## UNIT-3

## CHAPTER 1

## 3.1 KNOWLEDGE INFERENCE

The object of a knowledge representation is to express knowledge in a computer tractable form, so that it can be used to enable our AI agents to perform well.

A knowledge **representation language** is defined by two aspects:

1. Syntax The syntax of a language defines which configurations of the components of the language constitute valid sentences.
2. Semantics The semantics defines which facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

This is a very general idea, and not restricted to natural language.

Suppose the language is arithmetic, then 'x', '³' and 'y' are components (or symbols or words) of the language the syntax says that 'x ³ y' is a valid sentence in the language, but '³ ³ x y' is not the semantics say that 'x ³ y' is false if y is bigger than x, and true otherwise A good knowledge representation system for any particular domain should possess the following properties:

1. Representational Adequacy – the ability to represent all the different kinds of knowledge that might be needed in that domain.
2. Inferential Adequacy – the ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.
3. Inferential Efficiency – the ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.
4. Acquisitional Efficiency – the ability to acquire new information easily. Ideally the agent should be able to control its own knowledge acquisition, but direct insertion of information by a 'knowledge engineer' would be acceptable.

In practice, the theoretical requirements for good knowledge representations can usually be achieved by dealing appropriately with a number of practical requirements:

1. The representations need to be complete – so that everything that could possibly need to be represented, can easily be represented.

2. They must be computable – implementable with standard computing procedures.

3. They should make the important objects and relations explicit and accessible – so that it is easy to see what is going on, and how the various components interact.

4. They should suppress irrelevant detail – so that rarely used details don't introduce necessary complications, but are still available when needed.

5. They should expose any natural constraints – so that it is easy to express how one object or relation influences another.

6. They should be transparent – so you can easily understand what is being said.

7. The implementation needs to be concise and fast – so that information can be stored, retrieved and manipulated rapidly.

A Knowledge representation formalism consists of collections of condition-action rules (Production Rules or Operators), a database which is modified in accordance with the rules, and a Production System Interpreter which controls the operation of the rules i.e The 'control mechanism' of a Production System, determining the order in which Production Rules are fired. A system that uses this form of knowledge representation is called a production system.

**3.2 Production Based System**

A production system consists of four basic components:

1. A set of rules of the form Ci ® Ai where Ci is the condition part and Ai is the action part. The condition determines when a given rule is applied, and the action determines what happens when it is applied.

2. One or more knowledge databases that contain whatever information is relevant for the given problem. Some parts of the database may be permanent, while others may temporary and only exist during the solution of the current problem. The information in the databases may be structured in any appropriate manner.

3. A control strategy that determines the order in which the rules are applied to the database, and provides a way of resolving any conflicts that can arise when several rules match at once.

4. A rule applier which is the computational system that implements the control strategy and applies the rules.

**Four classes of production systems:-**

1. A monotonic production system

2. A non monotonic production system

3.A partially commutative production system

4. A commutative production system.

**Advantages of production systems:-**

1. Production systems provide an excellent tool for structuring AI programs.

2. Production Systems are highly modular because the individual rules can be added, removed or modified independently.

3. The production rules are expressed in a natural form, so the statements contained in the knowledge base should the a recording of an expert thinking out loud.

**Disadvantages of Production Systems:-**

One important disadvantage is the fact that it may be very difficult to analyse the flow of control within a production system because the individual rules don't call each other.

Production systems describe the operations that can be performed in a search for a solution to the problem. They can be classified as follows.

Monotonic production system :-

A system in which the application of a rule never prevents the later application of another rule, that could have also been applied at the time the first rule was selected.

Partially commutative production system:-

A production system in which the application of a particular sequence of rules transforms state X into state Y, then any permutation of those rules that is allowable also transforms state x into state Y.

Theorem proving falls under monotonic partially communicative system. Blocks world and 8 puzzle problems like chemical analysis and synthesis come under monotonic, not partially commutative systems. Playing the game of bridge comes under non monotonic , not partially commutative system.

For any problem, several production systems exist. Some will be efficient than others. Though it may seem that there is no relationship between kinds of problems and kinds of production systems, in practice there is a definite relationship.

Partially commutative , monotonic production systems are useful for solving ignorable problems. These systems are important for man implementation standpoint because they can be implemented without the ability to backtrack to previous states, when it is discovered that an incorrect path was followed. Such systems increase the efficiency since it is not necessary to keep track of the changes made in the search process.

Monotonic partially commutative systems are useful for problems in which changes occur but can be reversed and in which the order of operation is not critical (ex: 8 puzzle problem).

Production systems that are not partially commutative are useful for many problems in which irreversible changes occur, such as chemical analysis. When dealing with such systems, the order in which operations are performed is very important and hence correct decisions have to be made at the first time itself.

### 3.3 Frame Based System

A frame is a data structure with typical knowledge about a particular object or concept. Frames, first proposed by Marvin Minsky in the 1970s.

Example : Boarding pass frames

| **QANTAS BOARDING PASS** | |
|---|---|
| ***Carrier*:** | *QANTAS AIRWAYS* |
| ***Name*:** | *MR N BLACK* |
| ***Flight*:** | *QF 612* |
| ***Date*:** | *29DEC* |
| ***Seat*:** | *23A* |
| ***From*:** | *HOBART* |

| **AIR NEW ZEALAND BOARDING PASS** | |
|---|---|
| ***Carrier*:** | *AIR NEW ZEALAND* |
| ***Name*:** | *MRS J WHITE* |
| ***Flight*:** | *NZ 0198* |
| ***Date*:** | *23NOV* |
| ***Seat*:** | *27K* |
| ***From*:** | *MELBOURNE* |

Each frame has its own name and a set of attributes associated with it. Name, weight, height and age are slots in the frame Person. Model, processor, memory and price are slots in the frame Computer. Each attribute or slot has a value attached to it.

Frames provide a natural way for the structured and concise representation of knowledge.

A frame provides a means of organising knowledge in slots to describe various attributes and characteristics of the object.

Frames are an application of object-oriented programming for expert systems.

Object-oriented programming is a programming method that uses objects as a basis for analysis, design and implementation.

In object-oriented programming, an object is defined as a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand. All objects have identity and are clearly distinguishable. Michael Black, Audi 5000 Turbo, IBM Aptiva S35 are examples of objects.

An object combines both data structure and its behaviour in a single entity. This is in sharp contrast to conventional programming, in which data structure and the program behaviour have concealed or vague connections.

When an object is created in an object-oriented programming language, we first assign a name to the object, then determine a set of attributes to describe the object's characteristics, and at last write procedures to specify the object's behaviour.

