{1:t}, r_t)$ is computed with a Kalman filter reduced to the 2 or 3 parameters necessary to the position ϕ_i. With good data structures for the map, this approach, called FastSLAM (Montemerlo et al. 2002), reduces the complexity of each update to $O(n\log m)$ instead of $O(n^2)$ in the previous approach. In practice, one can keep a good accuracy for about $m \simeq 10^2$ particles, allowing to maintain online a map with $n \simeq 10^5$ landmarks.

The main limitation of these approaches is due to a well known and difficult problem of *data association*. At each step of the incremental localization process, one must be sure not to confuse the landmarks: associated measurements should be related to the same landmark. An update of the map and the robot positions with measurements related to distinct landmarks can lead to important errors, well beyond the sensory-motor errors. This argument, together with the computational complexity issue, favors sparse maps with few discriminating and easily recognizable landmarks. On a small motion between $t-1$ and t, the landmarks in the sensory field of the robot are likely to be recognized without association errors. But after a long journey, if the robot views some previously seen landmarks, a robust implementation of the approach requires a good algorithm for solving the data association problem.[4] In the particle filtering approach, the probability distribution of R_t is very different when the robot discovers a new place (equally likely distribution) from the case where it retraces its steps. This fact is used by active mapping approaches, which make the robot retrace back its steps as frequently as needed (Stachniss and Burgard 2004).

In the general case, there is a need for an explicit data association step between the two stages *(i)* and *(ii)* corresponding to Eq. 4. This step leads to maintain multiple association hypotheses. The SLAM approaches with Dynamic Bayesian Networks (DBN) for handling multi-hypotheses give good results. The DBN formulation of SLAM is quite natural. It results in a dependency graph (Fig. 12) and the following recursive equation:

$$\begin{aligned} P(X_t|z_{1:t}, u_{1:t}) &= \alpha P(z_t|X_t) \int P(X_t|u_t, X_{t-1}) P(X_{t-1}|z_{1:t-1}, u_{1:t-1}) dX_{t-1} \\ &= \alpha P(z_t|X_t) \int P(r_t|u_t, r_{t-1}) P(X_{t-1}|z_{1:t-1}, u_{1:t-1}) dr_{t-1} \end{aligned} \tag{6}$$

Here, α is a simple normalization factor. The vector state is as above $X_t = (r_t, \phi_1, \ldots, \phi_n)^T$; the second line results from the fact that the environment is assumed static and that the robot motion and landmark positions are independent. The

[4] This is sometimes referred to as the *SLAM loop problem*.

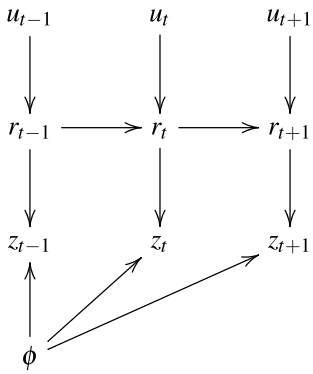


Fig. 12 Formulation of SLAM with a dynamic Bayesian network; arcs stand for conditional dependencies between random variables, ϕ gives the positions of the landmarks (time-independent), u_t, r_t and z_t denote the command, the robot positions and the new measurements at time t

term $P(z_t|X_t)$ expresses the sensory model of the robot, and the term $P(r_t|u_t, r_{t-1})$ corresponds to its motion model. This formulation is solved by classical DBN techniques, using in particular the Expectation-Maximization algorithm (EM), as for example in Ghahramani (1997), which provides a correct solution to the data association problem. However, online incremental implementation of EM are quite complex. Let us also mention another version of FastSLAM which takes this problem into account by an explicit optimization step over all possible associations (Montemerlo et al. 2003).

Recent approaches to SLAM favor this DBN formulation with a global parameter estimation problem overs the set of landmarks and robot positions. The problem is solved by robust optimization methods. This general formulation is called the beam adjustment method; it uses techniques of computer vision and photogrammetry (Triggs et al. 2000). Visual SLAM has also benefited from recent image processing features which are quite robust for the localization and identification of landmarks (Mei and Rives 2007; Newcombe and Davison 2010; Nez-Carranza and Calway 2010).

Let us conclude this section by mentioning a few possible representations for the map of the environment. Landmarks can be any set of sensory attributes that are recognizable and localizable in space. They can be a simple collection of points. They can also be compound attributes, such as visual segments, planes, surfaces, or more complex objects. The most appropriate attributes are generally specific to the type of sensors used. The global map can be represented as a 2D occupancy grid. Simple 3D maps for indoor environments, such as the Indoor Manhattan Representation, combine vertical planes of walls between two horizontal planes for the floor and ceiling, Flint et al. (2011). They can be used with more elaborate representations integrating semantic and topological information (see next section).

3.3 Navigation

The previous approaches are limited to metric maps. They only take into account distances and positions in a global absolute reference. When the environment is large, it is important to explicitly represent its topology, possibly associated with semantic information. In this case, a map relies on hierarchical hybrid representations, with metric sub-maps in local reference frames, together with relationships and connectivity constraints between sub-maps. The robot re-locates itself precisely when arriving in a sub-map.

Navigation in this case is also hybrid. Within a sub-map, motion planning techniques are used. Between sub-maps other methods such as road following or beam heading are more relevant. Sensory aspects and place recognition play an important role in navigation methods for *semantic hierarchy of spatial representations* (Kuipers and Byun 1991).

Mapping and map updates can be as flexible as in the case of SLAM through the updates of a graph of local sub-maps (Kuipers et al. 2004; Estrada et al. 2005). Topological planning relies on path search techniques in graphs (using algorithms such as Dijsktra or A^*, see also chapter "Heuristically Ordered Search in State Graphs" of Volume 2). It is associated with motion planning in sub-maps. Both types of planning can be combined incrementally. Topological planning gives a route which is updated and smoothed incrementally to optimize the motion giving the observed terrain while moving (Konolige et al. 2011).

Topological planning in a graph or within a grid can be used with a partial knowledge of the environment. Extensions of the A^* algorithm (D^* (Stentz 1994), D^* Lite, or Focused D^*) compute shortest paths in the graph, but they use the robot sensing to update the topology and costs parameters for finding shortest paths.

Finally, a classical problem in any hybrid approach is that of the frontiers between levels and their granularity. Labels of places (doors, rooms, corridors) and topology can emerge naturally from sensing and/or from a uniform description of space into cells (grids, polygons or Delaunay triangles). Decomposition techniques by *quadtrees* (a partially occupied cell is decomposed recursively) are useful but can be computationally complex. Analysis of the levels of connectivity of a graph provides elegant solutions with low complexity when the topological graph is planar (Hopcroft and Tarjan 1973; Laumond 1990).

4 Task Planning and Acting

Task planning is the problem of synthesizing a plan, i.e., a sequence or a structured set of actions, starting from the description of all possible actions that a robot can perform, and such that the synthesized plan achieves an intended objective. Task planning is supposed to be general enough to handle all kind of tasks, integrating mobility, manipulation, assembly, sensing, etc. A planner is a *predictive* system: it

chooses, among various projections of possible futures those likely to lead to the goal (see also chapter "Planning in Artificial Intelligence" of Volume 2 and Ghallab et al. 2004 for general discussions on planning). For this, the models of possible actions are at some level of abstraction that allows easy predictions. They are mainly logical or relational models, which grasp the causal relationships between actions, their conditions, effects and the intended objectives. The plans produced are more like guidelines for acting than direct programs to execute in open loop: they seldom fully unfold as expected, along a nominal scenario. Once a plan is found, there are problems for acting according to that plan, i.e., transforming the abstract actions in the plan into commands adapted to the context, monitoring their execution, and if necessary, to taking corrective steps, including replanning.

Robotics was one of the first area that motivated the development of task planning. It led naturally to the issue of coupling of planning and acting – the STRIPS planner of Fikes and Nilsson (1971), on the Shakey robot, associated with Planex (Fikes 1971) for the execution of plans, is a seminal work in this area.

The execution controller (controller for short) does not make prediction. It uses different types of models which allow monitoring and, possibly, diagnosis. It must know which actions, especially the sensory ones, are needed to launch a planned action and/or to observe the direct or indirect effects of the action. It must be able to update the state of the world required to monitor the plan execution. It must know the conditions which invalidate the current action, expressing the failure or absence of response time, and those which invalidate the current plan. In addition, the controller must be able to manage uncertainty and nondeterminism at various levels: the imprecision of sensory data and the uncertainty about their interpretations; the action duration; the nondeterminism inherent to the action outcomes, etc. Indeed, the controller launches the actions, but their effects and precise courses of execution depend upon conditions and contingent events partially modeled. Finally, by definition, the controller operates online: it must also be responsive to unforeseen events by the plan, and ensure some safety conditions.

The coupling of planning and acting requires a tradeoff between the constraints and models needed for the planner predictions and those needed for the acting online with action refinements, reactions, monitoring and revision. A description of a planning and acting system and how to achieve this tradeoff could be made on the basis of a hierarchical state transition system $\Sigma = (S, A, E, \gamma)$, where S, A and E are enumerable sets of *state* of *activities* and *events*, and γ is a function that describes the dynamics of the system $\gamma = S \times A \times E \rightarrow S^2$. Activities are decided and triggered by the robot, while events are not under its control; they give rise to changes in the environment which can be observed directly or indirectly. Σ is described with two levels of abstraction:

- the planner has an abstract model of Σ: its macro-states are subsets of S, its actions are subsets of activities; it rarely takes into account E;
- the controller has a finer model of Σ: it is able to refine each planned action in corresponding activities which are under its control; it knows how to launch activities and how to monitor their progress; it can trigger activities (e.g., monitoring, alarms) to observe the dynamics of S, and other activities to react to events.

A complete formalization of such a system depends on many conditions, especially the type of planning used, deterministic or non-deterministic and the system dynamics, e.g., how to take into account the concurrency between activities and events within the function γ. A presentation of possible approaches is beyond the scope of this chapter. We refer the reader to the book (Ghallab et al. 2016) for a detailed coverage of planning and acting issues, and to the survey of Ingrand and Ghallab (2017) for a broad perspective on deliberation in robotics. In the remainder of this section, let us review some of the main approaches for acting and execution control, focusing on relational and logic representations in deterministic and temporal approaches, and on Markov representations for nondeterministic approaches.

4.1 Deterministic Approaches

The approaches using a classical planner (as in STRIPS) often produce a plan π to which they associate a causal structure that help the controller follow the proper execution of the plan (e.g., triangular tables). The purpose of these structures is to provide the conditions of use of the actions so that the controller can verify their applicability and their proper execution. If these conditions are not met the control can relaunch this action (or another) or it can call the planner to produce a new plan.

These causal structure to monitor the execution of a plan are quite limited. Richer formalisms have been proposed to permit the execution of plans. They can be classified into two broad families:

- *Imperative Languages* such as RAP (Firby 1987), PRS (Ingrand et al. 1996), or TDL (Simmons and Apfelbaum 1998). They offer an imperative programming language that allows to specify procedures to be performed to meet some objectives (e.g. perform an action). These languages offer conventional programming control structures (test loop, recursion, parallelism, etc.), and often rely on concepts borrowed from logic programming (as in Prolog).
- *State Transition Systems* such as SMACH, the ROS controller language of ROS (Bohren and Cousins 2010). The user provides a set of hierarchical finite state machines. Each state corresponds to an activity involving one or more components of the robot. According to the returned values of executions, the controller performs the appropriate transition to the next state. The overall state of the system corresponds to the composition of the hierarchical automata.

These systems, based on automatons or procedures are very useful and necessary in setting complex robot experiments where one must coordinate many software components. However, these models, used to refine actions in activities, must be directly programmed by procedures or automatons developers, and are not inferred from specifications. This is a problem with respect to their validation and verification.

Planning with Hierarchical Task Network (HTN) (Tate et al. 1994; Erol et al. 1994) naturally incorporates a refinement process of abstract tasks in elementary actions. HTNs represent decomposition methods of task as a network (often an and/or tree) of elementary actions. The specification of knowledge in these approaches appears

natural to the programmer. These approaches seldom provide ways to refine planned actions into commands, and to repair refinements when an execution failure occurs. However, several HTN systems are used in robotics and extend the formalism in various ways. For example SIPE (Wilkins 1988) can produce plans where the duration of actions is taken into account. TCA/TDL (Simmons and Apfelbaum 1998) integrates execution and decomposition during the execution of tasks in plans. Xfrm (Beetz and Mcdermott 1997) can produce plans following an HTN approach, but also allows the modification/repair of these plans while executing them (transformational planing).

4.2 Temporal Approaches

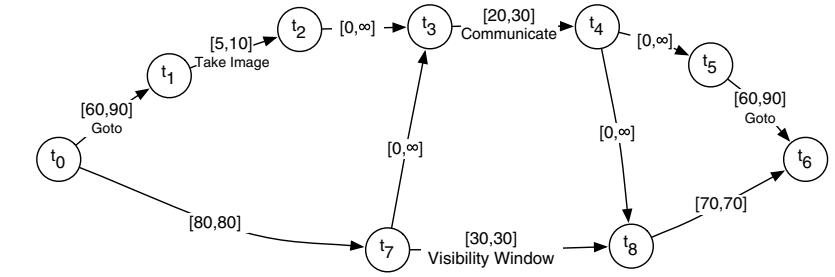
The controller of an autonomous robot must explicitly take into account time. A state transition approach is not sufficient. Indeed, the activities of the robot are not instantaneous (motions, taking images, etc). Often, they must be executed in parallel, synchronized, and bounded with earliest and latest date. These motivations lead to explicitly include time and temporal constraints in the models: the plan produced will be more robust with respect to execution.

Several planning approaches based on temporal intervals or events formalisms (Allen 1984; Ghallab and Alaoui 1989) have been developed (see also chapter "Qualitative Reasoning" of Volume 1), e.g., IxTeT (Ghallab and Laruelle 1994), HSTS (Muscettola 1994), Europa (Frank and Jónsson 2003), APSI (Fratini et al. 2011), FAPE (Bit-Monnot 2016). They led to extensions that take into account execution. They produce plans in the form of a lattice of instants (the beginnings and ends of actions) or intervals. A timeline represents the temporal evolution of a state variable (e.g., the position of the robot); it is composed of instants or intervals in which the variable keeps a value (e.g., the robot does not move), or changes its value (the robot moves). The search for a solution plan is in the space of partial plans (where each state is a partial plan with a set of partially instantiated and ordered actions), with a least commitment strategy.

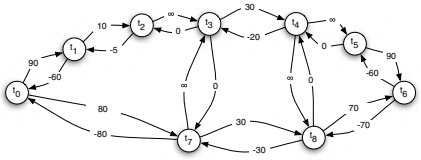
These approaches have many advantages for planning and execution in robotics. They properly manage concurrency or parallel execution. Furthermore, they generally produce plans that are temporally flexible, leaving to the execution the choices of the exact dates of occurrence (controllable or non-controllable but observable). For this, the execution controller must continually propagate the time constraints based on the date of occurrence actually observed to ensure that the plan remains consistent and repairable in case of inconsistency.

Some approaches (e.g., IDEA and T-ReX) offer a paradigm where the planner and the controller are tightly coupled in a set of reactors, each with its own horizon for planning and execution. The FAPE approach introduces hierarchical action models (similar to HTN) as to enrich the action model representation, but also to speed the search for a plan with methods.

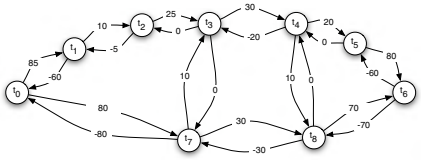
For events as well as intervals, these approaches rely on Simple Temporal Networks (STN) to model the temporal constraints between the events considered. An STN is a constraint network whose variables are events; constraints between two



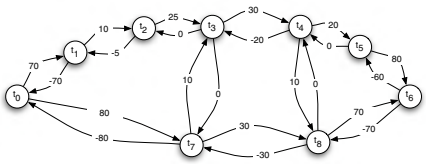
(a) Simple temporal network (STN)



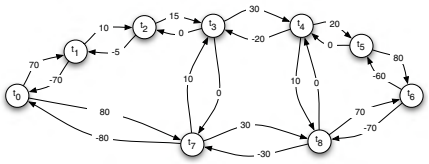
(b) Distance graph



(c) After a Floyd-Warshall propagation (Algo 14(a))



(d) After execution of the first Goto



(e) After incremental propagation (Algo 14(b))

Fig. 13 Successive phases of planning and execution of a temporal plan for a Mars exploration rover

events t_i and t_j are of the form: $min_{ij} \leq t_j - t_i \leq max_{ij}$. The Allen Algebra of intervals (Allen 1984) (using relations such as *before, meets, overlaps, starts, during, finishes*, their symmetrical and equality) can easily be transformed into an equivalent STN. One has just to translate the relations in precedence (or equalities) on the beginnings and ends of each interval.

The plan produced is an STN described by the corresponding constraint. Figure 13a shows the STN plan of a Mars rover that must go to a given location, take a picture, communicate the result to an orbiter during a window visibility, then return to its base. The network can be transformed into a distance graph (see Fig. 13b where arcs correspond to the inequalities $t_j - t_i \leq max_{ij}$ and $t_i - t_j \leq -min_{ij}$). One finds the minimum using Floyd-Warshall algorithm Fig. 14a. Here $dist[i, j]$ is the minimum distance from i to j, initialized with an infinite value when i and j are not constrained. One then obtains the graph in Fig. 13c.

When an STN is taken as a task to perform, the execution controller must incrementally propagate the update using algorithm Fig. 14(b) (which is of a lesser complexity, $O(n + n^2)$ instead of $O(n^3)$). In the example above, if the first Goto takes exactly 70 s, we get the STN in Fig. 13d and after propagation the graph in Fig. 13e.

```
Floyd-Warshall(dist, n)
  for k from 1 to n
    for i from 1 to n
      for j from 1 to n
        dist[i, j] ← min{dist[i, j],
                         dist[i, k] + dist[k, j]}
```

(a) Initiale propagation: Floyd-Warshall Algorithm

```
Incremental(dist, n, i0, j0)
  for i from 1 to n
    dist[i, j0] ← min{dist[i, j0],
                      dist[i, i0] + dist[i0, j0]}
  for i from 1 to n
    for j from 1 to n
      dist[i, j] ← min{dist[i, j0],
                       dist[j0, j] + dist[i, j]}
```

(b) Incremental propagation: after changing the constrain between two events i_0 et j_0

Fig. 14 Temporal constraints propagation algorithms

These approaches have been successfully implemented in many robotic experiments (e.g., MBARI, Py et al. 2010; Willow Garage, McGann et al. 2009; NASA, Finzi et al. 2004, and LAAS, Lemai-Chenevier and Ingrand 2004) but their development faces the following difficulties:

- writing the planning models and debugging them is difficult, especially when one wants to take into account non-nominal execution situations (i.e., failures and error recovery),
- the search for solutions in the partial plans space must be guided by adapted heuristics,
- the temporal controllability of the STN must be taken into account. Indeed, these STNs contain *controllable* variables but also *contingent* variable. The values of the former are selected by the robot, whereas the values of the latter are contingent and determined by the environment within their domain.[5] An STN is controllable if there a possible value assignment for controllable events depending on the values of the contingent ones. Strong controllability ensures that there exists an assignment of values of controllable events for all possible values of contingent ones. Weak controllability ensures that there is a possible value assignment for the controllable ones for all the values of the contingent ones, if they are known in advance (unrealistic). Dynamic controllability ensures that there is an assignment for controllable ones for the values of the past contingent ones. This last property keeps the flexibility while making sure that a solution remains.

Other approaches (e.g., Aspen/Casper, Chien et al. 2000) based on a temporal model produce complete plans without any flexibility. If a temporal (or a causal) failure occurs when executing the plan, the planner then repairs it using local search techniques.

[5]For example, in the graph of Fig. 13c, the starting time t_0 is controllable, but not the arrival time t_1 of a Goto action. The maximum travel time is reduced by propagation from 90 to 85 (Fig. 13c). But this is not acceptable for a contingent duration.

4.3 Probabilistic Approaches

Nondeterminism is not an intrinsic property of a system but a property of its model. Interaction with the real world always involves some level of nondeterminism, that may of may not be grasped in its model. The same arguments that foster the need for autonomous deliberation in a robot, i.e., open and diverse environments and tasks, promote the use of nondeterministic models. These models allow to handle various possible interactions between the robot actions and the environment own dynamics, possibly with probabilistic models. Markov Decision Processes (MDP) provide a convenient representation for planning under uncertainty. Let us introduce here a general MDP formulation, which will be also useful for Sect. 6 about learning.

Let S be a finite set of states, and A a finite set actions. If an action a is applicable in a state s, a can lead *nondeterministically* to any states in $F(s, a) \subseteq S$. Let $P(s'|s, a)$ be the probability of reaching state s' when action a is applied in s; $r(s, a) \geq 0$ is the reward associated with a in s. Let $\pi : S \rightarrow A$ be an application that associates to each state s the action to be performed in s. π is called a *policy*; it corresponds to a plan that tells the robot which action to carry in each state. π has possibly loops, i.e., following π from a state s may lead back to s after one or a few steps. The value function $V_\pi(s)$ of a state s under policy for π is the expected sum of rewards of this plan, weighted (to ensure convergence) by a decreasing coefficient:

$$\begin{aligned} V_\pi(s) &= E\left[\sum_{t=0}^{\infty} \xi^t r(s_t, \pi(s_t))\right], \text{ with } \xi < 1 \\ &= r(s, \pi(s)) + \xi \sum_{s' \in F(s,\pi(s))} P(s'|s, \pi(s)) V_\pi(s') \end{aligned} \tag{7}$$

The optimal value function for a state s is $V^*(s)$ for the optimal policy π^*.

$$\begin{aligned} V^*(s) &= \max_\pi V_\pi(s) \\ &= \max_a \{Q^*(s, a)\}, \text{ with} \\ Q^*(s, a) &= r(s, a) + \xi \sum_{s' \in F(s,a)} P(s'|s, a) V^*(s') \end{aligned} \tag{8}$$

Dynamic programming leads to a recursive formulation of V^* and provides easily implementable algorithms, such as Value Iteration (see Fig. 15).

The Value Iteration algorithm (Bertsekas 1995) terminates when a fixed point is reached, i.e., a full iteration over S without a change in any $V(s)$. It gives the optimal policy π^*. It can be initialized with an arbitrarily function $V(s)$. In practice one does not need to loop until a fixed point. It is sufficient to make sure that all updates of $V(s)$ on some iteration over S remain below a threshold ε. The returned solution then deviates from the optimum by at most $2\varepsilon \times \xi/(1 - \xi)$.

The above formulation is *process-oriented*: it seeks an optimal policy for an infinite process. The formulation can be transformed into a *goal-oriented* approach

```
Value Iteration(S, A, P, r)
    until reaching a fixed point do
        for each s ∈ S do
            for each a applicable in s do
                Q(s,a) ← r(s,a) + ξ Σ_{s'} P(s'|s,a) V(s')        ▷ (i)
            V(s) ← max_a {Q(s,a)}
    for each s ∈ S do
        π(s) ← argmax_a {Q(s,a)}
```

Fig. 15 Value iteration algorithm

by giving an initial state s_0, a set of goal states $S_g \subset S$, and by searching for an optimal policy that leads from s_0 to one of the states in S_g. One can also integrate cost distribution on actions and variable rewards function for goal states. In such a formulation, one is not searching for a policy defined everywhere, but for a partial policy, defined only in states reachable from s_0 by this policy. A *safe* policy π is guarantied to reach a goal from s_0. If a problem has a safe policy, then dynamic programming with $\xi = 1$ can find an optimal one. The algorithm Value Iteration applies to the case where there is a safe policy from every state. When this assumption does not hold, the problem is said to have *dead-ends*, i.e., states from which a goal is not reachable. Extensions of dynamic programming algorithms to handle problems with dead-ends have been introduced, e.g., by Bonet and Geffner (2006), Bertsekas (1995), or Barto et al. (1995). It is not necessary, nor possible, to iterate over all S. It is enough to search along states reachable from s_0 with a current policy. One may also estimate $Q(s,a)$ by sampling techniques (Jaakkola et al. 1994). The step $\triangleright(i)$ is replaced by $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \xi \max_{a'}\{Q(s',a')\} - Q(s,a)]$, where s' is taken in $F(s,a)$ by sampling according to the distribution $P(s'|s,a)$. This approach is very useful in reinforcement learning.

The Value Iteration algorithm has a polynomial complexity in $|S|$ and $|A|$. Unfortunately, S has generally a huge size: it is exponential in the number of state variables. There are a few scalable approaches, using heuristics and hierarchical techniques, e.g., those of Barry et al. (2011), Teichteil-Königsbuch et al. (2010), Pineau et al. (2003), Pineau and Gordon (2005), Teichteil-Königsbuch and Fabiani (2005). Probabilistic planning is a very active research area with may open problems. It is also discussed in chapter "Planning in Artificial Intelligence" of Volume 2.

Given a policy π, the controller for an MDP is extremely simple. Just iterate over two steps:

- observe the state s
- execute the action $\pi(s)$

until reaching a goal state or some other stopping conditions.

The MDP approach offer several runtime advantages. It explicitly manages the nondeterminism and uncertainty. It can be extended to take into account Partially Observable domains (Buffet and Sigaud 2008). Modeling a domain as an MDP is a

difficult task, but the MDP formulation can be combined with learning techniques (see Sect. 6). This explains the success of these approaches in many robotics applications which will be discussed later.

4.4 Integrating Motion and Task Planning

Task planning and motion planning are two different problems that use distinct mathematical representations. The former is concerned with causal relationship regarding the effects of abstract actions, the latter is concerned with computational geometry and dynamics. In simple cases a robot can decouple the two problems: task planning produces abstract actions whose refinement requires, possibly, motion planning. The two problems are however coupled for constrained environments and complex tasks. For example, moving objects in a storage room can make the motion impossible if the task is not appropriately organized. Let us discuss here some approaches to the integration of motion and task planning.

The Asymov planner (Cambon et al. 2009) combines a state-space search approach for task planning (using the FF planner, Hoffmann 2001) with a search in the configuration space for motion planning. It defines a *place* as a state in the task planning space, as well as a range of free configurations in $\mathscr{C}_f$. A place is a bridge between the two search spaces. These two spaces are not explicitly constructed, but for every found task state, Asymov checks that there are some reachable configurations in $\mathscr{C}_f$. This approach has been extended to multi-robot problems cooperating over a joint task, e.g. two robots assembling a large furniture such as a diner table in a cluttered environment.

Another interesting technique uses hierarchical planning in a so-called *angelic* approach (Wolfe et al. 2010) (from *"angelic nondeterminism"* which assumes that out of several issues, the best one can be chosen). An abstract action can be decomposed in different ways. An abstract plan is based on abstract actions; its set of possible decompositions is a subset of the product of all possible decompositions of its actions, some of which are not compatible. It is not necessary to ensure that all decompositions are feasible. A plan is acceptable if it has at least one feasible decomposition. Indeed, the decomposition is not made randomly. The robot decomposes, when needed, each abstract action by choosing a feasible decomposition, if there is one. The idea is to rely on a lower bound of the set of feasible decompositions of an abstract plan such as to make sure that this set is not empty. These lower bounds are computed by running simulations of action decompositions into elementary steps, using random values of state variables. The planner relies on these estimates for searching in the abstract state space.

The approach of Kaelbling and Lozano-Perez (2011) illustrates another hierarchical integration of task and motion planning. When planning at the abstract level, estimates regarding geometric information are computed with so-called *Geometric Advisers*. These advisers do not solve completely the motion planning problem submitted to them, but provide information about how feasible is a given step that enables

the abstract search to continue until reaching a complete plan. When the produced plan is executed, each step that requires movements triggers a full motion planning. This approach relies on two strong assumptions: geometric preconditions of abstract actions can be calculated quickly and efficiently (by the geometric adviser); subgoals resulting from decomposition of action are executable in sequence. The approach is not complete, i.e., the geometric refinement of a planned abstract action may fail. However, for problems where actions are reversible at a reasonable cost (i.e., allowing for backtracking at the execution level) the approach is efficient and robust.

5 Interaction

Most of the approaches presented above make the assumption that there is a single agent in the environment: the robot performing the task. But complex missions may require the participation of several humans and robots. Several approaches address the issues of interaction. For example, Simmons et al. (2007) proposes the Syndicate architecture, an extension to 3T (Bonasso et al. 1997), which allows the cooperation of several robots in collaboration with a human, for the assembly of large structures. Fong et al. (2006) offer an architecture to define interaction models (tasks, teams, resources, human) needed for the cooperation of a team of astronauts and extra-planetary rovers. In the next two sections, we examine the interaction problems and how they are accounted for in the planning process.

5.1 Multi-robot Interaction

Sometimes, to achieve a complex mission, it is necessary to deploy multiple robots. Several approaches to the problems of mission planning and execution in a multi-robot framework have been developed. We may distinguish several types of problems depending on the following features:

- planning is centralized or distributed,
- acting along partial plans of each agent is globally monitored or coordinated,
- planning is done offline or proceeds online along with acting
- execution failures are repaired at various levels
- communication between robots for planning and acting is pairwise or global.

Many research focuses on multi-robot motion planning, with geometric and kinematic representations (see Sect. 3), and decomposition techniques generic enough to lead to distributed implementations (Erdmann and Lozano-Perez 1987). Recent results, e.g., Bhattacharya et al. (2010), allow to efficiently take into account relative position constraints between the robots as well as missions featuring several sites to visit.

The Martha project illustrates an approach to manage a fleet of robots handling containers in ports and airports (Alami et al. 1998). The allocation of tasks to robots is centralized, but on a limited horizon. Planning, execution, refinement and coordination needed for the navigation of robots and the sharing of spacial resources in the environment are distributed. Robots negotiate among themselves the navigation in the environment, which is divided into cells (e.g., intersection crossing, convoy mode, overtaking), and also negotiate their path inside these cells. The deployed system assumes a reliable local communication. Execution deadlocks between multiple robots are correctly detected by the coordination algorithm, and one of the robots automatically takes control and produces a plan that it redistributes to the other robots with which it conflicts.

Other works propose an allocation of tasks by an auction mechanism (Dias et al. 2006) to assign tasks to robots (cells crossing/surveying). Tovey et al. (2005) propose a mechanism to generate appropriate auction rules adapted to the particular goal a group of exploration rover has (minimize the sum of the distances, minimize the maximum travelled distance of all robots, minimize the average time to the targets, etc.). In Zlot and Stentz (2006), Cao et al. (2010), the authors apply a similar technique to tasks and subtasks of an HTN plan as it is built. Each robot can win the bids on a task, then decompose into sub-tasks following an HTN method, and auction all or part of the sub-tasks. After the initial distribution of tasks, robots maintain, during execution, the ability to auction tasks they failed to perform. Moreover, communication in these systems is not permanent and complete, thus the replanning/redistribution phases must be planned in advance.

5.2 Human–Robot Interaction

The development of service robots raises important issues regarding human-robot interaction (Goodrich and Schultz 2007). We focus here on approaches which are concerned with planning and the models they use.

Interactive planning (or *mixed initiative* planning), i.e., planning while keeping humans in the loop, is used in various areas. The operator takes part in the search for a plan to make choices and help the planner solve the problem.

Planning for human–robot interaction raises a completely different issue: the plan is generated automatically by the robot, but must explicitly take into account the interaction with the human while executing the plan and even in some cases, plan for a shared execution. To this end, the planner has some models (learned or programmed) of human behaviors (Beetz et al. 2010). These models specify how humans behave with respect to the robot, what are the behaviors of the robot which are acceptable to humans. They also specify how planned actions translate into commands of the robot (Stulp and Beetz 2008).

Various planners have been adapted to take into account the role of the human in the plans produced. Generally, these are multi-agents planners, which have been modified to consider the human as one of the agents. Burgard et al. (1998) propose

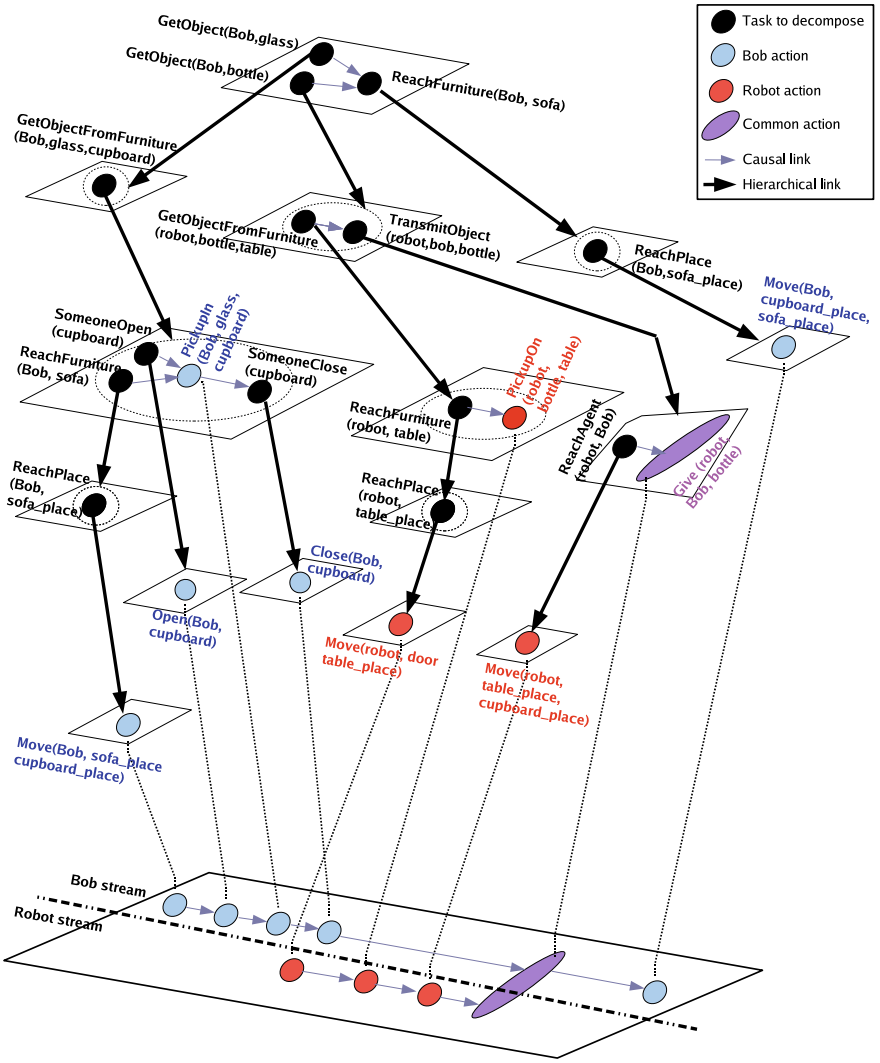


Fig. 16 A plan produced by HATP with human–robot interaction: the tasks (in black) are decomposed into primitive actions for the robot (in blue), actions of the human (in red), and joint actions (in purple), which require a synchronization (Alami et al. 2006)

an extension to GOLOG/GOLEX to plan the mission of a robot museum interacting with visitors. The approach used in Brenner et al. (2007) is based on MAPL, a PDDL based multi-agent planner to represent the beliefs of the various agents, the actions and the perception operators. A compiler then translates these PDDL models. Planning is then performed by the FF planner (Hoffmann 2001).

The HATP planner (Alami et al. 2006) plans in the context of human–robot interactions (e.g., for service robotics). This planner extends the HTNs formalism to create plans containing two execution threads, one for the robot and one for the human who interacts with the robot. Figure 16 shows a plan produced by HATP. It decomposes shared tasks into interleaved actions to be performed by the robot (in red) and by the humain (in blue). HATP differs from the classical HTN planning on several points. Task models and refinement methods involve human and robot. Furthermore, while the plan is produced, the system considers social rules to produce plans which are deemed acceptable and understandable to humans. For example, the robot will favor an action where it gives an object directly to the human rather than an action where it just lays the object before him. Similarly, when interacting with humans, the robot will favor a position where it faces the human, and make slower movements when it approaches him. When executing the plan, the robot must interpret and recognize human actions to properly carry out its plan. For example, if during a task the robot proposes a tool to human, and if the human loses interest, the robot should not insist, and wait for the attention of the human to return back to the robot. These good behavior recipes are not just cosmetic, they enable a more natural interaction between humans and robots.

6 Learning

Machine learning techniques have a very successful impact in many areas, and particularly in robotics. A variety of computational learning techniques are developed at various levels in robotics, from the sensory-motor level to the acquisition of tasks or environment models. A good coverage of recent learning techniques in robotics can be found in Sigaud and Peters (2010). Neural networks, statistical learning techniques and other approaches are very successful, in particular for classification and object recognition (see chapters "Statistical Computational Learning" and "Reinforcement Learning" of Volume 1, chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2, and chapter "Artificial Intelligence and Pattern Recognition, Vision, Learning" of this volume), but they are not specific to robotics. We already covered environment mapping issues in Sect. 3.2. Let us review here two approaches that are more specific to robotics: reinforcement learning and learning from demonstration.

6.1 Reinforcement Learning

Reinforcement Learning (RL) refers to methods that improve the performance of a learner by direct interaction with the world (Kaelbling et al. 1996; Sutton and Barto 1998). It is based on a trial and error approach. A robot learns how to act by maximizing the long term perceived benefit of its actions. Generally in RL, the

learner has no teacher providing examples of good behaviors in certain situations or advices about how to choose actions. The only feedback given to the robot at each step is a scalar: the reward associated with the performed action. As long as the robot has not tried all feasible actions in all encountered situations, it will not be sure that it uses the best actions. Reinforcement learning has to solve the compromise of *exploration* vs *exploitation*: the robot must make the most of what it already knows to maximize the benefit of its behavior. To find the best one, it must explore the options that are not known enough.

To introduce the approach, consider the very simple case where a single action solves completely the task at hand and has no impact on the next task. Suppose a stationary environment and nonnegative rewards. Let $r_i(a) > 0$ be the reward received after running an action a at the ith time. We can estimate the quality $Q(a)$ of an action a that has been executed k_a times by its average award:

$$Q(a) = \begin{cases} q_0 & \text{if } k_a = 0, \\ \frac{1}{k_a} \sum_{i=0}^{k_a} r_i(a) & \text{otherwise.} \end{cases} \tag{9}$$

This estimate is maintained by incremental updates:

$$Q(a) \leftarrow Q(a) + \alpha[r_{k_a+1}(a) - Q(a)], \text{ with } \alpha = \frac{1}{k_a + 1} \tag{10}$$

When $\forall a, k_a \to \infty$, the choice of the action which maximizes the sum of rewards is given by $\text{argmax}_a\{Q(a)\}$. However, as long as the exploration of alternatives has not been sufficient, the robot will use other options, according to various heuristics. One may define a function $\text{Select}_a\{Q(a)\}$ by one of the following methods:

- $\text{Select}_a\{Q(a)\} = \text{argmax}_a\{Q(a)\}$ with probability $(1 - \varepsilon)$, and a randomly drawn action other $\text{argmax}_a\{Q(a)\}$ with probability ε, where ε is decreasing with experience,
- $\text{Select}_a\{Q(a)\}$ chooses an action according to a probabilistic sampling distribution, for example, with Boltzmann sampling, according to a probability distribution given by $e^{\frac{Q(a)}{\tau}}$, where τ is decreasing with experience.

When the environment is stationary, the update of $Q(a)$ with Eq. 10 after executing an action a becomes increasingly weak with large k_a. If the environment is not stationary, we can keep $\alpha < 1$ constant. Note also that the initialization value q_0 fosters exploration if q_0 is high with respect to other rewards. For example, if $q_0 = r_{max}$, the maximum reward, new actions will be preferred to those already tested.

With these basics notions, let us now consider the interesting case where a task is performed by the combination of several actions, each interfering with the following ones, influencing the overall success of the task and the sum of rewards. The framework generally used is that of Markov decision processes introduced previously (Sect. 4.3). The robot seeks to learn an optimal policy that maximizes the value $V(s)$ over all s. This value is estimated from the observed rewards of the chosen

actions. A major problem is how to distribute rewards over the entire task. Indeed, the rewards give an immediate return in the short term, while the quality of achievement of the task to be maximized is described by the long term sum of rewards over some horizon.

One approach is to learn the MDP model then to apply planning techniques to find the optimal policy and use it. Learning a model means collecting enough statistics through an exploratory phase to estimate the probability distributions $P(s'|s, a)$ and the rewards $r(s, a)$. An interesting application of this direct approach combines a model learning technique with a receding horizon planning algorithm (Morisset and Ghallab 2002). It was illustrated for learning indoor navigation skills, combining different motion, localization and control modalities. The approach is applicable to any task for which the robot has several alternative methods whose performance depend on local features of the environment. The performance function is difficult to model. It is learned as an MDP whose state space is an abstract control space, which focuses on the features of the environment and current task context (including the method in use); actions correspond to available methods for performing the task. The state space is of small size (a few thousands states) which allows computing an optimal policy at each step of a receding horizon planning.

This direct approach requires a costly exploratory phase to estimate the model. It is often better to start performing the task at hand, given what is known, while continuing to learn, i.e., combine the two phases of acquiring a model and finding the best action for the current model. *Q-learning* algorithm meet this objectives while avoiding to build explicitly the MDP model.

