Chapter 6
Turing Machines

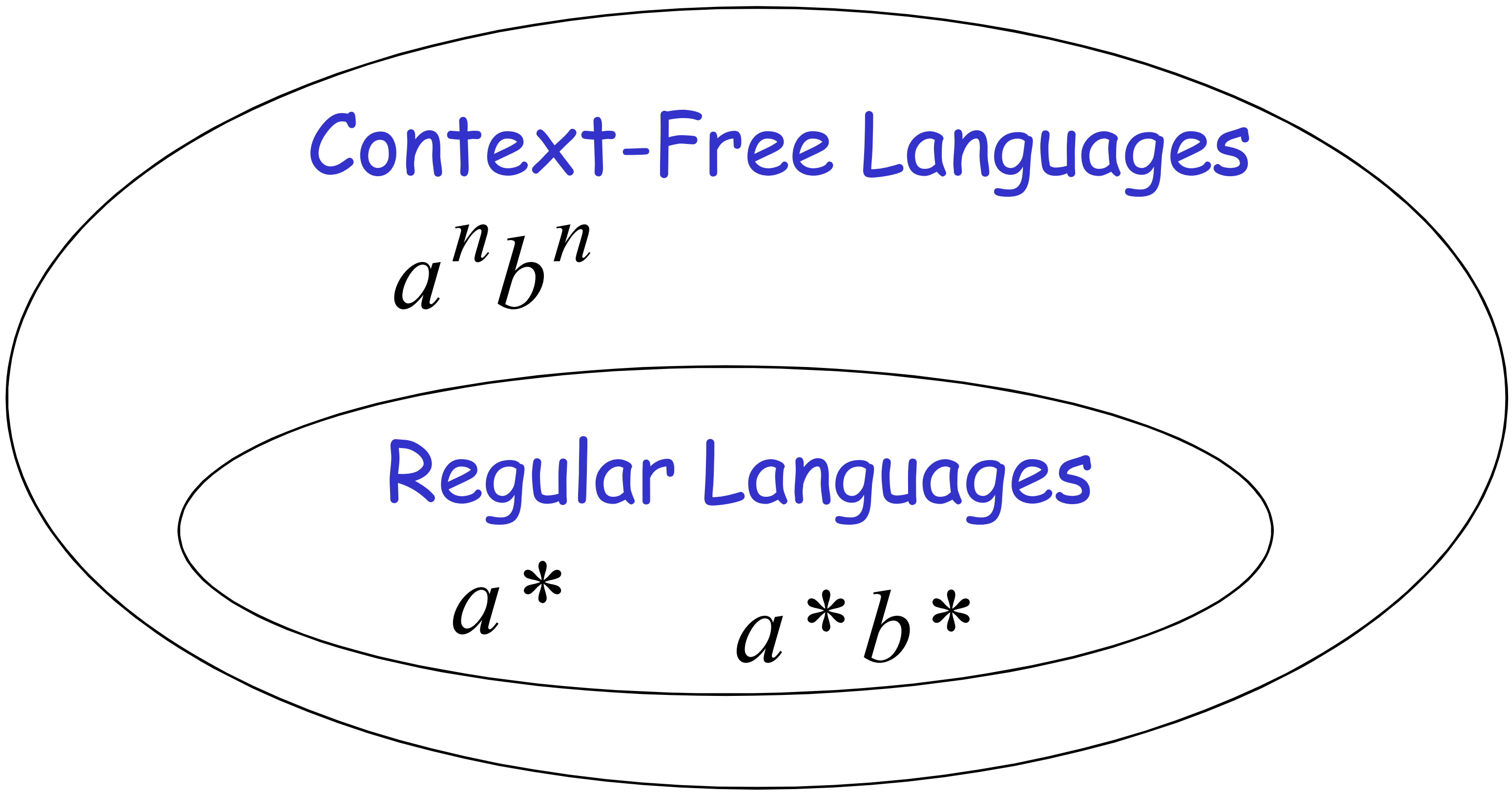
Turing Machines

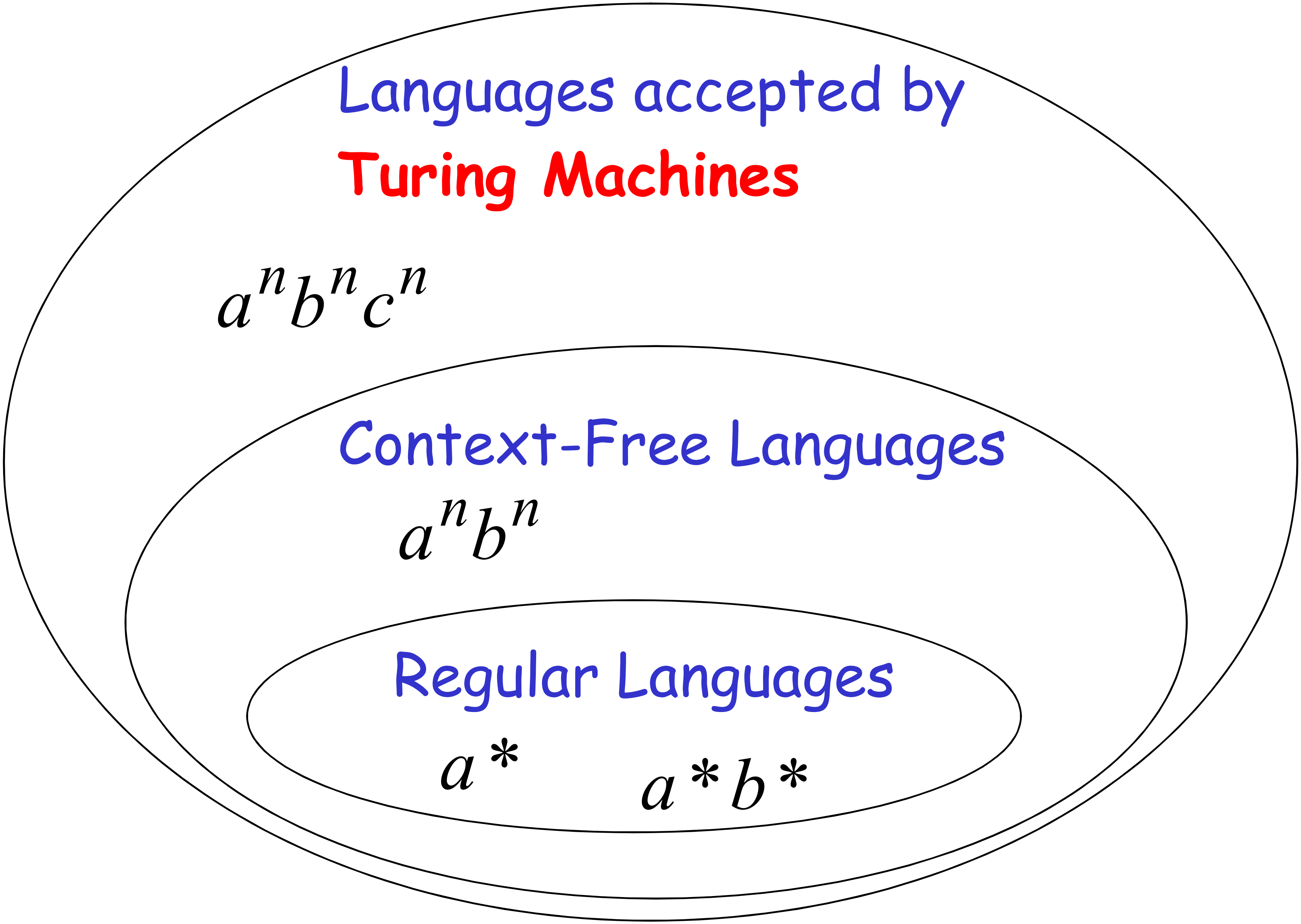
- Turing machines (TMs) were introduced by Alan Turing in 1936

- They are more powerful than both finite automata and pushdown automata. In fact, they are as powerful as any computer we have ever built.