A knowledge engineer refers to an object as a frame (the term, which has become the AI jargon).

**Frames as a knowledge representation technique**

The concept of a frame is defined by a collection of slots. Each slot describes a particular attribute or operation of the frame.

Slots are used to store values. A slot may contain a default value or a pointer to another frame, a set of rules or procedure by which the slot value is obtained.

**Typical information included in a slot**

**Frame name**.

Relationship of the frame to the other frames. The frame IBM Aptiva S35 might be a member of the class Computer, which in turn might belong to the class Hardware.

Slot value. A slot value can be symbolic, numeric or Boolean. For example, the slot Name has symbolic values, and the slot Age numeric values. Slot values can be assigned when the frame is created or during a session with the expert system.

Default slot value. The default value is taken to be true when no evidence to the contrary has been found. For example, a car frame might have four wheels and a chair frame four legs as default values in the corresponding slots.

Range of the slot value. The range of the slot value determines whether a particular object complies with the stereotype requirements defined by the frame. For example, the cost of a computer might be specified between $750 and $1500.

Procedural information. A slot can have a procedure attached to it, which is executed if the slot value is changed or needed.

Most frame based expert systems use two types of methods:

WHEN CHANGED and WHEN NEEDED

A WHEN CHANGED method is executed immediately when the value of its attribute changes.

A WHEN NEEDED method is used to obtain the attribute value only when it is needed.

A WHEN NEEDED method is executed when information associated with a particular attribute is needed for solving the problem, but the attribute value is undetermined.

Most frame based expert systems allow us to use a set of rules to evaluate information contained in frames.

How does an inference engine work in a frame based system?

In a rule based system, the inference engine links the rules contained in the knowledge base with data given in the database.

When the goal is set up, the inference engine searches the knowledge base to find a rule that has th goal in its consequent.

If such a rule is found and its IF part matches data in the database, the rule is fired and the specified object, the goal, obtains its value. If no rules are found that can derive a value for the goal, the system queries the user to supply that value.

In a frame based system, the inference engine also searches for the goal. But

In a frame based system, rules play an auxiliary role. Frames represent here a major source of knowledge and both methods and demons are used to add actions to the frames.

Thus the goal in a frame based system can be established either in a method or in a demon.

Difference between methods and demons:

A demon has an IF-THEN structure. It is executed whenever an attribute in the demon's IF statement changes its value. In this sense, demons and methods are very similar and the two terms are often used as synonyms.

However, methods are more appropriate if we need to write complex procedures. Demons on the other hand, are usually limited to IF-THEN statements.

**3.3 Inference**

Two control strategies: forward chaining and backward chaining

**Forward chaining:**

Working from the facts to a conclusion. Sometimes called the datadriven approach. To chain forward, match data in working memory against 'conditions' of rules in the rule-base. When one of them fires, this is liable to produce more data. So the cycle continues

**Backward chaining:**

Working from the conclusion to the facts. Sometimes called the goal-driven approach.

To chain backward, match a goal in working memory against 'conclusions' of rules in the rule-base.

When one of them fires, this is liable to produce more goals. So the cycle continues.

The choice of strategy depends on the nature of the problem. Assume the problem is to get from facts to a goal (e.g. symptoms to a diagnosis).

Backward chaining is the best choice if:

The goal is given in the problem statement, or can sensibly be guessed at the beginning of the consultation; or:

The system has been built so that it sometimes asks for pieces of data (e.g. "please now do the gram test on the patient's blood, and tell me the result"), rather than expecting all the facts to be presented to it.

This is because (especially in the medical domain) the test may be expensive, or unpleasant, or dangerous for the human participant so one would want to avoid doing such a test unless there was a good reason for it.

Forward chaining is the best choice if:

All the facts are provided with the problem statement; or:

There are many possible goals, and a smaller number of patterns of data; or:

There isn't any sensible way to guess what the goal is at the beginning of the consultation.

Note also that a backwards-chaining system tends to produce a sequence of questions which seems focussed and logical to the user, a forward-chaining system tends to produce a sequence which seems random & unconnected.

If it is important that the system should seem to behave like a human expert, backward chaining is probably the best choice.

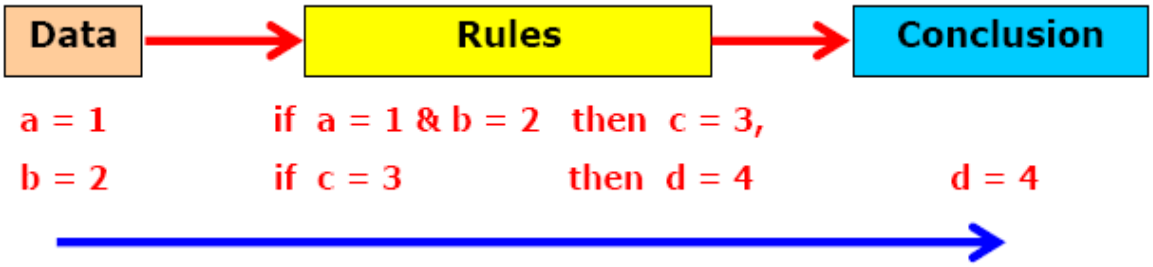
### 3.3.1 Forward Chaining Algorithm

Forward chaining is a techniques for drawing inferences from Rule base. Forward-chaining inference is often called data driven.

‡ The algorithm proceeds from a given situation to a desired goal,adding new assertions (facts) found.

‡ A forward-chaining, system compares data in the working memory against the conditions in the IF parts of the rules and determines which rule to fire.

‡ Data Driven




Example : Forward Channing

■ Given : A Rule base contains following Rule set

Rule 1: If A and C Then F

Rule 2: If A and E Then G

Rule 3: If B Then E

Rule 4: If G Then D

■ Problem : Prove

If A and B true Then D is true

Solution :

(i) Start with input given A, B is true and then

start at Rule 1 and go forward/down till a rule

“fires" is found.

First iteration :

(ii) Rule 3 fires : conclusion E is true

new knowledge found

(iii) No other rule fires;

end of first iteration.