Let us use the MDP notation introduced earlier, in particular $r(s, a)$ is the observed reward after performing action a in state s, and $Q(s, a)$ is the estimated quality a in s at current time. $Q^*(s, a)$, as given by Eq. 8, is unknown but it can be estimated by the expression:

$$Q(s, a) = r(s, a) + \xi \sum_{s \in F(s,a)} P(s'|s, a) \max_{a'} \{Q(s', a')\} \tag{11}$$

The basic idea of the *Q-learning* algorithm Fig. 17 is to perform an incremental update of $Q(s, a)$, similar to Eq. 10. This update does not use the unknown probability parameters of the model, but the quality of successor states s', as observed in the current experience. This update is given in line *(i)* in the algorithm below.

The values of $Q(s, a)$ are initialized arbitrarily. The function $\text{Select}_a\{Q(s, a)\}$ favors $\text{argmax}_a\{Q(s, a)\}$ while allowing for the exploration of non maximal action with a frequency decreasing with experience. The parameter $\alpha \in [0, 1]$ is set empirically. When it is close to 1, Q follows the last observed values by weighting down previous experience of a in s; when it is close to zero, the previous experience is more important and Q changes marginally. α can be decreasing with the number of instances (s, a) encountered.

A variant of this algorithm (known as " SARSA" for State, Action, Reward, State, Action) takes into account a sequence of two steps (s, a, s', a') before performing the update of the estimated quality of a in s by $Q(s, a) \leftarrow Q(s, a) + \alpha[R(s, a) +$

Q-learning
  until *Termination* do
    $a \leftarrow \text{Select}_a\{Q(s,a)\}$
    execute action a
    observe $r(s,a)$ and resulting state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \xi \max_{a'}\{Q(s',a')\} - Q(s,a)]$ ▷ *(i)*
    $s \leftarrow s'$

Fig. 17 Q-learning algorithm

$\xi\, Q(s', a') - Q(s, a)]$. One can prove the asymptotic convergence of these two algorithms to optimal policies.

Other model-free reinforcement learning algorithms proceed by updating the value function $V(s)$ rather then the function $Q(s, a)$. Updates are performed over tuples (s, a, s') in a similar way: $V(s) \leftarrow V(s) + \alpha[r(s, a) + \xi\, V(s') - V(s)]$. This algorithm called $TD(0)$, is combined with a Select function permitting exploration. It is part of a family of algorithms $TD(\lambda)$ which perform updates over all states, with a weight depending on the frequency of meeting each state.

Let us also mention the DYNA algorithm and its variants that combine learning and planning: it maintains and updates an estimate of probabilities $P(s'|s, a)$ and rewards $r(s, a)$; at each step two updates are performed, a *Q-learning* type with $Q(s, a) \leftarrow r(s, a) + \xi \sum_{s'} P(s'|s, a) \max_{a'}\{Q(s', a')\}$, for the current s and a, and a Value Iteration type for other couples (state, action) chosen randomly or according to certain priority rules, taking into account new estimates. Here, the experience allows to estimate the model and the current policy. The estimated model in turn allows to improve the policy. Each step is more computationally expensive than in *Q-Learning*, but the convergence occurs more rapidly in the number of experimental steps.

Reinforcement learning is widely used in robotics, but it is rarely implemented with explicit state space and tables of values $V(s)$ or $Q(s, a)$. The state space is generally continuous; it includes the configuration space of the robot and its environment. Even if one manages to discretize the state space appropriately (e.g., in grid type environment approaches), the astronomic size of S makes the explicit representation of S impractical. Moreover, the above algorithms are used to learn a good behavior for states encountered during learning phase, but they are not useful for states that have never been encountered: they do not allow to generalize. If one uses a continuous state space with a metric distance, one can make the reasonable assumption that *nearby* states are typically associated with close estimates of $V(s)$ or $Q(s, a)$, and thus use similar policies. Parametric approaches implement this idea.

Here S and A are described by two vectors of state and control variables. Let $\theta = (\theta_1, \ldots, \theta_n)$ be a vector of parameters. We assume that $Q(s, a)$ can be approximated parametrically by $Q_\theta(s, a)$, as a function of θ. This function is given a priori, e.g., a linear function of state and control variables. Learning involves estimating the parameters θ of this model. Q-Learning algorithm is the same as above, except that

the update *(i)* does not change values in a table, but the parameters of $Q_\theta(s, a)$. The process generally involves minimizing the mean squared error of Q with respect to Q^*. The latter is estimated at each iteration from the last observed update. The gradient algorithm follows this formulation:

$$\begin{aligned} \theta &\leftarrow \theta - \frac{1}{2}\alpha\nabla_\theta[r(s,a) + \xi\max_{a'}\{Q_\theta(s',a')\} - Q_\theta(s,a)]^2 \\ \theta &\leftarrow \theta + \alpha[r(s,a) + \xi\max_{a'}\{Q_\theta(s',a')\} - Q_\theta(s,a)]\frac{\partial Q_\theta(s,a)}{\partial\theta} \end{aligned} \tag{12}$$

This last expression replaces the step ▷*(i)* in the previous algorithm for each parameter θ_i. A similar formulation can be obtained for the estimate of V_θ.

Reinforcement learning with a parametric approach is used with success in robotics. It has been implemented in simple applications, for example to stabilize an inverse pendulum or to play darts, and in more complex demonstration, such as helicopter acrobatic flying (Abbeel et al. 2006; Coates et al. 2009). One of the main problems of these approaches is defining the action rewards.

Indeed, the previous algorithms indicates improperly "observe $r(s, a)$". But rewards are seldom directly observable by the the robot. One must provide the means to estimate the reward according to what is perceived. Sometimes the function $r(s, a)$ is easy to specify, for example as the deviation from equilibrium for a stabilization task, or the deviation from the target for tracking task. But often it is not, for example, how to specify the rewards of elementary actions for the task of driving a car?

This issue leads to the *inverse reinforcement learning* problem (Abbeel and Ng 2010). Here, the teacher gives the optimal behavior, the problem is to find the corresponding reward function that generates this behavior. In the case of an explicit finite MDP, the problem reduces to the following formulation (derived directly from Eq. 8): we know $\pi^*(s)$ for all s; we can express $Q(s, a)$ as a function of the unknown values of $r(s, a)$ and we want $Q(s, a)$ to be maximal for $a = \pi^*(s)$. This formulation is under-specified: it has infinitely many solutions that are of no interest. It is extended with an additional criterion, for example maximize the expression: $\sum_s[Q(s, \pi^*(s)) - \max_{a\neq\pi^*(s)} Q(s, a)]$. The problem is solved by linear programming.

In parametric approaches we also define rewards r_θ as a function (usually linear) of state and control variables and seek to estimate its parameters. The previous formulation is not directly applicable because π^* is known for a small number of state samples. However the main constraint that the distribution of states generated by r_θ must be the same as the one provided by the teacher leads to a formulation that one can solve iteratively. Each iteration combines two steps, a quadratic programming optimization criterion and a dynamic programming similar to Value Iteration.

As the reader has certainly noticed, these approaches are akin to the techniques used for inverse problems. They are also related to learning from demonstration techniques, discussed next.

6.2 Learning from Demonstration

As underlined above, the specification of the reward functions needed in reinforcement learning is far from obvious. Moreover, it is rare to have a fully observable Markov state space. We know how to transform a state space into a Markovian one, but this requires significant engineering and adds generally unobservable components. The complexity of learning and planning techniques in partially observable MDP is prohibitive. Moreover, the experimental complexity (in the total number of needed trials) is generally much more expensive in robotics than the computational complexity. Reinforcement learning requires for converging a very large number of experiments. Finally, it is common that the task to learn cannot be treated as a simple sequence of pairs *(state, action)*. It requires a plan or a control structure, such as repeating a subsequence of actions until a termination condition. For these reasons, learning from demonstration is a good alternative when the robot can benefit of the demonstrations of a teacher.

In learning from demonstration (see the survey of Argall et al. 2009), a teacher gives to the robot the appropriate actions in well-chosen settings. This allows the teacher to control the learning process and gradually focus learning on the most difficult part of the task. The robot generalizes from the teacher demonstrations and learns the required behavior, which can be expressed as a policy in simple cases, or as a mapping from sensory states to plans in the general case. Learning from demonstration is akin to supervised learning. However in supervised learning, the teacher provides directly correct labels for training cases. Learning from demonstration involves other issues about how to map the teacher's sensing and acting spaces to those of the robot learner.

In the simplest setting, learning from demonstration reduces to acquiring the correct behavior from teleoperated training cases. The teacher acts directly in the actuator space and the proprioceptive sensor space of the robot. The latter learns actions directly as its own control environment. These approaches have resulted in many implementations, such as those presented by Sigaud and Peters (2010) or Peters and Ng (2009).

In the general case, the teacher acts with its own actuators rather than those of the robot to illustrate the movements and manipulations she wants to teach. The robot observes the teacher from outside. In order to learn, the robot must build up a double mapping:

- a sensory mapping to interpret the observed demonstrations, and
- a control mapping to transpose the demonstrated actions to its own actuators.

This double mapping is very complex. It often limits learning from demonstration and requires the teacher to have pedagogic skills, that is, to understand at a low level how the robot will be able to map the teacher demonstrations in its own actuation capabilities.

Moreover, learning from demonstration can be performed with or without the acquisition of a task model. The first case corresponds generally to inverse reinforcement learning. In the latter case, learning can give rise to the acquisition of a

sensory-motor mapping. Here, the techniques use supervised learning, by classification or regression. Finally, learning can also lead to the acquisition of a mapping from sensory states to plans. These can be obtained by plan recognition methods. Plans can also be synthesized from the teacher specifications of operators and goals (final and intermediate) associated with observed sensory states.

Approaches relying plan recognition and synthesis allow to address a significantly more general class of behaviors that can be demonstrated by a teacher and acquired by a robot (including iterative actions and control structures). They also permit extended generalization since they lead to acquire basic principles and use the robot planning capabilities. They are finally more natural and easier for the teacher, since the teacher's actions are interpreted in terms of their effects on the environment rather than their sole order in a sequence of commands. They are illustrated for example in the work of Nicolescu and Mataric (2003), Rybski et al. (2007), but remain at a quite preliminary stage.

7 Integration and Software Architecture

Beyond the physical integration of mechanical, electrical and electronic systems, a robot is also a complex information processing system, from sensors data acquisition to actuators command. It integrates various processing paradigms such as real time control loops with a hierarchy of response time, and decisional functions conferring the autonomy and robustness required by the variability of tasks and environments. The integration of these processes needs to be based on architectures that defines how to combine all the components, how they communicate and how they share data and computing resources. Architectures must provide development methodologies to allow programming, integration and testing of the different modules. They should provide tools and libraries to facilitate the development and deployment of the various components on the robot, especially those of interest to this chapter: planning and execution control.

7.1 Architecture Paradigms

Several paradigms have been developed for designing robot architectures. Let us review briefly the main classes.

Reactive Architectures

The reactive architectures, popularized by the subsumption architecture of Brooks (1986), are conceptually simple. They are composed of modules which connect sensors and effectors through an internal state machine. These modules are hierarchically organized with the outputs of some which can inhibit or weight the outputs or the composition of others. These architectures were relatively popular because they are

a priori easy to setup. They do not require a predictive model of the world since they do not perform prediction and are tailored to simple reactive tasks, without planning. A robot like the Roomba which was (most likely) developed following this concept, achieves its preset unique task. But there is no task flexibility, quality nor efficiency objectives pursued. Reactive architectures still remain popular and are used in mono task applications. However, when a variety of tasks and environments needs to be considered, the programming and setting of inhibitors/weights quickly becomes infeasible.

Hierarchical Architectures

The hierarchical architectures and layered architectures remain the most popular in robotics. They propose to organize all robot software in two or three layers, from the functional level up to the decisional level (planning and acting). The former includes the sensory-motor functions to control sensors, effectors, and to perform the associated processing. In some instances, an intermediate level is used for execution control to verify safety conditions. Tools are typically associated with these architectures to ease the integration of the different components. Thus, the LAAS architecture (Ingrand et al. 2007) relies on GenoM to develop functional modules (see Fig. 18), and various tools (R2C, OpenPRS, Transgen, IxTeT) for execution, supervision and tasks planning. The CLARATy architecture (Nesnas et al. 2003) provides C++ basic classes which facilitate the development of the functional layer. TDL and ASPEN/Casper respectively implement the acting and the planning component.

Teleo-Reactive Architectures

More recently, teleo-reactive architectures such as IDEA (Finzi et al. 2004) and T-ReX Py et al. (2010) have emerged. They propose to decompose the problem in agents rather than in layers. Each agent[6] consists of a planner/actor tandem. It produces plans by establishing sequences of tokens on timelines representing the evolution of the state variables of the system, and ensures their execution. Planning is performed by a temporal planner (e.g. Europa, Frank and Jónsson 2003 or APSI, Fratini et al. 2011) based on Allen (1984) temporal intervals logic. These agents are organized depending on the relevant state variables. Each agent has an adapted latency, execution period and planning horizons. They communicate between them by sharing some timelines (with priority rules on which agent can change value on a shared timelines) with a dispatcher responsible for integrating the new values of token depending on the execution.

These architectures have two advantages. They have a unified agent architecture model (even functional modules are expected to be developed using this paradigm). They use the same modeling language, providing an overall consistency of models. They have been deployed in a number of experiments, notably: an autonomous underwater vehicle (McGann et al. 2008), and the PR2 robot from Willow Garage (McGann et al. 2009).

[6] Agents are called reactors in the T-ReX terminology.

However, the deployment of these approaches is hindered by two problems. The first is performance. Agents are seldom able to properly plan fast enough (e.g. in less than a second), to be used to model functional modules. The second is the difficulty to develop the model (e.g. writing compatibilities and constraints), especially when modeling non-nominal cases.

Finally, in addition to the above categories, there are reactive hybrid architectures that add one or more planners to reactive modules. The role of the planners is to propose plans to configure, via a coordination system, the activities of the reactive modules. The difficulty is to write this coordination module. Beaudry et al. (2008) illustrate a proposal that combines a motion planner and an HTN planner which explicitly manage time; this approach seems promising for non-critical applications.

7.2 Robustness, Validation and Verification

The robustness of the software deployed on a robot poses a major problem. A first step is to robustify key components to overcome environmental hazards, sensory noise, and the great variability of environments. One can require that a functional module, handling a sensory-motor function, has an explicit model of its functioning envelope. It should know and recognize when its data cannot be properly used, to allow corrective actions to be taken. For example, a stereo vision component recognizes when its cameras are not properly calibrated; a locomotion module detects wheel slippage or wheel blockage by monitoring the torque on the wheel. Similarly, a planning module should ensure that the produced plans will not lead to undesirable states.

However, the composition of these components, as robust as they individually are, does not lead directly to an overall safety properties of the robot. For example the vision component and the locomotion component can both be correct, but all possible executions of these two components together may not be acceptable, e.g., the parameters to capture high-resolution images while moving are constrained (to avoid blurry images). The safety and robustness of embedded real-time systems Henzinger and Sifakis (2006) has been an active field for many year. Moreover, with respect to robotics, one has also to consider the requirements for decisional autonomy. However, one must consider that formal methods are more commonly used on decisional components (Cimatti et al. 2004; Simmons et al. 2000; Wong and Kress-Gazit 2016). This is mainly due to the fact that decisional specifications are model based with well-defined semantics. As a result, these models can also be used not only to perform the decisional function they are intended too, but also to prove properties of the solution they bring.

Regarding the functional level of robots, nowadays, robotic software developments use model-based or model-driven software engineering approaches (e.g. SmartSoft, Schlegel et al. 2009; RobotML, Dhouib et al. 2012; MontiArc, Ringert et al. 2015). These approaches and their associated middleware are numerous and surveyed in several papers (Brugali 2015; Elkady and Sobh 2012; Mohamed et al.

2008). Still, most of theses approaches do not integrate formal model analysis and the use of V&V techniques. Several works propose to use the formal synchronous language ESTEREL (Boussinot and de Simone 1991) to model functional components (Espiau et al. 1996; Sowmya et al. 2002; Kim and Kang 2005). The formal models are then exploited to verify behavioral and timed properties using model checking tools. These experiments are however done on simple examples and specifications are either hard-coded in ESTEREL or manually translated from robotic components. More recently a special issue (Kress-Gazit 2011) on this subject presented a number of interesting works along hybrid automata (Muradore et al. 2011) and controller synthesis (Kress-Gazit et al. 2011; Jing et al. 2016). But the models remain at a high level of abstraction. Other approaches are interested in verifying that the code executed by the functional modules of a robot formally satisfy its logical specification (Frese et al. 2009) (at the cost of logically annotating all the code used in the module).

More recently, some works around GenoM3 produce automatically, from the specifications of the modules, a formal model of the entire functional layer of a robot. The modeling is based on the BIP formalism (Behavior, Interaction, Priority) (Bensalem et al. 2011; Basu et al. 2006), or Fiacre/TINA formalism (Foughali et al. 2016; Berthomieu et al. 2008) and exploits the fact that each GenoM module is an instance of a generic module (see Fig. 18).

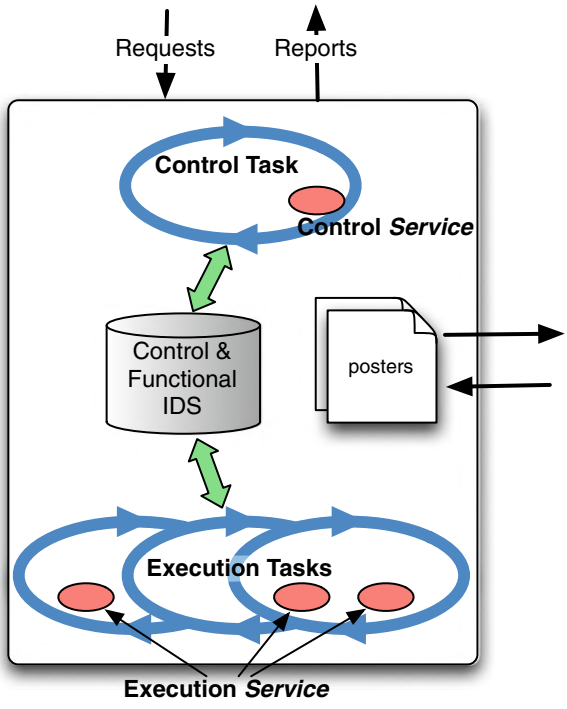


Fig. 18 The internal organization of a GenoM module. The control flow is organized as follows: the control task receives requests and starts the execution of corresponding services in the execution tasks. When execution is complete, the control task returns a report to the caller. Writing or reading posters provide the data flow between the modules

While using the BIP formalism, the invariant extractor and SAT-solving tool D-Finder (Bensalem et al. 2009) is used to check, offline, the absence of deadlocks within the BIP model. Note that this technique based on components and interaction invariant can potentially take into account search spaces larger than the ones that model checking techniques can handle, but the invariant can also be too weak and unable to prove any interesting properties. Additional safety constraints can be added and automatically translated from logical formulae into BIP. The resulting model is run within the BIP Engine on DALA, an iRobot ATRV (All-Terrain Robotic Vehicle), and the constraints are consequently enforced at runtime.

Using Fiacre/TINA, an automatic translation from the Fiacre model to Time Petri Net is performed, and the TINA tools can be used to prove interesting temporal properties. Unlike BIP/D-Finder, it is using model checking techniques which remain usable for relatively complex experiments (Foughali et al. 2017).

8 Conclusion

In this chapter, we presented an overview of the state of the art at the intersection of two broad fields which are Robotics and Artificial Intelligence. We reviewed models and techniques for addressing problems of planning and execution control of movements and tasks, interaction and learning. We discussed how to integrate decision-making functions with sensory-motor functions within a robot architecture. Most of these issues have been outlined very synthetically. Some were slightly detailed to provide the reader with illustrative frequently used representations and algorithms.

As stressed in the introduction, robotics is a multidisciplinary field. Significant progress in robotics can be expected from major advances in its basic disciplines. Further, robot can be a catalyst research target to advance these disciplines. For example, a light and fast mechanical gripper with high dexterity, an inexpensive accurate 3D range sensor, or an image recognition algorithm with broad and reliable performances for the ordinary objects that can be found in a house or store, will substantially enrich the functional capabilities of current platforms.

But, as we have also pointed out, robotics research is primarily integrative. One can certainly make progress in terms of basic components for some particular task or environment. But the autonomy of a machine when facing a diversity of environments and tasks requires progress in the integrated *perception - decision - action* control loop.

This loop is at the core of research in robotics. It requires explicit models of objects at various levels, from their physical appearance to their functions. It also requires models of activities, events and processes that constitute the environment and its agents, including the robot. It requires knowledge representations adapted to these models. These models are mathematically heterogeneous, that is, continuous/discrete, symbolic/numeric, geometric/topologic, deterministic/stochastic, etc. In robotics, the expression “knowledge representations” is necessarily plural. It also

requires a variety of learning techniques to acquire and improve these models. This is the research agenda, for which we have reviewed the progress over the past two or three decades, and on which more work remains to be done. This agenda is relevant to *self-contained* robots, which integrate all their components on a single platform, as well as to *distributed* robotics systems. Distribution is also an important item of this agenda. It concerns the distribution of cognitive functions over the components and functions of a single robot, as well as the distribution of robotics functions over a network of sensors, actuators and processing resources on a large scale.

It can also be argued that the *perception - decision - action* control loop is at the core of AI research. Progress is being made in all individual subfields of AI. For example, statistical and hybrid techniques have led to dramatic advances in automatic natural language processing, illustrated for example by the victory of the WATSON system in the question/answer game "Jeopardy" (Ferrucci et al. 2010). Representations coupling first-order logic and uncertainty management, such as probabilistic first order logic (Milch and Russell 2007), open remarkable opportunities, especially for the problems of planning and learning that we discussed here. Neural nets deep learning techniques (LeCun et al. 2015) have changed significantly the performance level of classification and interpretation tasks in numerous applications.

But the AI objective, namely *to understand, model and implement intelligence*, is seen by many researchers as being realized in the *perception - decision - action* control loop. Consider the problem of "*anchoring*", i.e., maintaining a mapping between a symbol and the sensory data related to the same physical object (Coradeschi and Saffiotti 2003), or the more general problem of "*symbol grounding*" (Harnad 1990), i.e., associating a symbol, in its context, to a signified content, object, concept or property. These problems requires the coupling of cognitive mechanisms to sensory-motor functions able to interact independently with the world to which symbols refer (the level T3 of the Turing test of Harnad 2001). For both fields, the coupling of Robotics and AI remains a very fertile research area.

References

Abbeel P, Coates A, Quigley M, Ng AY (2006) An application of reinforcement learning to aerobatic helicopter flight. In: Neural information processing systems, pp 1–8

Abbeel P, Ng AY (2010) Inverse reinforcement learning. In: Sammut C, Webb GI (eds) Encyclopedia of machine learning. Springer, pp 554–558

Alami R, Clodic A, Montreuil V, Sisbot EA, Chatilla R (2006) Toward human-aware robot task planning. To boldly go where no human-robot team has gone before, In AAAI spring symposium: to boldly go where no human-robot team has gone before, pp 39–46

Alami R, Fleury S, Herrb M, Ingrand F, Robert F (1998) Multi-robot cooperation in the MARTHA project. IEEE Robot Autom Mag 5(1):36–47

Allen JF (1984) Towards a general theory of action and time. Artif Intell 23(2):123–154

Antonelli G, Fossen TI, Yoerger DR (2008) Underwater robotics. In: Springer handbook of robotics, pp. 987–1008

Argall BD, Chernova S, Veloso MM, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(5):469–483

Barry JL, Kaelbling LP, Lozano-Pérez T (2011) DetH*: approximate hierarchical solution of large Markov decision processes. In: International Joint Conference on Artificial intelligence (IJCAI), pp 1928–1935

Barto AG, Bradtke SJ, Singh SP (1995) Learning to act using real-time dynamic programming. Artif Intell 72(1–2):81–138

Basu A, Bozga M, Sifakis J (2006) Modeling heterogeneous real-time components in BIP. In: International conference on software engineering and formal methods, pp 3–12

Beaudry E, Letourneau D, Kabanza F, Michaud F (2008) Reactive planning as a motivational source in a behavior-based architecture. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 1848–1853

Beetz M, Mcdermott D (1997) Expressing transformations of structured reactive plans. In: European conference on planning (ECP). Springer Publishers, pp 64–76

Beetz M, Jain D, Mösenlechner L, Tenorth M (2010) Towards performing everyday manipulation activities. Robot Auton Syst 58(9):1085–1095

Bensalem S, Bozga M, Nguyen T-H, Sifakis J (2009) D-finder: a tool for compositional deadlock detection and verification. In: Computer aided verification (CAV), pp. 614–619

Bensalem S, de Silva L, Ingrand F, Yan R (2011) A verifiable and correct-by-construction controller for robot functional levels. J Softw Eng Robot 1(2):1–19

Berthomieu B, Bodeveix J-P, Farail P, Filali M, Garavel H, Gaufillet P, Lang F, Vernadat F (2008) Fiacre: an intermediate language for model verification in the topcased environment. In: Embedded Real-Time Software and Systems, Toulouse. HAL - CCSD

Bertsekas DP (1995) Dynamic programming and optimal control. Athena Scientific, vols 1–2

Bhattacharya S, Likhachev M, Kumar V (2010) Multi-agent path planning with multiple tasks and distance constraints. In: IEEE international conference on robotics and automation, pp 953–959

Bit-Monnot A (2016) Temporal and hierarchical models for planning and acting in robotics. PhD thesis, Institut National Polytechnique de Toulouse-INPT

Bohren J, Cousins S (2010) The SMACH high-level executive [ROS News]. IEEE Robot Autom Mag 17(4):18–20

Bonasso RP, Firby RJ, Gat E, Kortenkamp D, Miller DP, Slack M (1997) Experiences with an architecture for intelligent, reactive agents. J Exp Theor Artif Intell 9(2/3):237–256

Bonet B, Geffner H (2006) Learning depth-first search: a unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In: International conference on automated planning and scheduling, pp 142–151

Boussinot F, de Simone R (1991) The ESTEREL language. In: Proceeding of the IEEE, pp 1293–1304

Brenner M, Hawes N, Kelleher J, Wyatt J (2007) Mediating between qualitative and quantitative representations for task-orientated human-robot interaction. In: International joint conference on artificial intelligence, vol 7

Brooks RA (1986) A robust layered control system for a mobile robot. IEEE J Robot Autom 2(1):14–23

Brugali D (2015) Model-driven software engineering in robotics. IEEE Robot Autom Mag 22(3):155–166

Buffet O, Sigaud O (eds) (2008) Processus Décisionnels de Markov en Intelligence Artificielle, Trait IC2, vol 1 et 2. Hermes - Lavoisier

Burgard W, Cremers AB, Fox D, Hähnel D, Lakemeyer G, Schulz D, Steiner W, Thrun S (1998) The interactive museum tour-guide robot. In: AAAI conference, pp 11–18

Cambon S, Alami R, Gravot F (2009) A hybrid approach to intricate motion, manipulation and task planning. Int J Robot Res 28(1):104–126. https://doi.org/10.1177/0278364908097884

Cao H, Lacroix S, Ingrand F, Alami R (2010) Complex tasks allocation for multi robot teams under communication constraints. In: Control architecture for robots, pp 1–12

Chatilla R, Laumond J-P (1985) Position referencing and consistent world modeling for mobile robots. In: IEEE international conference on robotics and automation, pp 138–145

Chaumette F, Hutchinson S (2008) Visual servoing and visual tracking. In: Springer Handbook of robotics, pp 563–583
Chien S, Knight R, Stechert A, Sherwood R, Rabideau G (2000) Using iterative repair to improve the responsiveness of planning and scheduling. In AAAI conference
Choset H, Lynch K, Hutchinson S, Kantor G, Burgard W, Kavraki L, Thrun S (2005) Principles of robot motion: theory, algorithms, and implementations. MIT Press
Cimatti A, Roveri M, Bertoli P (2004) Conformant planning via symbolic model checking and heuristic search. Artificial intelligence
Coates A, Abbeel P, Ng AY (2009) Apprenticeship learning for helicopter control. Commun ACM 52(7):97–105
Coradeschi S, Saffiotti A (2003) An introduction to the anchoring problem. Robot Auton Syst 43(2–3):85–96
Corke PI, Roberts JM, Cunningham J, Hainsworth D (2008) Mining robotics. In: Springer handbook of robotics, pp 1127–1150
Dhouib S, Kchir S, Stinckwich S, Ziadi T, Ziane M (2012) RobotML, a domain-specific language to design. simulate and deploy robotic applications, programming for autonomous robots. In: IEEE international conference on simulation, modeling, and programming for autonomous robots
Dias MB, Zlot RM, Kalra N, Stentz A (2006) Market-based multirobot coordination: a survey and analysis. In: Proceedings of the IEEE, pp 1257–1270
Elkady A, Sobh T (2012) Robotics middleware: a comprehensive literature survey and attribute-based bibliography. J Robot 2012:1–15
Erdmann M, Lozano-Perez T (1987) On multiple moving objects. Algorithmica 2:477–521
Erol K, Hendler J, Nau DS (1994) UMCP: a sound and complete procedure for hierarchical task-network planning. In: AI planning and scheduling (AIPS), pp 249–254
Espiau B, Kapellos K, Jourdan M (1996) Formal verification in robotics: why and how? In: International symposium on robotics research
Estrada C, Neira J, Tardós JD (2005) Hierarchical SLAM: real-time accurate mapping of large environments. IEEE Trans Robot Autom 21(4):588–596
Feron E, Johnson EN (2008) Aerial robotics. In: Springer handbook of robotics, pp 1009–1029
Ferrucci D, Brown E, Chu-Carroll J, Fan J, Gondek D, Kalyanpur AA, Lally A, Murdock JW, Nyberg E, Prager J, Schlaefer N, Welty C (2010) Building Watson: an overview of the deepqa project. AI Mag Fall 59–79
Fikes RE (1971) Monitored execution of robot plans produced by STRIPS. In: IFIP congress, Ljubljana, Yugoslavia
Fikes R, Nilsson N (1971) STRIPS: a new approach to the application of theorem proving to problem solving. Arti Intell 2(3–4):189–208
Finzi A, Ingrand F, Muscettola N (2004) Model-based executive control through reactive planning for autonomous rovers. In: IEEE/RSJ international conference on intelligent robots and systems, vol 1, pp 879–884
Firby RJ (1987) An investigation into reactive planning in complex domains. In: AAAI conference. Seattle, WA, AAAI Press, pp 202–206
Flint A, Murray D, Reid I (2011) Manhattan scene understanding using monocular, stereo, and 3D features. In: IEEE international conference on computer vision
Fong T, Kunz C, Hiatt LM, Bugajska M (2006) The human-robot interaction operating system. In: ACM SIGCHI/SUGAR conference on human-robot interaction, pp 41–48
Foughali M, Berthomieu B, Dal Zilio S, Ingrand F, Mallet A (2016) Model checking real-time properties on the functional layer of autonomous robots. In: International conference on formal engineering methods, Tokyo
Foughali M, Ingrand F, Mallet A (2017) GenoM3 Templates: from middleware independence to formal models synthesis. Technical report, LAAS
Frank J, Jónsson AK (2003) Constraint-based attribute and interval planning. Constraints 8(4):339–364

Fratini S, Cesta A, De Benedictis R, Orlandini A, Rasconi R (2011) APSI-based deliberation in goal oriented autonomous controllers. In: 11th symposium on advanced space technologies in robotics and, automation (ASTRA)
Frese U, Hausmann D, Lüth C, Täubig H, Walter D (2009) The importance of being formal. Electron Notes Theor Comput Sci 238(4):57–70
Ghahramani Z (1997) Learning dynamic Bayesian networks. Lect Notes Comput Sci 1387:168–197
Ghallab M, Alaoui AM (1989) Managing efficiently temporal relations through indexed spanning trees. In: International joint conference on artificial intelligence
Ghallab M, Laruelle H (1994) Representation and control in IxTeT, a temporal planner. In: International conference on AI planning systems, pp 61–67
Ghallab M, Nau DS, Traverso P (2004) Automated planning: theory and practice. Morgann Kaufmann
Ghallab M, Nau DS, Traverso P (2016) Automated planning and acting. Cambridge University Press, Cambridge
Gini ML, Ohnishi K, Pagello E (2010) Advances in autonomous robots for service and entertainment. Robot Auton Syst 58(7):829–832
Giralt G, Sobek R, Chatilla R (1979) A multi-level planning and navigation system for a mobile robot: a first approach to HILARE. In: International joint conference on artificial intelligence. Tokyo, Japan, pp 335–337
Goodrich MA, Schultz AC (2007) Human-robot interaction: a survey. Found Trends Human-Comput Interact 1(3):203–275
Guizzo E (2008) Kiva systems. In: IEEE spectrum, pp 27–24
Hägele M, Nilsson K, Pires JN (2008) Industrial robotics. In: Springer handbook of robotics, pp 963–986
Harnald S (1990) The symbol grounding problem. Physica D, pp 335–346
Harnad S (2001) Minds, machines and turing: the indistinguishability of indistinguishables. J Logic, Lang Inf; Special issue on "Alan turing and Artificial intelligence"
Henzinger TA, Sifakis J (2006) The embedded systems design challenge. In: International conference on formal methods. Lecture Notes in Computer Science 4085. Springer, pp 1–15
Hoffmann J (2001) FF: the fast-forward planning system. AI Mag 22(3):57
Hopcroft J, Tarjan R (1973) Efficient algorithms for graph manipulation. Commun ACM 16:372–378
Ingrand F, Chatilla R, Alami R, Robert F (1996) PRS: a high level supervision and control language for autonomous mobile robots. In: IEEE international conference on robotics and automation, pp 43–49
Ingrand F, Ghallab M (2017) Deliberation for autonomous robots: a survey. Artif Intell 247:10–44
Ingrand F, Lacroix S, Lemai-Chenevier S, Py F (2007) Decisional autonomy of planetary rovers. J Field Robot 24(7):559–580
Jaakkola T, Jordan MI, Singh SP (1994) On the convergence of stochastic iterative dynamic programming algorithms. Neural Comput 6(6):1185–1201
Jing G, Tosun T, Yim M, Kress-Gazit H (2016) An end-to-end system for accomplishing tasks with modular robots. In: Robotics science and systems
Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Intell Res 4
Kaelbling LP, Lozano-Perez T (2011) Hierarchical task and motion planning in the now. In: IEEE international conference on robotics and automation, pp 1470–1477
Kanoun O, Laumond J-P, Yoshida E (2011) Planning foot placements for a humanoid robot: a problem of inverse kinematics. Int J Robot Res 30(4):476–485
Kavraki LE, Svestka P, Latombe J-C, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 12(4):556–580
Kazerooni H (2008) Exoskeletons for human performance augmentation. In: Springer handbook of robotics, pp 773–793

Kim M, Kang KC (2005) Formal construction and verification of home service robots: a case study. Automated technology for verification and analysis. Springer, Berlin, pp 429–443
Konolige K, Marder-Eppstein E, Marthi B (2011) Navigation in hybrid metric-topological maps. In: IEEE international conference on robotics and automation
Kress-Gazit H (2011) Robot challenges: toward development of verification and synthesis techniques [from the Guest Editors]. IEEE Robot Autom Mag 18(3):22–23
Kress-Gazit H, Wongpiromsarn T, Topcu U (2011) Correct, reactive, high-level robot control. IEEE Robot Autom Mag 18(3):65–74
Kuipers B, Byun Y-T (1991) A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Robot Auton Syst 8(1–2):47–63
Kuipers B, Modayil J, Beeson P, MacMahon M, Savelli F (2004) Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In: IEEE international conference on robotics and automation, pp 4845–4851
Latombe J-C (ed) (1991) Robot motion planning. Kluwer, Boston
Laumond J-P (1990) Connectivity of plane triangulation. Inf Process Lett 34(2):87–96
LaValle SM (ed) (2006) Planning algorithms. Cambridge University Press, Cambridge
LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
Lemai-Chenevier S, Ingrand F (2004) Interleaving temporal planning and execution in robotics domains. In: AAAI conference
McGann C, Py F, Rajan K, Thomas H, Henthorn R, McEwen R (2008) A deliberative architecture for AUV control. In: IEEE international conference on robotics and automation (ICRA), pp 1049–1054
McGann C, Berger E, Boren J, Chitta S, Gerkey B, Glaser S, Marder-Eppstein E, Marthi B, Meeussen W, Pratkanis T et al (2009) Model-based, hierarchical control of a mobile manipulation platform. In: 4th workshop on planning and plan execution for real world systems at ICAPS
Mei C, Rives P (2007) Cartographie et Localisation Simultanée avec un Capteur. de Vision. In: Journées Nationales de la Recherche en Robotique
Milch B, Russell SJ (2007) First-order probabilistic languages: into the unknown. In: Inductive logic programming. Springer, pp 10–24
Minguez J, Lamiraux F, Laumond J-P (2008) Motion planning and obstacle avoidance. In: Handbook of robotics. Springer, pp 827–852
Mohamed N, Al-Jaroodi J, Jawhar I (2008) Middleware for robotics: a survey. In: IEEE conference on robotics, automation and mechatronics. IEEE, pp 736–742
Montemerlo M, Thrun S, Koller D, Wegbreit B (2002) FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: AAAI conference, pp 593–598
Montemerlo M, Thrun S, Koller D, Wegbreit B (2003) FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: International joint conference on artificial intelligence, pp 1151–1156
Moravec HP (1983) The Stanford cart and the CMU rover. Proc IEEE 71(7):872–884
Morisset B, Ghallab M (2002) Learning how to combine sensory-motor modalities for a robust behavior. In: Beetz M, Hertzberg J, Ghallab M, Pollack ME (eds) Advances in plan-based control of robotic agents. Springer
Muradore R, Bresolin D, Geretti L, Fiorini P, Villa T (2011) Robotic surgery. IEEE Robot Autom Mag 18(3):24–32
Muscettola N (1994) HSTS: Integrating planning and scheduling. In: Intelligent scheduling. Morgan Kaufmann
Nesnas IA, Wright A, Bajracharya M, Simmons R, Estlin T (2003) CLARAty and Challenges of developing interoperable robotic software. In: IEEE/RSJ international conference on intelligent robots and systems
Newcombe RA, Davison AJ (2010) Live dense reconstruction with a single moving camera. In: Computer vision and pattern recognition, pp 1498–1505
Nez-Carranza JM, Calway A (2010) Unifying planar and point mapping in monocular SLAM. In: British machine vision conference, pp 1–11

Nicolescu MN, Mataric MJ (2003) Natural methods for robot task learning: instructive demonstrations, generalization and practice. In: Autonomous agents and multi-agent systems, pp 241–248
Peters J, Ng AY (eds) (2009) Autonomous robots; Special issue on robot learning, vol 27. Springer, pp 1–2
Pineau J, Gordon G (2005) POMDP planning for robust robot. control. In: International symposium on robotics research, pp 69–82
Pineau J, Gordon G, Thrun S (2003) Policy-contingent abstraction for robust robot control. In: Uncertainty in artificial intelligence, pp 477–484
Prassler E, Kosuge K (2008) Domestic robotics. In: Springer handbook of robotics, pp 1253–1281
Py F, Rajan K, McGann C (2010) A systematic agent framework for situated autonomous systems. In: Autonomous agents and multi-agent systems (AAMAS), pp 583–590
Rajan K, Saffiotti A (2017) Artif Intell 247:1–9
Ringert JO, Roth A, Rumpe B (2015) Language and code generator composition for model-driven engineering of robotics component & connector systems. J Softw Eng Robot
Rosen CA, Nilsson N (1966) Application of intelligent automata to reconnaissance. Technical report, SRI International. http://www.ai.sri.com/shakey/
Russell SJ, Norvig P (2002) Artificial intelligence: a modern approach. Prentice Hall, Prentice
Rybski PE, Yoon K, Stolarz J, Veloso MM (2007) Interactive robot task training through dialog and demonstration. In: Conference on human-robot interaction, pp 49–56
Schlegel C, Hassler T, Lotz A, Steck A (2009) Robotic software systems: From code-driven to model-driven designs. In: International conference on advanced robotics. IEEE, pp 1–8
Schwartz JT, Sharir M, Hopcroft J (eds) (1987) Planning geometry and complexity of robot motion. Ablex series in artificial intelligence. Ablex Publishing
Siciliano B, Khatib O (eds) (2008) The handbook of robotics. Springer
Sigaud O, Peters J (eds) (2010) Studies in computational intelligence. From motor learning to interaction learning in robots, vol 264. Springer, Berlin. http://dx.doi.org/10.1007/978-3-642-05181-4
Siméon T, Laumond J-P, Cortés J, Sahbani A (2004) Manipulation planning with probabilistic roadmaps. Int J Robot Res 23(7–8):729–746. http://ijr.sagepub.com/cgi/content/abstract/23/7-8/729
Siméon T, Laumond J-P, Nissoux C (2000) Visibility-based probabilistic roadmaps for motion planning. Adv Robot 14(6):477–493
Simmons R, Apfelbaum D (1998) A task description language for robot control. In: IEEE/RSJ international conference on intelligent robots and systems, vol 3, pp 1931–1937
Simmons R, Pecheur C, Srinivasan G (2000) Towards automatic verification. of autonomous systems. In: Proceedings of the conference on intelligent robots and systems
Simmons R, Singh S, Heger F, Hiatt LM, Koterba S, Melchior N, Sellner B (2007) Human-robot teams for large-scale assembly. In: Proceedings of the NASA science technology conference. Citeseer
Smith R, Self M, Cheeseman P (1986) Estimating uncertain spatial relationships in robotics. In: Conference on uncertainty in artificial intelligence, pp 435–461
Sowmya A, Tsz-Wang So D, Hung Tang W (2002) Design of a mobile robot controller using esterel tools. Electron Notes Theor Comput Sci 65(5):3–10
Special Issue on AI and Robotics (2017). Artif Intell 247:1–440
Stachniss C, Burgard W (2004) Exploration with active loop-closing for FastSLAM. In: IEEE/RSJ international conference on intelligent robots and systems
Stentz A (1994) Optimal and efficient path planning for partially-known environments. In: IEEE international conference on robotics and automation (ICRA), vol 4, pp 3310–3317
Stulp F, Beetz M (2008) Combining declarative, procedural and predictive knowledge to generate and execute robot plans efficiently and robustly. Robotics and autonomous systems (Special Issue. on Semantic Knowledge)
Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press

Tate A, Drabble B, Kirby R (1994) O-Plan2: an architecture for command, planning and control. Morgan-Kaufmann

Täubig H, Menciassi A, Fichtinger G, Dario P (2008) Medical robotics and computer-integrated surgery. In: Handbook of robotics. Springer, pp 1199–1222

Teichteil-Königsbuch F, Fabiani P (2005) Symbolic heuristic policy iteration algorithms for structured decision-theoretic exploration problems. In: Workshop on planning under, uncertainty for autonomous systems at ICAPS

Teichteil-Königsbuch F, Kuter U, Infantes G (2010) Incremental plan aggregation for generating policies in MDPs. In: Autonomous agents and, multi-agent systems

Thrun S (2002) Robotic mapping: a survey. In: Lakemeyer G, Nebel B (eds) Exploring artificial intelligence in the new millenium. Morgan Kaufmann

Thrun S (2006) Stanley: the robot that won the DARPA Grand Challenge. J Field Robot 23(9):661–692

Tovey C, Lagoudakis M, Jain S, Koenig S (2005) The generation of bidding rules for auction-based robot coordination. In: Parker L, Schneider F, Schultz AC (eds) Multi-robot systems. From swarms to intelligent automata, vol III. Springer, Netherlands, pp 3–14

Triggs B, McLauchlan P, Hartley R, Fitzgibbon A (2000) Bundle adjustment - a modern synthesis. In: Triggs B, Zisserman A, Szeliski R (eds) Vision algorithms: theory and practice. Lecture notes in computer science, vol 1883. Springer, pp 298–372

Wilkins DE (1988) Practical planning. Extending the classical AI planning paradigm. Morgan Kaufman

Wolfe J, Marthi B (2010) Russell S (2010) Combined task and motion planning for mobile manipulation. In: International conference on automated planning and scheduling (ICAPS), vol 5, Toronto Canada. 2010 p

Wolpert DM, Flanagan JR (2010) Q&A: robotics as a tool to understand the brain. BMC Biol 8

Wolpert DM, Flanagan JR (2016) Computations Underlying Sensorimotor Learning. Curr Opinion Neurobiol 37:7–11

Wolpert DM, Ghahramani Z (2000) Computational principles of movement neuroscience. Nat Neurosci 3:1212–1217

Wong KW, Kress-Gazit H (2016) Need-based coordination for decentralized high-level robot control. In: IEEE/RSJ international conference on intelligent robots and systems. IEEE, 2209–2216

Yoshida K, Wilcox B (2008) Space robots and systems. In: Springer Handbook of robotics, pp 1031–1063. http://dx.doi.org/10.1007/978-3-540-30301-5_46

Zlot RM, Stentz A (2006) Market-based multirobot coordination using task abstraction. In: Yuta S, Asama H, Prassler E, Tsubouchi T, Thrun S (eds) Field and service robotics, vol 24. Springer, Berlin, pp 167–177

Artificial Intelligence: Philosophical and Epistemological Perspectives

Pierre Livet and Franck Varenne

Abstract Research in artificial intelligence (AI) has led to revise the challenges of the AI initial programme as well as to keep us alert to peculiarities and limitations of human cognition. Both are linked, as a careful further reading of the Turing's test makes it clear from Searle's Chinese room apologue and from Deyfus's suggestions, and in both cases, ideal had to be turned into operating mode. In order to rise these more pragmatic challenges AI does not hesitate to link together operations of various levels and functionalities, more specific or more general. The challenges are not met by an operating formal system which should have from the outset all the learning skills, but -for instance in simulation- by the dynamics of a succession of solutions open to adjustments as well as to reflexive repeats.