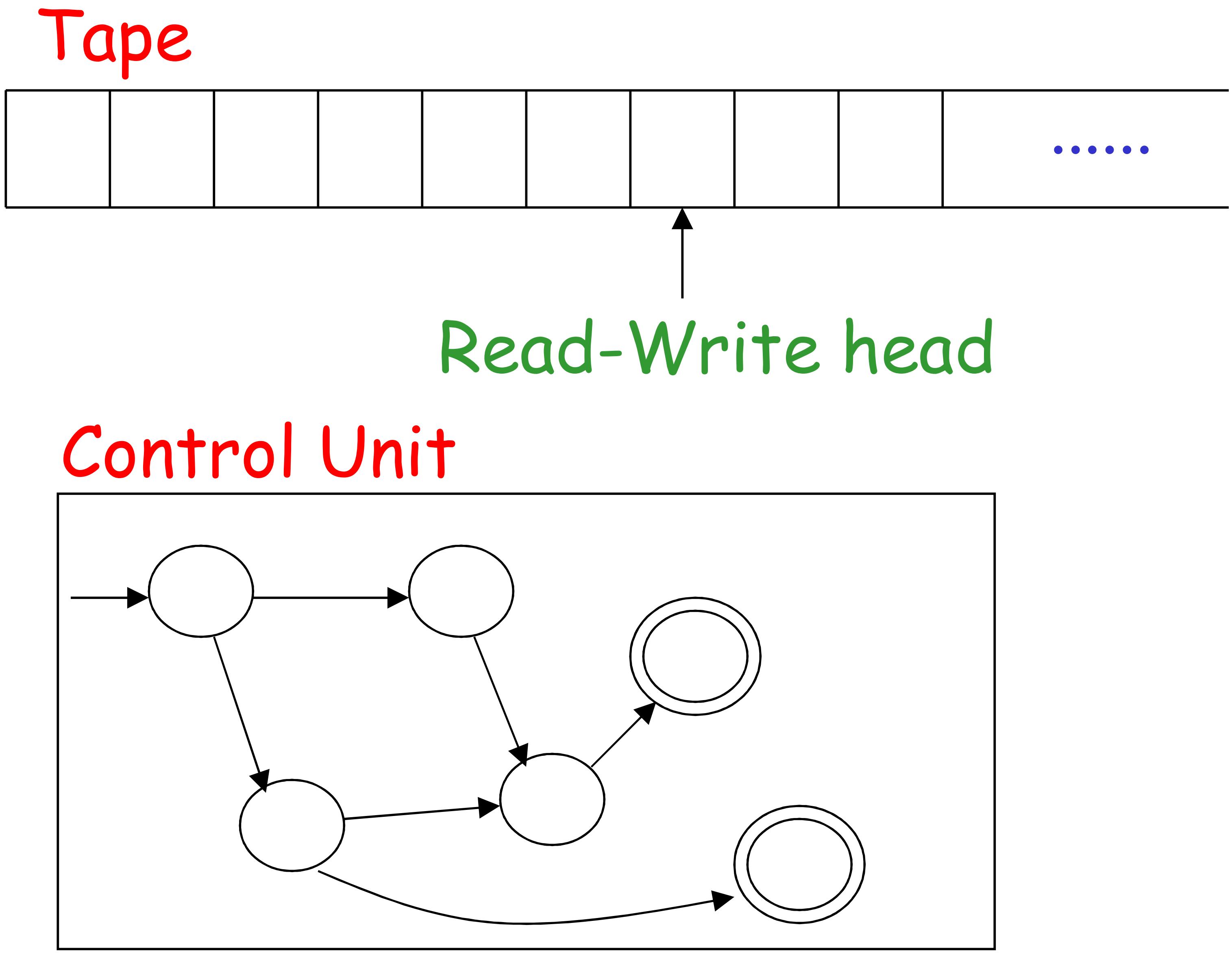
The Language Hierarchy

$a^n b^n c^n$?