(iv) Goal not found;

new knowledge found at (ii);

go for second iteration

Second iteration :

(v) Rule 2 fires : conclusion G is true

new knowledge found

(vi) Rule 4 fires : conclusion D is true

Goal found;

Proved

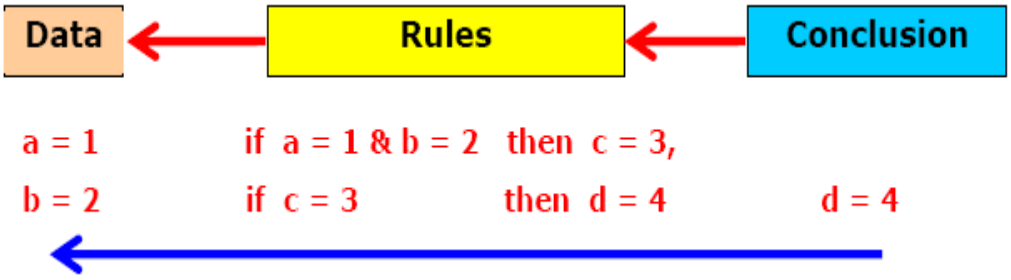
### 3.3.2 Backward Chaining Algorithm

Backward chaining is a techniques for drawing inferences from Rule base. Backward-chaining inference is often called goal driven.

‡ The algorithm proceeds from desired goal, adding new assertions found.

‡ A backward-chaining, system looks for the action in the THEN clause of the rules that matches the specified goal.

Goal Driven




Example : Backward Channing

■ Given : Rule base contains following Rule set

Rule 1: If A and C Then F

Rule 2: If A and E Then G

Rule 3: If B Then E

Rule 4: If G Then D

■ Problem : Prove

If A and B true Then D is true

Solution :

(i) Start with goal ie D is true

go backward/up till a rule "fires" is found.

First iteration :

(ii) Rule 4 fires :

new sub goal to prove G is true

go backward

(iii) Rule 2 "fires"; conclusion: A is true

new sub goal to prove E is true

go backward;

(iv) no other rule fires; end of first iteration.

new sub goal found at

(iii)go for second iteration

Second iteration :

(v) Rule 3 fires :

conclusion B is true (2nd input found)

both inputs A and B ascertained

Proved

## CHAPTER-2

### 3.4 Fuzzy Logic

Fuzzy Logic (FL) is a method of reasoning that resembles human reasoning. The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO.

The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human's YES or NO.

The inventor of fuzzy logic, Lotfi Zadeh, observed that unlike computers, the human decision making includes a range of possibilities between YES and NO, such as −

| |
|---|
| CERTAINLY YES |
| POSSIBLY YES |
| CANNOT SAY |
| POSSIBLY NO |
| CERTAINLY NO |

The fuzzy logic works on the levels of possibilities of input to achieve the definite output.

**Implementation**

- It can be implemented in systems with various sizes and capabilities ranging from small micro-controllers to large, networked, workstation-based control systems.
- It can be implemented in hardware, software, or a combination of both.

**Why Fuzzy Logic?**

Fuzzy logic is useful for commercial and practical purposes.

- It can control machines and consumer products.
- It may not give accurate reasoning, but acceptable reasoning.
- Fuzzy logic helps to deal with the uncertainty in engineering.

**Fuzzy Logic Systems Architecture**

It has four main parts as shown −

- Fuzzification Module − It transforms the system inputs, which are crisp numbers, into fuzzy sets. It splits the input signal into five steps such as −

LP x is Large Positive

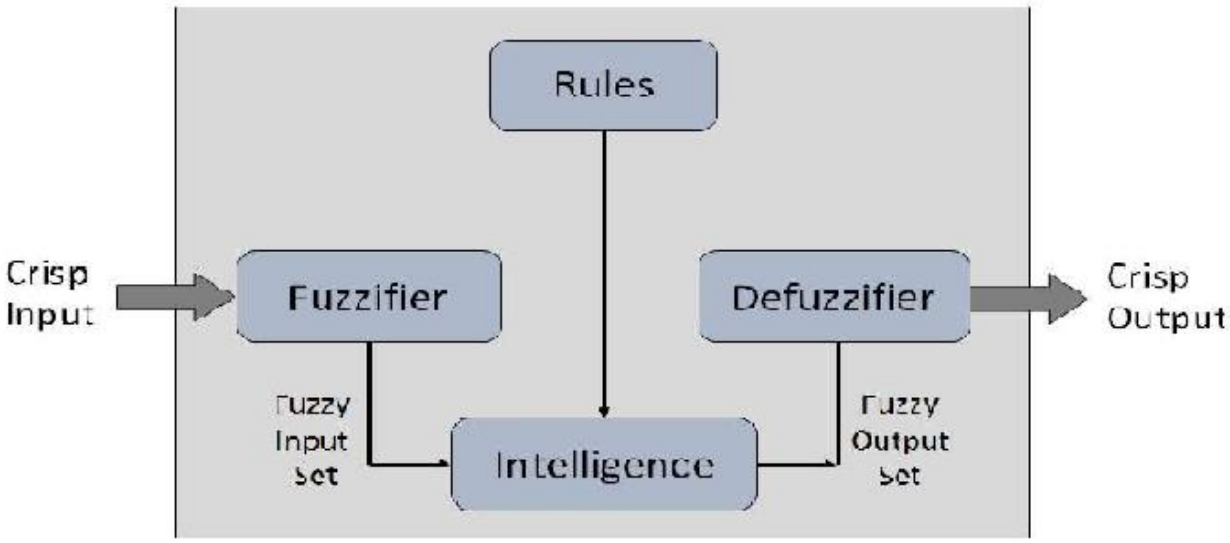
MP x is Medium Positive

S x is Small

MN x is Medium Negative

LN x is Large Negative

- Knowledge Base − It stores IF-THEN rules provided by experts.
- Inference Engine − It simulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN rules.
- Defuzzification Module − It transforms the fuzzy set obtained by the inference engine into a crisp value.



The membership functions work on fuzzy sets of variables.