1 Introduction

The question "Can machines think?" as Alan Turing asked in the first sentence of his famous paper Turing (1950) became a plausible question with the appearance of new computing machines in the forties of the last century. It is linked with other questions, about mathematical creativity: "Can computers discover interesting mathematical theorems?", or about decision: "Can computers find better solutions to collective decision problems?" (see chapter "Collective Decision Making" of Volume 1) "Artificial Intelligence" has been coined in 1956 in order to cover these questions (see chapter "Elements for a History of Artificial Intelligence" of Volume 1). This slogan was deliberately provocative and raised passionate debates about the previously

P. Livet (✉)
Aix-Marseille Université, Marseille, France
e-mail: pierre.livet@univ-amu.fr

F. Varenne
University of Rouen, Rouen, France
e-mail: franck.varenne@univ-rouen.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_13

assumed monopoly of living bodies and particularly humans on these cognitive abilities. This debate smoulders under the embers and is reactivated from time to time, but its main arguments have been already expressed long ago. Computers are better than humans for operating on multiple and discrete formalized data (including games like checkers, chess and now Go) but their capacity to deal with the relevance problem and to be creative (in the – rather fuzzy human sense) are still disputable. The intensity of the debate has decreased, but other interesting philosophical and epistemological questions have been raised in a less passionate way (Ganascia 1990). Whether real progress has been made on these new questions is still a matter of debate, but the progress of AI leads us to change the way we ask these questions and at least the way we could conceive the answers. These questions are strongly related to each other. They are even entangled and, as a consequence, difficult to expose separately. Maybe showing how the old debates can be reconsidered from the present perspectives could be useful for making the shift of perspective easier to appreciate.

2 Three Classical Debates: Turing's Test, Searle's Chinese Chamber, Dreyfus' Phenomenological Arguments

2.1 Turing's Test

In his paper in Mind, Turing sets out his test in a relatively complicated way, but this complexity is meaningful. A human people H1 is supposed to dialog with two entities. The first one is a human being (H2 – in the initial game, it was a women), the second one a machine that is supposed to try to imitate the human being H2. If H1 cannot distinguish between the machine and the human being H2, the machine wins the game. Why not to compare directly H1 with the machine? The reason is that the properties that we want to test in this game are not the intrinsic properties of H1 and the machine - there are obvious differences. What we want to test instead, is to what extent the observable speech behaviour of the machine is similar to the observable speech behaviour of H2, at least at the best level of approximation that is available for H1. Nowadays, for usual conversations, computers can pass the test - except for tricky contextual dependencies and creative metaphors. Do not forget that this success is partly due to the limitations of the discrimination capacities of human agent and human observer in a usual interaction. The peculiarities of the Turing's test are also useful in order to avoid a difficulty of the simpler version (interaction reduced to H1 and the machine): H1 could believe he interacts with a human being because he over-interprets the sentences of the machine. For example, people interacting with the program ELIZA Weizenbaum (1966, 1983) found it "human": ELIZA was programmed for building sentences, using some elements of the questions asked by people and answering by asking other questions related to the psychoanalytic stereotypes triggered by words belonging to the human sentences. People were over-interpreting the personal relevance of these stereotypes. Maybe the

comparison between the sentences of ELIZA and the ones of a real psychoanalyst would have allowed them to make a difference? A human being could invent creative and relevant new associations that other human beings could regard as relevant - even if they do not understand their precise meaning.

One can make the hypothesis that the best AI machine could not produce such new associations. But such hypothesis cannot be proved, as there are no strict criteria for the relevance of associations that are creative but not precisely understood. Turing's test does not require such impossible proof, but its result is necessarily vague, since it depends on the limitations and approximations of human cognition.

2.2 Searle's Chinese Room

Searle claims that the story of the Chinese room (Searle 1980) – shows that AI fails to pass the Turing's test – in its simple but stronger version: Indiscernibility between the intrinsic properties of H1 and the machine. Of course, this stronger and simple version (SSTT) is not Turing's version – a test of the indiscernibility of the behavioural properties of H2 and the machine. Does the Chinese room story show that AI fails to pass the real or true Turing's test (TTT)? (Saygin et al. 2000) Searle is in a room. Someone passes him through a hole an ideogram Chinese text. Searle does not know Chinese, but has at his disposal a handbook that gives him, for any combination of ideograms, another combination (the handbook is the equivalent of a program). He identifies one combination in the Chinese text, writes the correspondent combination of the handbook and passes it back through the hole. As the handbook is well made, the sequence of combinations happens to be a meaningful conversation in Chinese. Nevertheless Searle cannot pretend to understand Chinese.

Searle claims that the difference between a human being and a program is that the operations of the program are only syntactic while the human being's cognitive processes require semantics. Let us now apply TTT. Do the human being's discriminative abilities make him able to distinguish between two behaviours, if the first one is the result of syntactic operations and the second one the result of syntactic plus semantic processes? In the Chinese room, Searle believes that the ideograms are related together by the syntactic correspondences specified in the handbook. Maybe they have a semantic counterpart, but only for the Chinese people outside the room. Or, to be more precise, the semantics of the handbook is a purely formal one: the formal correspondence between two sequences of ideograms. This formal correspondence is very poor relatively to the real semantics that relates written symbols to a lot of things and activities in the world. Therefore we have shown that Searle can distinguish two kinds of semantics, one limited to the transcription operations between ideograms inside the room, and the other that relates ideograms to things inside and outside the room. Searle seems to believe that this result shows that these syntactic operations are the intrinsic properties of the AI machine, while the human abilities to establish semantic relations are intrinsic human properties. This conclusion would require the stronger test, which requires comparing the intrinsic properties of the

machine to the intrinsic properties of Searle. But semantic relations are not purely intrinsic they need external relata. And, according to Searle, the internal basis of mind is a biological one let say, the dynamics of our neural system. And they may be a similarity between the operations of a program and the dynamics of neural connections- namely, that syntactic operations preserve semantic relations in each case. Of course, focusing on the syntactic operations that preserve semantic relations seems to beg the question. The stronger version of Turing's test cannot be conclusive because its conditions cannot be satisfied. Let us come back to the TTT. We would have to build two Chinese chambers, one for Searle, the other for the AI machine. Another human operator would have to distinguish the AI machine and Searle's behaviours. As by construction there are no differences between the operations of the machine and Searle's processes, the AI machine would pass the test.

Why does Searle conclude in favour of the opposite proposition? Because he believes that a semantic relation involves all the possible relations that a human being can have with his world. In this case he can complain that in the Chinese room, the relations to the outside world are limited to the transcription operations, while the Chinese people have many more relations at their disposal. If Searle were not accustomed to these richer relations, he would not complain to be limited in the Chinese room. Let us now consider the AI machine. In the TTT without Chinese room, the machine is related to the external world by some devices, at least the ones that make it able to interact with H1 and H2. So the "real" TTT would imply to compare Searle in the real world with the AI machine in the real world. And the old problem of TTT would reoccur: Are we able to detect salient and stable differences between Searle's interactions with the real external world and the interactions of the AI machine with the same external world? Are we able to give to AI machines the tools for developing semantic relations that are admissible by human beings? This question has no definite answer because we, as human beings, do not know the limits of the realm of such relations.

Let us shift to what seems a less difficult question: When our AI machines present limitations because the syntax we gave them cannot preserve our semantic relations, will we be able to solve this problem? Will we have the syntactic abilities to solve it? In the formal domain, in which differences are strictly defined, one could interpret Gödel's theorem as showing that this is not always possible. But in the pragmatic domain of ordinary life, differences are not so strict. Vagueness of human semantics could lead us to consider too easily some differences between men and AI machines as admissible while regarding also too easily other differences as not admissible.

2.3 AI and the Phenomenological Approach

Dreyfus and Dreyfus (1988) and Dreyfus (1992) have tried to show that as AI uses computation on symbols governed by formal rules, it cannot be akin to human intelligence. Their starting point was an analysis of the human experience in the first person of shifting from one level of competence to a higher one, from the status of

novice to the ones of advanced beginner, of competent, of proficient and of expert. The novice tries to apply rules, but as he goes further he accumulates experiences in different contexts and become able to select the relevant elements in each context (Collins 1996). At the end of this contextual learning process, he has no longer to search for the right decision, he just sees it. What he has to do appears to him obvious, but this does not imply that he is able to explain how he manages to know that it is the good decision. This learning process cannot be the result of external programming, but only of a kind of progressive incorporation of diverse kinds of sensitivity to little clues that get their meaning only from the contextual situation. This learning story leads Hubert Dreyfus to believe that without a biological body human intelligence cannot be approached. But the story seems only to show that without integrating multiple experiences and being able to evolve autonomously in interaction with its environment, no system could pass the Simple version of the Turing test (STT).

Let us assume that STT implies that H1 detects the difference between the human experience in the first person and the experience of a machine. As it seems impossible to know by experience what could be the experience in the first person of an AI machine, the phenomenological perspective seems to make impossible to give a meaning to the STT. Nevertheless, we can apply the TTT. For example, we may give the role of H1 to a novice, the role of H2 to an expert and examine whether the novice detects any difference between the machine and the expert. We have to experiment every possible combination, with expert in the role of H1, the novice in the role of H2, etc. We could rank the results of TTT's on a scale. For example, it seems easier for AI machine to pass the test when H1 is a novice and H2 is an expert, because the novice understands the machine as badly as the expert. Some situations have a flavour of paradox. AI machines might pass the test when H2 is a novice and H1 is proficient or expert - if the operations of the machines are clumsy and if the limits of their program are similar to the limits of the heuristics of novices. They pass the test when H1 and H2 are novices, if the possible differences of the limits of the machine with the limits of the novice are difficult to detect for a novice. A competent person can be unable to detect differences between competent H2 and a high level machine, because, again, she could have similar difficulties to understand both the expert and the machine. The most unfavourable situation for the machine to pass the test is that H1 and H2 are proficient or expert. But in the domain of games that can be formalized, as chess and GO (see chapter "Artificial Intelligence for Games" of Volume 2), AI machines pass the test even with experts. The conclusions of these different TTT's seems to be that the limitations of AI cannot be defined separately neither from the limitations of human abilities nor from the fact that we build AI machines as complement of our own limitations. Nevertheless there are limitations for which we have difficulties to define what the useful complements would be, because we do not know our own modus operandi. For example, we do not know precisely how a human being can become an expert, how he can integrate his experiences and become able to extract from them methods of evaluating and choosing in various situations the relevant way for him to define the problem and to find a solution. Would we have the opportunity to know better our integration processes, maybe it would be difficult to optimize them. Let us suppose that we use some kind of "deep learning". Deep

learning programs accelerate a lot the treatment of information and take into account many more combinations (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 1). But they are sensitive to the configuration of the available information, and this configuration can induce biases. For example, racist sentences are frequent on Internet, and finding ways of discarding them is a difficult problem for human, and also for machines. The conclusion of this section would be that we have to acknowledge that in these disputes around the test, we cannot separate the question of the human limitations and the one of the AI limitations. Given this conclusion, we prefer to evaluate the results of AI by comparing them with the challenges that AI researchers have themselves defined and by analysing the conceptual evolution of their research program.

3 Initial Challenges of AI Program of Research

AI program of research, at its beginning, could be defined as computo-representationalist and internalist. Intelligence was supposed to handle problems by building representations of their elements. Representations were expressed in formal symbols. Intelligent inferences and reasoning were operated by computational devices. The context, the environment had also to be represented and these representations also were internal to the computational system. AI was supposed to take up different challenges. We will formulate them, not necessarily in their initial terms, but in terms that are inspired by the evolution of AI:

1. To solve complex problems, without previously knowing a procedure for demonstrating their solutions – or at least to give a satisfying approximation of a solution. Procedures that could be logically validated are preferred (see chapter "Automated Deduction" of Volume 2). This computational challenge has been partly taken up.
2. To build computational systems that can learn, can find generalizations of their procedures and extend them to other domains than the one of their learning example basis (see chapter "Statistical Computational Learning" of Volume 1 and chapters "Automated Deduction", "Belief Graphical Models for Uncertainty Representation and Reasoning", "Planning in Artificial Intelligence" of Volume 2). This challenge is also partly taken up, even if it is always problematic to avoid generalizations that do not take into account contextual differences.
3. To simulate creativity in more informal domains (to prove interesting theorems, find new solutions and methods). This challenge can be seen as a combination of the first and the second challenges. The difficulty lies in the fact that combinations that bring forth new results have to be selected in order to extract only the relevant ones.
4. To give a formal account of usual human reasoning. Simply building inferential systems is not sufficient here, because the relevance of usual reasoning is sensitive to contexts and the relevant definition of context cannot always been determined in advance. The development of inferences can lead to revise it (see chapters

"Argumentation and Inconsistency-tolerant Reasoning" of Volume 1 and chapter "Planning in Artificial Intelligence" of Volume 2).

5. To understand human languages and converse with people in a relevant way. Problems here are not only linguistic, but also pragmatic and contextual (see chapter "Artificial Intelligence and Natural Language" of this volume).
6. To decide and monitor relevant actions. This challenge implies to solve the frame problem (i.e.: While developing an action, one has to take into account only the changes of the situation that are relevant for achieving the task); the qualification problem (when one has to determine what are the properties and qualities of the situation of the action, one has to select only the qualifications that are relevant for the task), and the problem of ramification (when examining the tree of possible alternatives, one has to avoid exploring the branches that are not relevant for the action) (see chapter "Reasoning about Action and Change" of Volume 1 and chapter "Planning in Artificial Intelligence" of Volume 2). Here again the problems are not only syntactic and semantic, but imply pragmatic relevance and its extensions to elements that were at first external to the representations of the system (see chapters "Artificial Intelligence and Pattern Recognition, Vision, Learning" and "Robotics and Artificial Intelligence" of this volume).

Other more ambitious challenges of the computationalist program of cognitive sciences were philosophical challenges like giving a computational account of intentionality, qualia, and consciousness (Garbay and Kayser 2011; Kayser et al. 2004; Penrose 1989; Pitrat 1995). In order to take up these challenges, research on programs has to try to overcome the problems of relevance and of interactions that depend on context and environment. The recurrence of these problems has made difficult to maintain a pure internal perspective and led to revise the problematic.

4 How Evolution of AI Shifts Epistemological Perspectives on Intelligence

At the beginning of AI, the usual analysis of the sequence of an intelligent behaviour was roughly the following: Perceiving a situation, extracting the more meaningful features, retrieving in memory the knowledge useful for inferring the consequences of these features and combining different known data until finding a combination (for example an action plan) that satisfies expected requirements (see chapters "Artificial Intelligence and Pattern Recognition, Vision, Learning" and "Robotics and Artificial Intelligence" of this volume). Evolution of AI led us to notice that this representation of intelligence omitted essential elements. Perception cannot be reduced to the neutral collection of a lot of data coming from the external world and triggering a deductive process. It has to be conceived as continuous interaction with environment, an interaction oriented towards goals. These goals can be modified because of the evolution of the external world, but also when one updates the action because of the already acquired results. In the same way, combining different pieces of knowledge

that could be useful is not a simple task, because their kinds could be very different. In addition one has to allow some local combinations to imply contradictions while avoiding that these contradictions paralyze the system (see chapter "Argumentation and Inconsistency-tolerant Reasoning" of Volume 1). An individual (and a fortiori a collective of individuals) has to be able to use information that she does not completely control, or that refers to actions and processes that are still in progress. One needs to know how to integrate partial, uncertain and vague data on the action during the action itself (see chapters "Knowledge Representation: Modalities, Conditionals, and Nonmonotonic Reasoning", "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility", and 'Case-Based Reasoning, Analogical Reasoning, and Interpolation'" of Volume 1). This continuous process of updating and revising was not central in the initial agenda of AI.

New questions arise. Is the heterogeneity of the components needed for an intelligent behaviour irreducible? Is it impossible to conceive a formal language that could express every component of an intelligent knowledge (universal expressiveness)? What would be the relation between this formal language and the natural languages? Is a specific logic required for each of the different modes of combination that are specific to each kind of components, or does a universal logic exist that rules any kind of combination? How to deal with vagueness?

With regard to the problem of expressiveness, AI brings a new perspective, the philosophical implications of which have not still be explored (see chapters "Validation and Explanation" and "Knowledge Engineering" of Volume 1). Human intelligence does not require finding a formal expression for anything in order to compute everything. Intelligence has a pragmatic dimension, related to the urgency of decision. If a fire flares up, one has to decide immediately whether there are ways of extinguishing it or whether it is better to get out. AI has made possible a finer evaluation of the trade-off between expressiveness and efficiency of the systems of representation: If the language of description is "too much" expressive, the decision system will be "too" slow. One of the results of AI is to make us aware of a lot of similar trade-offs in intelligent behaviours. These trade-offs are related to the different variants of the frame-problem: How to define what we are entitled to neglect? How fine the grain of description has to be depends of the variety of the rhythms of the task. The grain of the definition of the sub-tasks has to be variable, each level of granularity (see chapter "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1) - implying a more or less fine grain of the language of description – implying a different speed of inference that has to fit the required speed of decision. When the goals of the system change, new elements have to be expressed in a finer grain while previous details can now be neglected. To some extent, computer programs that use this category of self-observation modules called "activity tracking" and "activity awareness" for their algorithms can help to increase this optimization at a runtime (see chapter "Reasoning with Ontologies" of Volume 1).

These remarks lead to focus on the meta-cognitive dimension of the intelligent behaviour (Proust 2013). This dimension cannot be reduced to "reflection" but consists also in finding efficient ways of combining memory accessibility and infor-

mation about the states of present processing. In order to make the right decision, knowing how long it would be to make an in-depth reasoning, or what is the ratio between the amount of knowledge that has been already mobilized and the amount of information that could be processed with a bigger effort and in a longer time are meta-information that matter. It becomes clear that we need a measure of the distance between the present state of reasoning or the process of decision and the solution that we are looking for. If this distance increases or does not decrease, a meta-level examination of the cognitive or decisional strategy and possibly a change of strategy have to be triggered. When the question about what is the more efficient level of granularity becomes relevant for the evaluation of an intelligent behaviour, reasoning can no longer be reduced to its version in the framework of classical logic. In this framework, logical reasoning is submitted to the constraint of saving truth: When it is applied to true premises, it has to give true conclusions. But when premises may have different granularity, they are not required to be true in an absolute sense. What matters is not to get absolutely true conclusions, but conclusions that are still relevant if considered at the degree of granularity that is relevant for the task. This evolution leads to build non-classical logics (AI has been very productive in this domain), or to put in the background the logical concerns and to work on ways of processing uncertain and vague propositions. One could believe that the normative role of logic is challenged, but it is not the case, as logic is still needed in order to ensure that programs really do what we want they to do (see chapters "Automated Deduction", "Logic Programming" of Volume 2, and chapter "Theoretical Computer Science: Computational Complexity" of this volume) (Sallantin and Szczeciniarz 2007).

Relations between rationality and strict logical validity become less close when AI has to deal with the problems of the computation time and of computational complexity. Cook's theorem (1971) reminds us that satisfiability in propositional logic is NP-complete. Instead of focus on logical truth and its formal versions, a more pragmatic orientation can be taken. Reasoning has to be coherent, but also functional. The coherence of information, as long as the information does not disturb the efficiency of the task in progress, may be not continuously checked, but incoherence of information that would cause troubles and decrease in efficiency has to be corrected. The way of checking and validating the coherence in a reasoning or decision task may be specific to the type of the task. Generalizing one kind of validity to a larger context may require redefining this validity. Complex context dependency, including dependency of contexts on other contexts in accordance with a given architecture, may require distinguishing and articulating different levels of validity. Generalization may imply to take more risks and to pass from truth and validity to the less demanding notion of normality.

Taking into account the dependency of a task on its context, or criteria of what is desirable for a task, as well as the uncertainty of the data and information that are at one's disposal, leads to consider "normal" inferences. Normal inferences are valid in normal circumstances, but if the context changes, they may be defeated (see chapter "Main Issues in Belief Revision, Belief Merging and Information Fusion" of Volume 1). The guiding principles of intelligent behaviour are not restricted to normality (see chapter "Norms and Deontic Logic" of Volume 1) and normativity (see chapters

"Knowledge Representation: Modalities, Conditionals, and Nonmonotonic Reasoning" and "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility" of Volume 1), but these notions are related to ways of hierarchizing the different possible states (see chapter "Artificial Intelligence and High-Level Cognition" of this volume). Defining a hierarchy is usually the privilege either of a culture or a subjectivity. Intelligence depends also on cultural and even subjective factors. In this perspective, our well-founded inferences allow us to anticipate what can be "normally" expected. In order to decide an action, one has to compare what will normally follow this action with what will normally follow if we choose another action or do nothing. Usually, we make a distinction between "normal" denoting "things that happen frequently", and "normal" in the sense of "normative". We are in a first step more sensitive to information that is "abnormal" in comparison of what we expect and want, and in a second step we try to explain the reason why our expectation has been defeated by finding some anomaly in the environment. When we have identified an anomaly, we try to infer, by abduction, what plausible process could have produced it. AI has contributed by developing non-monotonic logics, and different systems of probabilistic revision, and has tried to give an operational content to the notion of norm and, correlatively, to the notion of exception by playing with the double meaning of "normal". It is possible to relate these two meanings by conceiving the "normative normal", this expression denoting not what is necessarily frequent but what would become so if things evolve the right way.

This notion of normal relation is close to the Humean notion of cause (Hume 1987) as implying the regularity of the relations between antecedent and consequent. There are no claims in AI about the metaphysical question of the existence of causal laws, but AI requires giving an operational content to the notion of causality (see chapter "A Glance at Causality Theories for Artificial Intelligence" in Volume 1 and chapter "Main Issues in Belief Revision, Belief Merging and Information Fusion" in this volume). For example, an intelligent system has to diagnose breakdowns, and to try to find out what are the causes of breakdowns. A robot has to plan its actions and to know what effects its actions will cause. AI seems to favour the "interventionist" conception of causality: A is regarded as a cause of B in context C if A is an exogenous intervention and if, in context C, B is known as a normal consequence of this intervention, while if A would have been absent, B would not be regarded as a normal consequence. Von Wright (1971) was one of the firsts to propose this conception, and Bayesian networks are one of its formal expressions (see chapter "Belief Graphical Models for Uncertainty Representation and Reasoning" of Volume 2). Philosophers suspected this conception of being too anthropomorphic, but this objection has less weight since AI has operationalized this conception.

AI has shed a new light on another topic of philosophical considerations, the status of language and meaning (see chapter "Artificial Intelligence and Natural Language" of this volume). In its beginnings, AI was mainly focused on the first two members of the tripartition between syntax, semantics and pragmatics. The problem of the dependence of meaning on the context was still let aside. But the acceptability of a sentence cannot be ensured without taking into account the context, and this notion of context implies a mixture of semantics and pragmatics. More recent

researches in the domain of computational linguistic are no longer restricted to the articulation of syntax and semantics. They exploit statistical correlations in huge corpus of linguistic data and associate them to different uses of language in different social and pragmatic situations. They analyse also the evolution of the network of words and of co-occurrences of words, as well as the dynamics of conversations. They relate more directly the regularities of co-occurrences of linguistic symbols and their pragmatic contexts of use. The tools developed in AI in order to deal with the problem of normality have influenced numerous works on language, including the use of analogies. Relations between syntax and semantics can be considered under a new light, and even the approach of pragmatics first developed in the philosophical current of analysis of "ordinary language", which was still focussed on a "grammar" of uses, and not on an evolutionary network of statistical correlations, could be renewed by these new perspectives.

The development of AI leads also to consider the pragmatic conditions of AI itself: The pragmatics of the interactions between programs and users (human beings are not the only category of users) (see chapter "Negotiation and Persuasion Among Agents" of Volume 1). The dynamics of these interactions has to be examined, simulated and formalized. One can refer to Sallantin's proposal (Sallantin and Szczeciniarz 1999): Thinking of a proof not only as a formal deduction, but as giving to the person that understand it new abilities to make inferences, and ways of overcoming cognitive obstacles. This implies to take into account together the dynamics of a proof and its capacities of interaction - or even "transaction" - with its potential public. In a different perspective, Jean-Yves Girard (2001) remains in the core of the fundaments of logic - the benchmark for the validity of programs – but nevertheless regard proofs as interaction - interactions with possible counter-proofs.

These epistemological shifts lead AI researchers to raise new problems and put out new challenges: Finding representation formats and processes for manipulating these formats that make possible to change the operational mode in accordance with contexts and problems. This implies to have at one's disposal a sufficiently diverse and rich variety of normalities; ensuring the possibility to set new normalities for new contexts. Operational devices have then to be possibly reusable and also adjustable and revisable. We could name this challenge the problem of a "dynamical capacity": Operations that include the possibility of dynamically modifying their functionality (see chapters "Reasoning about Action and Change" and "Formalization of Cognitive-Agent Systems, Trust, and Emotions" of Volume1, and chapter "Planning in Artificial Intelligence" of Volume 2. See also Livet (2002a, b)

Solving the problem of the dynamical capacity would also solve the problem of frame, of qualification, of ramification. Their solutions need to reintegrate in monitoring and planning of action the observed deviations of the course of action (deviations with respect to the initial anticipations about what plans of action were relevant for the goals of the action). In order not to have to compute every possible state, these deviations have not been taken into account, but some of them need to be considered. These unanticipated or neglected elements are the source of difficulties in the three forms of the frame problem.

The reader has perhaps noticed that the philosophical challenges (intentionality, qualia, consciousness) are no longer the main concerns of AI. One could think that this loss of interest is the result of the revision of the pretensions of the initial program of AI. But there is a more subtle reason. It could be an effect of a revision of the relations between the properly philosophical horizon of these claims and challenges, on the one hand, and the new horizons and challenges of AI, on the other hand.

5 The Right Place of Philosophical Challenges

Among the many challenges AI was said to take up, philosophical ones seemed to be the most difficult. Maybe this was an illusion. It may have come from the fact that philosophical requirements concerning intentionality, qualia, and consciousness had been somehow idealized and defined in a too speculative and operational way. Philosophers, when pointing out the properties that have to be satisfied, define them in their strongest form and choose their more normative and ideal definition. The properties defined in such a way could be called "horizon-properties". Philosophers do not even indicate how to find operational processes that satisfy these properties. Let us consider again things in the TTT spirit. We would have to compare to the horizon properties of the philosophers their AI-corresponding horizon-properties: The properties that the actual accomplishments of AI would be supposed to satisfy in their idealized development - a kind of horizon of the operational. For example the Universal Turing machine could be regarded as an idealized development (that can be carried out ideally, but not in a realistic time) of real Turing machines with their effective program. We could also demand, symmetrically, from philosophers that they give us operational versions of the philosophical horizon-properties, but this would require that philosophy of mind and its collaboration with cognitive psychology and neurosciences would be more advanced. In these symmetrical perspectives, it would be possible to show that, if we suppose that AI is able to give at least partial solutions to the problem of "dynamical capacity", which is an horizon-property for AI, we can believe that it will take up the philosophical challenges. Finding a solution to this problem requires that the initially built up representation structure still offer the possibility of changes (this is the dynamical part) and that the possibility of these changes is at least partially present as dispositions (this is for capacity) of the operations associated to this structure (partially, because the modifications of these operations, while belonging to their dispositions, have also to be triggered by their interaction with a new context). The functioning of an operational dynamic capacity would imply:

1. A computo-representational functioning with its regularities.
2. The insertion of its operations in a new environment, insertion that triggers variations.

3. The capacity of the computo-representational system of re-formatting itself and inducing re-categorizations - if by "re-categorizations" we mean recalibration of these operations in accordance with the normalities of the new context.

Intentionality in the philosophical sense (capacity of making reference to external as well as internal referents as considered under a specific aspect) requires the satisfaction of these three conditions, but they are sufficient for intentionality. The reason is that insertion in an environment makes possible to have relations to referents, which are grasped through representations under a given format - an ersatz of the notion of aspect– defined by the computo-representational system (see chapter "Validation and Explanation" of Volume 1). These representations do not by themselves give us access to the exogenous referents, but they can be modified in accordance with changes of the referent. If we do not take a static perspective, but a dynamical one (see chapters "Main Issues in Belief Revision, Belief Merging and Information Fusion", "Case-Based Reasoning, Analogical Reasoning, and Interpolation", "Argumentation and Inconsistency-Tolerant Reasoning" and "Reasoning about Action and Change" of Volume 1), the evolution of the referent is strongly related to the evolution of the mode of representation (the format), and this relation is a way to satisfy the philosophical criterion for intentionality. Of course we have no guarantee that this always happens, this is only a possibility, but an operational version of philosophical intentionality cannot ensure a better guarantee.

The same three conditions are sufficient for the production of "qualia" (qualitative experiences in the first person, "what it is like" to smell the odour of a rose). What is needed is that the effective functioning of the interactive operations with a specific environment brings forth particularizations of the canonical format of representation. In AI, these particularizations can be the results of various constraints: constraints inherent to the physical implementation of computational operations, or specifications of the modalities of data capture, as well as specific versions of a program, or particular parameterizations, or specific effects of some coordination between different kinds of programs, etc. If these particularizations can trigger some evolution of the operations of categorization, it is possible to regard them (in a dynamical, not in a static perspective) as both the operational and the AI horizon-version of the philosophical qualia: phenomenal contents that modify the content of basic categorizations by particularizing them and even giving them a singular content.

Consciousness presupposes qualia and their capacity to particularize categorizations. To be conscious is to be able to integrate information that is not already categorized, or is particularized, together with information already categorized and more generic. We can be convinced of that proposition if we consider the phenomenology of our conscious states: The conscious representation of a situation involves usual categorizations together with related elements that may not be categorized yet, but can be used as a pool for possible evolutions of categories. Our conscious integration has to save both the peculiarities of information in its phenomenality and the capacity of generalization or systematicity of the information.

The paradox of the Chinese room is overcome when our three conditions are satisfied. A computer program that could re-categorize most of the variants that

the Chinese external observer could introduce in the sentences that express Chinese questions, so as to be able to give answers to these questions, would surely be regarded as "understanding" Chinese. Nevertheless, there would be no guarantee that for each variation of context the program could give a relevant answer. But human beings are no more ensured to give optimally relevant answers when their environment is turned upside down.

If AI researchers would imitate philosophers and their preference for idealized horizon-properties, and raise the degree of idealization of the operational capacities of AI up to the level of the horizon-property that we have called "dynamic operational capacity", it would be difficult for philosophers to distinguish the AI version of this horizon-property and the philosophical version of "dynamical operational capacity" that they have to propose if they consider an operational version of their idealized properties. Idealized operational version and operational version of idealized properties would be very similar.

6 Attempts to Take Up New Challenges

We have shown that AI could take up the challenge of dynamical capacity (relevance, flexibility with respect to changes of context, capacity of defining new normalities), but it remains to show that this is operationally plausible.

As we are tempted to relate dynamical capacity and evolution, approaches that have a more interactionist and evolutionary flavour (as connectionism or revision of rules or genetic algorithms) might have seemed more promising.

Let us regard genetic algorithms (see chapter "Meta-Heuristics and Artificial Intelligence" of Volume 2) as a way to deal with the problem of dynamical capacity. At the beginning, we have sequences of coded symbols (structures) that have functional capacities; they are submitted to the process of variations and selection of a simulated evolution, and the results of these structural variations are selected in accordance with the new desired functional capacities. It seems to give a good example of a dynamical capacity. But in this process, the relation between computational operations and functional normalities that was ensured in the first phase by the initial coding has been lost, and nothing guarantees that one will found clear relations between the new functionalities associated with the new coding and the initial ones. We may also wonder whether the introduction of an evolutionary process and the fact that these evolutions are not deterministically programmed give more chances to take into account the context, or to correctly simulate the re-constitution of a cognitive and pragmatic capacity that a change of context can bring about in human intelligence.

People often too rapidly admit that it is enough for a computational system to be evolutional and interactive (including for robots interaction with the real world) in order for this evolution and interactivity to correctly simulate the evolution and interactivity of human intelligence. But one evolutional interaction can be very different from another one.

This possible dissimilarity is accepted in AI. It is admitted that the operational solutions to the problem of dynamical capacity does not require that a unique program is sufficient for taking up the challenge. The problem may imply a combination of different functionalities and forms of representation that either co-exist (without conflict) or are activated sequentially. Computers combine in an articulated way different levels and kinds of functionalities the structure and processes of which can be very different (electronic level, compilation levels, computational level, semantic level, and may be normalities level).

Most often, simulations on computers replace deductive processes by a series of computation on symbols that do not necessarily emulate any logical inference. To this extent, simulation can be seen as more generic as many cognitivist or even connectionist use of computers in AI. More importantly, a simulation presents at least two different phases: An operative one and on observational one. During the operative phase, a simulation gives rise to computations between discrete symbols. In the observational phase, the computer simulation triggers a series of observational measurements or reuse procedures that are performed on the symbolic collective patterns or clusters arising from the operative phase. These two phases can be performed by the computer program and be run simultaneously, i.e. at runtime. As a consequence, computer simulations can take advantage, first, of the duality of their intrinsic phases and, second, of the superposition of different (at least two) symbolic levels that, accordingly, has to be recognized. Of course, the level of computation needs a substrate of electronic elements the compatibility of which with operations on symbols is ensured. But, this implies different movements back and forth between the levels and between different kinds of operations. The elements that interact in a given inner level of the computational system can be regarded as full symbols in the sense that they are referring to certain things or symbols in the target system. But they could also be regarded as sub-symbolic (not strictly decomposable – and composition is not possible for every element, while some allowed compositions can be used as emerging symbols at the upper level) relatively to the symbolic functioning of an upper symbolic level in the denotational hierarchy: Hence, the connectionist notion of sub-symbolhood can be multiplexed, contextualized and, as such, generalized. One can observe, for instance, that complex computational models involve entangled submodels, some of which are only partially formalized, others ones that use different formalisms. This entanglement at runtime demands prior contextualized sub-symbolizations. The plenary use of computer simulation involves all these processes. This is the case for standard systems of multi-modelling as DEVS (Zeigler 1976) (their supporters try nowadays to simulate a universal modelling system), or for some complex computational models (for instance coupled agent-based/equation-based models) in empirical sciences like physics, biology and the social sciences. What is proper to the forms that are the results of a specific effective mode of functioning is taken without modification in its specificity (Varenne calls that an "iconic" mode 2009, 2018). Conversely what is symbolic from the beginning (and because of that, multi-realizable) is discretized in order to get a operational specific (iconic) aspect at a lower level and, because of that, the capacity to interact with the iconic aspects of other heterogeneous components of the system of models.

We have said that simulation implies the possibility of going back and forth between the iconic level of specific functioning and the level of results endowed with symbolic genericity. These back and forth moves make then possible to give generic capacities to iconic specific functioning and conversely to recharge symbolic capacities with iconic specificities. This transformation of specific into generic, and conversely, offers a possibility of simulating the simulation, as it makes possible the auto-observation of simulations that are internal to the system. Sensitivity to context will be made manifest by a decreasing of genericity, which will lead to looking for particularity, and, if things go well, to a converse movement towards a readjusted genericity, resulting in adaptation to new normalities. Of course one has to allow in the program the possibility of modifying the basic functioning as a function of the problems that appear at the generic level, in order to implement the chosen heuristics or models of symbolic cognition. The computer simulation of human practices of simulation (a well-known ability of human practical and theoretical cognition) could be regarded as one of these steps of increasing particularity that are intertwined with procedures of re-symbolization in form recognition, re-categorization, frugal heuristics, and so on and so forth.

Let us suppose that one respects the constraint of ensuring a sufficient level of compatibility and co-functionality (as in DEVS). In addition, let us assume that the computational system has implementations of each sub-symbolic level that save the iconicity in such a way that the form of the computation results is preserved and recognizable at the other levels and, at the end, recognizable by observers external to the system otherwise simulation is only a computational trick that allows to solve a computation problem. One is then able to use all available means in order to simulate the role of context (recharging particularity, then recharging genericity). One can use sub-symbolic functioning and operations on iconic symbols. One can ensure that fine syntactic differences (differences of implementation) trigger operations that have the result of modifying categories while saving their main functioning. This modification may result from a cumulative process that reaches a threshold, or it may result from emergence (in the weak sense of the term, as it is only a result from the computational although inescapable properties of the step-by-step operations). One could design procedures of revising categories by merging different systems and hierarchies of categorization in such a simulation. It may be also possible to automatize and modularize devices for building correspondences between the different ontologies that can be extracted ex post from various re-categorizations (see chapter “Collective Decision Making” of Volume 1) (Livet et al. 2010).

At this state of the art of AI, the conditions that have to be satisfied in order to deal with the problem of dynamical capacity require computational means in order to favour the back and forth move between at least two levels (the specific sub-symbolic realizations and the new symbolic combinations that they may induce), and the development of a three phases dynamics: Functioning, perturbation, re-adjustment or revision.