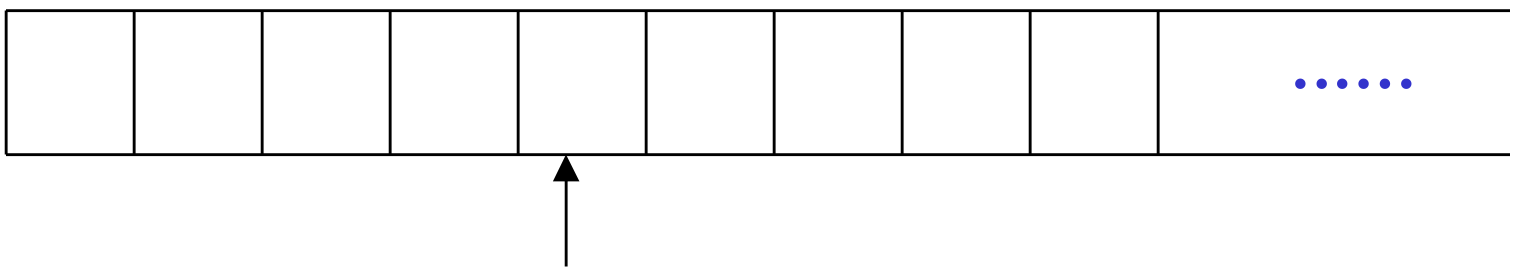

Languages accepted by
Turing Machines
$a^n b^n c^n$
Context-Free Languages
$a^n b^n$
Regular Languages
$a*$
$a*b*$

Basic design of Turing machine



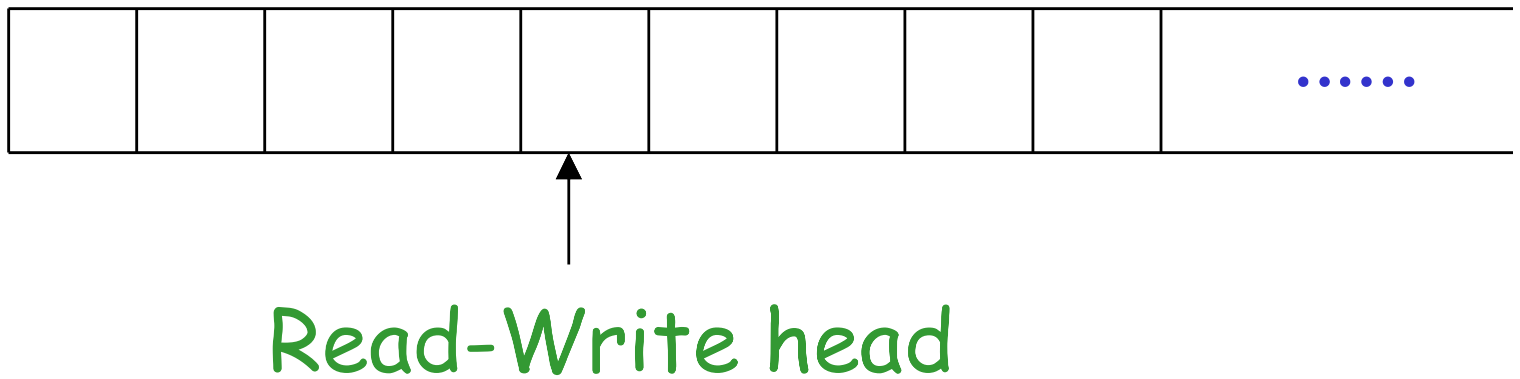
The Tape

No boundaries -- infinite length



Read-Write head

The head moves Left or Right



The head at each time step:

1. Reads a symbol
2. Writes a symbol
3. Moves Left or Right

TM instructions (transition functions)

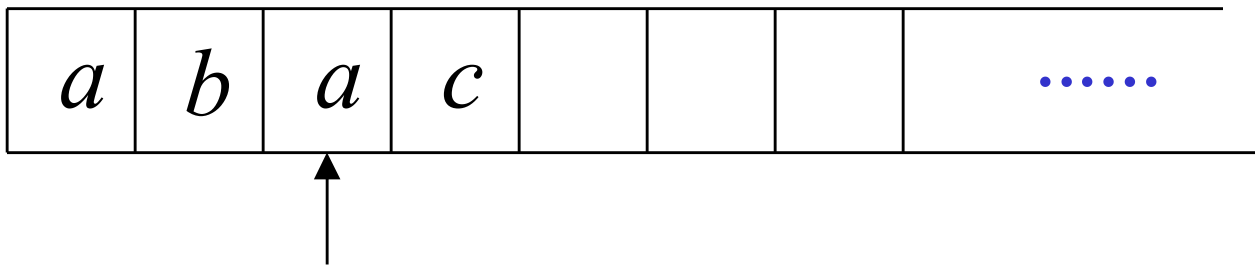
Each Turing machine instruction contains the following five parts:

- The **current** machine state.
- A tape symbol **read** from the current tape cell.
- A tape symbol to **write** into the current tape cell.
- The **next** machine state.
- A **direction** for the tape head to move.

Two inputs, three outputs: $T(i, a) = (b, j, R)$

Example:

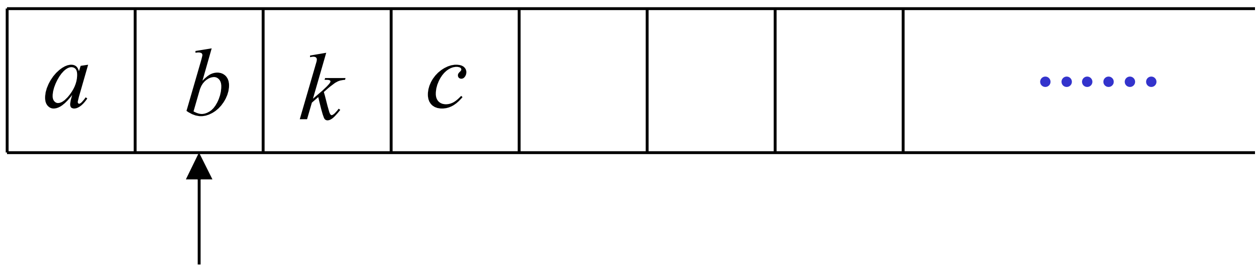
Time 0



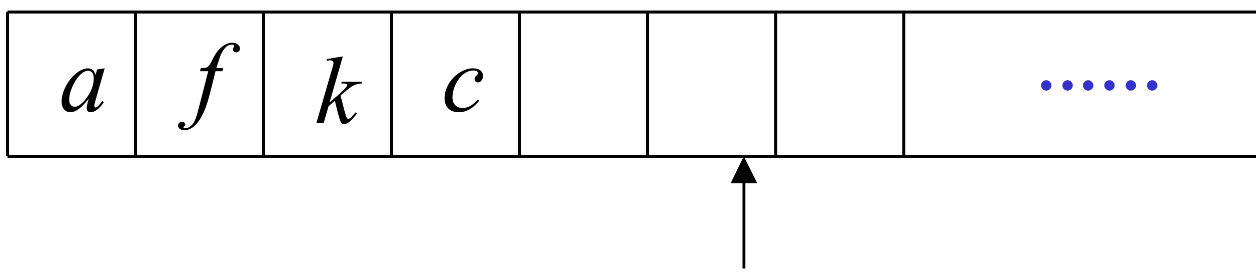
Time 1

1. Reads a
2. Writes k
3. Moves Left

Time 1

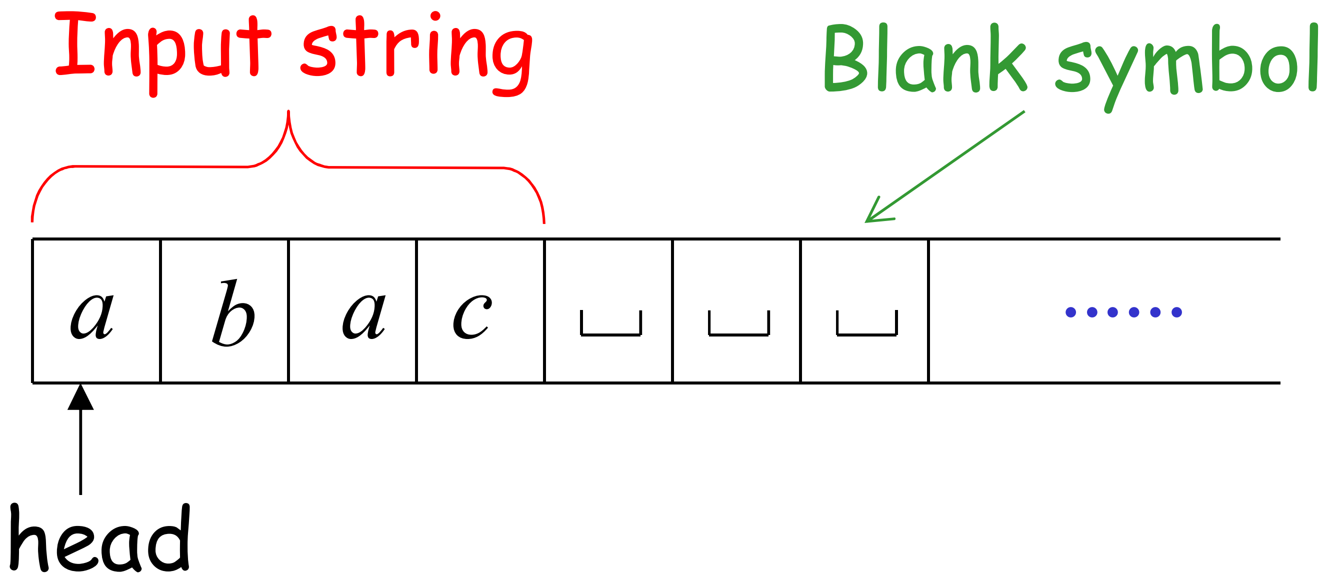


Time 2

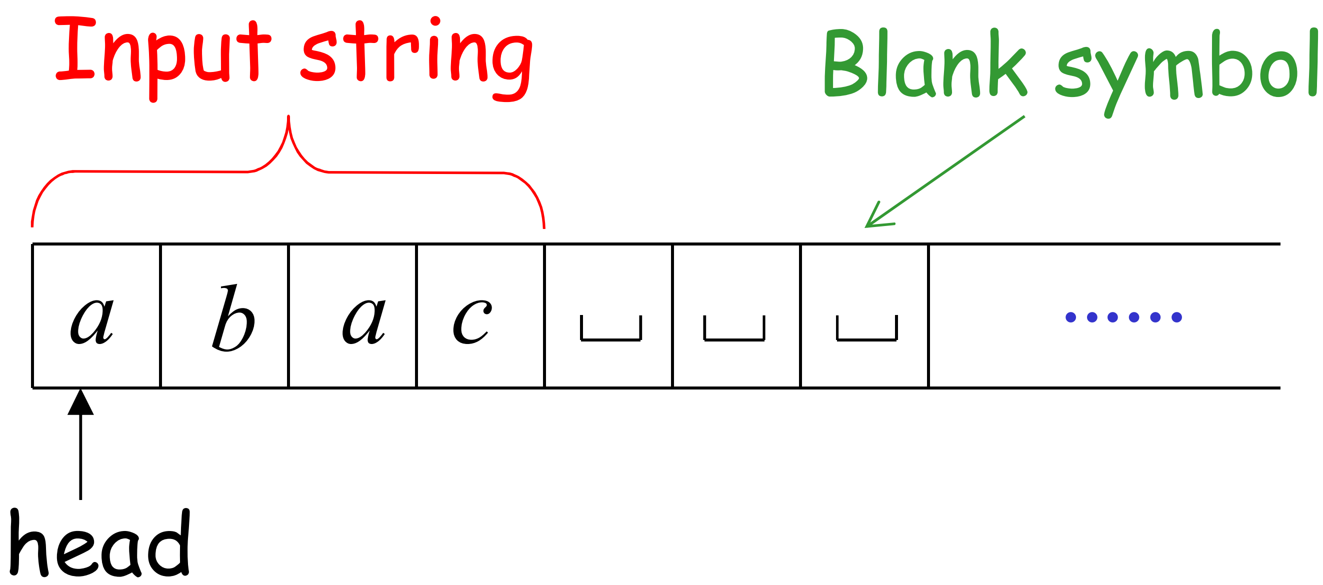