Membership Function

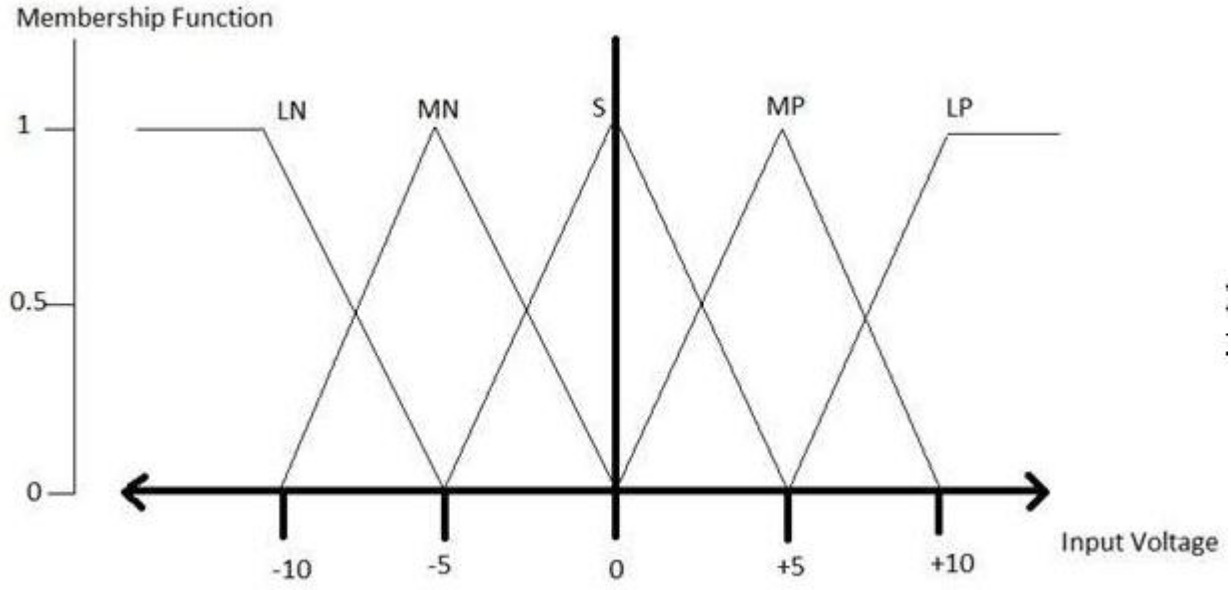
Membership functions allow you to quantify linguistic term and represent a fuzzy set graphically. A membership function for a fuzzy set A on the universe of discourse X is defined as $\mu A:X \rightarrow [0,1]$.

Here, each element of X is mapped to a value between 0 and 1. It is called membership value or degree of membership. It quantifies the degree of membership of the element in X to the fuzzy set A.

- x axis represents the universe of discourse.
- y axis represents the degrees of membership in the [0, 1] interval.

There can be multiple membership functions applicable to fuzzify a numerical value. Simple membership functions are used as use of complex functions does not add more precision in the output.

All membership functions for LP, MP, S, MN, and LN are shown as below −

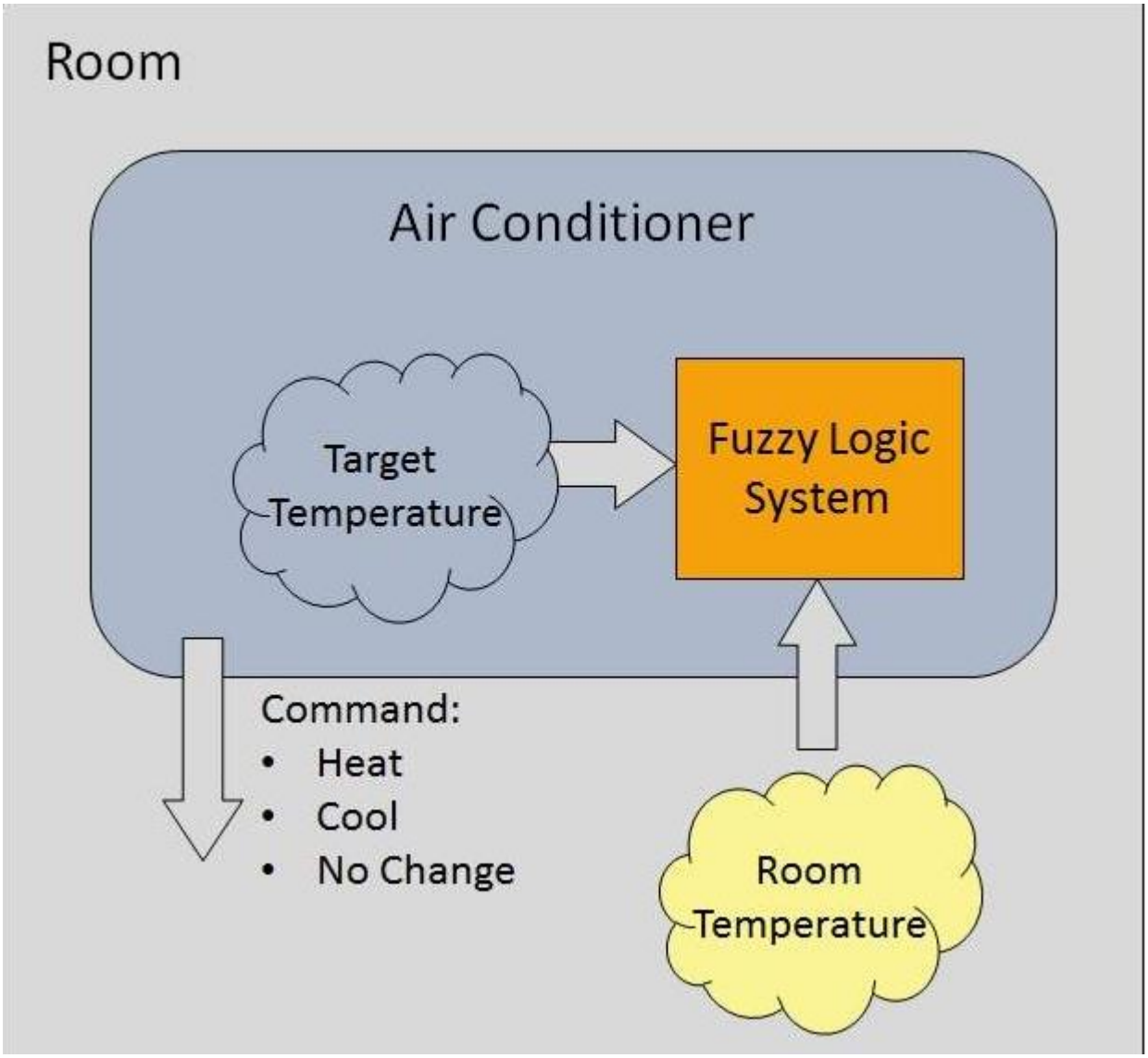



The triangular membership function shapes are most common among various other membership function shapes such as trapezoidal, singleton, and Gaussian.

Here, the input to 5-level fuzzifier varies from -10 volts to +10 volts. Hence the corresponding output also changes.

**Example of a Fuzzy Logic System**

Let us consider an air conditioning system with 5-lvel fuzzy logic system. This system adjusts the temperature of air conditioner by comparing the room temperature and the target temperature value.