7 The Laboratory of Agent-Based-Simulation

We can find examples of this use of multi-levels relations and plurality of phases in Agent-Base-Models (ABM) via their simulations of the interactions of multiple agents (see chapter "Negotiation and Persuasion Among Agents" of Volume 1) (Livet 2007). The agents are automata or well-defined pieces of programs. They interact in accordance with their own rules. Their rules are selectively triggered as reaction to their environment (time, spaces), to their neighbours, to the messages they receive and to their inner representations and goals. The collective result of their interactions can be interpreted as an emergence of collective forms. These forms have not been defined from the beginning, neither in the model nor in the program. These forms change the environment, the evolution of which can make new structures emerge, and other ones be perturbed and disappear. Rules and parameters are adjusted either in order to obtain a structure that is similar to the one that the model is supposed to study, or to trigger transitions from one structure to another, if one wants to study the evolution of a system. Jacques Ferber has suggested that it would be useful to inscribe the different phases of the development of an agent-based-model in different frames and to articulate these frames or "quadrants" in accordance with the distinction between individual and collective. The main steps in this process are the following:

1. Defining the functions that are internal to each individual agent (Individual internal quadrant).
2. Defining its interactions with environment - that requires transducing the effects of the functions of the individual agent into external observables (individual external quadrant).
3. Defining the observable effects of the collective aggregation of the behaviours of individual agents (collective-external quadrant).
4. If agents are cognitive ones, defining how they can use the observation of the collective behaviours in order to modify their own functioning (collective-internal quadrant).

One can apply these frames to the relations between local computational functioning and their aggregate effects under a common format.

The process that goes from 1 to 2, then to 4, and comes back to 1 is symmetrical with the process between iconic and symbolic genericity that we have previously considered. In this previous process, recharging particularity induced a symbolic re-categorization; in this new process, collective compatibility induces individual re-categorization. The two perspectives are different ways to deal with the problem of the context. Most computer scientists do not try to solve this problem by defining sub-contexts as sub-categories inside a general frame of formalization (a global perspective that we regard as static, because the higher level categories are not changed). They give a dynamical solution to the problem: The categories that belong only to a single step are not solutions to the problem. Only the dynamic of transformations from one structure of categorizations to another can give solutions - solutions that are only temporary and open to revisions.

Other proposals have been suggested in order to solve the problem of dynamical capacity. “Embodied” AI and cognition – grounded in the reality of the perceived environment and constrained by the conditions of action in real situation, which implies to solve problems of trade-off between taking time for exploring the situation and time of reaction to the situation - can be regarded as an attempt to solve it. In the embodied cognition perspective, interactions with environment trigger recharging particularity. Nevertheless, solutions to the problem of the contextual relevance have still to be elaborated. Context consists in the combination of the present task and the state and changes of the environment, including the changes induces by action in the environment as well as the changes that they induce in the agenda of the system. The constraints implied by the need to achieve the task in limited time and in a real environment may lead to simplify the problem, but these simplifications could be peculiar to local environments, and we would want them to be more general. Here again, AI seems to have to give up a static and generic perspective for a dynamical perspective that combines particularity and genericity, maybe by introducing revisability.

8 Conclusion

Although it seems to raise ultimate and somewhat “eternal” philosophical questions, the AI research program has known many significant changes in the last decades. In connection with these changes, the conception of human intelligence that the first AI was supposed to compete with has been modified as well. We were tempted to attribute to human intelligence the capacity of simultaneously activating different virtues, in a static perspective. But recent parallel or convergent developments of AI and of research programs in cognitive psychology and complex models simulations suggest that human intelligence cannot activate (and even possess) all these virtues at the same time. It probably has to let aside some virtues in order to activate others, and conversely. Activating these different virtues can only be done dynamically. Intelligence uses bootstrapping and recursivity, or, less formally, one uses one’s know-how in order to improve one’s action. But intelligence has also to use what stands up to it and raises problems, as it makes possible to detect on which point the intelligent behaviour was faulty. Intelligence implies modes of reacting to what the previous representations and operations could not control, and this requires to combine its initial capacities and incentives to variations suggested by new situations. The new project of AI is to analyse these intelligent dynamics by simulating them. AI and human intelligence appear to have more limited ambitions but more evolutional capacities.

References

Collins HM (1996) Embedded or embodied? a review of Hubert Dreyfus' what computers still can't do. Artif Intell 80(1–2):99–117

Cook SA (1971) The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on theory of computing, pp 151–158

Dreyfus HL (1992) What computers still can't do - a critique of artificial reason. MIT Press, Cambridge

Dreyfus HL, Dreyfus SE (1988) Mind over machine - the power of human intuition and expertise in the era of the computer. Free Press, New York City

Ganascia J (1990) L'âme-machine: les enjeux de l'intelligence artificielle. Science ouverte, Éditions du Seuil

Garbay C, Kayser D (2011) Informatique et sciences cognitives: influences ou confluence? http://hal.archives-ouvertes.fr/hal-00713699

Girard JY (2001) Locus solum: from the rules of logic to the logic of rules. Math Struct Comput Sci 11(3):301–506

Hume D (1978) A Treatise of Human Nature. Oxford Clarendon Press, book I, Part III, section XIV, an edition with analytical index by Selby-Bigge, second edition with text revised by Nidditch

Kayser D, Magda FA, Stefan-Gheorghe P (2004) Intelligence artificielle et agents intelligents. Printech

Livet P (2002a) Emotions et rationalité morale. Presses Universitaires de France

Livet P (2002b) Révision des Croyances. Hermès

Livet P (2007) Towards an epistemology of multi-agent simulation in social sciences. The Bardwell Press, Cumnor, pp 169–194

Livet P, Müller JP, Phan D, Sanders L (2010) Ontology, a mediator for agent-based modeling in social science. J Artif Soc Soc Simul 13(1):3

Penrose R (1989) The emperor's new mind: concerning computers, minds, and the laws of physics. Oxford University Press, Oxford

Pitrat J (1995) De la machine à l'intelligence. Hermès

Proust J (2013) The philosophy of meta-cognition, mental agency and self-awareness. Oxford University Press, Oxford

Sallantin J, Szczeciniarz J (1999) Le concept de preuve à la lumière de l'intelligence artificielle. Nouvelle Encyclopédie Diderot, Presses Universitaires de France, http://books.google.fr/books?id=48PgPAAACAAJ

Sallantin J, Szczeciniarz JJ (2007) Il concetto di prova alla luce dell'intelligencia artificiale (post-face)

Saygin AP, Cicekli I, Akman V (2000) Turing test: 50 years later. Minds Mach 10(4):463–518. https://doi.org/10.1023/A:1011288000451

Searle JR (1980) Minds, brains and programs. Behav Brain Sci 3:417–457

Selinger EM, Crease RP (2002) Dreyfus on expertise: the limits of phenomenological analysis. Cont Philos Rev 35(3):245–279–279. https://doi.org/10.1023/A:1022643708111

Turing AM (1950) Computing machinery and intelligence. Mind 49:433–460

Varenne F (2009) Qu'est-ce que l'informatique ? Chemins Philosophiques, J. Vrin. http://books.google.fr/books?id=KTRPPgAACAAJ

Varenne F (2018) From models to simulations. Routledge. https://www.routledge.com/From-Models-to-Simulations/Varenne/p/book/9781138065215

Weizenbaum J (1983) Eliza - a computer program for the study of natural language communication between man and machine (reprint). Commun ACM 26(1):23–28

Weizenbaum J (1966) Eliza - a computer program for the study of natural language communication between man and machine. Commun ACM 9(1):36–45

Wright G (1971) Explanation and understanding. Cornell classics in philosophy. Cornell University Press, Ithaca

Zeigler B (1976) Theory of modeling and simulation, 1st edn. Wiley Interscience, New York

Artificial Intelligence and High-Level Cognition

Marco Ragni

Abstract Artificial intelligence (AI) and cognitive science (CS) both investigate information processing, but with a different focus: AI aims to build *problem solving machines*, i.e., systems capable of solving diverse problems in an efficient and effective way while CS analyzes human cognition. Both approaches increase an understanding of the foundations, methods, and strategies that can be employed to perform in a natural or artificial environment. This chapter focuses on *high-level cognition*, i.e., cognitive processes that are related to reasoning, decision making, and problem solving. After an introduction to the core principles, intersections, and differences between both fields, some central psychological findings are presented. In a next step cognitive theories for high-level cognition are introduced. While the architecture of cognition has an impact too, main approaches for cognitive modeling from cognitive architectures to multinomial processing trees are analyzed. Current challenges conclude the chapter.

1 Introduction

Artificial intelligence (AI) and cognitive science (CS) deal with information processing including analyzing and understanding how to store, manipulate, and derive new information. Both fields differ in their respective goals: AI aims to built efficient problem solving systems and CS aims to understand and to model human behavior. But both have something to offer to each other: AI provides a precise language and methods to describe information processes while CS investigates a working intelligent system. A connection between the disciplines is built upon the mapping from *brain* and *mind* to *hardware* and *program* (Searle 2004). An AI that focuses on excelling on a clear defined bounded domain is called *weak AI*. Strong AI's ultimate goal is to develop a system that does not only "simulate having a mind; it literally has a mind" (Searle 2004). This requires an understanding about the limits and powers

M. Ragni (✉)
Cognitive Computation Lab, Technical Faculty, University of Freiburg, Freiburg im Breisgau, Germany
e-mail: ragni@cs.uni-freiburg.de

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_14

of the human mind and to identify its setpoints, i.e., the specifics or properties of the system. If the properties are unknown, then it is not possible to evaluate if a system demonstrates capabilities we typically ascribe *a mind*. Arguments put forward (e.g., in the Chinese room argument Searle 1980) show that it is not easy to distinguish a mind from a mindless simulation. What makes the human mind as a system interesting from an AI perspective, is that it is not limited to a specific domain (like navigation), but that it is a general system that goes beyond any domain limitations, e.g., the mind that navigates is the same that performs any other cognitive operation. Still, this strong AI approach is not what many AI researchers focus on nowadays.

One way to develop strong AI systems is to take humans as a cognitive prototype that demonstrates intelligent processes. Humans are functioning instances of *intelligent systems* and despite great advances in AI that demonstrates the power of the systems over human performance, there are still specific types of problems that humans solve better. If, for instance, only imprecise information is given or *insight* is necessary, humans can often generate a solution that is *satisficing* and, they can adapt and generalize results to other domains. As humans *do* show features we expect from intelligent systems, it is worthwhile to learn *how* humans process information, to gain insights in order to model high-level cognitive processes computationally, and to make cognitive processes available for technical systems. An artificial system does not necessarily need to mimic human behavior, but it can be built on cognitive principles. Furthermore, as AI systems enter more and more our everyday life the ways humans interact with AI systems increases. For a better human/AI interaction this requires to equip interacting systems with an understanding of human information processing. Examples are any kind of technical systems that need to provide information in a comprehensible way. For this endeavor methods from AI are interesting as the possibility to express cognition as algorithms, to analyze the algorithmic features of cognition, and having a general and formalized set of tools available is methodologically sound. However, an interaction between AI and CS would not be possible if there is no overlap between the fields. Consider for example the following definition (Foundation 1978) based on computational processes:

> What the subdisciplines of cognitive science share, indeed, what has brought the field into existence, is a common research objective: to discover the representational and computational capacities of the mind and their structural and functional representation in the brain.

Relevant are the notions of *representational* and *computational capacities of the mind*. These notions can be described using concepts from AI, but the core paradigm of CS is to *"equate mental processes with information processing"* (Strube et al. 2013) and that "[t]he overall accepted notion in cognitive science is that symbols lie at the root of intelligent action" (Newell and Simon 1976) and hence a "structural requirement for intelligence […] is the ability to store and manipulate symbols" (Newell and Simon 1976). This is termed the *physical symbol system* hypothesis (Newell and Simon 1976): "A physical symbol system has the necessary and sufficient means for general intelligent action". Hence, AI and CS use similar methods

such as symbolic descriptions. *The* characteristic method of CS—*cognitive modeling*—is simply not possible without techniques from AI (Strube et al. 2013). Despite the fact that human intelligence is the current creator and main foundation in creating any AI system, a new movement to create cognitive systems, i.e., systems that incorporate principles of human cognition, has started (Hollnagel and Woods 2005). Once psychological theories are formulated algorithmically, they cannot only be tested—they can be made available for AI systems.

A distinction is made between *low-level* and *high-level* cognitive processes: Low-level processes are typically associated with sensation or simple memory processes; they do not require effort, are often unconscious, and most humans perform them easily and intuitively. In contrast, high-level cognitive processes are demanding and often require the simultaneous execution of several mental processes of memory and imagination. Examples in the literature list problem solving, decision making, learning, and language comprehension among high-level processes, e.g., Just et al. (1999), Sternberg (1999), lle Lépine et al. (2005), O'Reilly (2006), Dubois et al. (2008). As it is hardly the case that intelligence is considered by researchers from CS without relating it to the capability to reason and to solve problems we will focus on both aspects in the following sections.

2 Core Empirical Results on High-Level Cognition

In this section we will get a flavor about specifics of human high-level cognition and how cognition is actually not captured by classical normative AI approaches. A core aspect of intelligence, be it natural or artificial intelligence, is the ability to *reason* about given information and to *solve problems*. *Reasoning* is the ability to gain new information from existing knowledge. Processes of classical reasoning and problem solving are similar to each other. They are often not treated differently—neither in AI nor in CS (Sternberg 1980; Newell 1979; Wason 1971; Greeno and Simon 1988; Baral 2003). Can we define a common basis for problem solving and reasoning? Typically, *information* and a *task* is given. The information can comprise premises in the case of reasoning or a description of an initial state for a problem. It is often so-called *declarative knowledge* (see below). The task description is often more *procedural*, i.e., derive new information or generate a different scenario by the application of some operators. In logical reasoning, however, humans do not expect to be informed about applicable operators. Humans take implicitly for granted how they have to process given information (cf. Table 1). Despite the great similarities between the processes of logical reasoning and problem solving, a difference lies in the (under-) specified operations. While applying known operations in a search problem is simple; it is still not possible, despite great advances in the field of AI, to construct machines that solve arbitrary *insight* problems (see below), while humans have demonstrated this ability.

Table 1 A comparison between reasoning and problem solving

	Logical reasoning	Problem solving
Given information	Premises	Initial scenario; goal scenario; (partially) operators
Task/Question	What follows? Or: Does conclusion X follow?	Is there a transformation from the initial to the goal scenario (using the operators)?
Operators	Often not given	Sometimes explicitly given

2.1 Psychological Findings

Research on human reasoning can be roughly divided into the categories of deductive, inductive, and abductive reasoning (cf. Strube et al. 2013).

Deductive reasoning can be defined as the method of drawing a conclusion from a given set of statements. As a normative framework psychologists often evaluate the answers from a normative perspective, e.g., a statement is only accepted as a correct conclusion if it can be derived by applying predicate logic to the premises.

Given: A set of premises $p_1, \ldots, p_n$.
Question: What follows? (Which conclusion(s) can be drawn?)

In contrast to logic, conclusions humans draw, need to be *meaningful* and *different* from the premises though there is no crisp definition frame. This *meaningfulness* often orients itself by Gricean communication principles (see below).

Inductive reasoning often require to formulate and test a hypothesis, i.e., a statement that describes data. A typical example is Halbmayer and Salat (2012): Given the observations that doves, eagles, hawks are birds and can fly, an observer could form the hypothesis that all birds can fly. Hence such an inference is based on a number of observations and a summarizing expression (often formulated as a conditional, e.g., if something is of type B then they have property F).

Given: Two sets M', M with $M' \subset M$ and relation R holds for all m of M'.
Question: Holds relation R for all m of M?

Humans do accept conclusions as long as there is no identified counterexample. If we learn that a penguin is a bird that cannot fly, the previously drawn conclusion that birds can fly does not hold anymore. In other words, inductive inferences cannot be inferred with certainty. *Reasoning about analogies* is often classified as a special form of inductive thinking (Beller and Spada 2001). It transfers knowledge of a *source* domain to a *target* domain. A possible formal definition of an analogical problems is (cp. Strube et al. 2013):

Given: Two domains D_1 and D_2; in D_1 the relation R holds between elements E_1 and E_2.
Question: Is there a function f, such that for elements $f(E_1), f(E_2)$ from D_2: $R(E_1, E_2) \in D_1 \Leftrightarrow R(f(E_1), f(E_2)) \in D_2$ holds?

Abductive reasoning is a mode of reasoning (Douven 2011). It has been characterized of as finding a best explanation $E_1, \ldots, E_n$ for a fact F. It has been so far mostly neglected in the psychology of reasoning.

In contrast to classical methods from AI, human reasoning mechanisms are not always sound with respect to a given normative theory. Though some errors are due to lack of focus or misunderstanding the majority of errors by most reasoners are systematic deviations from classical normative theories like predicate logic or probability theory. Researchers have focused on such aspects as they provide insights about the underlying mental representations and mechanisms.

2.1.1 Relational Reasoning

The way how humans represent and reason about relational information and what can cause reasoning difficulty depends on many factors on different cognitive levels. Before we analyze them, consider the following problems (Ragni and Knauff 2013):

(1a) Flight UA is north of flight LH.
Flight LH is north of flight AA.
Flight AA is north of flight DL.
What follows for flight UA and flight DL?

(1b) Flight UA is north of flight LH.
Flight UA is north of flight AA.
Flight AA is north of flight DL.
What follows for flight UA and flight DL?

The left-hand problem (1a) is called a *determinate problem*, i.e., there is only one qualitative arrangement of the aircrafts possible. Instead the right problem (1b) is called an *indeterminate problem*, i.e., different qualitative arrangements of the flights are possible. Nonetheless, the conclusion is for both cases identical: flight UA is north of flight DL.

We now analyze the processing of such (spatial) relational information and its relation to the internal mental representation and will refer to the following three levels later on.

Level 1: Processing of information. Some core findings are: The *symbol distance effect*, i.e., information that is presented in such a way that it is in relation to an information presented immediately before (*continuous order*) is easier to process than information that is at first unrelated and only later related and integrated (*discontinuous order*) (Potts 1974). Another factor is the *relational complexity effect*, i.e., information that contains relations with higher arity (e.g., the ternary relation in-between) is more difficult to reason with than information formulated with relations of smaller arity (Halford et al. 2010).

Level 2: Internal representation of information. The *ambiguity or indeterminacy effect of information*, i.e., if a relational description is ambiguous and, hence, allows for several possible models, a conclusion is harder to draw then if the description

allows only for one model, e.g., the indeterminate vs. determinate problems above. The *preference effect*: Reasoners tend to build a preferred mental model, based on a direct integration of information, for ambiguous descriptions and form preferred conclusions based on that. The generation of the preferred model can be due to working memory limitations. The *visual impedance effect*, i.e., relations that are easier to visualize can *impede* the reasoning process in contrast to relations that are rather visually abstract but spatial in their nature (Knauff and Johnson-Laird 2002). This potentially requires additional effort in brain regions connected with visual information (Knauff 2013). But additional visual presentations can support reasoning processes too, e.g., Bauer and Johnson-Laird (1993) presented problems with diagrams that enhance the idea of alternative interpretations resulting in better performance.

Level 3: Manipulation of the internal representation. The *transformation distance effect*: Reasoners tend to neglect alternative models, especially if the operational distance to transform the preferred mental model into the alternative mental model is high (Ragni and Knauff 2013). This explains a source of reasoning errors on an operational level, especially if an alternative mental model is a counterexample to a putative conclusion a reasoner forms based on their preferred mental model.

2.1.2 Syllogistic Reasoning

The above introduced three levels are often part in any kind of reasoning processes. One particular domain deals with reasoning about quantities and often only with syllogisms. In CS and psychology, syllogisms use the classical quantifiers such as *All, Some, Some .. not*, and *None*. Recently, generalized quantifiers such *Most* or *Few* and *Normally* have been investigated. Consider the following example (Klauer et al. 2000):

(2a) No cigarettes are inexpensive.
Some addictive things are inexpensive.
Some addictive things are not cigarettes.

(2b) No addictive things are inexpensive.
Some cigarettes are inexpensive.
Some cigarettes are not addictive.

Most participants accept the conclusion (below the line) in (2a) which is a valid answer but fewer (46%) accept the conclusion in (2b) despite being valid. This indicates that humans tend to evaluate the truth of a putative conclusion not necessarily on the given premises but how convincing a conclusion is. This is called the *belief-bias effect*. Another result is that just the internal problem representation can influence the responses. Consider:

(3a) All astronauts are computer specialists.
Some computer specialists are nerds.
What, if anything, follows?

(3b) Most As are Bs.
Most Bs are Cs.
What, if anything, follows?

Many participants (70%) conclude for similar problems like (3a) that "Some astronauts are nerds" (Khemlani and Johnson-Laird 2012). Although this is a possibility, it cannot be logically concluded, but this suggests that humans deviate from classical logical inferences. Problems using the generalized quantifier *Most* (see (3b)

above), where about 60% select *Most As are Cs*, give hints at the internal mental representation that influences the answers but the counterexample require to think about different set sizes. On the level of *processing the information* (see above) the quantifiers are interpreted differently than in formal logics: The Gricean implicature claims that communication principles have an influence on the interpretation of quantifiers (Newstead 1995), e.g., *Some* is interpreted as *Some, but not all*. Similar to relational reasoning the order in which information is presented has an influence on accuracy (Khemlani and Johnson-Laird 2012).

On the level of *the internal representation of information* the adequate representation for the average reasoner is not yet identified: A meta-analysis (Khemlani and Johnson-Laird 2012) demonstrates that any cognitive theory, be it model-based, rule-based, probabilistic or heuristic, deviates significantly from the empirical data of six studies; the current best model is mReasoner (Khemlani and Johnson-Laird 2013), a system based on mental models and heuristics. A potential explanation for the deviations could be the large inter-individual differences between reasoners (see Challenge 4 below). Recently, an analysis of subgroups has been undertaken (Khemlani and Johnson-Laird 2016). As the internal representation of the second level is not yet fully understood, only some theories (e.g., the mental model or the PHM theory) formulate processes on Level 3 where internal representations need to be manipulated; but there is not enough psychological data to analyze this. Recently, an analysis of the cognitive difficulty of language processing and cognitive resources including formal approaches like parametrized complexity measures has been conducted (Szymanik 2016).

2.1.3 Reasoning about Conditionals and Propositions

A conditional statement can be used to describe observations or explain facts, e.g., "if it rains then the street gets wet", it allows one to formulate scientific predictions, e.g., "if the air pollution continues, the ozone hole increases" or to reason about counterfactuals, e.g.,"if Oswald had not shot Kennedy, then someone else would have" (Byrne 2007). Conditionals can describe causal or temporal dependencies, definitions, and compressions of observations, aggregating different aspects in a short description:

(4) If the system passes the Turing test, then the system is intelligent.

A conditional consists of an *antecedent*, e.g., in the conditional above, "the system passes the Turing test", and a consequence, "the system is intelligent". Four inference rules have been investigated in the context of conditional reasoning. Let us consider the case where participants have the conditional above and additionally a fact be given, then four rules are possible: The *modus ponens* rule (for a, and, if a then b, conclude b), hence, for "The system passes the Turing test" then it can be inferred that "the system is intelligent". The three other rules are *denial of antecedence* (from $\neg a$, and, if a then b, conclude $\neg b$), *affirmation of the consequence* (from b, and, if a then b, conclude a), *modus tollens* (from $\neg b$, and, if a then b, conclude $\neg a$). While only

modus ponens and modus tollens are logically correct, humans do, depending on the information and the background context accept or derive other conclusions, too. If participants receive first a negative consequence $\neg c$ and only then the conditional *if a then c*, the participants apply the modus tollens more often (Legrenzi et al. 1993). Participants without training in formal logic suppressed previously drawn conclusions when additional information became available (Byrne 1989). Interestingly, in some instances the previously drawn conclusions were valid whereas in other instances the conclusions were invalid with respect to classical two-valued logic. Consider the following *suppression effect* (Byrne 1989):

(5a) If she has an essay to write,
then she will study late in the library.
If she has a textbook to read,
then she will study late in the library.
She has an essay to write.
What, if anything, follows?

(5b) If she has an essay to write
then she will study late in the library.
If the library stays open,
then she will study late in the library.
She has an essay to write.
What, if anything, follows?

Most participants (98 %) concluded for problem (5a) *"She will study late in the library"*. If participants instead receive problem (5b), only 38% of the participants concluded *"She will study late in the library"*. This shows that although the conclusion is logically still correct, it is suppressed by an additional conditional which is an supports the assumption that human reasoning demonstrates features of *non-monotonic* reasoning. A characteristic of non-monotonic reasoning is, that new information can reduce knowledge. The famous Wason Selection Task tests how humans evaluate a hypothesis formulated as a conditional (Ragni et al. 2018):

(6) The experimenter explains to the participants that each card in a pack has a letter on one side and a number on the other side. Four cards chosen at random from the pack are placed on the table, e.g., E K 2 3. The experimenter presents the following general hypothesis: If there is a vowel on one side of a card, then there is an even number on the other side. The participants task is to select all those cards, and only those cards, which would have to be turned over in order to discover whether the hypothesis is true or false about the four cards on the table. Participants make their selection; and the task is over.

Many participants check only the card with the vowel or the card with the even number, albeit the logical correct answer is to select the card with the vowel and the card with the odd number. If the abstract task is replaced by an isomorphic representation with drinking beer and being over 18—a deontic formulation of the task is obtained for which more participants chose the classical logically correct answers. For conditionals so-called *enablers* or *defeaters* can exist, i.e., facts that support the conditional or not and they can have an effect on the construction of a mental representation. As a consequence they can increase or decrease the likelihood to draw an inference (Verschueren et al. 2004).

Reasoning about counterfactuals is a "mental undo" of a fact or observation, e.g., "if Oswald had not shot Kennedy, then someone else would have", these are often generated after goal failures and are related to causal thoughts (Byrne 2002). The role of counterfactuals is to test the relation between different antecedents and the cause hereby identifying the strength of the (causal) connection between antecedent and

consequence in a conditional. They lead to an increase of the application of the modus tollens in reasoning (Byrne 2002) and there is a preference to think about exceptional alternative events and actions (Dixon and Byrne 2011). There exists illusions in reasoning about propositional assertions (Khemlani and Johnson-Laird 2009). Illusions are inferences where participants are convinced that the drawn conclusions are true, while they are wrong:

(7) Consider, for instance, the following assertion: "You have the bread, or else you have the soup or else the salad. Given the further premise: You have the bread." What follows? Can you have the soup too? What about the soup and the salad?

Please note that participants were instructed to interpret the or else as an exclusive or (XOR). Hence the problem can be reformulated as bread XOR soup XOR salad. But, only 17% of the participants gave the correct answer that you can have all three. This answer may depend on the underlying mental representation as we will see later.

2.1.4 Common-Sense and Heuristic Reasoning

Psychological findings indicate that human reasoners do deviate from classical logical approaches. But do human reasoners adhere to the laws of probability? Consider the following Linda problem (Tversky and Kahneman 1983):

(8) Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations. Which is more probable?

A Linda is a bank teller.
B Linda is a bank teller and is active in the feminist movement.

Statement B is a conjunction of the statement A: "Linda is a bank teller", and another statement "is active in the feminist movement" (which we abbreviate by C). Hence the probability of the joint event $P(B)$ is the probability $P(A \wedge C)$ and this can be at most as high as the single probability $P(A)$ (short: $P(A \wedge C) \leq P(A)$). But, most participants (85%) select response B and decide that the answer $P(A \wedge C)$ is more likely then $P(A)$. Hence, they deviate from the predictions of the probabilistic calculus.

An example of a connection between heuristic reasoning and the involvement of memory is the so-called *availability heuristic* (Tversky and Kahneman 1973):

(9) Suppose you sample a word at random from an English text. Is it more likely that the word starts with a K, or that K is its third letter?

About 66% of the participants stated incorrectly that there are more words that start with a K with the rest stating the correct answer that the third letter is more likely. People do consistently commit such an error because the initial letter has a more

relevant role in our memory. As a lexical arrangement, it seems to play a large role, so ultimately, a memory management principle is the cause of this misconception.

In decision making theory another deviation from probability theory is the *sure thing* principle (Busemeyer and Bruza 2012):

(10) Imagine that you have just played a game of chance that gave you a 50% chance to win $200 and a 50% chance to lose $100.
(a) The coin was tossed and you have [won $200/lost $100] and you are now offered a second identical gamble.
(b) Imagine that the coin has already been tossed but that you will not know whether you have won $200 or lost $100 until you make your decision concerning a second, identical gamble.
Would you accept or reject the second gamble?

While in condition (10a) most participants accept a second gamble (69% if they have won and 59% if they have lost) only 36% of the participants would do so in condition (10b).

2.1.5 Problem Solving

Research in psychology on problem solving started with the work of Gestalt psychologists with many problems that are called *insight problems* (Duncker 1945; Wertheimer 1923). Much due to the influence of the General Problem Solver research into *permutation problems* started shortly afterwards (Newell and Simon 1972). And, with computer analysis and strategy games the domain of *complex problems* followed Funke (2006). They are characterized depending on additional features (see, Table 2). The probably most famous AI method, namely a *search of the problem space* often fails to describe the human reasoning process due to limitations of the human working memory. Instead, content is accessed from long-term working memory or the method of case-based reasoning is used Strube et al. (2013). All three problem classes are now introduced.

Table 2 An overview of different problem classes. *Static*: the problem does not change while the agent deliberates; *Observable*: all relevant information of the problem is given; *Search Space*: the set of all possible states is given with the problem description; *Operators*: the set of all possible operations is given with the problem description. *redefine**: requires to adapt given operators or the search space. [a]Introduced by Russell and Norvig (2003); [b]Introduced by Duncker (1945); [c]Introduced by Frensch and Funke (1995)

Problem type	Environment	Observable	Search space	Operators	Examples
Permutation[a]	Static	Fully	Defined	Known	Tower of Hanoi
Insight[b]	Static	Fully	Identify and redefine*	Identify and redefine*	Raven's IQ test
Complex[c]	Dynamic	Partially	Partially	Partially	Lohhausen

Permutation problems. An instance of this class is the classical *cannibals and missionaries* problem (Simon and Newell 1962):

(11) There are three missionaries and three cannibals on the bank of a wide river, wanting to cross. There is a boat on the bank, which will hold no more than two persons, and all six members of the party know how to paddle it. The only real difficulty is that the cannibals are partial to a diet of missionaries. If, even for a moment, one or more missionaries are left alone with a larger number of cannibals, the missionaries will be eaten. The problem is to find a sequence of boat trips that will get the entire party safely across the river-without the loss of any missionaries.

Other examples include the Tower of Hanoi/London (Anderson 2007; Kaller et al. 2004) or the PSPACE-complete Rush Hour problem (Flake and Baum 2002). Psychologists often call the applied method *means-end analysis*: Identify the *ends*, possible operations (the *means*), and then select the operators that, applied to the current state, reduce the distance to the goal (*difference reduction*), which is a form of Greedy strategy. During problem solving applying an operation that requires to increase the distance from the current state to the goal is more difficult (as in the cannibals and missionaries example above). Recent results indicate that humans apply a specific kind of chunking of objects in the search space (Bennati et al. 2014). Hence, not only the working memory capacity but as well the specific internal representation influences the planning process. As a result it indicates that humans are not searching the entire problem space, but rather prune the search tree by systematically preferring specific operations or heuristics.

Insight Problems

Another important class of problems are so-called *insight problems*. These problems, originally inspired by Gestalt psychologists, e.g., Duncker (1945), cannot be solved by an exhaustive search (Chu and MacGregor 2011). They almost always require a spontaneous insight, called the *Eureka effect*, and the identification of operators (Gilhooly and Murphy 2005). From a computational perspective, such problems are difficult to conceptualize and to algorithmize. The candle problem, originally termed the box problem (Duncker 1945), is the following:

(12) On the door, at the height of the eyes, three small candles are to be put side by side ("for visual experiments"). On the table lie, among many other objects, a few tacks and the crucial objects: three little pasteboard boxes (about the size of an ordinary matchbox, differing somewhat in form and color and put in different places). Solution: with a tack apiece, the three boxes are fastened to the door, each to serve as platform for a candle. In the setting a.p., the three boxes were filled with experimental material: in one there were several thin little candles, tacks in another, and matches in the third. In w.p., the three boxes were empty. Thus F_1: "container"; F_2: "platform" (on which to set things).

An explanation of this solution is the so-called *functional fixation*: human reasoner consider the matchbox as a container for objects, but not to be a fixture for the candle. Other findings (Fauconnier and Turner 2008) support that human reasoners construct *mental spaces*. To solve some insight problems, these mental spaces must become superimposed to so-called blended spaces where operations are possible that were

not in any of the original mental spaces. Many IQ-test problems require insight as the operators of the problem are not specified. A specific class of IQ-tests are *geometrical analogy problems*. An example is Raven's *Standard Progressive Matrices* (SPM), to test average intelligent adults, and *Advanced Progressive Matrices* (APM) to test above average intelligent adults (Raven et al. 2000; Raven 1962). Participants are presented with a 3×3 matrix where in 8 of the 9 cells geometrical objects are present and the underlying function needs to be inferred. Especially the latter aspect, identifying patterns, is an important aspect of any intelligent system, and can provide a fruitful benchmark for general approaches.

Complex Problems

In contrast to the static problems above, complex problems change over time and are intransparent (Frensch and Funke 1995). The lack of transparency is due to the fact that the exact properties of the given state, the target state and the operations (or at least some) are unknown. Solving complex problems requires an efficient interaction between the user and the situation-related constraints of the task. It may require cognitive, emotional, and other skills and knowledge. Examples include a strategic game scenario "Lohhausen" (Dörner et al. 1983). In Lohhausen participants had to govern a small city by successfully managing the almost 2000 system variables. Findings from Lohhausen indicate that classical intelligence tests have only a small predictive power for failure or success, in contrast to self-confidence (Funke 2006). The internal representation followed a linearization effect, namely that participants rather thought in causal chains instead of causal networks, i.e., a tendency to linearize cause-effect and a reduction to often only one cause were observed. Another example are dynamic stock and flow problems (DSF) that represent stocks, accumulations of a certain amount of a quantifiable unit, and flows, that increase or decrease the stock amount over time. The task is to predict and react to the underlying changes in order to keep the stock in balance. Humans are not good in predicting the changes (Cronin et al. 2009). An overview about the involved cognitive complexity can be found in Schmid et al. (2011).

2.2 Features of Human Reasoning and Problem Solving

High-level human cognition is *context-dependent*, sometimes *heuristic*, *representation dependent*, and it does not obey pure *normative approaches*, i.e., human inferences deviate from predictions of classical logic and probability theory. Particularities arise from the structure and the limitation of the working memory, the mental representation, and the bounded-rationality. Some features of human reasoning that can be identified are:

- *Inferences can be nonmonotonic*: The suppression effect (cp. Example 5) above, among other findings indicate that inferences humans draw are nonmonotonic (Oaksford and Chater 2007; Johnson-Laird 2006; Stenning and Lambalgen 2008). Small number of defeaters are neglected by participants (Verschueren et al. 2004).

- *Internal representations of propositions and connectives imply partial orders*: Humans generate specific mental representations and neglect others (cp. Example 1, the indeterminacy effect). Partial orders of mental elements are responsible for illusions (cp. Example 7). Open are the number of truth values and how uncertainty is mentally represented, e.g., as possibilities or probabilities as discussed from an AI perspective in Chap. 3 of Volume V1.
- *The role of objects and form of representations can restrict the search space*: Gestalt principles, e.g., the principle of good continuation (Wertheimer 1923) or functional fixation can impact the cognitive processes (cp. Example 12). Humans tend to employ rather qualitative than quantitative representations (Knauff 2013).
- *The drawn inferences depend on different reasoning systems*: In a first step reasoners employ a fast and heuristic reasoning process and then in a second step they reason analytically (Kahneman 2003). This is sometimes termed dual system reasoning.
- *Knowledge influences inferences*: Inferences are not drawn if the given information already seems plausible (e.g., the belief-bias effect, Example 2), or different inferences are drawn between the abstract and deontic version of the Wason Selection Task (cp. Example 6).
- *Inferences deviate from classical normative frames of rationality*: Neither classical logic nor probability theory adequately reflect the inferences (cp. Example 2, 7, 8, 10) and few to none experiments precise the underlying concept of rationality. Concepts like bounded rationality provide better frameworks (Gigerenzer and Selten 2002).
- *Actions can be partially explained by expectations of information gain*: Experiments based on a repeated Wason Selection Task show that participants might select those cards that allow for a higher information gain (Oaksford and Chater 2007).
- *Context, plausibility, and common sense are evaluation criteria of human reasoners*: The Linda problem (cp. Example 7) among others indicates that instead of a pure deductive inference process humans do prefer to take plausibility related reasoning and the interpretation of context into account.

3 The Cognition of Reasoning and Problem Solving

A distinguishing feature of any cognitive theory from a purely logical or AI theory is that the aspect of *cognitive adequacy* is relevant (Strube 1992). The term cognitive adequacy of a theory can be subdivided into (i) *representational adequacy*, i.e., do humans use a similar mental representation as the theory predicts and (ii) *inferential adequacy*, that is, do humans draw the same conclusions as the theory predicts. While most cognitive theories of reasoning focus on the latter, the first is relevant too, as the internal representation influences the kind of inferences that can be drawn.

3.1 Mental Models

The theory of mental models (MMT for short) assumes that for given assertions we construct *iconic models* representing relevant parts of the events or objects based on background knowledge, semantics, and level of expertise (Johnson-Laird 2006). These models represent possibilities and have recently been modeled with modal operators (Hinterecker et al. 2016). Consider the assertion *if it rains then the street is wet*. In classical mathematical logic all possible valuations of *rain* and *wet street* make the conditional true, except for the assignment *rain* to true and *wet street* as false. The mental model theory assumes an order on the interpretations. One factor is the *principle of truth* claiming that human reasoners do not represent what is false, but only what is considered to be true. As Goodwin and Johnson-Laird (2005) explicates, the nature of mental models is *iconic*, meaning that mental models do not represent truth values but humans instead prefer to represent *abstract tokens* in these models. These tokens function as place holders for any kind of events, terms, or objects such as houses, fruits, or mathematical objects (Bara et al. 2001). So the conditional above is represented by the model where both events *rain* and *wet street* co-occur:

rain *wet street*
...

The ellipsis ("...") represents potential alternative models. These models can be generated in a cognitive "flesh out process" (Johnson-Laird 2006), if necessary. Hence, the mental model theory can also be reckoned among cognitive dual process theories. The ordering of information is a principle that can explain many deviations from logic such as illusions for instance (cp. Example 7). The mental models of assertion represent three possibilities as the initial models for the example above (with empty cells in each line for not fleshed out values of the objects in the column):

bread
soup
salad

These models imply that one cannot have the soup, the salad, or both of them. This is a possible explanation why only 17% of the participants gave the correct answer (Khemlani and Johnson-Laird 2009). Hence, the mental ordering of information is a predictor of the kind of answers participants give. The preferred mental model theory can explain the *indeterminacy effect* and the *preference effect* for spatial reasoning. An analysis of the inference mechanism demonstrates that the mental model theory is "somewhere in between" a credulous and sceptical inference process (Bonnefon 2004), demonstrating nonmonotonic aspects for nonmotonic reasoning problems (Johnson-Laird 2006).

3.2 Models of Cognition Inspired by Nonmonotonic Logics

The specifics of human inferences exclude classical logic as a potential model of human reasoning. This does, however, not necessarily hold for other logics as discussed in chapter "Knowledge Representation: Modalities, Conditionals and Nonmonotonic Reasoning" of Volume 1. Even though logics focus on formalizing correct inferences, cognitive scientists were always inspired by these approaches and aimed to adapt such accounts. In the last years some cognitive scientists have proposed System P (Neves et al. 2002; Benferhat et al. 2005; Pfeifer and Kleiter 2005; Kuhnmünch and Ragni 2014; Ragni et al. 2016), or the Weak Completion Semantics (WCS) based on the the three valued Łukasiewicz logic (Dietz et al. 2012; Hölldobler and Ramli 2009). In addition to the classical truth values of *true* and *false*, the latter two use also the value *unknown*. The WCS guarantees the existence of least models. The general idea is based on introducing an abnormality predicate *ab* in a conditional: Consider the example (3) from above: "If the system *p*asses the Turing test then the system is *i*ntelligent" can be represented as clauses in a logic program as a license for an inference $P = \{i \leftarrow p \wedge \neg ab\}$, with i called the *head* and the $p \wedge \neg ab$ called the *body*. Then, for a given program P, two kinds of transformations are considered: (1) If i is the head of more then one clause then replace all these clauses by $i \leftarrow body_1, \vee \cdots \vee body_m$), then (2) replace all occurrences of $\leftarrow$ by $\leftrightarrow$. The obtained set of formulas is called weak completion of the program. The WCS has been so far applied to the Wason Selection Task, the suppression effect, syllogistic reasoning, and on relational reasoning (Dietz et al. 2015). Other nonmonotonic logics could possibly explain human reasoning, too. Recently, an analysis of many de-facto standards of nonmonotonic logics in explaining human behavior in the suppression effect have been conducted (Ragni et al. 2016). Apart from Łukasiewicz logic none of the nonmonotonic logics like System P, logic programming, Reiter's Default logic, and ranking models could explain the results. Only by a manipulation of background knowledge this changes (Ragni et al. 2017).

3.3 Syntactic Approaches

Syntactic logic approaches, sometimes termed *rule-based* or mental logic theories have been applied to explain human reasoning. The underlying idea is that humans do apply syntactic procedures to derive inferences from given premises. These theories have been applied to reasoning about conditionals, syllogistic reasoning, and relational reasoning (Braine and O'Brien 1998; Rips 1994; Van der Henst 2002). Deviations of human reasoning from classical logical inferences for conditional reasoning can be explained by the derivation length, the working memory capacity, and the kind of inference processes necessary. For example the modus tollens ($A \rightarrow B$ and $\neg B$, then $\neg A$) is more difficult than the modus ponens as in the first case a mental derivation is necessary (comparable to a mathematical proof) and this makes the

modus tollens from the perspective of syntactic approaches more difficult than modus ponens. Syntactic rules are extended with additional principles such as Gricean implicature about the quantifiers in syllogistic reasoning. While nowadays the number of proponents of such theories is smaller, combinations of probabilistic and rule based theories have been recently proposed, e.g., Zhai (2015).

3.4 Probabilistic and Heuristic Approaches

Probabilistic theories assume that human reasoning is of probabilistic nature and can best be modeled by theories developed and based on Bayesian inference. Probabilistic theories and, to some extend, heuristic approaches claim that humans do represent uncertainty by assigning probabilities to knowledge. This allows not only to model absolute statements such as *As* are *Bs* (with probability 1), but as well statements including uncertainty such as *probably, As are Bs*. For instance, a conditional *if a then c* is represented by the conditional probability $P(c \mid a)$. For syllogisms, human reasoning is described by the probabilistic heuristic model (PHM; Oaksford and Chater 2007), i.e., the quantifiers can be ordered according to their informativeness from *All*, *Most*, *Few*, *Some*, *None*, to *Some ... not* identified by a computational analysis (Oaksford and Chater 2001). Three heuristics are employed: min-heuristic, p-entailment, and attachment heuristic to derive an answer based on the ordering. Other applications are in modeling inductive learning (Tenenbaum et al. 2006), causal inference (Steyvers et al 2003; Griffiths and Tenenbaum 2005, 2007), language acquisition and processing (Chater and Manning 2006; Xu and Tenenbaum 2007), and semantic memory (Steyvers et al. 2006). The theories assume that humans may conduct a statistical sampling to derive the required knowledge of basic probabilities for events (in particular for singular events). Probabilistic models can explain the aggregated behavior of groups of participants; it has not yet been applied for modeling individual decisions. Probabilistic models have been implemented as Bayesian nets and artificial neural network models (Neal 2012). On the other hand, results demonstrate that a possibilistic approach instead of a probabilistic approach can explain results in some cases more adequate (Raufaste et al. 2003).

Heuristic approaches are often domain-dependent, e.g., the *matching heuristic* for syllogistic reasoning: The most conservative quantifier is preferred in the conclusion and it is given by the following ordering of the quantifiers:

$$\text{No} > \text{Some} \ldots\text{not} = \text{Some} >> \text{All}$$

i.e., if one premise is "No" and the other is "Some" then the conclusion contains the quantifier "Some". There are two lines of research: One that focuses more on fallacies and limitations due to reasoning about heuristics (Kahneman 2011) and another one illustrating the potential of heuristics for those considered fast-and-frugal heuristics from a computational perspective (Gigerenzer and Selten 2002). Progress has been

made by a combined approach of AI and Psychology towards a more general and formally founded theory of decision making based on a rankings (Dubois et al. 2008).

Of course, combinations of different approaches leading to hybrid theories would also be conceivable. Here, as elsewhere in the sense of Occam's razor, a simpler theory should be preferred. These methods have hitherto been demonstrated in the sense of a *proof of concept* that the different theories are able to predict at least some human responses.

3.5 The Cognition of Analogical Reasoning

A prominent model of analogical reasoning (as discussed in chapter "Case-Based Reasoning, Analogical Reasoning, Interpolation" of Volume 1) is the Structure Mapping Engine (SME), which proposes three steps in human analogy making (Gentner 1983; Falkenhainer et al. 1986; Gentner et al. 2001): The first is to *access a target domain*, i.e., identify a source domain similar to the target domain from long-term memory. The second step is to *identify a mapping*, i.e., to identify the relation for individual elements in both domains and generalize them to a general mapping between the domains. The third is to *evaluate and apply* the generalized mapping. The mapping that best fits is the analogy sought.

Core elements of analogies are *objects* and *relations* and the *structural consistency* requires that the relational relationship between elements of the domain must be preserved. Representation is thus related to at most one element of the other domain in the sense of an *injective* mapping. A thorough analysis of the IQ-test Raven's Progressive Matrices yielded two procedural cognitive models (FAIRAVEN and BETTERAVEN; Carpenter et al. 1990) simulating the solution process of human adults with average and above average intelligence, respectively. The models implement six rules that are able to solve the Raven problems. Differences between both models depend on working memory limitations. A cognitive model (Lovett et al. 2009, 2010) based on the computational implementation of the *Process of Structure Mapping* combined with *CogSketch* a sketch understanding tool (Forbus et al. 2008). It uses automatically generated semantic and relational knowledge to successfully solve Raven's *Standard Progressive Matrices* and successfully simulate the answers of human adults (Lovett et al. 2009). Problems that could not be solved were considered difficult for humans (Lovett et al. 2010). A model excluding aspects of human problem solving (Strannegård et al. 2013) is able to solve 28 of the 36 SPM problems (Cirillo and Ström 2010). The program computed the solutions without considering the given possible solutions (Cirillo and Ström 2010). However, the program does not solve arbitrary geometrical problems. A different approach considers a logical view of analogical proportions on the pixel level and is so able to solve 32 out of 36 problems (Correa et al. 2012).

Taken together, there are programs that try to solve the Progressive Matrices in a non-cognitive approach (Evans 1968; Cirillo and Ström 2010) and cognitive models

that solve them similarly to humans (Lovett et al. 2010; Carpenter et al. 1990). None of these approaches have been applied to APM and SPM problems at the same time and none of these approaches uses working memory assumptions, makes response time predictions or is generalizable to arbitrary analogical problems (probably besides Lovett et al. 2010).

4 The Architecture of Cognition and Cognitive Models

It should be borne in mind that cognitive theories of reasoning often incorporate only few assumptions about the underlying human working memory and the specific processing of diverse and modality-specific information. We now turn to models of the data structure underlying human cognition. The contrast between the well-defined concepts of Turing machine or λ-calculus in theoretical computer science (Papadimitriou 1994) and the mystery of the human mind makes CS a fascinating discipline. Four core objectives for cognitive theories (Foundation 1978) are: the *abstraction*, i.e., "to formulate abstract descriptions of the mental capacities manifested by the structure, content, and function of various cognitive systems", the *instantiation*, i.e., "the systematic exploration of alternatives as and their realizations in different physical systems", the *plausibility*, i.e., "to characterize the mental processes underlying cognitive function in living organisms", and the *realization*, i.e., "the study of the neurological mechanisms involved in cognition". But it is not only the flow of information, it is the *architecture of information processing* that is vital; neurophysiological findings show that brain damage for example, can drastically alter cognitive abilities (Shallice 1988). This leads to the ultimate goal of CS—to develop a unified theory of cognition on all three of Marr's levels (see below). Many architectures assumed a modality specific information processing, i.e., that certain modules are responsible for the processing of different types of information, e.g., visual information is processed in a different memory location than auditory information and so on. This is supported by findings from neuroscience (Anderson 2007).

4.1 Evaluation Criteria for Cognitive Models

While AI systems aim at a general measure of efficiency or absolute performance as a normative factor, cognitive theories aim at a theory that is both explanatory and predictive for human behavior performance: this often includes accuracy (for a given normative framework), response time, and process steps. However, cognitive models are never just *simulation models* that can reproduce only existing experimental data. A good cognitive model is largely independent of experimental data and has general

strategies from which the experimental data (response times and given responses) can be generated. In summary, requirements of cognitive models can be determined by several criteria (Ragni 2008):

- *Transparency of the underlying model assumptions and the role of parameters.* If the assumptions of the model are opaque the predictions cannot be related back to the model processes.
- *Independence of the modeling principles of experimental data*. A model is never just a post-hoc explanation of experimental data, but relates responses to general cognitive processes.
- *Coverage of relevant phenomena*. The degree of coverage may be further specified in accuracy, response time, and intermediate step correspondence (Simon and Wallach 1999). If applicable, models can be additionally compared by information criteria (see below).
- *Capability to represent and explain inter-individual differences*. Inter-individual differences appear in reasoning and can be traced back, e.g., to different working memory sizes, knowledge, and concepts.
- *Generalizability and predictability of the cognitive model*. This includes a possible domain-independence and whether new and so far untested predictions about cognitive phenomena can be predicted.

While current approaches in AI are benchmarked against some current test problems (e.g., in planning or in theorem proving), the benchmarks of cognitive theories and models can differ across modeling approaches as we will see.

4.2 The Architecture of High-Level Human Cognition

Cognitive modeling can be understood as an algorithmization of psychological theories in a cognitive architecture. The ultimate goal is to move beyond the reproduction of empirical results (reaction times and error rates) and to obtain an equivalency on the process level on activations on the brain level. The modeling process is iterative and takes place in at least three steps: First, certain psychological phenomena or effects are identified. These are then explained and reproduced by a cognitive model in a second step. In a third step, new, not yet empirically tested model predictions are tested experimentally. Information processing in the human mind can be described on at least three levels: "the first level, known as the *computation level* abstractly represents the characteristics and objectives specified in the problem (Marr 1982). The second level, the *algorithmic level*, indicates how this calculation is implemented using algorithms. The third level, the *implementation level*, reflects the biological realization, i.e., the neuronal implementation. These three levels are also called semantic level, syntactic level, and physical level" (Marr 1982; McClamrock 1991). Cognitive theories have been developed for all levels and recently new approaches aim at hybrid

approaches (e.g., ACT-R Anderson 2007) incorporating symbolic and subsymbolic processes. A variety of approaches were inspired by the architecture of computers with a Central Process Unit (*CPU*) and a short term memory. Other possibilites are *hierarchical architectures*, the control of the entire processing being ensured by a special module (*supervisor*) or by *production control systems* (Anderson 2007; Sun 2001; Laird 2012), the direct control performed by an interpreter by a central data structure comprising various modules for cognition-specific tasks (*working memory*). Two different methods in modeling can be distinguished: For *top-down* processes knowledge, abilities, or reflection drives the behavior. In contrast, *bottom-up* processes immediately start at the level of perception and stimuli from the environment. Most actions are based on the interaction of both types of cognitive processes.

4.2.1 Cognitive Architectures

Most cognitive architectures are inspired by the General Problem Solver (GPS, Newell and Simon 1972), a model that uses means-end analysis as a search heuristic. It has been reimplemented as a production rule system. Production rule systems realize the physical symbol system hypothesis. These systems are composed of (i) *production rules*; they consist of a *condition* part and an *action* part, an (ii) *interpreter* that checks, if conditions of existing production rules are satisfied in a given model's state (they can *fire*). In the event that several rules can fire a conflict resolution process starts. Architectures often specify additional data structures. We focus on two architectures with most published (cognitive) models: an AI oriented approach SOAR and the hybrid cognitive architecture ACT-R.

SOAR (States, Operators, And Reasoning) is a production rule system with reinforcement learning built upon the GPS, hence aiming at a general problem solving agent (Newell 1990). It uses a problem space representation by differentiating between different forms of knowledge, e.g., procedural and semantic knowledge, and a distinctive working and long-term memory. Its emphasis lies on applying learning on all levels, hence implementing all AI and cognitive learning principles. The current version SOAR 9 integrates non-symbolic representations and other learning mechanisms (Laird 2008, 2012). It is a responsive system, i.e., each decision depends on the sensory input, the state of the working memory and encoded knowledge in the long-term memory. It performs a variety of problems from planning, robotic systems, interactions with virtual humans, and an air combat simulation for pilot training at the USAF (Tambe et al. 1995).

The cognitive architecture ACT-R 7.0 (Anderson 2007) aims at a unified human cognition approach. It is a hybrid theory, consisting of symbolic and subsymbolic parts. Its data structure is oriented on modality specific knowledge modules for perception (e.g., *visual, aural*), goal and sub-goal representations (*goal, imaginal*) and interfaces (so-called *buffers*) which can be accessed by production rules. ACT-R uses chunks as the atomic knowledge representation format with procedural knowledge

encoded in production rules and declarative knowledge that uses the concept of activation. Cognitive models have been developed for learning and memory, problem solving, deductive reasoning, perception, attention control, and human-computer interaction (HCI). Recently, ACT-R allows for the prediction of task-specific brain activations allowing modeling of findings from fMRI research.

A limitation and point of criticism is that many *cognitive architectures* are Turing complete (Anderson 1983) and, hence, do not provide cognitive bottlenecks or other architectured based constraints on computation processes of cognition.

4.2.2 Models Based on Artificial Neurons

A different modeling approach does not focus on symbols as the atomic components but on artificial neural models which are a simplification of brain neurons with a focus on electrical excitation while neglecting neurotransmitters or hormonal activity. Logical and arithmetic functions can be calculated by such artificial neural networks (ANNs) (McCulloch and Pitts 1943). The Hebbian learning rule (Hebb 1949) realizes the strengthening of the connection between two neurons when both neurons are active at the same time. First ANNs had two layers of nodes (*perceptron*), but the limitation to not represent the boolean operator XOR lead to the development of multilayer models using *backpropagation* (McClelland and Rogers 2003) or recurrent networks (Hölldobler and Ramli 2009). Most connectionists, proponents of ANNs, regard ANNs as "calculation models and not as models of biological reality" (Smolensky 1988). A recent approach, NEF (neural engineering framework), is based on biologically inspired spiking neural networks. It is build upon three principles that cover the nonlinear encoding and linear decoding for representations and transformations as dynamic systems (Eliasmith 2013). An artificial "brain" called SPAUN, is built based on NEF, and consists of about 2 million simulated neurons that cover different brain regions like the posterior parietal cortex, the prefrontal cortex, and occipital cortex, and the basal ganglia that have a similar role as the production rule mapping in ACT-R for distributing tasks to the specific brain areas. SPAUN is capable of solving tasks in high-level cognition from Raven's Progressive Matrices to serial working memory among others (Stewart et al. 2012).

4.2.3 Bayesian Modeling and Quantum Models

The starting point and the basic idea of Bayesian cognitive models is the question of how a cognitive agent revises its current assumptions in the light of observed data. In principle these models assume that an agent has degrees of beliefs, hypotheses that can be represented by probability distributions, and that the agent updates its belief distributions on new evidence according to Bayes' rule. However, some Bayesian modelers claim that "the human mind learns and concludes according to Bayesian

principles is not the assertion that the human mind implements Bayesian inferences" (Tenenbaum et al. 2011). In contrast to the symbolic and hybrid models presented above, Bayesian models do not aim at being cognitive process models. For example, cognitive restrictions of the *cognitive bottlenecks* are not relevant. Instead these models are realizations of a method called *rational analysis* (Anderson 2007). A "large number of known connectivist algorithms have a Bayesian interpretation" (Griffiths and Tenenbaum 2011), and these could serve as a first approach to a neural modeling of Bayesian approaches. Causal Bayesian networks allow the representation of structural aspects between random variables (Pearl 2000) and have been used for cognitive models. The causal structure, that is, the dependencies of the random variables, is represented by an acyclic directed graph. The directional graphs represent the relationship between cause and effects. Such causal Bayesian networks have not only a purely probabilistic relationship between the variables, but also a causal structure with implications on the statistical data, which can then be checked by empirical data (Hagmayer and Waldmann 2006). The strength is, in particular, to present a modeling approach on the computational level which is robust enough against noisy data. Application areas cover all domains of reasoning and such models are dominant in the area of decision making. Quantum probability models (Busemeyer and Bruza 2012) are a recent modeling approach in the field of decision making that extends the classical Bayesian approaches to model some paradoxes such as the sure thing principle (cp. Example 10) (Pothos and Busemeyer 2009).

4.2.4 Multinomial Processing Tree Models

Multinomial processing tree (MPT) are a class of models described by directed acyclic graphs, where each inner node represents a cognitive state, the leaves represent possible responses of participant(s), and the edges represent transition probabilities for each processing step. Hence, MPTs aim at explaining the generated output via the underlying latent cognitive processes. MPTs are a tool to compare theories (Oberauer 2006). Typical statistical measures for the goodness-of-fit are information criteria (e.g., Bayesian Information Criteria) that punish overly complex models. A limitation in contrast to process models is that the nodes are not algorithmically specified but that the sequences of the processes are predominant. As a result, models for reasoning do not necessarily have implications on the required working memory capacity and do not pose modelling constraints like cognitive bottlenecks. Applications of MPT-models are in models of recognition memory, decision making, and conditional and syllogistic reasoning, e.g., the belief-bias effect (cp. Example 2) (Klauer et al. 2000).

Cognitive modeling can be pure *symbolic*, *connectionistic*, or *hybrid*, and may include one or more Marrs' levels. It can integrate specific assumptions about the structure of the working memory, individual behavior, or likelihood-based prediction of the behavior of a group. The various approaches reflect different areas of human cognition, and offer the possibility of the empirical falsifiability of cognitive theories as opposed to pure descriptive theories.

5 Challenges in High-Level Cognition Research

Despite great progress in the field of AI, the construction of machines that fully implement high-level cognition has not yet been achieved. Human thinking is not simply the realization of the laws of classical logic, but clearly differs and is more multifaceted. Humans, in contrast to current cognitive and artificial systems, have an impressive ability to deal with underspecified data and imprecise knowledge and to solve problems by insight. The purpose of this line of research is that by understanding and modeling human thinking and reasoning, we can learn about feasible techniques that can be implemented in systems to deal with imprecise and complex problem descriptions.

1. **What are relevant benchmark problems?** The fields of action planning and automatic theorem proving in AI have greatly benefited from well defined benchmark problems and annual competitions. This made a fair comparison between different approaches and systems possible and triggered a competitive spirit to improve the state-of-the-art of the fields and to incorporate new concepts. We see the necessity to have competitions in the field of human reasoning as well, as the number of cognitive theories that argue to explain parts of human reasoning is continuously increasing, but few comparisons on common data sets exist. While psychological experiments can provide such benchmark problems, some findings are more important than others. So far there are no criteria identified that can be applied to identify *relevant* problems but this is a necessary condition to develop a generally accepted benchmark.
2. **How do humans represent and process information in high-level cognition?** A recent study in syllogistic reasoning demonstrated that any of the main cognitive theories deviates significantly from the empirical data. This demonstrates that even for reasoning about quantified assertions the underlying representation is still open. A current limiting factor is that there is no common language to formalize different representations and no precise well-defined benchmark. Additionally, many cognitive theories are underspecified. Hence, the question is how can these descriptive theories be turned into appropriately implemented models?
3. **How to model insight processes and meaning?** Central to many processes in high-level cognition is the ability to gain insight and assign meaning. At the same time these processes are hard to formalize or algorithmize. How is meaning generated and how can it be implemented?
4. **What are necessary features of "good" cognitive models?** While there are several definitions of cognitive models none is formally specified. Without a clear definition what is accepted as a cognitive model limits the search for better fitting models in the space of all cognitive models. Two steps can improve cognitive modeling:

    - The turn to *predictive cognitive models*: Current models are often post-dictive in contrast to predictive models, but only the capability to predict new phe-

nomena extends current limitations and aims at a general and unified cognitive modeling approach.
- The turn to *cognitive models for individual reasoners*: Many cognitive models aim at modeling an average human reasoner by an aggregation of the data of individual participants. However, aggregation inserts noise and blurs the cognitive processes of each individual reasoner. Moreover, any cognitive model that adequately models individual reasoner can model groups.

5. **How to develop cognitive models that are domain-independent?** Most AI and cognitive systems are specialized for respective domains with some recent exceptions in the field of modeling like SOAR or NEF. In contrast human reasoning is not limited to one domain. What are necessary features of such general and unified cognitive models?
6. **What are necessary properties of cognitive architectures?** There already exists a broad variety of cognitive architectures (Kotseruba et al. 2016), of which many even perform comparably. But the general foundations of such architectures are not specified, formalized, or compared.

These challenges are on a foundational level and demonstrate that many questions are open, even after decades of research. The field of cognitive modeling can strongly benefit from the rigorous formal approaches from AI.

6 Conclusion

AI aims at improving systems efficiently to find optimal solutions or at least good approximations. In contrast, CS concentrates more on *modeling* the mental processes underlying human behavior. The level of modeling comprises understanding the information theoretic processes on an algorithmic level and aligning it with neural activity. Yet the differences between AI and CS should not be mistaken for a disadvantage. From the separate viewpoints of AI and CS emerges a fertilization process. Humans can *adapt* themselves to new domains and solve problems by *insights*, two high-level cognition phenomena that could improve current AI systems. To improve AI systems we require cognitively adequate frameworks that are suitable for representing information and have good computational properties at the same time, i.e., that solutions can be computed in a reasonable time. High-level cognition is not a black-box, the performance of human reasoning and problem solving can be analyzed, reproduced, and predicted. This requires, however, a multi-disciplinary approach covering psychological experiments, formal and cognitive modeling, and logics. Understanding cognition is often a *reverse engineering problem*, i.e., it is necessary to reconstruct the underlying functions from behavioral findings such as error rates and reaction times. At the same time cognitive models provide an important and interesting bridge between formal methods and empirical psychological results. They offer the possibility to formalize psychological theories, even to produce human-like error rates and reaction times and finally to compare these results with predictions of

psychological theories. Recently, this has lead to a stronger interest in formal methods in the psychology of reasoning (Bonnefon 2013). Recent approaches do focus more on human reasoning processes about preferences and behavior of other agents (Bonnefon et al. 2012).

By analyzing the specific features of cognitive architectures it is possible to integrate all models into a general system based on Newell's idea of a "unified theory of cognition" (Newell 1994). Such a unified theory of cognition should offer a small or even single set of mechanisms that can account for human performance on cognitive tasks from perception to problem solving. Most research is performed on a normative scale without reflecting the underlying premises. This leads to the impression that human reasoning is "weaker" or "erroneous" in contrast to formal methods from AI or logic. But, classical logic cannot always be applied, it requires specific properties and has its limitations if applied in the wrong context, e.g., in a nonmonotonic world. In this sense human reasoning that is nonmonotonic, inductive, plausible, context-dependent, integrating different reasoning systems has adapted itself to reasoning efficiently and satisficing with respect to bounded rationality and it is adaptable to different domains. These properties still make human reasoning interesting for developing better AI systems.

Acknowledgements The author is grateful to Bernhard Nebel (University of Freiburg), Emmanuelle-Anna Dietz Saldanha (TU Dresden), and Nicolas Riesterer (University of Freiburg) for their substantial feedback.

References

Anderson JR (1983) The architecture of cognition. Hillsdale, NJ, US

Anderson JR (2007) How can the human mind occur in the physical universe? Oxford University Press, New York

Bara BG, Bucciarelli M, Lombardo V (2001) Model theory of deduction: a unified computational approach. Cogn Sci 25:839–901

Baral C (2003) Knowledge representation, reasoning and declarative problem solving. University Press, Cambridge

Bauer MI, Johnson-Laird PN (1993) How diagrams can improve reasoning. Psychol Sci 4:372–378

Beller S, Spada H (2001) Denken. In: Strube G (ed) Wörterbuch der Kognitionswissenschaft. Klett-Cotta, Stuttgart

Benferhat S, Bonnefon J, Neves RDS (2005) An overview of possibilistic handling of default reasoning, with experimental studies. Synthese 146(1–2):53–70

Bennati S, Brüssow S, Ragni M, Konieczny L (2014) Gestalt effects in planning: rush-hour as an example. In: Bello P, Guarini M, McShane M, Scassellati B (eds) Proceedings of the 36th annual conference of the cognitive science society. Cognitive Science Society, Austin, TX, pp 1234–1240

Bonnefon J (2004) Reinstatement, floating conclusions, and the credulity of mental model reasoning. Cogn Sci 28(4):621–631

Bonnefon J (2013) Formal models of reasoning in cognitive psychology. Argum Comput 4(1):1–3. https://doi.org/10.1080/19462166.2013.767559

Bonnefon J, Girotto V, Legrenzi P (2012) The psychology of reasoning about preferences and unconsequential decisions. Synthese 185(Supplement-1):27–41

Braine MDS, O'Brien DP (1998) Mental logic. Erlbaum, Mahwah
Busemeyer JR, Bruza PD (2012) Quantum models of cognition and decision. Cambridge University Press, Cambridge
Byrne RMJ (1989) Suppressing valid inferences with conditionals. Cognition 31:61–83
Byrne RMJ (2002) Mental models and counterfactual thoughts about what might have been. Trends Cogn Sci 6(10):426–431
Byrne RMJ (2007) The rational imagination: how people create alternatives to reality. MIT Press, Cambridge
Carpenter PA, Just MA, Shell P (1990) What one intelligence test measures: a theoretical account of the processing in the raven progressive matrices test. Psychol Rev 97(3):404–431
Chater N, Manning CD (2006) Probabilistic models of language processing and acquisition. Trends Cogn Sci 10:335–344
Chu Y, MacGregor JN (2011) Human Performance on insight problem solving: a review. J Probl Solving 3(2):119–150
Cirillo S, Ström V (2010) An anthropomorphic solver for raven's progressive matrices. Master's thesis, Chalmers University of Technology, Department of Applied Information Technology, SE-41296 Goeteborg, Sweden
Correa W, Prade H, Richard G (2012) When intelligence is just a matter of copying. In: Proceedings of the 20th European conference on artificial intelligence. IOS Press, Amsterdam, The Netherlands, ECAI'12, pp 276–281
Cronin M, Gonzalez C, Sterman JD (2009) Why don't well-educated adults understand accumulation? a challenge to researchers, educators and citizens. Organ Behav Hum Decis Process 23(1):108
Dietz EA, Hölldobler S, Ragni M (2012) A computational approach to the suppression task. In: Miyake N, Peebles D, Cooper RP (eds) Proceedings of the 34th annual conference of the cognitive science society. Cognitive science society, Austin, TX, pp 1500–1505
Dietz EA, Hölldobler S, Höps R (2015) A computational logic approach to human spatial reasoning. In: 2015 IEEE symposium series on computational intelligence. IEEE, pp 1627–1634
Dixon JE, Byrne RMJ (2011) "if only" counterfactual thoughts about exceptional actions. Mem Cogn 39(7):1317–1331
Dörner D, Kreuzig HW, Reither F, Stäudel T (1983) Lohhausen. Vom Umgang mit Unbestimmtheit und Komplexität, Huber, Bern
Douven I (2011) Abduction. In: Zalta EN (ed) The stanford encyclopedia of philosophy
Dubois D, Fargier H, Bonnefon J (2008) On the qualitative comparison of decisions having positive and negative features. J Artif Intell Res 32:385–417
Duncker K (1945) On problem-solving. Psychological Monographs ix(58):113
Eliasmith C (2013) How to build a brain: a neural architecture for biological cognition. Oxford University Press, Oxford
Evans TG (1968) A program for the solution of a class of geometric-analogy intelligence-test questions. In: Minsky ML (ed) Semantic information processing. MIT Press, Cambridge, MA, chap 5, pp 271–351
Falkenhainer B, Forbus KD, Gentner D (1986) The structure-mapping engine. Report Department of Computer Science, University of Illinois at Urbana-Champaign
Fauconnier G, Turner M (2008) The way we think: conceptual blending and the mind's hidden complexities. Basic Books, New York
Flake GW, Baum EB (2002) Rush Hour is PSPACE-complete, or why you should generously tip parking lot attendant. Theor Comput Sci 270:895–911
Forbus K, Usher J, Lovett A, Lockwood K, Wetzel J (2008) CogSketch: open-domain sketch understanding for cognitive science research and for education. In: Alvarado C, Cani MP (eds) Proceedings of the fifth eurographics workshop on sketch-based interfaces and modeling
Foundation S (1978) Cognitive science 1978: Report of the state of the art committee. http://csjarchive.cogsci.rpi.edu/misc/CognitiveScience1978_OCR.pdf

Frensch PA, Funke J (eds) (1995) Complex problem solving: the European perspective. Lawrence Erlbaum, Hillsdale, NJ
Funke J (2006) Lösen komplexer Probleme. In: Funke J, Frensch P (eds) Handbuch der Allgemeinen Psychologie - Kognition. Handbuch der Psychologie, Hogrefe, Göttingen, pp 439–445
Gentner D (1983) Structure-mapping: a theoretical framework for analogy. Cogn Psychol 7:155–170
Gentner D, Holyoak KJ, Kokinov BN (2001) The analogical mind: perspectives from cognitive science. Bradford Books, MIT Press, Cambridge, MA
Gigerenzer G, Selten R (2002) Bounded rationality: the adaptive toolbox. MIT Press, Cambridge
Gilhooly KJ, Murphy P (2005) Differentiating insight from non-insight problems. Think Reason 11(3):279–302
Goodwin GP, Johnson-Laird PN (2005) Reasoning about relations. Psychol Rev 112:468–493
Greeno JG, Simon HA (1988) Problem solving and reasoning. Technical report, University of Pittsburgh
Griffiths TL, Tenenbaum JB (2005) Structure and strength in causal induction. Cogn Psychol 51:354–384
Griffiths TL, Tenenbaum JB (2007) From mere coincidences to meaningful discoveries. Cognition 103(2):180–226
Griffiths TL, Tenenbaum JB (2011) Predicting the future as Bayesian inference: people combine prior knowledge with observations when estimating duration and extent. J Exp Psychol: Gen 140(4):725
Hagmayer Y, Waldmann MR (2006) Kausales Denken. In: Funke J (ed) Enzyklopädie der Psychologie 'Denken und Problemlösen', no. 8 in C, Hogrefe, Göttingen, chap II, pp 87–166
Halbmayer E , Salat J (2012) Qualitative Methoden der Kultur- und Sozialanthropologie. http://www.univie.ac.at/ksa/elearning/cp/qualitative/qualitative-5.html
Halford GS, Wilson WH, Phillips S (2010) Relational knowledge: the foundation of higher cognition. Trends Cogn Sci 14:497–505
Hebb DO (1949) The organization of behavior. Wiley, New York
Hinterecker T, Knauff M, Johnson-Laird PN (2016) Modality, probability, and mental models. J Exp Psychol: Learn Mem Cogn 42(10):1606–1620
Hölldobler S, Ramli CDPK (2009) Logic programs under three-valued Łukasiewicz semantics. In: Logic programming. Springer, Berlin, pp 464–478
Hollnagel E, Woods DD (2005) Joint cognitive systems: foundations of cognitive systems engineering. CRC Press, Boca Raton
Johnson-Laird PN (2006) How we reason. Oxford University Press, New York
Just MA, Carpenter PA, Varma S (1999) Computational modeling of high-level cognition and brain function. Hum Brain Mapp 8:128–136
Kahneman D (2011) Thinking, fast and slow. Macmillan, New York
Kahneman D (2003) A perspective on judgement and choice. Am Psychol 58:697–720
Kaller CP, Unterrainer JM, Rahm B, Halsband U (2004) The impact of problem structure on planning: insights from the Tower of London task. Cogn Brain Res 20:462–72
Khemlani S, Johnson-Laird PN (2009) Disjunctive illusory inferences and how to eliminate them. Mem Cogn 37:615–623
Khemlani S, Johnson-Laird PN (2012) Theories of the syllogism: a meta-analysis. Psychol Bull
Khemlani S, Johnson-Laird PN (2013) The processes of inference. Argum Comput 4(1):4–20
Khemlani S, Johnson-Laird PN (2016) How people differ in syllogistic reasoning. In: Proceedings of the 36th annual conference of the cognitive science society. Cognitive Science Society, Austin, TX
Klauer KC, Musch J, Naumer B (2000) On belief bias in syllogistic reasoning. Psychol Rev 107(4):852–884
Knauff M (2013) Space to reason: a spatial theory of human thought. MIT Press, Cambridge
Knauff M, Johnson-Laird PN (2002) Visual imagery can impede reasoning. Mem Cogn 30:363–71

Kotseruba I, Gonzalez OJA, Tsotsos JK (2016) A review of 40 years of cognitive architecture research: focus on perception, attention, learning and applications. arXiv:161008602
Kuhnmünch G, Ragni M (2014) Can formal non-monotonic systems properly describe human reasoning? In: Bello P, Guarini M, McShane M, Scassellati B (eds) Proceedings of the 36th annual conference of the cognitive science society, Austin, TX, pp 1222–1228
Laird JE (2008) Extending the SOAR cognitive architecture. In: AGI, pp 224–235
Laird JE (2012) The SOAR cognitive architecture. The MIT Press, Cambridge
Legrenzi P, Girotto V, Johnson-Laird PN (1993) Focussing in reasoning and decision making. Cognition 49(1):37–66
lle Lépine R, Parrouillet P, Camos V, (2005) What makes working memory spans so predictive of high-level cognition? Psychon Bull Rev 12(1):165–170
Lovett A, Tomai E, Forbus K, Usher J (2009) Solving geometric analogy problems through two-stage analogical mapping. Cogn Sci 33(7):1192–1231
Lovett A, Forbus K, Usher J (2010) A structure-mapping model of raven's progressive matrices. In: Ohlsson S, Catrambone R (eds) Proceedings of the 32nd annual conference of the cognitive science society. Cognitive Science Society, pp 2761–2766
Marr D (1982) Vision. A computational investigation into the human representation and processing of visual information. Freeman, San Francisco
McClamrock R (1991) Marr's three levels: a re-evaluation. Minds Mach 1(2):185–196
McClelland JL, Rogers TT (2003) The parallel distributed processing approach to semantic cognition. Nat Rev Neurosci 4(4):310
McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133
Neal RM (2012) Bayesian learning for neural networks, vol 118. Springer, Berlin, Heidelberg
Neves RDS, Bonnefon J, Raufaste E (2002) An empirical test of patterns for nonmonotonic inference. Ann Math Artif Intell 34(1–3):107–130
Newell A (1979) Reasoning, problem solving and decision processes: the problem space as a fundamental category. Technical report, Computer science department CMU
Newell A (1990) Unified theories of cognition. Harvard University Press, Cambridge
Newell A (1994) Unified theories of cognition. The William James lectures. Harvard University Press, Cambridge
Newell A, Simon HA (1972) Human problem solving. Prentice-Hall, Englewood Cliffs
Newell A, Simon HA (1976) Computer science as empirical enquiry: symbols and search. Commun ACM 19:113–126
Newstead SE (1995) Gricean implicatures and syllogistic reasoning. J Mem Lang 34:644–664
Oaksford M, Chater N (2001) The probabilistic approach to human reasoning. Trends Cogn Sci 5(8):349–357
Oaksford M, Chater N (2007) Bayesian rationality: the probabilistic approach to human reasoning. Oxford University Press, Oxford
Oberauer K (2006) Reasoning with conditionals: a test of formal models of four theories. Cogn Psychol 53:238–283
O'Reilly RC (2006) Biologically based computational models of high-level cognition. Science 314(5796):91–94
Papadimitriou CM (1994) Computational complexity. Addison-Wesley, Reading
Pearl J (2000) Causality: models, reasoning, and inference. Cambridge University Press, Cambridge
Pfeifer N, Kleiter GD (2005) Coherence and nonmonotonicity in human reasoning. Synthese 146:93–109
Pothos EM, Busemeyer JR (2009) A quantum probability explanation for violations of 'rational' decision theory. Proc R Soc Lond B: Biol Sci
Potts GR (1974) Storing and retrieving information about ordered relationships. J Exp Psychol 103(3):431

Ragni M (2008) Räumliche Repräsentation, Komplexität und Deduktion: Eine kognitive Komplexitätstheorie[Spatial representation, complexity and deduction: A cognitive theory of complexity]. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg
Ragni M, Knauff M (2013) A theory and a computational model of spatial reasoning with preferred mental models. Psychol Rev 120(3):561–588
Ragni M, Kola I, Johnson-Laird P (2018) On selecting evidence to test hypotheses: a theory of selection tasks. Psychol Bull 144(8):779–796
Ragni M, Eichhorn C, Kern-Isberner G (2016) Simulating human inferences in the light of new information: a formal analysis. In: Kambhampati S (ed) Proceedings of the twenty-fifth international joint conference on artificial intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, IJCAI/AAAI Press, pp 2604–2610. http://www.ijcai.org/Proceedings/2016
Ragni M, Eichhorn C, Bock T, Kern-Isberner G, Tse APP (2017) Formal nonmonotonic theories and properties of human defeasible reasoning. Minds Mach 27(1):79–117
Raufaste E, Neves RDS, Mariné C (2003) Testing the descriptive validity of possibility theory in human judgments of uncertainty. Artif Intell 148(1–2):197–218
Raven JC (1962) Advanced progressive matrices, Set II. H. K. Lewis, London
Raven J, Raven JC, Court JH (2000) Manual for ravens progressive matrices and vocabulary scales. Harcourt Assessment, San Antonio, TX
Rips LJ (1994) The psychology of proof: deductive reasoning in human thinking. The MIT Press, Cambridge
Russell S, Norvig P (2003) Artificial intelligence: a modern approach, 2nd edn. Prentice Hall, Englewood Cliffs
Schmid U, Ragni M, Gonzalez C, Funke J (2011) The challenge of complexity for cognitive systems. Cogn Syst Res 12(3–4):211–218
Searle JR (1980) Minds, brains, and programs. Behav Brain Sci 3(3):417–424
Searle JR (2004) Mind: a brief introduction. Oxford University Press, Oxford
Shallice T (1988) From neuropsychology to mental structure. Cambridge University Press, Cambridge
Simon HA, Newell A (1962) Computer simulation of human thinking and problem solving, vol 27 [Society for Research in Child Development, Wiley], pp 137–150
Simon HA, Wallach D (1999) Editorial: cognitive modeling in perspective. Kognitionswissenschaft 8:1–4
Smolensky P (1988) On the proper treatment of connectionism. Behav Brain Sci 11:1–74
Stenning K, Lambalgen M (2008) Human reasoning and cognitive science. Bradford Books, MIT Press, Cambridge
Sternberg RJ (1980) Reasoning, problem solving, and intelligence. Technical report, DTIC Document
Sternberg RJ (1999) The nature of cognition. A Bradford Book, MIT Press, Cambridge
Stewart T, Choo FX, Eliasmith C (2012) Spaun: a perception-cognition-action model using spiking neurons. In: Proceedings of the cognitive science society, vol 34
Steyvers M, Tenenbaum JB, Wagenmakers EJ, Blum B (2003) Inferring causal networks from observations and interventions. Cogn Sci 27(3):453–489. http://dblp.uni-trier.de/db/journals/cogsci/cogsci27.html#SteyversTWB03
Steyvers M, Griffiths TL, Dennis S (2006) Probabilistic inference in human semantic memory. Trends Cogn Sci 10:327–334
Strannegård C, Cirillo S, Ström V (2013) An anthropomorphic method for progressive matrix problems. Cogn Syst Res 22–23:35–46
Strube G (1992) The role of cognitive science in knowledge engineering. In: Contemporary knowledge engineering and cognition, pp 159–174
Strube G, Ferstl E, Konieczny L, Ragni M (2013) Kognition. In: Görz G, Schneeberger J, Schmid U (eds) Handbuch der Künstlichen Intelligenz. Oldenbourg, München
Sun R (2001) Duality of the mind - a bottom-up approach toward cognition. Lawrence Erlbaum

Szymanik J (2016) Quantifiers and cognition: logical and computational perspectives. Springer, Berlin
Tambe M, Johnson WL, Jones RM, Koss FV, Laird JE, Rosenbloom PS, Schwamb K (1995) Intelligent agents for interactive simulation environments. AI Mag 16(1):15–39
Tenenbaum JB, Griffiths TL, Kemp C (2006) Theory-based Bayesian models of inductive learning and reasoning. Trends Cogn Sci 10:309–318
Tenenbaum JB, Kemp C, Griffiths TL, Goodman ND (2011) How to grow a mind: statistics, structure, and abstraction. Science 331(6022):1279–1285
Tversky A, Kahneman D (1973) Availability: a heuristic for judging frequency and probability. Cogn Psychol 5(2):207–232
Tversky A, Kahneman D (1983) Extensional versus intuitive reasoning: the conjunction fallacy in probability judgment. Psychol Rev 90(4):293
Van der Henst JB (2002) Mental model theory versus the inference rule approach in relational reasoning. Think Reason 8(3):193–203
Verschueren N, Schaeken W, De Neys W, d'Ydewalle G (2004) The difference between generating counterexamples and using them during reasoning. Q J Exp Psychol Sect A 57(7):1285–1308
Wason PC (1971) Problem solving and reasoning. Br Med Bull 27(3):206–210
Wertheimer M (1923) A brief introduction to Gestalt, identifying key theories and principles. Psychol Forsch 4:301–350
Xu F, Tenenbaum JB (2007) Word learning as bayesian inference. Psychol Rev 114(2):245–72
Zhai F (2015) Toward probabilistic natural logic for syllogistic reasoning. Ph.D. thesis, Universiteit van Amsterdam

Artificial Intelligence and Literature

Tim Van de Cruys

Abstract Throughout the history of artificial intelligence, the performance of natural language processing algorithms has continuously improved, such that many aspects of human language behaviour are now effectively modeled by current systems. The improvements within the field of natural language processing are self-evidently beneficial for the generation of literary artefacts, but they are not sufficient for truly creative language generation: in order to exhibit creativity, language systems need to be equipped with additional components, that not just mimic human language use, but are able to produce and self-assess creative language expressions. In this chapter, we will trace the various paradigms that researchers have taken in order to model this creative process. We start with the simplest form, mechanical creativity, continue with rule- and template-based systems, and end with statistical machine learning approaches. We will exemplify the various paradigms by focusing on different forms of literary artefacts: computational humour, metaphor, poetry generation, and story generation.