**Algorithm**

- Define linguistic variables and terms.
- Construct membership functions for them.
- Construct knowledge base of rules.
- Convert crisp data into fuzzy data sets using membership functions. (fuzzification)
- Evaluate rules in the rule base. (interface engine)
- Combine results from each rule. (interface engine)
- Convert output data into non-fuzzy values. (defuzzification)

**Logic Development**

Step 1: Define linguistic variables and terms

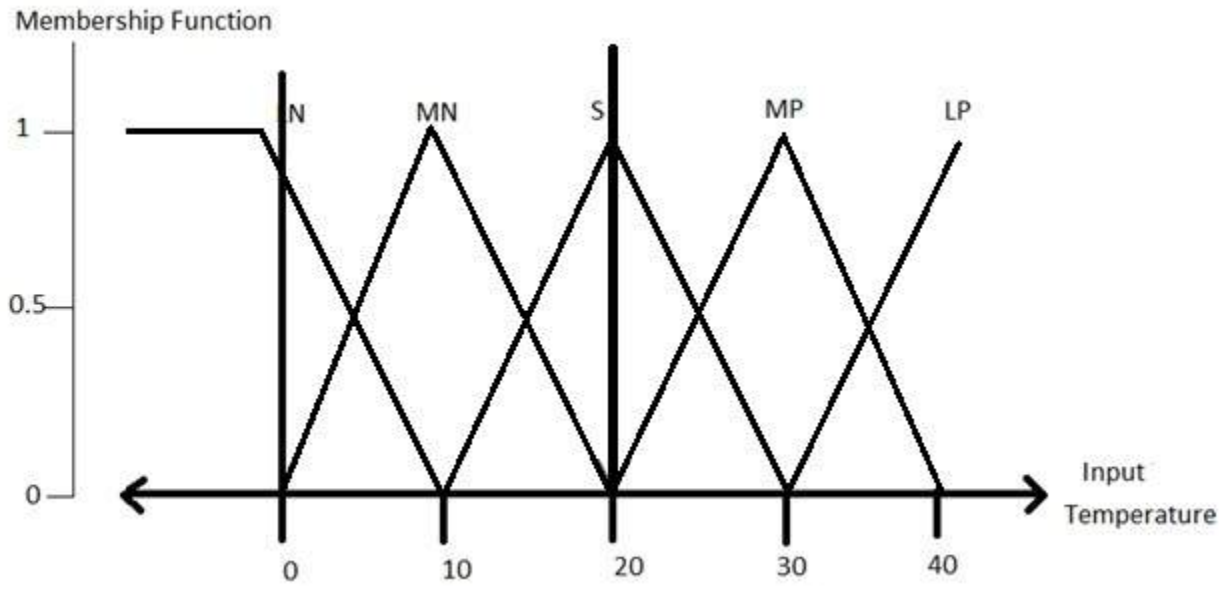
Linguistic variables are input and output variables in the form of simple words or sentences. For room temperature, cold, warm, hot, etc., are linguistic terms.

Temperature (t) = {very-cold, cold, warm, very-warm, hot}

Every member of this set is a linguistic term and it can cover some portion of overall temperature values.

Step 2: Construct membership functions for them

The membership functions of temperature variable are as shown −




Step3: Construct knowledge base rules

Create a matrix of room temperature values versus target temperature values that an air conditioning system is expected to provide.

| RoomTemp. /Target | Very_Cold | Cold | Warm | Hot | Very_Hot |
|---|---|---|---|---|---|
| Very_Cold | No_Change | Heat | Heat | Heat | Heat |
| Cold | Cool | No_Change | Heat | Heat | Heat |
| Warm | Cool | Cool | No_Change | Heat | Heat |
| Hot | Cool | Cool | Cool | No_Change | Heat |
| Very_Hot | Cool | Cool | Cool | Cool | No_Change |

Build a set of rules into the knowledge base in the form of IF-THEN-ELSE structures.

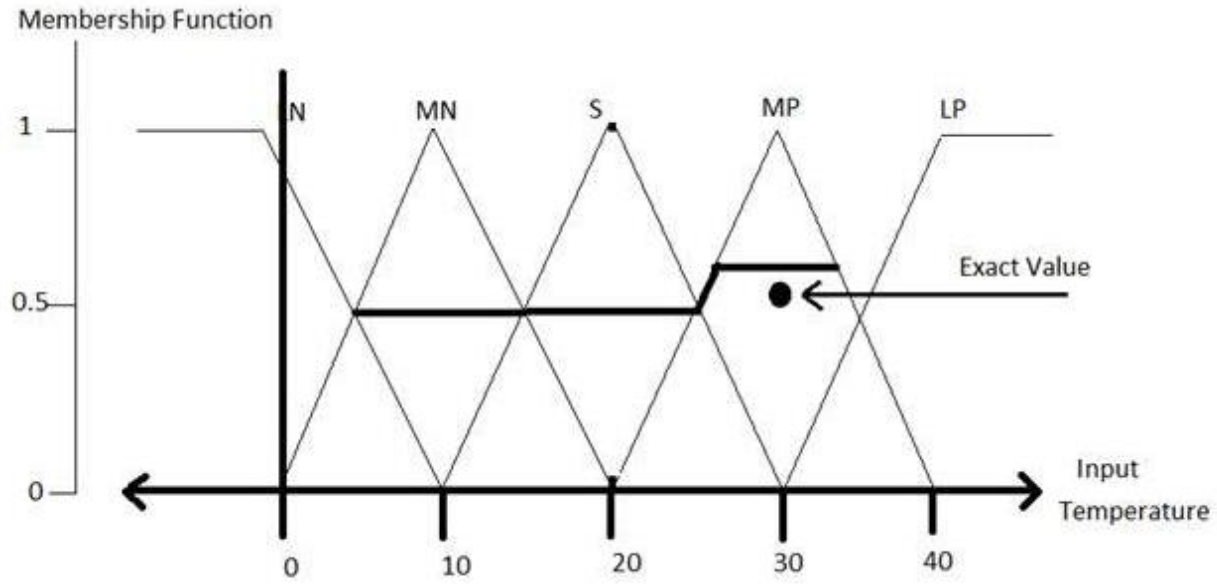
| Sr. No. | Condition | Action |
|---|---|---|
| 1 | IF temperature=(Cold OR Very_Cold) AND target=Warm THEN | Heat |
| 2 | IF temperature=(Hot OR Very_Hot) AND target=Warm THEN | Cool |
| 3 | IF (temperature=Warm) AND (target=Warm) THEN | No_Change |

Step 4: Obtain fuzzy value

Fuzzy set operations perform evaluation of rules. The operations used for OR and AND are Max and Min respectively. Combine all results of evaluation to form a final result. This result is a fuzzy value.

Step 5: Perform defuzzification

Defuzzification is then performed according to membership function for output variable.