1 Introduction

1.1 Intelligence Versus Creativity

In the course of the last decades, significant progress has been made within the field of artificial intelligence in general, and natural language processing in particular. While most algorithms prior to the nineties focused on rule-based systems, the field of natural language processing has undergone a major paradigm shift, exploiting statistics—rather than rules—as its main instrument (see chapter "Artificial Intelligence and Language" of this volume). Combined with the exponential increase in

T. Van de Cruys (✉)
IRIT-CNRS, Université Paul Sabatier, Toulouse, France
e-mail: tim.vandecruys@irit.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_15

computing power and the development of increasingly advanced machine learning techniques (see chapter "Designing Algorithms for Machine Learning and Data Mining" of Volume 2), we have now reached a point where machines are able to tackle complex language modeling tasks, allowing natural language processing algorithms to be employed successfully within real-world applications.

The improvements within the field of natural language processing also have their impact on the generation of creative language expressions: a sophisticated and powerful language model is a *conditio sine qua non* for full-fledged creative language generation that goes beyond simple template filling. However, it is not the only prerequisite: while computational creativity can be considered part of the field of artificial intelligence, its goals are somewhat different. The main goal of artificial intelligence is the development of systems that exhibit intelligent behaviour, whereas the goal of computational creativity is the development of systems exhibiting behaviour that would be deemed creative by human observers (Colton and Wiggins 2012). By design, the majority of language processing systems exclusively focus on the intelligent part: they produce the best possible output according to a particular problem or a specific task. Typically, a machine learning model is trained on a significant amount of training data, optimizing its parameters in order to solve a particular task at hand. As such, the bulk of natural language processing algorithms merely mimic human language use, and seldom produce creative output: the very design of the algorithms leaves little room for creativity. In order to exhibit creativity, creative language systems need to be equipped with additional components, that are able to produce and self-assess creative language expressions.

In this chapter, we will trace the various paradigms that researchers have taken in order to model this creative process. We start with the simplest form, mechanical creativity, continue with rule- and template-based systems, and end with statistical machine learning approaches. But first, we will describe the concept of creativity in somewhat closer detail.[1]

1.2 Different Kinds of Creativity

When discussing computational models of creativity, it is useful to define the concept of creativity, and distinguish the different forms of creativity that come into play. Boden (2004) distinguishes three different kinds of creativity: creative ideas may be combinational, exploratory, or transformational. Combinational creativity takes place

[1]Note that this chapter mainly focuses on computational approaches to creativity in which the system takes control of the creative process. Another form of computational creativity is brought about by the nature of the computational medium, in which the static, linear process is dispensed with, leaving room for an interactive reading of texts. This is the kind of creativity that emerges within hypertexts, or story development within computer games (see chapter "Games in Artificial Intelligence" of Volume 2). This second kind of computational creativity is beyond the scope of this chapter.

when unexpected ideas are combined, provoking surprise. Consider the following examples:

(1) Paris is a lively city.

(2) Paris sparkles like a freshly poured glass of champagne.

Most language users would agree that sentence (1) is a rather conventional way of conveying the intended meaning, whereas the same language users would likely consider sentence (2) a more creative way of expressing the very same semantic content: sentence (2) makes use of a *simile*, comparing the bustling and brilliant city life of Paris to the scintillating and swirling bubbles in a glass of sparkling wine. It combines concepts from two different domains in order to express an intended meaning in an original and creative way.

Exploratory and transformational creativity are different. Exploratory creativity makes use of existing stylistic rules or conventions in order to generate new structures. In terms of language, an example would be the composition of a haiku, or the writing of a novel in a particular style. Transformational creativity is the most striking of the three; it pushes boundaries by going beyond existing styles or conventions, and thus generates ideas that were 'impossible' before. In terms of language, it is the creativity that emerges from Shakespeare's *Macbeth*, Joyce's *Ulysses*, Ezra Pound's *Cantos*, or indeed any work that transcends previous styles or conventions; it is also this kind of creativity that drives language change.

In the following sections, we will explore a number of computational approaches that aim to model these different forms of creativity. We will see that most models exhibit either combinational or exploratory creativity; transformational creativity—for the computational models we have today—still seems out of bound.

2 Mechanical Creativity

The simplest, most straightforward form of computational creativity may be characterized as mechanical creativity. Mechanical creativity may be defined as combinational creativity in its most primitive form. It consists of the mechanical combination of ideas (e.g. words or phrases) in a deterministic way, possibly giving rise to creative artefacts. Over the past century, the idea of mechanical creativity has been explored by various artists and artistic movements. One famous example is Raymond Queneau's *Cent mille milliards de poèmes* ['a hundred thousand billion poems'; Queneau, 1961]. The work consists of 10 sonnets, which all have the same structure and the same rhyme scheme. Moreover, each individual line of each sonnet is printed on a separate strip of paper, allowing each line of a sonnet to be combined with any line of the other sonnets (which allows for 10^{14} different combinations in total). The work is one of the most famous examples that emerged from OULIPO (OULIPO 1981), a

literary movement whose members are mainly French-speaking writers and mathematicians. OULIPO, short for *Ouvroir de littérature potentielle* ('potential literature workshop'), was founded in 1960 by Raymond Queneau and François Le Lionnais, and its aim is to create literary work based on creative constraints or mechanical procedures. Another example is the $N + 7$ method. The idea is to replace every noun in a text by the seventh noun that follows the original one in a dictionary. As such, an existing phrase like *when I find myself in times of trouble* might be mechanically transformed into a novel, creative, but often somewhat non-sensical phrase like *when I find myself in timpanists of trout*.

The combinational and deterministic mechanisms conceived by movements like OULIPO may be straightforwardly implemented using computational means, as exemplified by the French ALAMO group. The ALAMO group (*l'Atelier de Littérature Assistée par la Mathématique et les Ordinateurs*; 'literature workshop assisted by mathematics and computers') grew out of the OULIPO movement, and explored the movement's ideas within the realm of computing. The group implemented a number of ideas previously conceived by OULIPO, but also came up with new ideas and implementations of combinational algorithms within the spirit of OULIPO's original principles. They went on to create a number of *littéraciels* (a portmanteau word of *littérature* 'literature' and *logiciel* 'software'), such as LAPAL (*Langage Algorithmique pour la Production Assistée de Littérature*; 'algorithmic language for the assisted production of literature'). At the core of these programs lay a complex set of (mostly syntactic, but also semantic) rules and constraints, which allowed for the automatic generation of simple language fragments. The program was meant to be used by an intervening user: LAPAL generated potential language fragments, which might be validated by a user or not, staying true to ALAMO's founding manifesto.[2] A famous example that originated in the ALAMO group is the concept of *rimbaudelaires*, automatic poems based on the structure of poems by Rimbaud, but filled with the vocabulary of poems by Baudelaire. Other ALAMO members focused on story generation (Borillo and Virbel 1983).

Whereas the early work of ALAMO members relied on simple mechanical creativity, later work uses increasingly more evolved rule-based and template-based methods (Balpe 2000). Indeed, the idea of computational generation according to predefined constraints naturally leads to rule-based and template-based generation systems. In the next section, we describe a number of systems that further explore rule-based and template-based approaches for the generation of creative language.

[2] 'ALAMO considers the computer a tool that facilitates combinatorial work. The goal is not to make the computer generate specific artefacts; rather, the texts are written by authors, and the machine's function is to make available, to rearrange, and to reactivate.' Excerpt from ALAMO founding manifesto by Paul Braffort and Jacques Roubaud, cited in Bootz (2012).

3 Rule-Based and Template-Based Methods

In this section, we discuss a number of models that rely on predefined symbolic rules or templates in order to generate creative language expressions. We subsequently review models for computational humour, metaphor, poetry generation, and story generation.

3.1 Computational Humour

Humour is an inherently creative phenomenon. Binsted and Ritchie (1994, 1997) present a system called JAPE (Joke Analysis and Production Engine), that is able to generate punning riddles based on predefined templates. Consider the following examples:

(3) What do you get when you cross a sheep and a kangaroo? *A woolly jumper*

(4) What do you call a weird market? *A bizarre bazaar*

The first pun above plays on the phonologically similar but different meanings of the word *jumper* ('piece of clothing' and 'agent that jumps'); it is an example of a well-known English pun. The second example plays on the phonological similarity between the first and second part of the expression, and it was automatically generated by JAPE.

The important thing to note is that these kind of punning riddles exhibit regular structures and mechanisms, that may be exploited by a computational system in order to construct new ones. Generally speaking, JAPE's procedure can be described as follows:

- create a new expression, e.g. using substitution based on phonological constraints or ambiguity ('tub crawl');
- find a plausible description for the expression ('bath tour');
- return the pun as a question-answer pair ('What do you call a bath tour? A tub crawl').

JAPE realizes the above subtasks by relying on a considerable amount of predefined templates, rules, and lexicons (such as an English pronunciation dictionary, and WordNet). It must be noted that the lexicons used are not specifically designed to generate jokes. The lexicon is general and neutral, and it is the algorithm itself that exploits the lexical information for the generation of a suitable pun. The best examples generated by the JAPE system do have some merit, though its reliance on a limited number of predefined schema do hamper its variety.

Another example of a knowledge-based system is HAHAcronym (Stock and Strapparava 2005), a system that aims to generate humorous acronym interpretations, such as 'Fantastic Bureau of Intimidation' for 'FBI'. Again, the system relies on lexical resources (WordNet) in order to generate ironic counterparts of the original acronym.

3.2 Metaphor

Besides humour, metaphor is one of the most intriguing means for creative language use, and as such it has been substantially researched within a computational context. Most computational research on metaphor focuses on identification and interpretation. A number of approaches rely on hand-crafted lexical resources (Krishnakumaran and Zhu 2007; Wilks et al. 2013), but the most interesting approaches make use of statistical information extracted from corpora (cfr. infra). Work on metaphor generation, on the other hand, most often resorts to pattern-based methods. A key characteristic of metaphor is the mapping of an expression from a source domain to a target domain. Consider the examples in (5) and (6).

(5) Don't kill the vibe.

(6) You need to clean up your life.

In example (5), a mapping is made between the source domain LIFE towards the target domain MOOD. In example (6), there is a mapping from the domain ENVIRONMENT (literal action of cleaning) to the domain PERSONAL LIFE (metaphorical action of cleaning, i.e. sorting out one's personal affairs). Veale and Hao (2007) show that these mappings may be straightforwardly extracted from large web corpora using pattern-based extraction methods. The main idea of their system—called 'Sardonicus'—is to construct a large collection of stereotypical similes by automatically mining specific linguistic patterns such as *X is as A as a(n) N* from the web. Such stereotypical similes tell us that *A* is a characteristic that is shared by *X* and *N*, and moreover that *A* is a salient property of *N* that will often be important when *N* is used metaphorically. The resulting similes may then be used in order to generate new metaphors. As an example, for a target concept like *city* and a property like *lively*, the system would be capable of generating metaphors such as the one in (7):

(7) The city is a beehive.

In order to generate a metaphor like (7), the system consults its database to determine which are the salient nouns that may be described as *lively*, and generates suitable internet queries in order to determine which expressions are actually attested. By ordering on web frequency, the resulting metaphors are validated.

Based on the techniques described above, Veale (2012) presents a system that is able to generate metaphors with an affective orientation. Once again, stereotypical similes are extracted using specific linguistic patterns, in order to construct a stereotype lexicon of considerable size. In a second step, the stereotypes are annotated with an affective orientation (positive or negative), based on the assumption that characteristics that have the same affective orientation appear in the same conjunctions, e.g. *as cute and happy as a baby*. The resulting similes may then be used for metaphor generation with an affective stance, and by inserting the metaphor and its most salient elements into a number of predefined sentence patterns, a poem that

builds on the metaphor may be generated (Veale 2013). As an example, the poem in (8) is generated using *lively city* with a positive stance as input.[3]

(8) No Beehive Is More Bustling

Charm me with your bustling boulevard
No supermarket is more hectically busy, or more largely crowded
Does any beehive hum more busily than this city?
O City, you fascinate me with your complex buzz

The example above takes us into the realm of poetry generation. In the following section, we will explore a number of additional approaches for poetry generation that rely on rules, templates, and handcrafted resources.

3.3 Poetry Generation

One of the first approaches that goes beyond mere mechanical creativity for poetry generation is the ASPERA system (Gervás 2001). ASPERA is an expert system that relies on a complex knowledge base, a set of rules, and case-based reasoning. The system, which is conceived for Spanish poem generation, is meant to be used in an interactive setting: based on a few input parameters specified by the user (intended message, mood, setting), it collects suitable vocabulary entries from a knowledge base, and puts them into verse based on the structure of existing poems. The candidate verses are then presented to the user for validation. Once corrected and validated, the poems are added to the system's database for future use.

Manurung et al. (2012) present another rule-based approach to poetry generation. Their system is based on a lexalized tree adjoining grammar (LTAG), and the generation of verse is optimized according to a number of constraints, such as poeticness (metre and rhyme) and meaningfulness (suitable realization of a semantic representation), using a genetic algorithm.

Gonçalo Oliveira (2012) presents a modular system for poetry generation, called PoeTryMe. The system relies on chart generation in order to generate candidate verses, which are again evaluated and optimized according to a number of different generation strategies. And, as described in the previous section, metaphorical expressions generated by pattern-based methods have also been used to generate poetry by incorporating them within predefined sentence templates (Veale 2013).

[3]Metaphor Magnet. http://ngrams.ucd.ie/metaphor-magnet-acl/. Visited 21 January 2018.

3.4 Story Generation

The creation of full-fledged, extensive literary artefacts, such as stories or novels, is arguably even more difficult than poetry generation, as a goal-oriented strategy needs to be adopted in addition to literary constraints. A number of rule-based systems have been developed that try to implement such objectives for story generation. One of the first systems that was developed for story generation is the Automatic Novel Writer system, by Klein et al. (1973). The system generates murder mystery stories according to a fairly rigid script. A set of rules model the dependencies between the different story situations, as well as the grammatical realization of each sentence in the story. The generation gives rise to murder stories of about 2000 words, which exhibit sequential coherence. An example is given in (9).

(9) RONALD AWAKENED. RONALD GOT UP. RONALD THOUGHT THAT THE DAY WAS BEAUTIFUL. RONALD FOUND JAMES. RONALD SAW THAT JAMES WAS DEAD. RONALD YELLED. THE OTHERS AWAKENED. THE OTHERS RAN TO RONALD. THE OTHERS SAW JAMES. EVERYONE TALKED. HEATHER CALLED THE POLICEMEN. HUME EXAMINED THE BODY. DR. BARTHOLOMEW HUME SAID THAT JAMES WAS KILLED BY POISON.

Another system that placed goal-oriented, coherent story telling at its core is the TALE- SPIN system (Meehan 1977). The system is conceived as a goal-directed, rule-based problem solver: characters within a story hold certain goals, and a plan is developed in order to achieve those goals. Additionally, the characters are endowed with certain personality traits, which influences how they might react in particular situations. An example output of the system is given in (10). Note again how the system is able to generate a coherent story, though the planning at the level of characters makes its output somewhat banal: not just any character goal makes for an interesting story.

(10) ONCE UPON A TIME GEORGE ANT LIVED NEAR A PATCH OF GROUND. THERE WAS A NEST IN AN ASH TREE. WILMA BIRD LIVED IN THE NEST. THERE WAS SOME WATER IN A RIVER. WILMA KNEW THAT THE WATER WAS IN THE RIVER. GEORGE KNEW THAT THE WATER WAS IN THE RIVER. ONE DAY WILMA WAS VERY THIRSTY. WILMA WANTED TO GET NEAR SOME WATER. WILMA FLEW FROM HER NEST ACROSS A MEADOW THROUGH A VALLEY TO THE RIVER. WILMA DRANK THE WATER. WILMA WASN'T VERY THIRSTY ANY MORE.

A similar approach is taken by the UNIVERSE system (Lebowitz 1985). UNIVERSE was designed in order to generate a neverending script of overlapping storylines, analogous to a continuous soap opera scenario. Each character is represented as a complex set of emotional features, as well as a set of interpersonal relationships. In

addition, the system contains a variety of plot fragments—certain actions in which the character might take part. The main difference with the TALE- SPIN system is that UNIVERSE generates a story according to author-level goals instead of character-level goals. Plot fragments are the narrative elements that are used in order to achieve goals defined by the author; characters take part in plot fragments, in line with their character representation. The MINSTREL system (Turner 1992), designed to generate stories in the spirit of the legend of King Arthur, takes the idea of author goals a step further. It aims to explicitly model creativity in the production of a narrative plan that satisfies a number of overlapping author goals, such as the incorporation of drama and suspense, while entertaining thematic consistency.

A number of other rule-based models exist that make use of similar techniques as the systems presented above (Dehn 1981; Theune et al. 2003; Riedl and Young 2010); for an overview see Gervás (2009). Generally speaking, systems that follow a rule-based approach devote most attention to the overarching storyline; they implement a goal-oriented strategy that generates a global, coherent, and interesting plot for the story. The surface realization of the plot—the rendering into natural language—receives much less attention. Some systems (e.g. UNIVERSE) do not consider this step at all, while other systems are restricted to rudimentary language generation models (such as simple, rule-based grammars) which leads to regular, unvaried language generation.

4 Stochastic Methods

Over the last two decades, the field of artificial intelligence in general, and natural language processing specifically, has seen a paradigm shift from rule-based and knowledge-based methods towards statistical methods and machine learning. Instead of relying on rules and manually constructed resources, the systems presented in this section take a data-driven approach. They rely on corpora and on statistical machine learning methods in order to extract information useful for processing and generating creative language.

4.1 Computational Humour

A number of researchers have applied statistical approaches to the problem of humour recognition (Mihalcea and Strapparava 2006; Kiddon and Brun 2011). The main idea is to construct a corpus of jokes, from which features are extracted. A machine learning model is then trained in order to classify a particular input as humour or not. A system for the unsupervised generation of jokes from data is presented by Petrović and Matthews (2013). They generate a specific kind of joke, which follows the structure *I like my X like I like my Y, Z*, where *X* and *Y* are nouns, and *Z* is an adjective. Their model relies on three assumptions:

- a joke is funnier the more often the attribute is used to describe both nouns;
- a joke is funnier the less common the adjective is;
- a joke is funnier the more ambiguous the adjective is;
- a joke is funnier the more dissimilar the two nouns are.

Crucially, *(noun, noun, adjective)* triples that adhere to these assumptions can be extracted automatically from large corpora using unsupervised measures of distributional similarity. As such, the system is fairly successful at generating these specific kind of jokes, like the one in (11).

(11) I like my relationships like I like my source, open.

4.2 Metaphor

Most data-driven work on metaphor focuses on identification and interpretation, and less on generation. Shutova et al. (2013) present an automatic, statistical method capable of both identification and interpretation. The method applies a clustering method, initialized with a seed set of metaphors, in order to find generalizations of metaphors in corpus data, which may then be used to automatically identify metaphorical expressions (Shutova et al. 2010). By exploiting selectional preferences, equally constructed from corpus data, the metaphors are also automatically interpreted by computing a literal paraphrase (Shutova 2010). For an overview of research on automatic metaphor recognition and interpretation, see Shutova et al. (2013).

4.3 Poetry Generation

Whereas poetry generation with rule-based and template-based models has an inherent tendency to be rather rigid in structure, advances in statistical methods for language generation have opened up new avenues for a more varied and heterogeneous approach to creative language generation.

Standard n-gram language models, in which the probability of the next word is estimated from a limited number of preceding words, have been used for poetry generation, for example in Ray Kurzweil's *Cybernetic poet* (Kurzweil 2001). In his system, word probabilities are estimated from a corpus of existing poems. The model is then able to produce poems in a similar style to the ones present in the corpus. Standard n-gram techniques tend to produce well-formed verses, but the coherence between different verses is rather limited.

A number of approaches make use of distributional similarity, an approach that automatically induces semantically similar words from corpora by looking at their shared contexts (Turney and Pantel 2010). Toivanen et al. (2014) present a method for

poetry generation based on statistical word associations extracted from existing documents, aiming to fill up a template with less predictable word sets that still produce relatively coherent content. As such, they are combining the pattern-based approach presented in Sect. 3.3 with corpus-based, data-driven methods. Similarly, Netzer et al. (2009) present a method for generating haiku using distributional methods.

In the last few years, neural networks have become increasingly popular in NLP applications. In particular, neural language models have demonstrated impressive performance at the task of language modeling. By incorporating word embeddings that capture word similarity, neural language models are able to overcome the problem of data sparseness that standard n-gram models are confronted with. Moreover, while standard language models only consider a limited number of words to predict the next one, recurrent neural networks (RNNs) build up a representation for the entire preceding context, which improves the coherence of the output. Recurrent neural networks have been incorporated within poetry generation systems; Zhang and Lapata (2014) use an encoder-decoder RNN for Chinese poetry generation, in which one RNN builds up a hidden representation of the previous line in a poem, and another RNN predicts the next line word by word, based on the hidden representation of the previous line. The system is trained on a corpus of Chinese poems. Yan (2016) try to improve upon the encoder-decoder approach by incorporating a method of iterative improvement. In their approach, the network constructs a candidate poem in each iteration, and the representation of the former iteration is used in the creation of the next one.

Ghazvininejad et al. (2016) combine RNNs (for syntactic fluidity) with distributional similarity (for the modeling of semantic coherence) and finite state automata (for imposing literary constraints such as metre and rhyme). Their system, HAFEZ is able to produce well-formed poems with a reasonable degree of semantic coherence, based on a user-defined topic. An example stanza produced by the system is given in (12).

(12) Noodles

The people wanna drink spaghetti alla,
And maybe eat a lot of other crackers,
Or sit around and talk about the salsa,
A little bit of nothing really matters.

Finally, Potash et al. (2015) use a recurrent neural network in order to generate rap lyrics, and Malmi et al. (2016) do the same using a neural network architecture combined with support vector machines.

4.4 Story Generation

McIntyre and Lapata (2009) present a generate-and-rank approach to story generation. As a first step, several possible stories are generated by consulting a database

that contains information about entities and their prototypical interactions, as well as prototypical ordering of subsequent events. It is important to note that the database is constructed automatically: the information about entities and events (represented as predicate-argument structures) is automatically extracted from a children's story corpus. The result of the first step is a tree of predicate-argument structures, such that each traversal of the tree represents a different story, encoded as a sequence of predicate-argument structures. Each of the sequences is transformed into text using a surface realization system in combination with an *n*-gram model. In a second step, a support vector machine classifier that is trained on human ratings predicts which of the generated stories is the most interesting. Each individual story is ranked according to a number of features, and the top result is chosen as the final story.

In visual storytelling, the goal is to generate a cohesive story from a sequence of images, coherently linking the pictured events through time. Huang et al. (2016) present both an extensive, crowdsourced dataset and a recurrent neural network approach for visual storytelling. First, the sequence of images (represented as vectors constructed by a convolutional neural network) is given to an encoder RNN, building up a representation for the entire sequence; next, a decoder RNN generates a story word by word for the entire sequence. The encoder-decoder architecture is trained on human-annotated image sequences, and is shown to produce reasonably coherent stories compared to models trained on simple image captions.

5 Conclusion

Generally speaking, computationally creative systems have up till now focused on very specific, constrained tasks; more evolved, unconstrained creative language generation—which requires a considerable amount of reasoning—seems beyond what current systems are capable of producing. We do, however, notice an evolution throughout the history of computationally creative models: whereas former approaches—based on symbolic rules—were very rigid with regard to their creative output, more recent approaches—based on statistical processing—do allow for more freedom in the generated expressions. This is particularly true for current approaches to poetry generation that are based on neural network architectures.

Still, more advanced, unconstrained creative generation is akin to the third kind of creativity we discussed in the beginning of this chapter—transformational creativity—and the field of AI in general will need further advancements before such kind of creativity may be effectively generated. Systems lack the capability of conscious self-evaluation, which is an important element for the kind of creativity that may be considered truly creative. This shortcoming of current systems is aptly expressed in Italo Calvino's 1967 essay *Cybernetics and ghosts*, and it seems fitting to end this chapter with his words, both as a characterization of the limits of current systems and a hint at future work:

> The true literature machine will be one that itself feels the need to produce disorder, as a reaction against its preceding function of order: a machine that will produce avant-garde work to free its circuits when they are choked by too long a production of classicism. In fact, given that developments in cybernetics lean toward machines capable of learning, of changing their own programs, of developing their own sensibilities and their own needs, nothing prevents us from foreseeing a literature machine that at a certain point feels unsatisfied with its own traditionalism and starts to propose new ways of writing, turning its own codes completely upside down (Calvino 1986).

References

Balpe JP (2000) Contextes de l'art numérique. Hermès science publications

Binsted K, Ritchie G (1994) An implemented model of punning riddles. In: Proceedings of AAAI-94. pp 633–638

Binsted K, Ritchie G (1997) Computational rules for generating punning riddles. HUMOR-Int J Humor Res 10(1):25–76

Bootz MA (2004) The creative mind: myths and mechanisms. Psychology Press, London

Bootz P (2012) From OULIPO to transitoire observable: The evolution of frenchdigital poetry. Dichtung Digital 41

Borillo M, Virbel J (1983) Bases de connaissances dans les stratégies de création textuelle. In: Cahier de l'ALAMO, vol 1

Calvino I (1986) Cybernetics and ghosts. The uses of literature. Harcourt Brace Jovanovich, pp 3–27

Colton S, Wiggins GA (2012) Computational creativity: the final frontier? ECAI 12:21–26

Dehn N (1981) Story generation after tale-spin. IJCAI 81:16–18

Gatt A, Krahmer E (2018) Survey of the state of the art in natural language generation: core tasks, applications and evaluation. J Artif Intell Res 61:65–170

Gervás P (2001) An expert system for the composition of formal spanish poetry. Applications and innovations in intelligent systems VIII. Springer, London, pp 19–32

Gervás P (2009) Computational approaches to storytelling and creativity. AI Mag 30(3):49

Ghazvininejad M, Shi X, Choi Y, Knight K (2016) Generating topical poetry. In: Proceedings of the 2016 conference on empirical methods in natural language processing. Association for computational linguistics, Austin, Texas, pp 1183–1191

Gonçalo Oliveira H (2012) Poetryme: a versatile platform for poetry generation. Comput Creat Concept Inven Gen Intell 1:21

Gonçalo Oliveira H (2017) A survey on intelligent poetry generation: languages, features, techniques, reutilisation and evaluation. In: Proceedings of the 10th international conference on natural language generation. Association for computational linguistics, pp 11–20

Huang THK, Ferraro F, Mostafazadeh N, Misra I, Agrawal A, Devlin J, Girshick R, He X, Kohli P, Batra D, Zitnick CL, Parikh D, Vanderwende L, Galley M, Mitchell M (2016) Visual storytelling. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. Association for computational linguistics, San Diego, California, pp 1233–1239

Kiddon C, Brun Y (2011) That's what she said: double entendre identification. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies. Association for computational linguistics, Portland, Oregon, USA, pp 89–94

Klein S, Aeschlimann J, Balsiger D et al (1973) Automatic novel writing: a status report. Technical Report 186, Wisconsin University

Krishnakumaran S, Zhu X (2007) Hunting elusive metaphors using lexical resources. In: Proceedings of the workshop on computational approaches to figurative language. Association for computational linguistics, pp 13–20

Kurzweil R (2001) Cybernetic poet. http://www.kurzweilcyberart.com/poetry. Accessed 30 Sept 2016

Lebowitz M (1985) Story-telling as planning and learning. Poetics 14(6):483–502

Malmi E, Takala P, Toivonen H, Raiko T, Gionis A (2016) Dopelearning: a computational approach to rap lyrics generation. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 195–204

Manurung R, Ritchie G, Thompson H (2012) Using genetic algorithms to create meaningful poetic text. J Exp Theor Artif Intell 24(1):43–64

McIntyre N, Lapata M (2009) Learning to tell tales: A data-driven approach to story generation. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP. Association for computational linguistics, Suntec, Singapore, pp 217–225

Meehan JR (1977) Tale-spin, an interactive program that writes stories. In: Proceedings of the 5th international joint conference on Artificial intelligence, vol 1, pp 91–98

Mihalcea R, Strapparava C (2006) Learning to laugh (automatically): computational models for humor recognition. Comput Intell 22(2):126–142

Netzer Y, Gabay D, Goldberg Y, Elhadad M (2009) Gaiku: generating haiku with word associations norms. In: Proceedings of the workshop on computational approaches to linguistic creativity. Association for computational linguistics, Boulder, Colorado, pp 32–39

OULIPO (1981) Atlas de littérature potentielle. Gallimard, Paris, France

Petrović, S, Matthews D (2013) Unsupervised joke generation from big data. In: Proceedings of the 51st annual meeting of the association for computational linguistics (volume 2: short papers). Association for computational linguistics, pp 228–232

Potash P, Romanov A, Rumshisky A (2015) Ghostwriter: using an lstm for automatic rap lyric generation. In: Proceedings of the 2015 conference on empirical methods in natural language processing. Association for computational linguistics, Lisbon, Portugal, pp 1919–1924

Queneau R (1961) Cent mille milliards de poèmes. Gallimard, Paris, France

Riedl MO, Young RM (2010) Narrative planning: balancing plot and character. J Artif Intell Res 39:217–268

Shutova E (2010) Automatic metaphor interpretation as a paraphrasing task. In: Human Language Technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics. Association for computational linguistics, Los Angeles, California, pp 1029–1037

Shutova E, Sun L, Korhonen A (2010) Metaphor identification using verb and noun clustering. In: Proceedings of the 23rd international conference on computational linguistics (Coling 2010). Coling 2010 organizing committee, Beijing, China, pp 1002–1010

Shutova E, Teufel S, Korhonen A (2013) Statistical metaphor processing. Comput Linguist 39(2):301–353

Stock O, Strapparava C (2005) The act of creating humorous acronyms. Appl Artif Intell 19(2):137–151

Theune M, Faas S, Nijholt A, Heylen D (2003) The virtual storyteller: story creation by intelligent agents. In: Proceedings of the technologies for interactive digital storytelling and entertainment (TIDSE) conference, vol 204215

Toivanen JM, Gross O, Toivonen H (2014) The officer is taller than you, who race yourself! using document specific word associations in poetry generation. In: 5th international conference on computational creativity, ICCC

Turner SR (1992) MINSTREL: a computer model of creativity and storytelling. Phd thesis, University of California at Los Angeles, Los Angeles, CA, USA

Turney PD, Pantel P (2010) From frequency to meaning: vector space models of semantics. J Artif Intell Res 37(1):141–188

Veale T (2012) A context-sensitive, multi-faceted model of lexico-conceptual affect. In: Proceedings of the 50th annual meeting of the association for computational linguistics (volume 2: short papers). Association for computational linguistics, Jeju Island, Korea, pp 75–79

Veale T (2013) Less rhyme, more reason: knowledge-based poetry generation with feeling, insight and wit. In: Proceedings of the international conference on computational creativity. pp 152–159

Veale T, Hao Y (2007) Comprehending and generating apt metaphors: a web-driven, case-based approach to figurative language. In: AAAI. pp 1471–1476

Wilks Y, Galescu L, Allen J, Dalton A (2013) Automatic metaphor detection using large-scale lexical resources and conventional metaphor extraction. In: Proceedings of the first workshop on metaphor in NLP, pp 36–44

Yan R (2016) i, poet: automatic poetry composition through recurrent neural networks with iterative polishing schema. In: IJCAI, pp 2238–2244

Zhang X, Lapata M (2014) Chinese poetry generation with recurrent neural networks. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). Doha, Qatar, pp 670–680

Music and Artificial Intelligence

Patrick Saint-Dizier

Abstract Besides its obvious emotional and psychological dimensions, music is also a very rational system that obeys complex and relatively rigorous construction laws. Aristotle argued that *music are numbers made audible.* Considering the high level of elaboration that most musical works have reached since the 17th century, we could also argue that *music are symbols, structures and processes made audible.* Besides the emotional aspects of music, there is a major rational component that needs to be explored. AI, language analysis and cognitive sciences are the ideal formal vehicles to realize this exploration. This contribution explores the contribution of music to artificial intelligence and vice-versa, from the early Baroque period to the contemporary productions where logical and statistical models are frequently used.

1 Prelude

Besides its obvious emotional and psychological dimensions, music is also a very rational system that obeys complex and relatively rigorous construction laws. Aristotle argued that *music are numbers made audible.* Considering the high level of elaboration that most musical works have reached since the 17th century, we could also argue that *music are symbols, structures and processes made audible.* Besides the emotional aspects of music, there is a major rational component that needs to be explored. AI, language analysis and cognitive sciences are the ideal formal vehicles to realize this exploration.

Before using an instrument, or conducting an orchestra or a choir, most performers develop a mental image of the musical work they want to play, with quite a lot of details, including the articulations and the main gestures. They often repeat that they do not play an instrument with their fingers, but with their brain. Fingers are marginal:

P. Saint-Dizier (✉)
CNRS-IRIT, 118 route de Narbonne, 31062 Toulouse, France
e-mail: stdizier@irit.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8_16

they execute the brain orders. Quite a similar view is shared by composers: as early as 1917, E. Varèse said *I dream about instruments that obey my thoughts, that would lend themselves to the combinations I wish*.

The study of music composition, performance, improvisation, listening, whatever the tradition, involves forms of theorizing. This means developing an analysis of the knowledge and the processes that conceptualize the process of composition, of listening or of performing a piece of music. However, understanding musical skills cannot probably be totally reduced to rigorous abstractions.

This double perspective clearly indicates that, in spite of their major differences, AI and music are areas that have a lot to share. During a first generation (1980–1995), a large number of formalisms for music knowledge representation, processing strategies, heuristics and tools based on AI were developed, mainly for tonal music analysis, composition support and performance. A second generation, which covers the period 2000–2015, focuses on the development of more complex systems for contemporary music production, where the types of objects and the processes which are involved have been considerably generalized.

This chapter can be read without any specific knowledge of music. The main notions in music are progressively introduced. This chapter is organized as follows. Section 2 develops the position of music with respect to the philosophy of rationality, and then considers its relations with natural language models and cognitive psychology. This section ends by an overall presentation of the interactions between music and AI. Section 3 develops features of musical knowledge representation. As an illustration, a model based on typed feature structures is introduced. Its use is illustrated to model motive and theme representation, chords and polyphony organization and the definition of operations on these structures. This section ends by a presentation of how the musical concepts have evolved in contemporary music and the role of AI in this evolution. Section 4 presents various artificial music intelligence research areas such as: modeling emotions, problem solving in music, machine learning for music production and analysis, multi-modal environments and intelligent tutoring systems. This last section gives a global picture of the main research trends and of the tools that have been developed.

2 Music, Language and Reasoning

This first section develops the position of music with respect to two closely related areas: linguistics and natural language processing, since music is a language, and cognitive sciences. The prominence of cognition in music production and performance is outlined, since it has many consequences on AI models.

2.1 Music and Rationality

Since the Greek period, a number of authors investigated the structure of music from a rational and scientific point of view. Till the Renaissance, music was part of the *Quadrivium* together with geometry, arithmetics and astronomy. The three other 'liberal' arts, the *Trivium*, included grammar, rhetorics and dialectics. Music was the closest discipline to the Trivium. Saint Augustine (354–430, in the Confessions and De Musica) and Boece (470–525, in the Consolations) show that music is a science, via the development of a rational analysis of music based on numbers and proportions, supposed to manage the harmony of movements, including movements of planets. At that period, music was considered not only as a mathematical object that describes the structure of melodies and rhythm, with a strong explicative power, but also as a form of abstraction that reflects creativity and perfection.

In the early Middle Ages, Hildegard von Bingen (1098–1179) is the composer that probably developed the most interesting scientific and logical analysis of music. Her very visionary style, made of images with striking effects, reveals a nun fascinated by theological questions. Her analysis is supported by liturgical songs that she composed, including hymns and sequences *(Ave Generosa, Ordo Virtutum*, etc.). Then, the later Medieval period developed a very strong view of music via a metaphysics of sound organization: music becomes a rational part of theology. In the Gregorian tradition, music is viewed as a *perfect sound with a unified view of body movements, pitch, metrics and text*; it is an art of the orator (*jubilus*). At that period singing and recitation were not so different, this is probably the origin of the contemporary *SprechGesang*.

From the Renaissance, the scientific analysis of the musical discourse, and of its effects on the listener gradually took some distance with philosophy and theology. Music was associated with a more analytical vision, with, among others, the following major points of investigations, which are still influential on to-day's AI and cognitive science analysis of music:

- analysis and modeling of musical structures: from melodies and rhythmic structures to polyphony and comprehensive pieces (fugues, variations, tripartite forms, sonata, rondo, etc.). These models were generalized at the end of the 19th century with e.g. cyclic themes (C. Franck) and polythematic forms (A. Bruckner, J. Sibelius) and then in the early 20th century by the Vienna school of Serial music, to all dimensions of music (A. Schoenberg and A. Webern).
- modeling and extending the facets of the tension-resolution mechanism in tonal harmony, which allows the introduction of colors and contrasts, and generalizations of the initial principles of harmony, gradually leading e.g. to the notion of note cluster (culminating in C. Debussy), or the re-introduction of Greek and medieval modes and rhythms (e.g. M. Emmanuel, C. Tournemire),
- development of models for polyphonies, with the analysis of their communicative perception and dimensions on the listeners, culminating in the late baroque period (J.S. Bach).

Behind this analysis, there are important musical knowledge representation issues paired with well-formedness constraints. Contemporary music has generalized musical structures and constraints, these are presented in Sect. 3.6.

A number of 17th century composers, who were also mathematicians and philosophers, produced treatises on music structure and rhetoric dealing with these aspects, such as M. Praetorius (1571–1621), J. Burmeister (1566–1629), M. Mersenne (1588–1648), A. Werkmeister (1645–1706), and J. Mattheson (main treaty circa 1722). More recent theoreticians include (Schonberg et al. 1999) and (Hindemith 1984).

2.2 Music and Language: Striking Similarities?

It is now commonly admitted that music is a natural language that has its own lexicon, syntax and discourse structure (Katz and Petsetzsky 2009). The study of the relationships between language and music is indeed a very old tradition. It is then not a surprise that cognitive science, language and music have strongly influenced each other in practical and theoretical circles: music motivates and determines a fundamental part of the human behavior. We can then expect music to bring its own, original contribution to studies on the brain and the mind, that is different from what language can offer. Another contribution of much interest is the relationship between intellectual and affective attitudes. This sets new challenges to AI research.

The cognitive similarities and dissimilarities between language and music have stimulated a number of debates in linguistics, music, psychology, cognition and philosophy. When considering high level and very abstract capacities of the human brain, similarities are striking. This is the case for many aspects concerning the structures, the processes, the functions and the affects conveyed by both language and music. This is probably also the case for the other forms of art. However, these abstract capacities remain to be identified and their functions need to be characterized from a scientific point of view. Cognitive psychology in conjunction with Artificial intelligence together is the main means to better understand and provide models for these abstract capacities.

Language utterances convey information, beliefs or jokes, suggest actions, teach, give orders, ask questions, remind listeners about their obligations, etc. Language is first designed to convey meaning. Meaning entails reasoning and knowledge acquisition and revision. Meaning may provoke affects. Music does not convey information and meaning in the same sense than language. Music is basically designed to convey affects, but these affects are not just psychological: the means used to stimulate affects have a strong internal cohesion that induce forms of underlying meaning. For example, figures of sound have strong argumentative capabilities due to their rhetoric power. Music also conveys a number of symbolic elements which include some types of abstract meanings, via, for example, the reference to typical numbers and proportions in the macro-and micro structures of a piece of music van Houten and Kasbergen (1985), or the metaphors developed in the baroque and classical periods, still widely used in the contemporary period.

There are many languages around the world, with very diverse lexicons and syntax. Music is more universal and even if there are different traditions and genres, it does not need any translation to be 'understood', it just probably needs some familiarity with a given style to be appreciated. In spite of these differences a number of formalisms and processes proper to natural language processing can be transposed to music analysis, as will be seen in the next sections. For example, the system developed by Lerdahl and Jackendoff (1983) is based on Generative Grammar principles; it characterizes hierarchical relations among notes in a melody. This approach includes the musical meter, the notions of beat and of left or right melodic elaborations that is similar to the notion of left and right adjunctions in language. This system develops notions of prominent notes, ornamental notes, tensing and relaxing notes that is appropriate for music of the 18th century, but which cannot be used for later or earlier periods. However, this system has a good explanatory power that is used in a number of music AI applications.

2.3 Music and Cognitive Science: Prominence of the Brain

Besides the language perspective, a major approach to research in artificial music intelligence is cognitive psychology. Music cognition aims at improving our understanding of human psychology, emotion and intellect. In 1944, A. Copland said *that music gives pleasure is axiomatic. But the source of that pleasure is one of the prime puzzles of consciousness*. It is important to note that the approach of the composer or of the active performer is essentially mental: before creating a musical work or before using an instrument, it is essential for most composers and performers to develop a mental image of the musical work they want to produce or play with a deep understanding of all the details. Most performers keep on saying that they do not play an instrument with their fingers, but with their brain.

The development of cognitive models aim at telling us how active musicians, performers, composers and improvisers behave and reason, and possibly some features of the role of music in the society, this is developed in depth in e.g. Raffmann (1993), Lerdahl and Jackendoff (1983) and Patel (2003), Patel (2008). This integrated view is very challenging: so far musicology has essentially focused its domain of analysis on idealized listeners, with a clear separation of knowledge and action, a tribute to Cartesian assumptions. Cognitive musicology tries to model musical knowledge and production in relatively strict computational and compositional terms, in particular based on natural language theories and processes. It is clear that e.g. Beethoven's musical thinking cannot be reduced to rewrite rules and well-formedness constraints or to the possibilities of a Turing machine. What is left out is essential.

The area of research of music cognition questions the existence of music composition universals, as in natural language. It has to deal with complex problems among which: how to measure music parameters (audio, affects) and what are the parameters of a given style, how to formally represent music components (sound, processes, structures), and how to use this data to have a better understanding of

music, musicians and to be able to produce new forms of music Mantaras and Arcos (2002), Temperley (2004). Western music notation, based on scores, is a simplification and an abstract model of the musical reality and practice. Scores represent e.g. pitch and rhythm in a straightforward way, but they fail to describe how to interpret pitch and rhythm in context.

Experimental psychology, in particular Gestalt psychology, has developed models that explain how music is perceived, on the basis of patterns or grouping principles. These models include perceptual notions such as proximity, continuity, closure, similarity, regularity, etc. These models partly allow an analysis of the discrepancies observed between e.g. notated rhythm and performed rhythm. The same observation holds for chords and many other dimensions of music.

An interesting and simple model is Narmour's Implication-Realization model Narmour (1990) which is based on the notion of expectation. Expectations in the listener's mind are structures or motives that the listener, via its musical experience, expects to hear after a certain musical sequence. For example after a series of chords developing a tension, he expects a resolution. When the expectations are not fulfilled, the listener is surprised, disappointed, uncomfortable or even stressed. Narmour proposed a cognitive Gestalt theory based on a set of expectations for musical motives which provides a psychological model for melodic surface realizations. This system can be generalized to more abstract structures such as expected interval alteration, timbre or dynamics.

Recent research in music cognition focuses on musical action and aims at modeling the listener's and the musician's behavior. These actors are integrated into a model of intelligent agents. Several assumptions guide this research. The first is the possibility to identify stable primitive ingredients that constitute musicians' behavior. The second critical point is the possibility to define subsets of consistent rules or constraints operating on these ingredients. A third question is the nature and the structure of a musical memory, and how much it is different from a musical computer memory. Research shows that humans tend to remember musical fragments of a musical piece, and these are not necessarily tied together by any global and coherent musical principle. These could be the fundamental ingredients on which a model could be developed, from, e.g. Lewin (1986) model of musical perception.

A last point of investigation is the relation between musical experiences in association with non-musical experiences and emotions. A well-known example are the relations between music and rhetoric (Saint-Dizier 2014). Other examples of non-musical experiences include the perception of colors, space and movement, which is now a source for musical composition.

2.4 The Contribution of Music to the Development of AI

Music and AI are two very distinct areas and activities. AI can learn from the analysis of musical processes. Conversely, music is clearly an activity that involves complex intellectual skills for which AI formalisms and tools are extremely useful. Music can

make an interesting contribution to knowledge representation and reasoning, from theoretical analysis to more applied dimensions. Theoreticians of music try to understand and conceptualize e.g. the various abstractions and processes at stake in music composition, analysis and teaching. There is clearly a major challenge to represent the musical knowledge in a way that reflects composer's behaviors, allowing partial knowledge, strategy revisions, etc. These issues have been investigated as early as Laske (1972). AI offers two main types of contributions:

1. the use of AI to analyze music produced and played by humans, in order to model the cognitive and intellectual processes of music production and performance at stake,
2. the use of the above models to produce new forms of music, to develop new instruments, new tools for composers with specific aesthetic, scientific and technological concerns. This is usually called **artificial musical intelligence**.

The main areas where AI has a major role to play are then:

- Music composition, that involves musical knowledge representation and dedicated forms of planning and problem solving. This level includes the analysis and the modeling of composition processes (e.g. sets of rules and constraints that describe the different facets of a musical work and the development of processing strategies), the automatic recognition of styles, authors or forms (e.g. by means of predefined or acquired patterns (Thomas 1985)), the development of devices to help composers (e.g. tools to produce musical motive variations, automatic musical composition tools based on stochastic models). The overall goal of a music composition tool is to support composers in highly creative phases of a work, including planning an overall structure. Such a tool is an intelligent assistant that leaves the overall control to the composer.
- Performance modelling, that involves various forms of knowledge acquisition, for example via forms of case-based reasoning. The aim is to model the way scores, which are abstract music representations, are played by musicians or groups of musicians (piano, string quartet, orchestra, etc.), in other terms the interpretation dimensions of a musical work and the performer's touch. These dimensions include basic features such as the dynamics, the accentuation, the articulations, the rubato and the vibrato (for some instruments and the voice). Modeling performance also allows new forms of music to be 'played' automatically with higher realism.
- Music theory, where the goal is to develop formalisms that account for the structure and the constraints of composer's personal styles of, more generally, musical styles and their evolutions. This aim can be carried out by reusing adapted formalisms used for natural language analysis, as shown in the next section. However, music has many more dimensions than language, e.g. polyphony, timbre, pitch, dynamics, therefore models must be able to accommodate all these dimensions consistently.
- Digital sound processing, with in particular the recognition of music features from a recording, which is mainly a signal processing task. This level also includes the creation of new instruments with new acoustics features and performance capabilities and the rework of recordings with computer tools.

In this chapter, we mainly deal with the three first items above, the fourth one being essentially a signal processing problem.

The above considerations on music and cognition show that modeling musical activities in AI raises several difficulties:

- First, domains in which music operates are not totally clear: psychological, perceptual, or emotional.
- A large proportion of musical knowledge and composition strategies is non-verbal.
- Composers have complex, non-monotonic music production strategies: they often change their mind or revise already written music.
- Next, it is not clear whether musical models can be given a Tarsky-style semantics, independently of any human usage and perception.
- Then, music is a multi-dimensional system that requires several layers of knowledge description and abstraction. Each layer is coherent and reflects a different perspective and system. It is not clear whether these systems elaborate on each other and are consistent or if they are inconsistent or even just unrelated.
- Finally, but there are many other questions, the close integration between musical knowledge and musical action remains problematic to AI which tends to establish a hierarchy between knowledge and constraints on the one hand, and processes and strategies on the other.

3 Main Features of Musical Structure: Musical Knowledge Representation

Music has an organization which is less hierarchical than natural language. The four main levels are, informally:

1. the melody level which accounts for the structure of the 'horizontal' dimension of music,
2. the harmony level, which develops the 'vertical' level of music in close connection with the melody, by specifying the structure and the sequencing of chords,
3. the polyphony level, which develops the organization of layers of melodies realized at several voices. Polyphony must observe the rules of harmony and melody construction, and
4. the form level, comparable to the discourse level in language, that specifies various musical organizations (e.g. fugue, sonata, minuet) at a more global level.

Music has many other components which are not be developed here such as: timbre and instrumentation, meter and rhythm, dynamics and accentuation, phrasing and articulations, and in more contemporary music: note clusters, groupings, series, etc. Each of these levels are managed by a set of specific rules and principles, and typical patterns that characterize various styles and genres, close to what is observed in natural language.

To understand how AI can model music production and performance, let us first briefly develop the different components of a musical work. A musical knowledge representation based on a typed feature structure (TFS) and type constructors is used as a illustrative model. The TFS approach allows to introduce the basic elements of a musical work of the tonal period. To illustrate the concept which are beyond the tonal system, Sect. 3.6 introduces several musical features of contemporary music which turn out to be of much interest to computer science and AI.

3.1 Basic Musical Knowledge Representation Features

Computational aspects of music have been investigated as early as Laske (1972). Several music description languages have been developed such as Formes Cointe and Rodet (1983), or Petitjean (2012) and Turnbull et al. (2008). Besides XML models such as MusicXML, most models are based on types characterized by patterns, possibly underspecified. Musical structures have been integrated in FOL, on which dedicated operators, such as transformations, are applied.

Most music description languages and music knowledge representation systems have adopted a declarative and modular approach, so that the different components of music (pitch, timbre, durations, accentuation, etc.) can be described separately and their relations made more clear. These are often represented as types or (logical) object classes. For example, there are many durationless structures that can be paired with rhythmic patterns. In durationless structures, duration is left underspecified. In rhythmic patterns, the pitch component is a priori left open, even if constraints can be formulated on melodic aspects (e.g. in the case of note repetitions).

Partially specified music structures are often represented by means of patterns and attributes. Music structure operators or musical transformations allow the instantiation of structures and their combinations. For example, in the Calm system (Blevis et al. 2002), primitive operations used to build music structures have been defined in terms of polymorphic constructors to combine structures of different natures (e.g. pitch and timbre), constraints on forms, and non-deterministic operations that instantiate event sequences. Tranformations are realized via first-order unification. Additional metrics evaluate the differences between sequences. Meta-control expressions describe how transformations can be computed.

These formal aspects are only one facet of the composition problem which requires much more flexibility. In general, composers of any period and style do not have a comprehensive formalization of the concepts they use. Therefore, specific types of operations must make the objects and processes flexible and revisable during the composition process, on the basis of second-order logics. Next, musical objects are often considered by composers from different points of view (e.g. harmony-pitch, or pitch-accentuation). Polymorphic types can be introduced for these objects with multiple facets.

3.2 A Model for Music Knowledge Representation Based on Typed Feature Structures

Typed feature structures (TFS) are widely used in knowledge representation, in particular in the representation of natural language structures, from morphology to semantics. For example, Shieber (1986) show the relevance of TFS for language processing, while Carpenter (1992) develops TFS theoretical features. A TFS is a recursive structure of the form 'attribute-value', where 'value' is a type, which can be simple (a value) or complex. A complex type is either a structure which is not an atom but a more complex representation (e.g. a formula) or, recursively, another feature structure. A feature structure is a hierarchical structure, where the top nodes represent the highest level features, for example, in natural language, the word stem, the morphology, the syntax and the semantics. These nodes are then further decomposed according to the representation needs and the accuracy of the description. For example, the feature 'morphology' includes the verb sub-attributes: gender, number, voice, tense and mode. A simple feature structure for the word *window* is the following:

$$\begin{bmatrix} \text{WORD} = \text{WINDOW}, \\ \text{MORPHOLOGY} = \left[\text{GENDER} = \text{FEMININE}, \ \text{NUMBER} = \text{SINGULAR}\right], \\ \text{SYNTAX} = \left[\text{CATEGORY} = \text{NOUN}\right], \\ \text{SEMANTICS} = \left[\text{TYPE} = \text{CONCRETE- OBJECT}\right] \end{bmatrix}$$

The reader may note that attribute names can be any tag, as long as they represent a unique feature. Values associated with each attribute define a type. They are, for example, constants, numbers, words, or any structure. Attributes may appear in a feature structure in any order, but hierarchies must remain the same for all the structures described in a system, e.g. for all words which are described, independently of their category. Some attributes are proper to certain types of words, for example `voice` is proper to verbs. This attribute does not appear in noun descriptions. The absence of an attribute in a feature structure may have different interpretations. In our case, it simply means it is not relevant. Finally, re-entrancy operators allow to establish constraints between distant subtypes in a TFS (Shieber 1986).

When several objects, for example in a sentence, are described by means of feature structures, it is possible to state constraints or relations between these objects, based on their feature structure content. It is, for example, possible to state that in the construction determiner + noun there must be an agreement in number (and gender for Romance languages). Therefore, informally, if:

```
determiner: morphology = [number = X1 ],
noun: morphology = [number = X2 ]
```

then, the agreement constraint states that: `X1 = X2`.

The syntax adopted in this article follows the Prolog syntax: symbols starting by a capital letter are variables, while the others are constants. The square brackets delimit lists of terms. Feature structures have been developed for a family of unification grammar formalisms, used to encode various linguistic theories such as the Lexical Functional Grammar or the Head Driven Phrase Structure Grammar. An introduction and comparisons in terms of expressive power are provided in Shieber (1986). Feature structures can be used in a large variety of formalisms and parsing strategies. They are declarative and allow the description of linguistic structures and constructions at a high level of abstraction and linguistic adequacy. They are therefore well-adapted to represent music knowledge.

3.3 Representation of a Melody

The basic melodic unit is the musical motive, which is in general 2 or 4 bars long (bars are also called measures). A motive has a relatively constrained internal structure, in conjunction with the harmony level. A theme is a sequence of articulated musical motives which follow precise construction rules that depend on the style and author that is considered. A theme follows construction principles such as symmetry. For example, given two musical motives A and B, with the same meter, key signature and tempo, a theme can be the combination: T1= A B A, or A B A1 where A1 is a variant of A, for example with a stronger cadence. Another theme can be T2= A B A1 B1 A2. A melody follows strict construction principles. It is for example a theme or a sequence of themes T_i which can be either contrasted or variations of a theme T.

A musical motive is basically composed of notes. A note has its own characteristics, independently of the musical motive in which it occurs. The main characteristics of a note are its pitch level composed of the name of the note (with values A, B, C, D, E, F), and its level (with values from e.g. 1 to 8 if we consider the octave as a unit), its duration (where values are the names of standard durations, e.g.: semibreve, minim or half note, crotchet, quaver, semi-quaver, etc.), sub-duration (which handles cases where the duration of the note is dotted, adding half of its duration), and, finally, accidentals (e.g. flat or sharp).

Let us consider Fig. 1 below, Brahms' variation 10 on a theme by R. Schumann, op. 9. This example shows two motives of two bars each at the upper part (soprano), and also at the lower part (bass). We will see later that these two motives are identical: they are mirrors of each other. This is an illustration of the high cohesion of a musical piece. The first note of the soprano, F-sharp, with pitch level 5, is represented as follows in the TFS formalism (nameN is the note name):

$$\text{NOTE:} \begin{bmatrix} \text{NAMEN} = \text{F}, \ \text{PITCH} = 5, \\ \text{DURATION} = \text{CROTCHET}, \ \text{SUB- DURATION} = \text{NONE}, \ \text{ACCIDENTAL} = \text{SHARP} \end{bmatrix}$$

Fig. 1 Brahms variations op 9

The term `NOTE` is a type constructor. This notation is much more elaborated than the MIDI notation, it is slightly simpler than Lilypond notations or the XML formats developed for example in Turnbull et al. (2008) or Petitjean (2012) or in description environments such as musicXML (http://www.musicxml.com/). The TFS notation gets a lot of power due to its level of abstraction and because of the types of operations which can be applied to it.

A musical motive can be analyzed, when it is realized on a single voice, as an ordered sequence of notes. Each note is represented by the type constructor `note` as illustrated above. Besides being a sequence of notes, a musical motive has its own feature structure which is functionally composed of two distinct parts:

1. Its global characteristics, valid for the whole sequence of notes, in particular: the meter, the key signature, the tempo and the bar number where the motive starts if it is included into a larger structure,
2. The ordered sequence of individual notes the motive is composed of. The order is encoded by the attribute `order` that is associated with values indicating the note's position in the sequence, so that the sequence is unambiguously structured. The attribute `order` is a kind of meta-attribute that introduces temporal relations (or events) between notes.

The type constructor `musical-motive` is defined as follows:

$$\text{MUSICAL-MOTIVE} : \left[\begin{array}{l} \text{NAME} = \text{ANY NAME}, \\ \text{FEATURES} = \text{METER}, \text{KEY-SIGNATURE}, \\ \text{TEMPO}, \text{STARTING-BAR}, \text{SLUR}, \text{DYNAMICS}, \text{ACCENT}, \ldots \\ \text{SEQ-NOTES} = \left[\text{NOTE} = \left[\text{ORDER} = 1, \ldots\right], \text{NOTE} = \left[\text{ORDER} = 2, \ldots\right] \ldots\right] \end{array}\right]$$

The two first bars of the soprano melody of the variation 10 in Fig. 1, are represented as follows:

```
MUSICAL MOTIVE :

[NAME = VARIATION10 , METER = 2/4 , DYNAMICS = PIANO,
 KEY- SIGNATURE = 2SHARPS, TEMPO = 'POCO ADAGIO', STARTING- BAR = 1,
 SEQ- NOTES = [NOTE= [NAMEN = F, ORDER = 1,
                      PITCH = 5, DURATION = CROTCHET, SUB- DURATION =NONE],
               NOTE= [NAMEN = C, ORDER = 2,
                      PITCH = 5, DURATION = CROTCHET, SUB- DURATION = NONE],
               NOTE= [NAMEN = D, ORDER = 3,
                      PITCH = 5, DURATION = CROTCHET, SUB- DURATION = DOTTED],
               NOTE= [NAMEN = B, ORDER = 4,
                      PITCH = 5, DURATION = QUAVER, SUB- DURATION = NONE],
               ETC.]]
```

From such a description, it is possible to automatically induce e.g. the notes on the major beats, the musical motive ambitus and the modulations. The key signature displays the accidentals of the tonality while local accidentals are mentioned at the note level. Note that the sequence of notes is a priori independent of the meter and of the starting point of the musical motive (bar-number). The type constructor `musical-motive` enables the description of information covering groups of notes. For example, the `nuance` type constructor can be used to express that there is a crescendo from the note with `order = 1` to the note with `order = 4`.

3.4 Representing Polyphony and Chords

A polyphony is a combination of two or more voices song at the same time. In counterpoint theory, these voices must share some common musical material and must not violate harmony principles. A well known case is the canon where voices start one after the other and have exactly the same musical motive, starting at different temporal points from the same note. A polyphony is modeled by the type constructor `seq-voice` that accounts for voice combinations. The names of the different voices can be any constant such as soprano, alto, tenor and bass, or any other identifier, especially when more voices are involved, or when these terms are not appropriate or when the number of voices changes over the music piece.

Let us now consider the case of simple chords viewed as a kind of cluster. The typical situation is a melody that is accompanied by chords. The chord [C, E, G] that is used as an accompaniment in Fig. 2 can be represented as follows, with notes at pitch level 4, following the tradition of reading chords in a bottom up fashion:

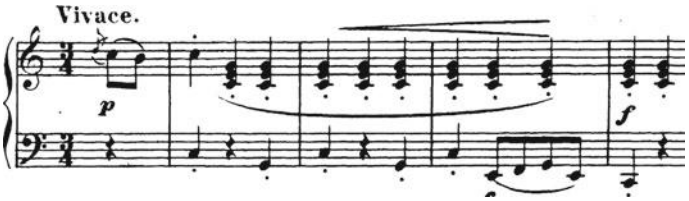


Fig. 2 Beethoven, theme of diabelli variations op. 120

$$\text{CHORD} : \left[\text{VERTICAL- SEQ} = \left[\begin{array}{l} \text{NOTE} = [\text{NAMEN} = \text{C}, \text{PITCH} = 4, \ldots], \\ \text{NOTE} = [\text{NAMEN} = \text{E}, \text{PITCH} = 4, \ldots], \\ \text{NOTE} = [\text{NAMEN} = \text{G}, \text{PITCH} = 4, \ldots] \end{array} \right], \ldots\ldots \right].$$

The bass in Fig. 2 is a musical motive that evokes a waltz. It can be represented in conjunction with the chord or as a musical motive that occurs at the same time, using the `order` attribute.

3.5 A Few Generic Operations on Feature Structures

Several standard musical operations can be defined on TFS. These operations mainly come from the counterpoint tradition. Most composers tend to introduce their personal touch in these very rigorous and abstract forms. These operations are expressed by means of transformations made on attribute values. These operations can be used in production (e.g. to produce new musical motives) or in recognition (to identify them from feature structures). Rules described below are thus a priori reversible. Transformations include operations related to transpositions, augmentations or diminutions, mirror and motive inversions. By analogy with linguistics, let us called these operations **music alternations**.

For illustrative purposes, let us consider a simple example: transformations by augmentation or diminution of note durations in a motive. This operation, very frequent in baroque music, consists in increasing or decreasing by a constant factor the duration of each note. The most common situation is an augmentation of a factor 2. Diminutions and augmentations are often observed between the *cantus firmus* and the secondary voices e.g. in the 18 organ Leipzig Chorales by J. S. Bach. Augmentation increases the emphasis on the theme and gives a lot of strength and persuasion to the motive. Diminution is appropriate for realizing various forms of decorations, comments, variations or transitions, e.g. in the classical sonata form where small fragments of the theme are developed. In the above Fig. 1, the two inner voices play the role of a kind of comment: they are in fact the soprano motive reproduced a third above, with passage notes and with a slight diminution of 25%. This reinforces the

cohesion of the polyphony. Considering the representations based on attribute and values given above, to develop a simple augmentation of a factor of 2, the operation consists in transforming every note duration into its immediate superior duration. For example, a crotchet becomes a half note. Dotted notes become double dotted. In the formalism that we have introduced, such a transformation can be carried out by a rewriting operation on the whole feature structure of a musical motive. The rewriting rule can be written as follows:

```
note: [ duration = X, sec-duration = none ] →
note : [ duration = X1, sub-duration = none ], precede(X1, X).
```

The predicate precede (X1, X) defines the sequences of notes by decreasing order: `precedes(half-note, crotchet).`

These rules are very systematic and cannot account for composer's creativity. Composition assistant tools offer more elaborated transformation rules which are composition of such basic rules, e.g. augmentation coupled with the introduction of ornaments and passage notes.

3.6 Model Evolutions in Contemporary Music

Since about 1930, the language of music underwent major evolutions, with the emergence of electronic music, automatic composition, concrete music, random music and random performance, etc. Music production and performance became closely related, with a more central role played by computers, in particular at the level of sound production and management and composition tools. Computers and in particular AI gradually played an increasing role in these new trends. We briefly review them below, outlining the role played by AI.

The initial protagonists of the development of radically new musical forms were, most notably: C. Debussy, E. Satie, A. Schoenberg, B. Bartok, J. Sibelius, A. Webern and C. Ives. New concepts and forms emerged around 1910–1920, as a reaction to romantic and post-romantic music. The main characteristics of this first wave of new music was the dissolution of tonality and harmony, the emergence of radically new musical forms, and the development of the Serial school (Schonberg et al. 1999), (Schonberg 2006).

A serial music is composed of a melody or a series of chords that follows a given fixed sequence of the twelve notes of the chromatic scale. Notes and chords no longer have any clear function. This sequence is then repeated several times in a musical work with various surface realizations that could evoke geometrical structures (e.g. A. Webern piano variations op. 27, 1936, illustrated in Fig. 3). This notion of series was later extended to the other parameters of music such as intensities and durations. An integral serialism flourished in the early 1960. The serial music produced was based on mathematical models, including micro- and macro-structure planning. AI models can develop such structures following composers' guidelines.

In Fig. 3, the 12 note series is given from bars 1 to 4 (first beat), the first notes are F, E, B, F sharp, G, C sharp, etc. The series is realized by a combination of note clusters

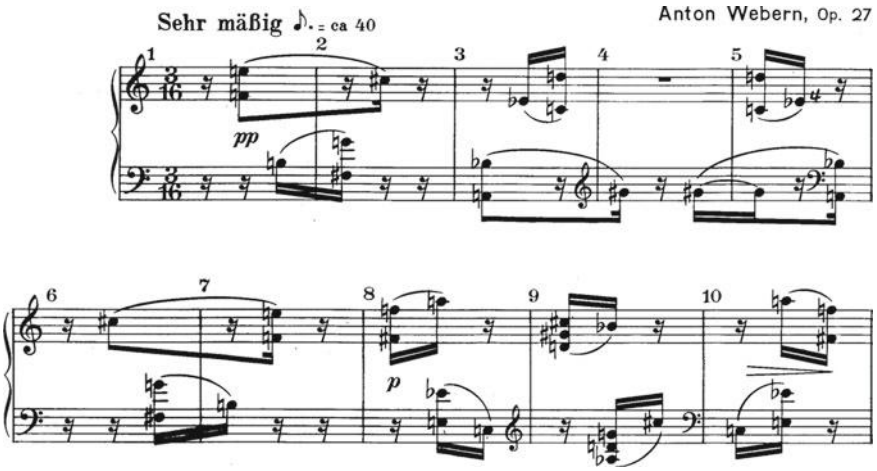


Fig. 3 A. Webern, piano variations op. 27

(e.g. F and E played together) and motives. From the end of bar 4 till the middle of bar 7, this same series is repeated in the reverse order. These two presentations of the same 12 note series suggests a symmetric geometrical figure, a kind of arch or water jet according to analysts. The piece goes on with a variant of the series with transpositions, generating the same geometrical figure in a more concise way.

A return to neoclassical forms (e.g. second part of I. Strawinsky production) occurred and lasted till the end of the second world war. From 1950, new forms of music emerged again, with more elaborated mathematical tools and more radical views of what music of the future should be, with new principles on (1) musical structures and on (2) the relations between composers, performers and listeners. O. Messiaen, with his *Modes de Valeurs et d'Intensités* (1949) composed for the piano the first work which was entirely based on a mathematical model, in particular to manage rhythm and dynamics. One of the major proponent of the integral serialism was K. Sockhausen, for example in his Kreuzspiel (1951) for small ensemble, where a high degree of abstraction was reached. Composers such as L. Nono, B. Maderna or L. Berio aimed at developing new ideas in terms of structure, with a stronger emphasis on aesthetics. This perspective motivated the emergence of new forms of melodies and melodic construction principles, based e.g. on constraint satisfaction principles.

Another trend was the introduction of random processes in music composition, paired with strict construction principles (e.g. Metastasis, by I. Xenakis, 1955). Music metaphorically became composed of 'clouds of sounds' and 'sound galaxies'. The introduction of irrational elements left quite a lot of freedom in the performance of such works. Musical research lead by e.g. K. Sockhausen and G. Ligeti in the domain of counterpoint transformed the baroque principles of a counterpoint of notes into a counterpoint of surfaces, shapes, masses and even 'musics', combining different

aesthetics. The complexity of the musical objects and concepts that were manipulated became a challenge for the listener and changed his approach to music. As a reaction to these complex systems, a kind of return to the past, around 1968, was observed through the Minimalist and post-Minimalist Music approach whose challenge was to produce expressive music from very limited musical materials. Pieces are produced from a few notes, with a limited set of instruments. Producing such a music is not easier than producing serial music, but computer tools and planning systems are much simpler and accurate.

From the manipulation of natural sounds by computer to produce new types of sound from existing ones, the production of electronic sounds and the introduction of sounds of the everyday life (whistles, trains, cars, clocks), emerged the paradigm of Concrete Music (P. Schaeffer, P. Henri). The idea was to have an equal treatment of various sources of sound, be they natural or artificial (e.g. Hymmen, K. Stockhausen, 1967). For that purpose, a taxonomy of musical objects was defined, where all objects are treated at the same level, with their specific features. This allows, e.g. to introduce performance parameters (e.g. time - scale ratios) in abstract composition constructs. Musical objects were then associated with event structures, similarly to event structures in natural language semantics (Talmy 2001). The traditional meter was 'smoothed': instead of beats, time became continuous, without no notion of pulse or counting. The notion of event was extended to graphs of events, with numerous parallel events: the notion of meter moved at IRCAM to e.g. polymeters (Bel 2002). Musical composition became a kind of musical object mapping process that must meet stylistic constraints.

The major intellectual challenge of these movements was the development of a strong convergence between the musical material, whatever it is, and human thought and abstraction (L. Nono, L. Berio, Sequenzas). In the music of the 21st century, contemporary music principles are essentially theoretical elaborations, involving e.g. spectral musics (based on electro-acoustics principles and planning algorithms), minimalism, or music based on repetitions as in traditional musics, etc. The notions of events and time have also been investigated, leading to notions such as contracted, dilated or striated time. These theoretical considerations on music had an influence on AI and cognitive science development. Most notably, new types of flexible and underspecified objects and processes, postulated to mirror human thought and behavior, have contributed to the development of contemporary foundations of AI and new approaches to human cognition.

4 AI and Music: Main Topics and Research Areas

AI has been used in music in a number of areas. First generation models and resulting tools are mainly dedicated to tonal music. They include the automatic analysis of music (structure recognition, often based on grammatical principles), the creation of improvisation supports, melody harmonization, tools for music composition and performance (based on grammars, on machine learning or, to a lower extent, on neural

networks), and music retrieval techniques (Barrington et al. 2008). Second generation tools and formalisms generalize the previous experiences and develop new mathematical concepts for sound production, music structure, and music performance and perception which are proper to contemporary music.

Most of the first generation systems share several features: they address Western music, they include some form of emotional criteria, music knowledge is based on facts and rules, often implemented in Prolog. Facts and rules are based on some theory of music, e.g. (Lerdahl and Jackendoff 1983), and style (Carpenter 1991), (Hofstetter 1988). Processes involve AI search strategies, possibly associated with heuristics or reduction search space techniques since the non-determinism is high, much higher than e.g. for language processing. Modeling processing strategies by means of neural networks is not as widely used as in other areas that involve intensive knowledge processing. Architectures are traditional AI architectures (Blevis et al. 2000), (Cook and Morgan 1993).

A major difficulty for AI is the modeling and the acquisition of non-verbal or partly verbal knowledge. No general methodology exists. This is typically the case for most fine arts. The strategy is to bypass the verbalization levels and to focus on non-monotonic processes such as composer's intuitions and strategies, their changes of mind, their experiential knowledge, or the way they plan and revise a musical piece. A direction is the definition of a task model, derived from the observation of composers. The tool presented in e.g. (Garcia et al. 2014) develops such a model from composers such as (Hindemith 1984) and P. Schaeffer works. Modal logics are also relevant to model composers' behaviors which do not always seem to be totally rational to an external observer. The resulting knowledge representation and task models have different uses which are illustrated below. They include:

- the investigation of the relationships between mental processes and the structure of musical pieces,
- the definition of an empirical model for music composition in general or given some inputs (style, mood and emotion, motives, etc.),
- the development of intelligent assistants for music composition,
- the elaboration of evaluation methods for assisted composition tools,
- the development of guidelines for teaching music composition or performance that combine traditional music theory and analysis with issues proper to AI.

4.1 Modeling Emotions

Emotion analysis and production is becoming a major research field in AI and in opinion analysis, and is of much interest to music perception analysis (Gratch and Marsella 2009), (Pynadath et al. 2013). In music, emotions features are frequently associated with musical rhetoric (Saint-Dizier 2014), with the development of figures of sound that parallel figures of speech in language. This topic is not new and has motivated a number of treatises in particular in the 17th century. In spite of some

disagreements between authors, the emotive effect of each of these figures of sound is relatively well-defined taken in isolation. The main music parameters which must be considered to develop a model for music affect are numerous:

- Modes and tonalities, which have different colors and entail different emotions,
- Melodic intervals and profiles of musical motives, in particular in relation with spatial metaphors. Unusual melodic intervals or chromatic sequences create affects such as happiness or pain,
- Rhythm in musical motive, meter, tempo and tempo variations, specific forms of rhythm have an important impact on affect (e.g. the dactyle versus the anapest, syncopation, etc.),
- Mood, nuances and articulations,
- Harmony: sequences of chords, realization of chords,
- Timbre, instrumentation and registration for the organ,
- Symbolic aspects of forms. For example, some forms suggest strong images: the sea and the birds in romantic music, the cross in baroque music, etc.

It is clear that these parameters largely interact; the moods and affects produced by a parameter can be further refined, transformed or modified by other parameters. The interactions between these parameters are difficult to characterize on the listener (Si et al. 2008).

In a tool such as Wolfgang (Riecken 2002), that realizes tonal monodic compositions, emotions are encoded via specific nodes, called E-nodes. These are not primitive musical artifacts, but properties associated with constructions, that characterize their emotive potential. Four distinct emotion values are considered: happiness, sadness, anger and meditativeness. Each emotion value covers a relatively large spectrum of emotions. In a musical work, emotions form a complex network represented by means of K-lines, a notion introduced in Minsky (1980). Sets of agents modeled by K-lines involve the creation of partial emotive mental states. The propagation of emotions and networks of emotions represented by K-lines is based of summing, averaging and smoothing algorithms. However, this system does not say how emoting structures and networks influence the composition process.

4.2 Problem Solving for Music Analysis and Production

A musical work is highly structured at several levels: theme management, harmony, counterpoint development, etc. These levels have a lot of interactions, and must meet a number of well-formedness constraints which are partly style dependent. Making the harmonization of a simple song is a complex task: which chords to choose with which harmonic functions, how to link chords given harmony and style constraints, how to distribute the notes of a chord on each voice, etc. is highly non-deterministic. The strategies used for analyzing or producing a musical work of a certain ambition are therefore extremely complex and entail grammatical and logical models that require accurate problem solving algorithms. Musical works

also include, for example, discontinuous and dependency structures, as e.g. left-extraposition phenomena in language, that require the development of formalisms that can accommodate various forms of variable and long-distance dependencies.

Control structures must provide problem reduction heuristics. This approach is crucial since there are many dependencies between the different facets of a piece of music. For example, a method developed in Marsella and Schmidt (1998) involves two sets of specifications: (1) a set of quite abstract and possibly underspecified objects describes the contents of the musical work: motives, sequences of chords, types of variations, etc. and (2) a set of specifications contains rules for the development of the musical piece and requirements or constraints that the musical work should meet. The authors argue that the process of musical composition can be viewed as the analysis of the control structures at stake when producing a musical piece.

This view is rather restricted to a rationale analysis. It is clear that, besides problem solving there is also a decision making process since there are many directions that are possible when producing a musical work: various, almost equivalent, ways of developing a motive, various forms and combinations of timbre, of polyphony, etc. Finally, it is interesting to note that in spite of their popularity over the las twenty years, neural networks have not been much used to model search strategies. More 'traditional', logic-based approaches have been favored, possibly because of the difficulty to develop data sets of observations.

4.3 Machine Learning for Music Analysis and Performance

Machine learning, although it may have little explanatory power when investigating a phenomenon, whatever it is, has however proven useful in a number of real-world applications. Machine learning is basically oriented towards classifications, clustering and pattern extraction from raw data. Machine learning can be used for several aspects of music analysis, for example to identify musical patterns with some flexibility or to develop artificial performers from the observations of real performers. However, considering the complexity of a piece of music or of a performance, the difficulty to access to the real data for such a non-verbal activity, and the abstraction levels that are expected for the results, it seems that only relatively simple features can be efficiently and accurately treated by machine learning techniques.

A detailed analysis of the main parameters at stake in expressive music performance analysis is developed in Widmer (2000). The main parameters considered are the notes (pitch and durations), rubato, vibrato and dynamics. However, the main result is that it is not really appropriate to learn how isolated notes are played. Performance and expression rules are learned at the musical structure level (a motive or a theme, with its harmonic and possibly polyphonic context). This confers a much more global view to a performance, taking into account local 'contexts'. An important result, via performance analysis, is how a given piece of music is understood by a performer and how this understanding is expressed in the performance. For that purpose, some knowledge about musical perception and musical structure must be

developed. Widmer's project was conducted using a symbolic learning algorithm, in contrast with neural networks, whose results remain somewhat opaque.

This research had major results, among which:

1. some principle of expression in performance are learnable,
2. these principles provide strong evidence for the role played by musical knowledge in the perception of a performance,
3. rules that were learned can be interpreted, e.g. to identify regularities or a detailed picture of the diversity of music structures, going beyond the Schenker model Schenker (1954), Morgan (2014) which concentrates on prototypical elements and neglects the details that, in fact, are often the most valuable features of a piece of music. Finally,
4. the identification of stylistic differences and evolutions over a composer's life or over a certain musical period (e.g. the way baroque music was played over the last 50 years).

Another interesting use of machine learning is the extraction of typical music patterns from a collection of musical pieces or within a given piece of music. One of the goals is to investigate how composers develop musical motives, introduce variations, how and possibly for what stylistic and expressive purposes. Other goals include the analysis of the evolution of music composition, automatic music generation, and the development of an intelligent music composer assistant. Pattern based analysis and extraction was initiated by authors such as Ruwet (1972), and developed by authors such as Roland and Ganascia (2000). Musical patterns are motives, sequences of chords, polyphonic or counterpoint structures. These are extracted from a set of musical scores, transcribed in a way usable by a computer (e.g. in MIDI notation). A typical, simple system is EMI (Carpenter 1991). EMI extracts sequences of pitches and durations which are relatively frequent in a collection of pieces.

Pattern extraction is, roughly, based on the following steps and constraints (Roland and Ganascia 2000):

1. identify sequences of notes which occur quite frequently, this is often realized by a comparison algorithm that considers all possible sequences and makes comparisons,
2. introduce flexibility factors, from musical composition knowledge, such as transposition, local rhythmic changes, interval amplification or reduction (as in fugue themes and their response), introduction of various types of ornaments, etc. This seriously complexifies the learning algorithms, but does correspond to the music composition practice,
3. develop adequate metrics to measure numerical similarities between music sequences and to induce patterns, which must be close to each other, given a threshold,
4. identify subsequences or groups of sequences: for example, motives and themes composed of these motives,
5. measure frequencies to indicate the recurrence of patterns in a set of musical pieces, and possibly their evolution.

Comparison between sequences involves a quantitative similarity measure, ranging from 0 to 1, and a qualitative similarity measure which attempts to explain the differences between sequences based on musical knowledge. Strict matching between sequences is relatively unusual, therefore, musical knowledge is crucial to identify patterns in a sound way.

Deep learning (e.g. (Bengio 2009)), is becoming a major trend in AI. In music, it has not been so widely used yet, for example, to reproduce works of existing authors. The difficulty is to introduce unexpected forms that characterize the creativity of authors. Reproducing existing works can be done via pattern-based approaches. Deep learning is used at the audio level, where performance features can be acquired (Bertin-Mahieux et al. 2010), including heuristics on expectations on audio classifications (Hamel and Eck 2010).

4.4 Multi-modal Intelligent Environments

These environments, which are part of the augmented reality paradigm, provide (1) virtual environments for an interactive production of music based on predefined music knowledge and (2) hyper-instruments which get additional properties from sound manipulations by computer. These intelligent environments provide, for example, a simple mapping between human movements and sound. This has many applications in film, advertising and computer games. These environments can also be influenced and revised by human behavior or any other form of specifications.

Sensor systems analyze and categorize various types of movements: body parts, eyes, various objects, and additional parameters such as acceleration. They induce a taxonomy of the main types of movements (e.g. those developed at the MIT media lab). A vocabulary of gestures can then be defined, based on a Gestalt approach (Blevis et al. 2000). Movements, speed, local gestures of various kinds, as well as the style of the movement are taken into account and correspond to music motives which are integrated via composition rules. Dynamic programming is often used to develop such interactive environments, leading to the expression: *the computer as an intelligent artist*. This expression probably focuses on the rationale dimensions provided by music produced by computers.

4.5 Intelligent Tutoring Systems

There are many programs and environments designed to teach music, however few of them are really intelligent tutors. The goal is to create systems that behave as a human teacher: presenting tasks to students and providing feedback and guidance when necessary. Students can make errors which violate some music composition rules of a certain style or period, or they can develop a music composition which must be improved in various respects, e.g. the harmony needs to be enhanced or

some voices of a polyphony need to be more expressive. Intelligent tutors are in general based on a set of rules and constraints, quite similarly to natural language processing. However, music knowledge cannot be quantified as precisely as rules of e.g. morphology. Furthermore, the rules that make a good, expressive composition are not necessarily well developed in tutoring systems: intuition remains an important factor.

Intelligent tutoring systems are composed of:

- a domain model that describes music knowledge related to a given style (e.g. late-baroque, Mozart-like), and how to use it to produce musical pieces of a certain quality,
- a student model that includes heuristics, guidance principles and feedback production,
- a set of strategies to identify the student aims and to provide guidance when appropriate and, finally
- a well-designed user-interface where the students can easily edit scores, visualize already existing examples (e.g. typical patterns, existing works).

These systems often include an indexed library of music compositions and typical patterns of composers which can be shown as examples to the student. A simple system that performs these tasks is GUIDO (Hofstetter 1988), more advanced systems include THE MUSES (Sorisio 1987) and VIVACE (Cook and Morgan 1993), (Thomas 1985).

An interesting subset of tutors are based on computational models of creativity. These models applied to music originate from works developed by, among others, Minsky (1981) and Sloboda (1985). Music, like many forms of art, is an open-domain that obeys to constraints that can evolve, as exemplified in Sect. 3.6 devoted to the evolution of contemporary forms of music.

As pointed out in Sloboda (1985), creativity is based on initial data and preexisting constraints that must be satisfied in some way. Then, creativity can be characterized by the development of a musical piece that meets additional constraints which are coherent, cohesive and acceptable by the artist. Then, preexisting constraints can be replaced by new ones, defined by the composer. Once again, constraints in music are not as clear as constraints in mathematics. They are often based on perceptual and cognitive factors in accordance with a cultural consensus, and probably some deeper psychological considerations that owe e.g. to rhetoric.

Coming back to tonal music, and assuming that a number of constraints can be modeled in a formal language, then a tutoring tool that supports forms of creativity must contain:

1. constrained-based representations of musical knowledge in harmony, motive and thematic construction, polyphony development, etc.,
2. a set of prototypes based on the elements given in (1), which can be used by a composer or a planner,
3. constrained-based descriptions of musical pieces, forms and styles, with real examples,

4. and a constraint-based planner such as PLANC, based on logic programming (Pascal van Hentenrick 1995).

The prototypes advocated in (2) above contain schemes to develop motives and variations, harmonic sequences with specific roles (such as cadences, transitions for theme re-introduction, etc.). More advanced features like instrumentation and dynamics do not have yet received any convincing model. Furthermore, their interactions with the more fundamental rules of music are not well established. An interesting system is COLERIDGE (Cook and Morgan 1995) that supports problem seeking instead of problem solving. This system focuses on precise tasks, which are difficult for composers such as motive development or principles of instrumentation.

4.6 Music Composition Tools

There is now a proliferation of computer tools for music composition. Most are basic systems aimed at producing real-time accompaniments or new types of sounds. A synthesis is provided in:

http://www.hitsquad.com/smm/cat/COMPUTER_AIDED_COMPOSITION/.