**Application Areas of Fuzzy Logic**

The key application areas of fuzzy logic are as given −

Automotive Systems

- Automatic Gearboxes
- Four-Wheel Steering
- Vehicle environment control

Consumer Electronic Goods

- Hi-Fi Systems
- Photocopiers
- Still and Video Cameras
- Television

Domestic Goods

- Microwave Ovens
- Refrigerators
- Toasters
- Vacuum Cleaners
- Washing Machines

Environment Control

- Air Conditioners/Dryers/Heaters
- Humidifiers

Advantages of FLSs

- Mathematical concepts within fuzzy reasoning are very simple.
- You can modify a FLS by just adding or deleting rules due to flexibility of fuzzy logic.
- Fuzzy logic Systems can take imprecise, distorted, noisy input information.
- FLSs are easy to construct and understand.
- Fuzzy logic is a solution to complex problems in all fields of life, including medicine, as it resembles human reasoning and decision making.

Disadvantages of FLSs

- There is no systematic approach to fuzzy system designing.
- They are understandable only when simple.
- They are suitable for the problems which do not need high accuracy.

### 3.5 Certainty Factor

A certainty factor (CF) is a numerical value that expresses a degree of subjective belief that a particular item is true. The item may be a fact or a rule. When probabilities are used attention must be paid to the underlying assumptions and probability distributions in order to show validity. Bayes' rule can be used to combine probability measures.

Suppose that a certainty is defined to be a real number between -1.0 and +1.0, where 1.0 represents complete certainty that an item is true and -1.0 represents complete certainty that an item is false. Here a CF of 0.0 indicates that no information is available about either the truth or the falsity of an item. Hence positive values indicate a degree of belief or evidence that an item is true, and negative values indicate the opposite belief. Moreover it is common to select a positive number that represents a minimum threshold of belief in the truth of an item. For example, 0.2 is a commonly chosen threshold value.

Form of certainty factors in ES

IF <evidence>
THEN <hypothesis> {cf }

cf represents belief in hypothesis H given that evidence E has occurred

It is based on 2 functions
i) Measure of belief MB(H, E)
ii) Measure of disbelief MD(H, E)

Indicate the degree to which belief/disbelief of hypothesis H is increased if evidence E were observed

Total strength of belief and disbelief in a hypothesis:

$$cf = \frac{MB(H, E) - MD(H, E)}{1 - \min[MB(H, E), MD(H, E)]}$$

**3.6 Bayesian networks**

- Represent dependencies among random variables
- Give a short specification of conditional probability distribution
- Many random variables are conditionally independent
- Simplifies computations
- Graphical representation
- DAG – causal relationships among random variables
- Allows inferences based on the network structure

**Definition of Bayesian networks**

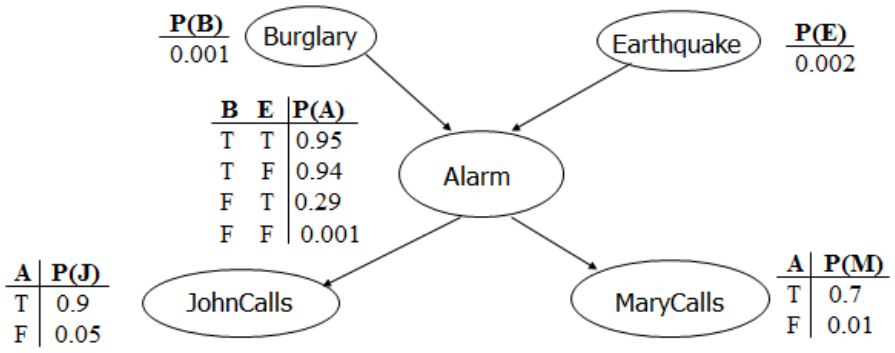
A BN is a DAG in which each node is annotated with quantitative probability information, namely:

- Nodes represent random variables (discrete or continuous)
- Directed links X→Y: X has a direct influence on Y, X is said to be a parent of Y
- each node X has an associated conditional probability table, $\mathbf{P(X_i \mid Parents(X_i))}$ that quantify the effects of the parents on the node

Example: Weather, Cavity, Toothache, Catch

- Weather, Cavity → Toothache, Cavity → Catch

**Example**




**Conditional probability table**

| B | E | P(A \| B, E) T | P(A \| B, E) F |
|---|---|---|---|
| T | T | 0.95 | 0.05 |
| T | F | 0.94 | 0.06 |
| F | T | 0.29 | 0.71 |
| F | F | 0.001 | 0.999 |

**Bayesian network semantics**

A) Represent a probability distribution

B) Specify conditional independence – build the network

A) each value of the probability distribution can be computed as:

$$\mathbf{P(X_1=x_1 \wedge \dots X_n=x_n) = P(x_1,\dots, x_n) = \Pi_{i=1,n}\, P(x_i \mid Parents(x_i))}$$

where Parents(xi) represent the specific values of Parents(Xi)

**Building the network**

$$\mathbf{P(X_1=x_1 \wedge \dots X_n=x_n) = P(x_1,\dots, x_n) =}$$

$$\mathbf{P(x_n \mid x_{n-1},\dots, x_1) * P(x_{n-1},\dots, x_1) = \dots =}$$

$$\mathbf{P(x_n \mid x_{n-1},\dots, x_1) * P(x_{n-1} \mid x_{n-2},\dots, x_1) * \dots P(x_2|x_1) * P(x_1) = \Pi_{i=1,n}\, P(x_i \mid x_{i-1},\dots, x_1)}$$

- We can see that $\mathbf{P(X_i \mid X_{i-1},\ldots, X_1) = P(x_i \mid Parents(X_i))}$ if $\mathbf{Parents(X_i) \subseteq \{ X_{i-1},\ldots, X_1\}}$
- The condition may be satisfied by labeling the nodes in an order consistent with a DAG
- Intuitively, the parents of a node $X_i$ must be all the nodes $X_{i-1},\ldots, X_1$ which have a direct influence on $X_i$.
- Pick a set of random variables that describe the problem
- Pick an ordering of those variables
- **while** there are still variables **repeat**

  (a) choose a variable $X_i$ and add a node associated to $X_i$

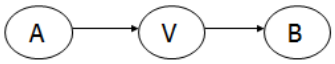
  (b) assign Parents($X_i$) $\leftarrow$ a minimal set of nodes that already exists in the network such that the conditional independence property is satisfied

  (c) define the conditional probability table for $X_i$
- Because each node is linked only to previous nodes $\rightarrow$ DAG
- P(MaryCalls | JohnCals, Alarm, Burglary, Earthquake) = P(MaryCalls | Alarm)

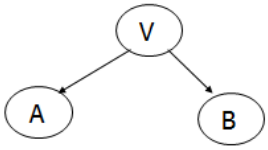
**Compactness of node ordering**

- Far more compact than a probability distribution
- Example of **locally structured system** (or ***sparse***): each component interacts directly only with a limited number of other components
- Associated usually with a linear growth in complexity rather than with an exponential one
- *The order of adding the nodes is important*
- The correct order in which to add nodes is to add the "root causes" first, then the variables they influence, and so on, until we reach the leaves

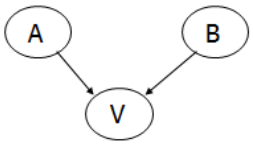
**Probabilistic Interfaces**



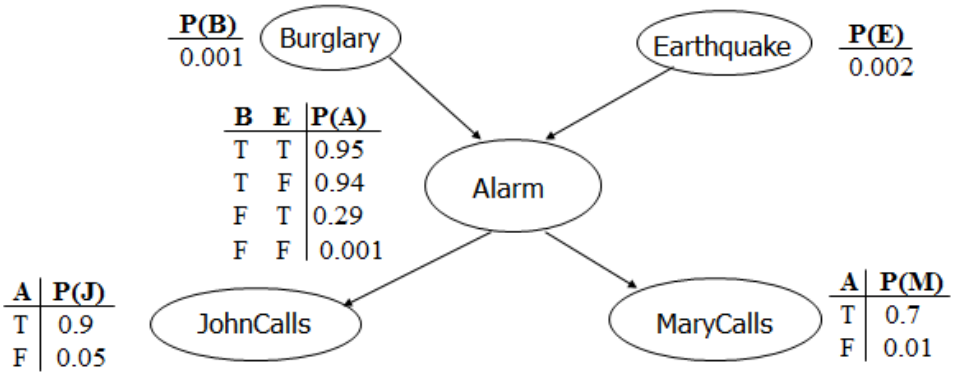

$P(A \wedge V \wedge B) = P(A) * P(V|A) * P(B|V)$




$P(A \wedge V \wedge B) = P(V) * P(A|V) * P(B|V)$




$P(A \wedge V \wedge B) = P(A) * P(B) * P(V|A,B)$




$P(J \wedge M \wedge A \wedge \sim B \wedge \sim E) =$

$P(J|A) * P(M|A) * P(A|\sim B \wedge \sim E) * P(\sim B) \wedge P(\sim E) = 0.9 * 0.7 * 0.001 * 0.999 * 0.998 = 0.00062$

$P(A|B) = P(A|B,E) * P(E|B) + P(A|B,\sim E) * P(\sim E|B) = P(A|B,E) * P(E) + P(A|B,\sim E) * P(\sim E)$

$= 0.95 * 0.002 + 0.94 * 0.998 = 0.94002$

**Different types of inferences**

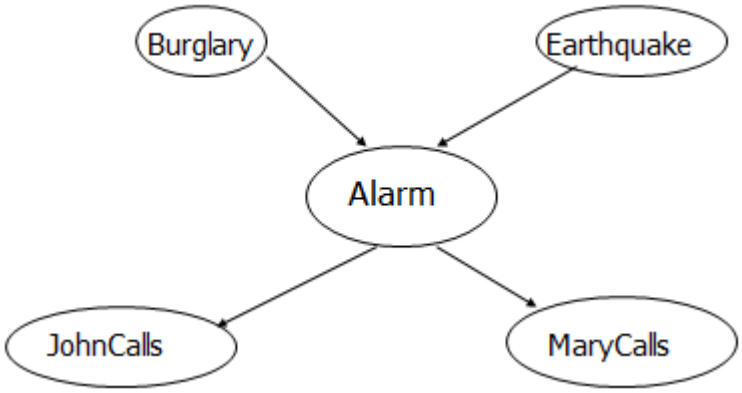
**Diagnosis inferences** (effect → cause)

P(Burglary | JohnCalls)

**Causal inferences** (cause → effect)

P(JohnCalls |Burglary),

P(MaryCalls | Burgalry)




**Intercausal inferences (between cause and common effects)**

P(Burglary | Alarm ∧Earthquake)

**Mixed inferences**

P(Alarm | JohnCalls ∧ ~Earthquake) → diag + causal

P(Burglary | JohnCalls ∧ ~ Earthquake) → diag + intercausal

### 3.7 Dempster-Shafer Theory

- Dempster-Shafer theory is an approach to combining evidence
- Dempster (1967) developed means for combining degrees of belief derived from independent items of evidence.
- His student, Glenn Shafer (1976), developed method for obtaining degrees of belief for one question from subjective probabilities for a related question

- People working in Expert Systems in the 1980s saw their approach as ideally suitable for such systems.
- Each fact has a degree of support, between 0 and 1:
  - ✓ 0 No support for the fact
  - ✓ 1 full support for the fact
- Differs from Bayesian approah in that:
  - ✓ Belief in a fact and its negation need not sum to 1.
  - ✓ Both values can be 0 (meaning no evidence for or against the fact)

**Set of possible conclusions**: Θ

Θ = { θ1 , θ2 , …, θn }

Where:

- ✓ Θ is the set of possible conclusions to be drawn
- ✓ Each θi is mutually exclusive: at most one has to be true.
- ✓ Θ is Exhaustive: At least one θi has to be true.

**Frame of discernment**

Θ = { θ1 , θ2 , …, θn }

- Bayes was concerned with evidence that supported single conclusions (e.g., evidence for each outcome θi in Θ):
- p(θi| E)
- D-S Theoryis concerned with evidences which support
- subsets of outcomes in Θ, e.g., θ1 v θ2 v θ3 or {θ1, θ2, θ3}
- The "frame of discernment" (or "Power set") of Θ is the set of all possible subsets of Θ:

  E.g., if Θ = {θ1, θ2, θ3}

- Then the frame of discernment of Θ is:

  ( Ø, θ1, θ2, θ3, {θ1, θ2}, {θ1, θ3}, {θ2, θ3}, { θ1, θ2, θ3} )

- Ø, the empty set, has a probability of 0, since one of the outcomes has to be true.
- Each of the other elements in the power set has a probability between 0 and 1.
- The probability of {θ1, θ2, θ3} is 1.0 since one has to be true.