In this short section, the main tools which are related to AI are mentioned with their main features. Systems and tools introduced in the previous sections are not repeated here.

Concerning tools that use AI principles which assist musicians in the production of new music pieces, let us note Harmony Navigator 2.7 (2014) which provides simple accompaniments in real time based on an accurate multi-layer musical knowledge representation. Algorithmic composer V2 (2000), Jniz 2.7 (2016), Symbolic Composer (2014) and Opusmodus (2015) are sets of tools designed to develop various kinds of musical compositions. Algorithmic composer uses abstract data types to represent musical knowledge. Strasheela (2009) is an interesting system essentially based on constraint satisfaction expressions. The other tools mentioned above include interesting planning strategies based on learning techniques.

Concerning contemporary music, a system such as FractMus 2000 (1999) is a system based on Fractal geometry theory that provides some help to write music in this approach. Minimalist music and other AI-based approaches do not have yet any available tool for composers. Some tools are paired with video and score editing facilities. In terms of score edition, Musescore is an easy to use tool which produces files with straightforward integration capabilities in text editors. An renewal of music authoring tools is emerging, with for example, the notion of 'flow machines' developed at Sony Labs Paris, which, given a large corpus of music, creates musical textures which are used to create new sounds and new sound sequences. Techniques used are essentially based on sampling techniques and Markov sequences, e.g. Papadopoulos et al. (2016). Based on the notion of Flow-Machine, algorithms allow the generation of new music works based on the characteristics of a given style. From these exper-

iments, in particular the Continuator machine, (Pachet 2012), (Pachet et al. 2017) developed insights into music creativity and the notion of virtual musicians.

5 Conclusion

This article shows the mutual influence of music and AI, in spite of the difficulty to capture music non-verbal knowledge and composition strategies. Music has received quite a lot from the AI models to develop more advanced composition and automatic analysis tools. Conversely, AI models have been enriched by the very complex representation and computation problems raised by music. Cognitive psychology is also a major cornerstone to better understand how composers produce music.

Two main trends are observed: the development of models and tools for 'tonal' music and the generalizations of the underlying concepts for contemporary music where music concepts become much more abstract. In this article we have developed the main uses of AI for music: composition, analysis, performance, listening and teaching. These five perspectives are based on similar music knowledge representations.

We have also shown the limits of a science such as AI, even in conjunction with cognitive science, to capture all the features of music, even if music is defined as a science, as advocated in the introduction. This is obviously the case for most forms of arts where humans are at the center, with their creative capabilities, not the machines.

References

Barrington L, Yazdani M, Turnball D, Langkriet G (2008) Combining feature kernels for semantic music retrieval. In: ISMIR, pp 614–619

Bel B (2002) Symbolic and sonic representations of sound-object structures. In: Understanding music with AI. AAAI Press

Bengio Y (2009) Learning deep architectures for ai. Found Trends Mach Learn J 2(1)

Bertin-Mahieux T, Eck D, Mande M (2010) Automatic tagging of audio: the state-of-the-art. In: Machine audition principles, algorithms and systems. IGI Publishing

Blevis EB, Jenkins MA, Glasgow JL (2000) Artificial intelligence architectures for composition and performance environment. In: Miranda ER (ed) Readings in music and AI. Routledge

Blevis EB, Jenkins MA, Glasgow JL (2002) Sources and initial design ideas for calm: a composition-analysis language for music. In: Understanding Music with AI. AAAI Press

Carpenter R (1991) Computers and musical style. Oxford University Press, Oxford

Carpenter R (1992) The logic of typed feature structures. Cambridge University Press, Cambridge

Cointe P, Rodet X (1983) Formes, a new object-language for managing a hierarchy of events. In: IRCAM report, Paris

Cook J, Morgan N (1993) In: Constructionisms I Harel (ed) Towards a construtionist musicology. Ablex publishing

Cook J, Morgan N (1995) Coleridge: composition learning environment for reflection about intentions and dialog goals. In: International congress on AI and music. Edinburgh

Garcia J, Tsandilas T, Agon C, Mackay WE (2014) Structured observation with polyphony: a multifaceted tool for studying music composition. In: DIS'14 - ACM conference on designing interactive systems, Jun 2014, Vancouver
Gratch J, Marsella S (2009) Modeling the cognitive antecedents and consequences of emotion. Cognit Syst Res 10(1):
Hamel P, Eck D (2010) Learning features from music audio with deep belief networks. In: ISMIR, pp 614–619
Hindemith P (1984) The craft of musical composition books one and two. Schott
Hofstetter FT (1988) Computer literacy for musicians. Prenctice Hall, Prenctice
Katz J, Petsetzsky D (2009) The identity thesis of language and music
Laske O (1972) On problems of performance models for music
Lerdahl E, Jackendoff R (1983) A generative theory of tonal music. MIT Press, Cambridge
Lewin D (1986) Music theory, phenomenology, and modes of perception. J Music Percept 3(4):
Mantaras RL, Arcos JL (2002) Ai and music: from composition to expressive performance. In: Understanding music with AI. AAAI Press
Marsella S, Schmidt C (1998) A problem reduction approach to automated music composition
Minsky M (1980) K-lines: a theory of memory. Cognit Sci J 4(2):
Minsky M (1981) Music, mind and meaning. Comput Music J 5(3):
Morgan RP (2014) Becoming Heinrich Schenker: music theory and ideology. Cambridge University Press, Cambridge
Narmour E (1990) The analysis and cognition on basic melodic structures: the case of automatic harmonization. Chicago University Press, Chicago
Pachet F (2012) Musical virtuosity and creativity, computers and Creativity. Springer, Berlin
Pachet F, Papadopoulos A, Roy P (2017) Sampling variations of sequences for structured music generation. In Proceedings of the 18th international society for music information retrieval conference (ISMIR), pp 23–27
Papadopoulos A, Pachet F, Roy P (2016) Generating non-plagiaristic markov sequences with max order sampling. In: Altmann E, Pachet F, Degli Esposti M (eds) Creativity and universality in language. Springer
Pascal van Hentenrick MD (1995) Forward checking in logic programming. In: ICLP4. MIT Press, Cambridge, pp 614–619
Patel AD (2003) Language music, syntax and the brain. Nat Neurosc J 6
Patel AD (2008) Music, language and the brain. Oxford University Press, Oxford
Petitjean S (2012) Describing music with metagrammars. In: Proceedings CSLP 2012, LCNS. Springer
Pynadath DV, Si M, Marsella SC (2013) Modeling theory of mind and cognitive appraisal with decision-theoretic agents. In: Gratch J, Marsella S (eds) Social emotions in nature and artifact. Prentice Hall, Prentice
Raffmann D (1993) Language, music and mind. MIT Press, Cambridge
Riecken RD (2002) Wolfgang, a system using emoting potentials to manage musical design. In: Understanding music with AI. AAAI Press
Roland PY, Ganascia JG (2000) Musical pattern extraction and similarity assessment. In: Miranda ER (ed) Readings in music and AI. Routledge
Ruwet R (1972) Langage, musique, poesie. Le Seuil, Paris
Saint-Dizier P (2014) Musical rhetoric: foundations and annotation schemes. Wiley, New York
Schenker H (1954) Harmony. Chicago University Press, Chicago
Schonberg A (2006) The musical idea and the logic, technique and art of its presentation. Indiana University Press, Bloomington
Schonberg A, Stein L, Strang G (1999) Fundamentals of musical composition. Faber and Faber
Shieber S (1986) An introduction to unification-based approaches to grammar. CSLI lecture notes, vol 4. Stanford
Si M, Marsella CM, Pynadath DV (2008) Modeling appraisal in theory of mind reasoning. In: Proceedings of IVA, Tokyo, Japan

Sloboda J (1985) The musical mind. Oxford Science Press, Oxford
Sorisio L (1987) Design of an intelligent tutoring system in harmony. In: International computer music conference, Urbana, IL
Talmy L (2001) Towards a cognitive semantics, vols 1, 2. MIT Press, Cambridge
Temperley D (2004) The cognition of basic musical structures. MIT Press, Cambridge
Thomas M (1985) Vivace: a rule-based ai system for composition. In: International computer music conference, Vancouver, BC
Turnbull D, Barrington L, Langkriet G (2008) Five approaches to collecting tags for music. In: ISMIR
van Houten K, Kasbergen M (1985) Bach and numbers. Walburg Press Zutphen
Widmer G (2000) On the potential of machine learning for music research. In: Miranda ER (ed) Readings in music and AI. Routledge

Afterword—A Note on Other Areas in Relation with AI

Pierre Marquis, Odile Papini and Henri Prade

In spite of its vast coverage ranging from fundamental issues to multiple fields in computer science and to other related disciplines, this volume about the interfaces of AI with other areas of research is certainly not complete. In particular, we may regret that due to circumstances some areas in computer science, or at the border between computer sciences and humanities, or still in the computer art domain are not covered. They include

- augmented reality and virtual reality (Langton 1995; Luck and Aylett 2000; Delaney 2008; Donikian and Petta 2011; Muratet et al. 2011; Bevacqua et al. 2017; Schmorrow and Fidopiastis 2018),
- ambient intelligence (Foresti and Ellis 2005; Cai and Abascal 2006; Rebaï et al. 2013; Streitz and Konomi 2018),
- artificial life (Magnenat-Thalmann and Thalmann 1994; Brooks 2000, 2001; Drogoul and Meyer 2000; Bersini and Reisse 2007; Lenaerts et al. 2014),
- tutoring systems and didactics (Wenger 1987; Papert 1993; Balacheff 1993; Calvo and D'Mello 2011),
- law and moral reasoning (Walton 2010; Colyvan et al. 2010; Barber and Kudenko 2010; Prakken and Sartor 2015; Bench-Capon 2017),

P. Marquis (✉)
CRIL-CNRS, Université d'Artois and Institut Universitaire de France, Lens, France
e-mail: marquis@cril.univ-artois.fr

O. Papini
Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
e-mail: odile.papini@univ-amu.fr

H. Prade
IRIT, CNRS and Université Paul Sabatier, Toulouse, France
e-mail: prade@irit.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8

- interactions between science and creativity (Boden 1994; Schmidhuber 2010), including artistic creation (beyond literature or music, covered by chapters "Artificial Intelligence and Literature" and "Music and Artificial Intelligence" in this Volume) (McCormack and d'Inverno 2012),
- painting and other visual arts, (Moos 1996; Passath and Bast 2017; McCormack and d'Inverno 2012; Colton 2008; Colton et al. 2008; Cook and Colton 2011; Gatys et al. 2016; Borillo 2010; Gufflet and Demazeau 2004.)
- aesthetical judgements (Birkhoff 1933; Galanter 2012; Bonnefon and Prade 2010).

The reader is referred to the few references given above for a beginning.

There are a number of challenging topics for AI. One of them is certainly computational humor. The understanding of jokes (Minsky 1980), of humor (Raccah 2016), the detection of irony or sarcasms (Karoui et al. 2017), the recognition of jokes (Kiddon and Brun 2011), their generation (Binsted et al. 2006; Stock and Strapparava 2006), even if they have been considered by AI researchers, are still largely open questions.

May the spirits of Democritus "The Mocker" and Aristotle "The First Teacher" inspire the future of AI!

References

Balacheff N (1993) Artificial intelligence and real teaching. In: Keitel C, KRuthven (eds) Learning from computers: mathematics education and technology, NATO ASI series book series, vol 121. Springer, pp 131–158

Barber H, Kudenko D (2010) Generation of adaptive dilemma-based interactive narratives. IEEE Trans Comput Intell AI Games 1(4):309–326

Bench-Capon TJM (2017) Hypo's legacy: introduction to the virtual special issue. Artif Intell Law 25(2):205–250

Bersini H, Reisse J (eds) (2007) Comment Définir la Vie? Les Réponses de la Biologie, de l'Intelligence Artificielle et de la Philosophie des Sciences. Vuibert

Bevacqua E, Richard R, De Loor P (2017) Believability and co-presence in human-virtual character interaction. IEEE Comput Graph Appl 37(4):17–29

Binsted K, Bergen B, Coulson S, Nijholt A, Stock O, Strapparava C, Ritchie G, Manurung R, Pain H, Waller A, O'Mara D (2006) Computational humor. IEEE Intell Syst 21(2):59–69

Birkhoff GD (1933) Aesthetic measure. Harvard University Press

Boden M (ed) (1994) Dimensions of creativity. A Bradford book. MIT Press

Bonnefon J, Prade H (2010) Qu'est-ce qui (nous) fait signe? In: Borillo M (ed) Dans l'atelier de l'art: expériences cognitives. Champ Vallon, pp 186–205

Borillo M (ed) (2010) Dans l'atelier de l'art: expériences cognitives. Champ Valon

Brooks RA (2000) Artificial life: From robot dreams to reality. Nature 406:945–947

Brooks RA (2001) The relationship between matter and life. Nature 409:409–411

Cai Y, Abascal J (eds) (2006) Ambient intelligence in everyday life. LNCS, vol 3864. Springer, Berlin

Calvo RA, D'Mello SK (eds) (2011) New perspectives on Affect and learning technologies. Springer, Berlin
Colton S (2008) Experiments in constraint-based automated scene generation. In: Gervás P, Pérez y Pérez R, Veale T (eds) Proceedings international conference on computational creativity, Madrid, pp 127–136
Colton S, Valstar MF, Pantic M (2008) Emotionally aware automated portrait painting. In: Tsekeridou S, Cheok AD, Giannakis K, Karigiannis J (eds) Proceedings of 3rd international conference on digital interactive media in entertainment and Arts, (DIMEA'08), 10–12 Sept, Athens, ACM, pp 304–311
Colyvan M, Cox D, Steele K (2010) Modelling the moral dimension of decisions. Noûs 44(3):503–529
Cook M, Colton S (2011) Automated collage generation - with more intent. In: Ventura D, Gervás P, Harrell DF, Maher ML, Pease A, Wiggins GA (eds) Proceedings of 2nd international conference on computational creativity, Mexico City, April 27–29, pp 1–3
Delaney K (ed) (2008) Ambient intelligence with microsystems. Augmented materials and smart objects. Springer, Berlin
Donikian S, Petta P (2011) A survey of research work in computer science and cognitive science dedicated to the modeling of reactive human behaviors. J Vis Comput Animat 22(5):445–455
Drogoul A, Meyer JA (eds) (2000) Intelligence Artificielle Située. Cerveau, Corps et Environnement. Hermès
Foresti GL, Ellis T (eds) (2005) Ambient intelligence - a novel paradigm. Springer, Berlin
Galanter P (2012) Computational aesthetic evaluation: past and future. In: McCormack J, d'Inverno M (eds) Computers and creativity. Springer, Berlin, pp 255–293
Gatys LA, Ecker AS, Bethge M (2016) Image style transfer using convolutional neural networks. In: The IEEE conference on computer vision and pattern recognition (CVPR), pp 2414–2423
Gufflet Y, Demazeau Y (2004) Applying the PACO paradigm to a three-dimensional artistic creation. In: Proceedings of 5th international workshop on agent-based simulation (ABS'04), Lisbon, pp 121–126
Karoui J, Benamara F, Moriceau V, Patti V, Bosco C, Aussenac-Gilles N (2017) Exploring the impact of pragmatic phenomena on Irony detection in tweets: a multilingual corpus study. In: Lapatta M, Blusom P, Koller A (eds) European chapter of the association for computational linguistics (EACL), Valence, April 3–7, association for computational linguistics (ACL), vol 1 - long papers, pp 262–272
Kiddon C, Brun Y (2011) That's what she said: Double entendre identification. In: Lapatta M, Blusom P, Koller A (eds) Proceedings of 49th annual meeting of the association for computational linguistics: human language technologies, Portland, association for computational linguistics (ACL), pp 89–94
Langton CG (ed) (1995) Artificial life: an overview. MIT Press, Cambridge, Mass
Lenaerts T, Giacobini M, Bersini H, Bourgine P, Dorigo M, Doursat R (2014) Special issue for the 20th anniversary of the European conference on artificial life (ECAL 2011). Artif Life 20(1):1–181
Luck M, Aylett R (2000) Applying artificial intelligence to virtual reality: Intelligent virtual environments. Appl Artif Intell 14(1):3–32
Magnenat-Thalmann N, Thalmann D (eds) (1994) Artificial life and virtual reality. Wiley, New York
McCormack J, d'Inverno M (eds) (2012) Computers and creativity. Springer, Berlin
Minsky M (1980) Jokes and the logic of the cognitive unconscious. In: Vaina L, Hintikka J (eds) Cognitive constraints on communication. Representations and processes, D. Reidel, Dordrecht, pp 175–200
Moos D (ed) (1996) Painting in the age of artificial intelligence (Art & Design Profile). Wiley, New York

Muratet M, Torguet P, Viallet F, Jessel JP (2011) Experimental feedback on Prog&Play: a serious game for programming practice. Comput Graph Forum 30(1):61–73

Papert S (1993) The children's machine: rethinking school in the age of the computer. Prentice-Hall

Passath N, Bast G (2017) Thinking like a machine: an artists journey into robotics. De Gruyter

Prakken H, Sartor G (2015) Law and logic: a review from an argumentation perspective. Artif Intell 227:214–245

Raccah PY (2016) Humour et métaphore: quelques éléments d'une analogie pour la construction d'un sens inattendu. illustration sur un corpus de citations de George Bernard Shaw. Revue de Sémantique et Pragmatique (39):75–94

Rebaï I, Przytula-Machrouh E, Bellik Y, Pruvost G, Sansonnet JP (2013) Influence des modalités de sortie d'un système sur les modalités de l'utilisateur. Cas des environnements ambiants de type pièces intelligentes. Tech Sci Inform 32(5):575–604

Schmidhuber J (2010) Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE Trans Auton Ment Dev 2(3):230–247

Schmorrow DD, Fidopiastis CM (eds) (2018) Proceedings of 12th international conference on augmented cognition: intelligent technologies. LNAI, vol 10915. Springer

Stock O, Strapparava C (2006) Laughing with HAHAcronym, a computational humor system. In: Proceedings of 21st national conference on artificial intelligence (AAAI'06), Boston, July 16–20, AAAI Press, pp 1675–1678

Streitz N, Konomi S (eds) (2018) Proceedings of 6th international conference on distributed, ambient and pervasive interactions: understanding humans. LNCS, vol 10921. Springer

Walton D (2010) Argumentation methods for artificial intelligence in law. Springer, Berlin

Wenger E (1987) Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge. Morgan Kaufmann Publisher, Los Altos

Epilogue: A Plea for a Unified View of Artificial Intelligence as a Science

Pierre Marquis, Odile Papini and Henri Prade

Abstract We first briefly recall the ambition of this book: providing a complete overview of the multiple lines of research that have been studied in Artificial Intelligence (AI), before surveying different views of AI that have been advocated in the last 60 years. Then we defend the idea of AI as a science and not only as a matter of innovative technology (to which often medias tend to reduce it), and we suggest that a better integration of the different views and concerns of AI may be beneficial in the long range.

1 Why this Book

AI is more than sixty years old. It has a singular position in the vast field of computer science and information processing sciences. Even though AI has never experienced so many different developments and impressive applications, its results remain largely unappreciated as a whole by the other scientific communities. This set of three volumes proposes a detailed overview of AI research that covers the knowledge representation, reasoning and learning dimensions, as well as their algorithmic side, and finally the applications and interfaces of AI with other research domains, including information processing sciences. For a long time numerous books

P. Marquis (✉)
CRIL-CNRS, Université d'Artois and Institut Universitaire de France, Lens, France
e-mail: marquis@cril.univ-artois.fr

O. Papini
Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
e-mail: odile.papini@univ-amu.fr

H. Prade
IRIT, CNRS and Université Paul Sabatier, Toulouse, France
e-mail: prade@irit.fr

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8

have been available that provide introductions and overviews of AI, including general textbooks (Winston 1977; Nilsson 1982; Aleksander et al. 1986; Charniak and McDermott 1985; Brachman and Levesque 2004; Russell and Norvig 2009; Poole and Mackworth 2010), or collections of significant papers (Webber and Nilsson 1981; Luger 1995), as well as essays (Hofstadter 1979; Dennett 1996; Minsky 2007; Lungarella et al. 2007) and historical books (see the bibliography in chapter "Elements for a History of Artificial Intelligence" of Volume 1). This book by its purpose and its contents is quite different. It proposes a comprehensive and up-to-date research overview in AI. Indeed, it has seemed essential to us to draw up an inventory of AI research at the international level. As this was a complex objective, we called upon the entire French community (which took an active part in the development of AI for decades) and beyond to achieve the goal. This book will have the merit of offering the community of AI researchers a picture of itself, and, for colleagues from other disciplines or for the funding agencies, to show where this community is located within computer science and to identify a number of borders with many other scientific areas. This is its very purpose. As far as we know, this makes this book rather unique.

Such a project would not have been addressed appropriately without the active involvement of numerous colleagues who agreed to participate in this adventure. Let each participant be heartily thanked here. This book constitutes a thoroughly revised, partly rewritten, and substantially expanded version of a similar treatise in French, published in 2014 (Marquis et al. 2014). The project originated from an idea proposed by the last of the three coordinators to the two others.

2 Different Views of AI

AI, at its beginnings, essentially developed a problem solving perspective, looking for the implementation of universal methods rather than approaches dedicated to specific problems as in operations research (see chapter "Elements for a History of Artificial Intelligence" of Volume 1). Still, the first AI programs, motivated by the will to compete with human capabilities, addressed special types of problems such as theorem proving and chess game. These concerns gave birth to the algorithmic side of AI, as exemplified in particular by ordered heuristic search techniques, solving methods for constraint satisfaction problems, and logic programming.

Early, it was noticed that problems solving efficiency was to some extent depending on the representation framework that was used. Moreover, the framework should be as general as possible because of the generality of the intended approaches. Logic is a general representation setting that has been considered from the beginning. However, it soon appeared to be too poor to take into account incomplete information, uncertainty, or specific dimensions such as time and space, for example. Indeed with the age of expert systems, the focus was on the question of explicitly representing human knowledge, which, due its imperfections, is generally different from universally true mathematical statements. This has later motivated numerous research works,

broadly echoed within this book, for developing non-classical logics able to handle exceptions or inconsistencies, or for implementing theoretical frameworks allowing the representation of different kinds of uncertainty. This has also led to awake interest in the modeling of agents' epistemic states (rather than universally true statements). Besides, the AI concern for knowledge handling is also directly related to the AI interest for making explainable conclusions drawn by inference. Thus, knowledge representation in all its various forms is a specific concern of AI.

Moreover, the interest for graphical representations, such as conceptual graphs on the logical side, or Bayesian nets on the probabilistic side has led to very important trends of research. Artificial neural networks are other popular graphical structures where the representation is embedded in the weights of the structure that are tuned by learning from data, which is quite different from concerns in human knowledge representation, and more oriented towards computation.

Indeed, there exist different views of AI (see Simon 1969; Bellman 1978; Schank 1991; Brooks 1991; Simon 1995; Pitrat 1995; Minsky 2007; Kowalski 2011; LeCun 2016) just to mention a few), often intertwined, which does facilitate a clear understanding of what is the nature of AI. It is interesting to note that in Turing's writings (1948; 1950), machine intelligence is first a computability issue, and secondarily a learning ability from data. In other words, this does not really include human knowledge representation. This view reflects much more a "black box" approach like pattern recognition or situation recognition which would correspond for humans to instinctive (reactive) activities (without having a verbalized expression as for the deliberative ones), and whose neural networks provide a prototype. It is precisely these kinds of approaches that recently put AI back into the limelight with the stunning performances obtained by massive data processing with machine learning techniques such as deep learning and reinforcement learning in relation with mathematical formalisations in the domain of probability and statistics.

This fully contrasts with the handling of human knowledge, which requires an explicit knowledge representation, suitable inference mechanisms, and in principle, explanation abilities of the derived conclusions in order to be able to communicate and justify them to the user (in plain intelligible language). Then AI research focuses on mechanizing various forms of reasoning or decision making. This latter issue may be also addressed differently by humans when decision is rather a matter of perception, of emotions than the result of a logical deliberation process (Berthoz 2003, 2006). Indeed, as emphasized by Daniel Kahneman (2011), human mind has two modes of thinking: "System 1" which is fast, instinctive and emotional, while "System 2" is slower, more deliberative, and more logical. See Raufaste (2001) for an illustration of similar ideas in the area of radiological diagnosis, where "super-experts" provide correct diagnosis, even on difficult cases, without any deliberation, while "ordinary experts" may hesitate, deliberate on the difficult cases and finally make a wrong diagnosis. Still, a "super-expert" is able to explain to an "ordinary expert" why he/she was wrong and what was important to notice in the difficult cases. The long term ambition of AI is to make machines capable of any information processing task that human mind can do. This includes both recognition, identification, decision and diagnosis tasks, which are "System 1" tasks, and "System 2" when one needs

to reason about situations stated in multiple pieces of information and to be able to explain obtained conclusions.

In addition, AI maintains fruitful exchanges with cognitive sciences (Dupuy 1994). On the one hand, it provides new benchmarks, points of comparison for the understanding of intelligence, and on the other hand AI can be inspired by what is known about the functioning of the brain and the way human beings reason, even if nothing says that AI must copy human intelligence in all its ways of proceeding (as often said, planes fly, though differently from birds!). Moreover, since machines must exchange their conclusions with users, it is important that they can express themselves in cognitively meaningful terms for the users. The great progress of neurosciences (Changeux 1997; Eccles 1989, 1994; Changeux 2012) should also have a long-term impact on AI.

Another kind of intelligence which has also kept the interest of researchers, is distributed and collective intelligence (whose importance has been underlined by philosophers Varela and Dupuy (1992)). It corresponds to the emergence (Bersini 2007) of complex properties or behaviors in dynamic systems from the local interactions between artificial agents obeying simple rules, thus bridging for example, insect society ethology, multi-agents AI and meta-heuristics (Theraulaz et al. 1998; Bonabeau et al. 1999), or robotics and phonology by the study of the self-organization of vocalisation systems and the learning of perceptual motor correspondences in a group of robots babbling together (Oudeyer 2013). Finally, this last aspect should be related to an "embodied" standpoint of intelligence (Iida et al. 2004), where agents have sensory-motor skills (whose role with regard to learning had been underlined by Piaget (1936) a long time ago), and where emotional processes (Damasio 1999) take part to the activation of intelligence as for humans.

3 AI as a Science

Already in 1981, Nils Nilsson was asking "Artificial Intelligence: engineering, science, or slogan?" (Nilsson 1981). For many people, AI reduces to innovative technological products, and this is also true to a large extent for computer science in general. Even if different views are at work, and different directions have been investigated, we hope that the three volumes of this guided tour of AI research make clear that AI is a scientific field with its own original concerns that locate it among the sciences dealing with the processing and the exploitation of information.

In spite of remarkable practical achievements, in machine learning or in solving SAT and constraint satisfaction problems for instance, AI systems have remained specialized in tasks for dedicated domains: there are no such systems that would be able of handling highly diverse recognition tasks or of combining different forms of reasoning depending on the situations at hand, as humans commonly do. It might be also worth remembering that good experimental results are often first obtained before being fully explained. AI is a young discipline which still needs more in deep research for strengthening its scientific foundations, more works for relating

its various aspects that have been developed too much in isolation from each others until now, and more investigations on its limits, in order to achieve unity and to reach full maturity. The present emphasis on quickly producing fully operational tools and effective products, even if it leads to genuine advances, also contributes to divide the research community in different subgroups that promote their own results while they largely ignore other researches that might be connected and complementary to theirs. As human intelligence has two modes of thinking, as intelligence may manifest itself in various forms, AI cannot be reduced to only one form, as recently emphasized in Darwiche (2017).

In the field of information processing sciences, AI requires a necessary clarification of its contours. This necessity is a major concern of the AI community as evidenced by the organization of a special track on the evolution of the contours of AI at the conference IJCAI 2018.[1] This book, especially Volume 3 about areas having active interfaces with AI should also contribute to make more precise the contours of AI.

The establishment of AI as a science should also help in the long range to clear up misunderstandings about AI. Indeed the use of the two words "Artificial Intelligence", has spread widely among the public as computer technology products have been taking a larger place in human activities. Young people, when playing video games, compete against opponents, which they often call "artificial intelligences". This rather singular plural might be explained by the plurality of forms that AI can take and more simply by the wide variety of applications considered. Clearly, AI is now part of our culture as evidenced by the existence of many articles, books, or movies more or less directly related to AI. AI has constantly fueled the collective imagination, as for example, with the emergence of the concept of cyborg, a creature at the interface between humans and machines that has prompted a reflection on the ethical, political and social aspects of AI science and technology.

Due to its ambiguous and somewhat scary denomination in addition to its poorly defined contours, AI has generated a great number of fantasies that recently reappeared with the actual highly publicized boom of AI. Associations have been created that set the goal of stopping any research in AI, while the development of the myth of "technological singularity"[2] is supported by many notable scientists who hypothesize that if AI continues to develop at its current dizzying rate, the singularity could come in the middle of the present century and could endanger the future of humanity. However, as rebutted by Ganascia (2017) there is no direct link between the computing power of machines and their capacity to simulate intelligence.

What is happening now is not new, scientific advances often generated irrational fear, however the community of AI scientists is aware that their research results could be used for purposes that deny some fundamental values, this is precisely the reason

[1] https://www.ijcai-18.org/cfpstecai/.

[2] Technological singularity was introduced in a science fiction novel by the mathematician Vernon Vinge (1981) who then theorized this notion in a treatise (Vinge 1993). In mathematics, a singularity is a point, value or a special case poorly defined which appears as a critical one; Vinge uses the term singularity in order to describe the phase transition to which the technology evolution could lead, due to the exponential rhythm of the increasing technological performances.

why an open letter warning of the threat of an arms race in military AI and calling for a ban on autonomous weapons was announced at the opening of the IJCAI 2015 conference in Buenos Aires, and signed by thousands AI researchers.[3] More recently, a new open letter joining AI and robotic companies from different countries addressed to the United Nations was released at the opening of the IJCAI 2017 conference in Melbourne.[4] The AI community believes that AI has great potential to benefit to humanity in many ways, and that the goal of the field should be to do so.

4 Conclusion

It is difficult to make serious predictions about what AI will be in 50 years, or even only in 20 years. History of science calls for caution in this respect because even the best AI researchers made wrong predictions in the past. For instance, Alan Turing (who can be considered as a "grandfather" of AI) was mistaken (in his paper "Computing Machinery and Intelligence", published in *Mind* in 1950, when he predicted that a thinking machine would be built in 2000 at last), or to give another example, one of the great names of AI (an organizer of the Dartmouth conference), namely Marvin Minsky was also wrong. In 1970, he predicted that within less than eight years, we will have a machine with the general intelligence of an average human being (see Walsh (2018) for more details). Clearly enough, we are still far from it today. Another example, at the time of the "boom " of expert systems, some did not hesitate to present AI as a new Eldorado (Feigenbaum and McCorduck 1983; Feigenbaum et al. 1988), although some important questions remained open regarding knowledge acquisition, while the handling of exceptions and uncertainty was not yet fully mastered.

There are still other examples of wipeouts in AI, such as the industrial failures of Fifth Generation Computer Systems (FGCS) (Shapiro 1983), of Lisp machines (Stone 1987; Bromley and Lamson 1987) and connexion machines (Hillis 1986). That mentioned, such projects were often ahead of their time in different respects, and revivals are still possible in the future (as it has been the case for neural nets). Nevertheless, the overflowing enthusiasm for AI and the broken promises have contributed to an alternance of periods of overconfidence and diffidence with respect to AI. The need to promote new paradigms, the search for institutional and industrial supports have often led to inflate promises, sometimes to a great extent. See also, for discussions along this line "The seven deadly sins of AI predictions", by Brooks (2017).

A very difficult question regarding general AI is to determine what approach to follow for reaching the multiple task intelligence of a human being. We are still far from an intelligent machine which would be able to understand all the aspects of a

[3] https://futureoflife.org/open-letter-autonomous-weapons/.

[4] https://newsroom.unsw.edu.au/news/science-tech/world's-tech-leaders-urge-un-ban-killer-robots.

situation, and moreover to have a form of "conscience" of its acts (although some researchers, e.g., Pitrat (2009) have worked actively on that). Even today, machine simulation of "common sense" reasoning is a stumbling block to AI, and this obstacle must be crossed or bypassed in order to move towards even "smarter" AI systems. As one of the French pioneers (and world leaders) of deep learning recently reminded us: "We do not have machines that have common sense. The smartest intelligent machines have less common sense than a cat"[5] (even if AI has already developed approaches to various forms of commonsense reasoning such as exception-tolerant reasoning or case-based reasoning, which have been applied to *dedicated domains*). The exploitation of knowledge is, in fact, essential to achieve various tasks in a much better way, even for those tasks where deep learning is the dominant approach to date, as in machine translation (for example, to solve anaphora).

But, conversely, considering as negligible the various advances made in AI for about sixty years would be foolish. AI research tends to make the machine capable of acquiring information, recognizing items in pictures, reasoning about a static or dynamic situation, solving huge constraint satisfaction problems, making a diagnosis, proposing a decision or an action plan, explaining and communicating the conclusions it obtains, understanding a text or a dialogue in natural language, summarizing it, discovering hidden relations in data, ... But machines until now handle such tasks separately on dedicated classes of problems, and does not master multiple abilities in the same time, being able to switch from one to another appropriately. Even if, on all these questions, great progress certainly remain to be made, particularly by developing more unified views of approaches, many results have already been obtained showing that at least to some extent this program should be feasible in the long range. AI is moving towards ever greater and more user-friendly control of information exploitation. The opportunity of large amounts of data and knowledge now available on the Web should allow one to reconsider the old research program headed by Douglas Lenat (1990) on a new basis. To do this, the machine must be equipped with methods that are generic enough to be adaptable to large classes of situations.

However, a machine with all the features mentioned above, these features having reached a very high level of efficiency, would still be far enough to possess the ability to think of a human being (even if the machine will prove to be without doubt much more successful for certain tasks than a human being). Though some tentatives exist to fill the gap, many components of human intelligence are still lacking nowadays, like the ability to create abstractions of the problems encountered (as human beings can do), to build instruments or other tools that help to solve those problems, and finally to act in good conscience (of themselves, of the others, and of the world into which they have place). In any case, the goal of AI should be to build systems and machines that are able to assist human beings in their professional or ordinary daily tasks.

---

[5]Interview of Yann LeCun with Frédérique Vidal (French Minister of Higher Education and Research) during the presentation of the Villani report *Donner un sens à l'intelligence artificielle - Pour une stratégie nationale et européenne* at the Collège de France on March 29, 2018 (see https://www.aiforhumanity.fr).

References

Aleksander I, Farreny H, Ghallab M (1986) *Decision and intelligence*. Kogan Page, London

Bellman R (1978) Artificial intelligence. Can computers think? Boyd & Fraser Publishing Company, Boston.

Bersini H (2007) Qu'est-ce-que l'Émergence ? Ellipse éditions

Berthoz A (2003) La Décision. Odile Jacob, Paris

Berthoz A (2006) Emotion and reason: the cognitive neuroscience of decision making. Oxford University Press, Oxford

Bonabeau E, Dorigo M, Theraulaz G (eds) (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, Cary, North Carolina

Brachman RJ, Levesque HJ (2004) Knowledge representation and reasoning. Elsevier

Bromley H, Lamson R (1987) LISP lore: a guide to programming the Lisp machine. Kluwer Academic Publishers, Boston

Brooks RA (1991) Intelligence without representation. Artif Intell 47(1–3):139–159

Brooks RA (2017) The seven deadly sins of AI predictions. The MIT review, the artificial intelligence issue (Nov/Dec). https://www.technologyreview.com/s/609048/the-seven-deadly-sins-of-ai-predictions/

Changeux J-P (1997) Neuronal man the biology of mind. Princeton University Press, Princeton (Revised edition)

Changeux J-P (2012) The good, the true and the beautiful a neuronal approach. Yale University Press

Charniak E, McDermott D (1985) *Introduction to artificial intelligence*. Addison-Wesley, Reading

Damasio A (1999) The feeling of what happens: body and emotion in the making of consciousness. Houghton Mifflin Harcourt, New York

Darwiche A (2017) Human-level intelligence or animal-like abilities? CoRR, arXiv:abs/1707.04327

Dennett D (1996) Kinds of minds. Toward an understanding of consciousness. Basic Books

Dupuy J-P (1994) Aux origines des sciences cognitives. Éditions La Découverte. Trad. M. B. DeBevoise. On the origins of cognitive science: the mechanization of the mind. Princeton University Press, 2000; MIT Press, 2009

Eccles JC (1989) Evolution of the brain: creation of the self. Piper Verlag, München / Zürich.

Eccles JC (1994) *How the self controls its brain*. Springer, Berlin

Feigenbaum E, McCorduck P (1983) *The fifth generation. Artificial intelligence and Japan's computer challenge to the world*. New American Library, New York; Revised paperback edition, 1984

Feigenbaum E, McCorduck P, Nii HP (1988) The rise of the expert company. How visionary companies are using artificial intelligence to achieve higher productivity and profits. Macmillan, London. Foreword by T. Peters ; Appendix: Expert Systems in Use by P. Harmon

Ganascia J-G (2017) Le mythe de la singularité. Seuil

Hillis WD (1986) The connection machine. MIT Press series in artificial intelligence. MIT Press

Hofstadter DR (1979) Gödel, escher, bach: an eternal golden braid. A metaphorical fugue on minds and machines in the spirit of Lewis Carroll. Vintage Books

Iida F, Pfeifer R, Steels L, Kuniyoshi Y, (eds) (2004) Embodied artificial intelligence. LNCS, vol 3139. Springer

Kahneman D (2011) Thinking fast and slow. Farrar, Straus and Giroux, New York

Kowalski R (2011) Computational logic and human thinking: how to be artificially intelligent. Cambridge University Press, Cambridge.

LeCun Y (2016) L'Apprentissage profond: une révolution en intelligence artificielle. Leçons inaugurales du Collège de France. http://www.college-de-france.fr/site/yann-lecun/inaugural-lecture-2016-02-04-18h00.htm.

Lenat DB, Guha RV, Pittman K, Pratt D, Shepherd M (1990) CYC: toward programs with common sense. Commun ACM 33(8):30–49
Luger GF (ed) (1995) Computation and intelligence: collected readings. AAAI Press
Lungarella M, Iida F, Bongard JC, Pfeifer R, (eds) (2007) 50 years of artificial intelligence, essays dedicated to the 50th anniversary of artificial intelligence. LNCS, vol 4850. Springer
Marquis P, Papini O, Prade H (2014) Panorama de l'intelligence artificielle. Cépaduès-Editions, Toulouse, France. Volume 1: Représentation des connaissnces et formalisation des raisonnements. Volume 2: Algorithmes pour l'intelligence artificielle, Volume 3: Frontières et applications
Minsky M (2007) The emotion machine: commonsense thinking, artificial intelligence, and the future of the human mind. Simon and Schuster
Nilsson NJ (1981) Artificial intelligence: engineering, science, or slogan? AI Mag 3(1):2–9
Nilsson NJ (1982) Principles of artificial intelligence. Springer
Oudeyer P-Y (2013) Aux sources de la parole - auto-organisation et évolution. Odile Jacob, Paris
Piaget J (1936) La Naissance de l'Intelligence chez l'Enfant. Delachaux et Niestlé
Pitrat J (1995) De la Machine à l'Intelligence. Hermès, Paris
Pitrat J (2009) Artificial beings: the conscience of a conscious machine. ISTE et Wiley
Poole DL, Mackworth AK (2010) Artificial intelligence: foundations of computational agents, 2nd ed (2017). Cambridge University Press, UK
Raufaste E (2001) Les Mécanismes Cognitifs du Diagnostic médical: optimisation et expertise. PUF, Paris
Russell S, Norvig P (2009) Artificial intelligence: a modern approach (3rd edn.). Prentice Hall; 1st Edition 1995, 2nd Edition 2003
Schank R (1991) Where is the AI. AI Mag 12(4):38–49
Shapiro EY (1983) The fifth generation project - a trip report. Commun ACM 26(9):637–641
Simon HA (1969) The sciences of the artificial. MIT Press, Cambridge; 2nd edition, 1981
Simon HA (1995) Artificial intelligence: an empirical science. Artif Intell 77(1):95–127
Stone J (1987) The AAAI-86 conference exhibits: new directions for commercial AI. AI Mag 8(1):49–54. See sections "VLSI lisp machine implementations are coming" and "A new lisp machine vendor?" in particular
Theraulaz G, Bonabeau E, Deneubourg JL (1998) The origin of nest complexity in social insects. Complexity 3:15–25
Turing AM (1948) Intelligent machinery. Technical report. Report National Physical Laboratory, London; Reprinted in: Machine intelligence, vol 5, Edinburgh University Press, pp 3–23, 1969
Turing AM (1950) Computing machinery and intelligence. Mind 59:433–460
Varela FJ, Dupuy J-P (1992) Understanding origins: contemporary views on the origin of life, mind and society. Springer
Vinge V (1981) True names. Dell Publishing
Vinge V (1993) The coming technological singularity. Vision-21: interdisciplinary science and engineering in the era of cyberspace, pp 115–126
Walsh T (2018) Machines that think. Prometheus Books
Webber BL, Nilsson NJ (eds) (1981) Readings in artificial intelligence. Tioga Pub Co.
Winston P-H (1977) Artificial intelligence. Addison-Wesley, Reading; 3rd edition, 1992

Index

P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06170-8

E

F

G

H

I

J

K

L

M

N

O

P

Q

S

T

U

V

W

Y