**Mass function m(A):**

- (where A is a member of the power set) = proportion of all evidence that supports this element of the power set.
- "The mass m(A) of a given member of the power set, A, expresses the proportion of all relevant and available evidence that supports the claim that the actual state belongs to A but to no particular subset of A."
- "The value of m(A) pertains only to the set A and makes no additional claims about any subsets of A, each of which has, by definition, its own mass.
- Each m(A) is between 0 and 1.
- All m(A) sum to 1.
- m(Ø) is 0 - at least one must be true.

  Interpretation of m({AvB})=0.3

- Means there is evidence for {AvB} that cannot be divided among more specific beliefs for A or B.

  **Example**

  - 4 people (B, J, S and K) are locked in a room when the lights go out.
  - When the lights come on, K is dead, stabbed with a knife.
  - Not suicide (stabbed in the back)
  - No-one entered the room.
  - Assume only one killer.

- Θ = { B, J, S}
- P(Θ) = (Ø, {B}, {J}, {S}, {B,J}, {B,S}, {J,S}, {B,J,S} )
- Detectives, after reviewing the crime-scene, assign mass probabilities to various elements of the power set:

| Event | Mass |
|---|---|
| No-one is guilty | 0 |
| B is guilty | 0.1 |
| J is guilty | 0.2 |
| S is guilty | 0.1 |
| either B or J is guilty | 0.1 |
| either B or S is guilty | 0.1 |
| either S or J is guilty | 0.3 |
| One of the 3 is guilty | 0.1 |

**Belief in A:**

The belief in an element A of the Power set is the sum of the masses of elements which are subsets of A (including A itself).

E.g., given A={q1, q2, q3}

Bel(A) = m(q1)+m(q2)+m(q3)+ m({q1, q2})+m({q2, q3})+m({q1, q3})+m({q1, q2, q3})

**Example**

- Given the mass assignments as assigned by the detectives:

| A | {B} | {J} | {S} | {B,J} | {B,S} | {J,S} | {B,J,S} |
|---|---|---|---|---|---|---|---|
| m(A) | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.1 |

- bel({B}) = m({B}) = 0.1
- bel({B,J}) = m({B})+m({J})+m({B,J}) =0.1+0.2+0.1=0.4
- Result:

| A | {B} | {J} | {S} | {B,J} | {B,S} | {J,S} | {B,J,S} |
|---|---|---|---|---|---|---|---|
| m(A) | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.1 |
| bel(A) | 0.1 | 0.2 | 0.1 | 0.4 | 0.3 | 0.6 | 1.0 |

**Plausibility of A: pl(A)**

The plausability of an element A, pl(A), is the sum of all the masses of the sets that intersect with the set A:

E.g. pl({B,J}) = m(B)+m(J)+m(B,J)+m(B,S) +m(J,S)+m(B,J,S) = 0.9

**All Results:**

| A | {B} | {J} | {S} | {B,J} | {B,S} | {J,S} | {B,J,S} |
|---|---|---|---|---|---|---|---|
| m(A) | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.1 |
| pl(A) | 0.4 | 0.7 | 0.6 | 0.9 | 0.8 | 0.9 | 1.0 |

**Disbelief (or Doubt) in A: dis(A)**

The disbelief in A is simply bel(¬A).

It is calculated by summing all masses of elements which do not intersect with A.

The plausibility of A is thus 1-dis(A):

pl(A) = 1- dis(A)

| A | {B} | {J} | {S} | {B,J} | {B,S} | {J,S} | {B,J,S} |
|---|---|---|---|---|---|---|---|
| m(A) | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.1 |
| dis(A) | 0.6 | 0.3 | 0.4 | 0.1 | 0.2 | 0.1 | 0 |
| pl(A) | 0.4 | 0.7 | 0.6 | 0.9 | 0.8 | 0.9 | 1.0 |

**Belief Interval of A:**

The certainty associated with a given subset A is defined by the belief interval:

[ bel(A) pl(A) ]

E.g. the belief interval of {B,S} is: [0.1 0.8]

| A | {B} | {J} | {S} | {B,J} | {B,S} | {J,S} | {B,J,S} |
|---|---|---|---|---|---|---|---|
| m(A) | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.1 |
| bel(A) | 0.1 | 0.2 | 0.1 | 0.4 | 0.3 | 0.6 | 1.0 |
| pl(A) | 0.4 | 0.7 | 0.6 | 0.9 | 0.8 | 0.9 | 1.0 |

**Belief Intervals & Probability**

The probability in A falls somewhere between bel(A) and pl(A).

- bel(A) represents the evidence we have for A directly So prob(A) cannot be less than this value.
- pl(A) represents the maximum share of the evidence we could possibly have, if, for all sets that intersect with A, the part that intersects is actually valid. So pl(A) is the maximum possible value of prob(A).

| A | {B} | {J} | {S} | {B,J} | {B,S} | {J,S} | {B,J,S} |
|---|---|---|---|---|---|---|---|
| m(A) | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.1 |
| bel(A) | 0.1 | 0.2 | 0.1 | 0.4 | 0.3 | 0.6 | 1.0 |
| pl(A) | 0.4 | 0.7 | 0.6 | 0.9 | 0.8 | 0.9 | 1.0 |

**Belief Intervals:**

Belief intervals allow Demspter-Shafer theory to reason about the degree of certainty or certainty of our beliefs.

- A small difference between belief and plausibility shows that we are certain about our belief.
- A large difference shows that we are uncertain about our belief.

However, even with a 0 interval, this does not mean we know which conclusion is right. Just how probable it is!

| A | {B} | {J} | {S} | {B,J} | {B,S} | {J,S} | {B,J,S} |
|---|---|---|---|---|---|---|---|
| m(A) | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.1 |
| bel(A) | 0.1 | 0.2 | 0.1 | 0.4 | 0.3 | 0.6 | 1.0 |
| pl(A) | 0.4 | 0.7 | 0.6 | 0.9 | 0.8 | 0.9 | 1.0 |

PART – A

1. Define NMR

2. Define Justifications

3. What is non monotonic inference?

4. Difference between JTMS and LTMS

5. Define Bayes theorem.

6. What do you mean by Rule based system?

7. Define fuzzy logic.

8. What is credit assignment problem?

9. Define Frame problems.

14. What do you understand by Default reasoning.

15. Define frames.

16. What are singular extensions?

17. What is a Bayesian network?

18. Define dumpster Shafer theory.

PART-B

1. Distinguish between

    a. Production Based System.

    b. Frame Based System

2. What is uncertainty? How to reason out in each situation? What are the various strategies under such case?

3. Write a note on a. Fuzzy reasoning b. Bayesian probability c. Certainty factors

4. What is certainty factor? Compute certainty factor based on hypothesis.

5. How does inference engine work in a frame based system.

6. Explain Bayesian Network.