1. Reads b
2. Writes f
3. Moves Right

The Input String

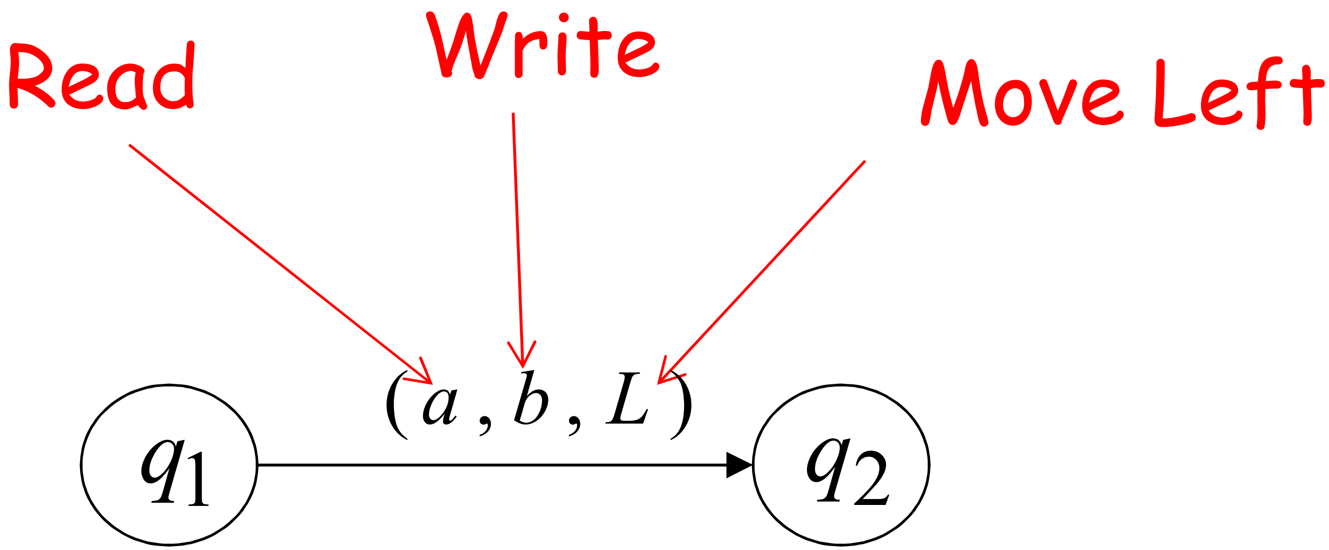


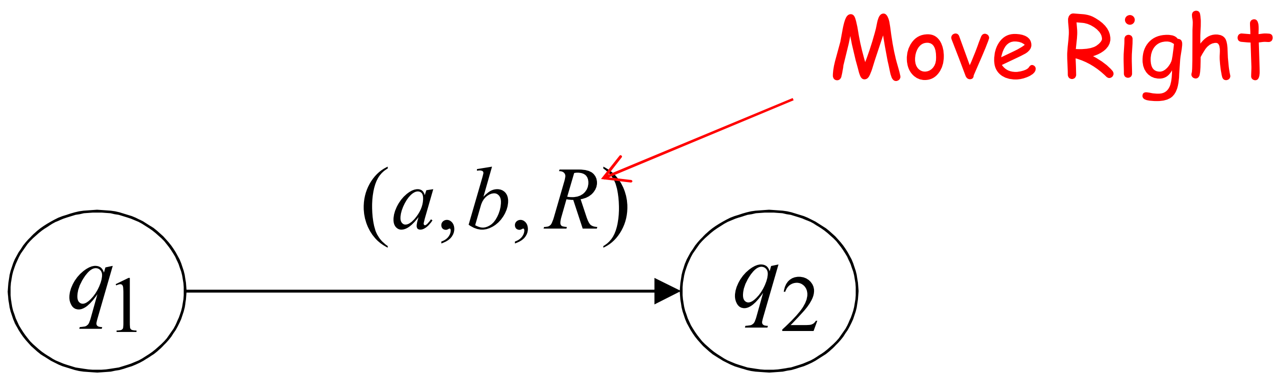
Head starts at the leftmost position of the input string



Remark: the input string is never empty

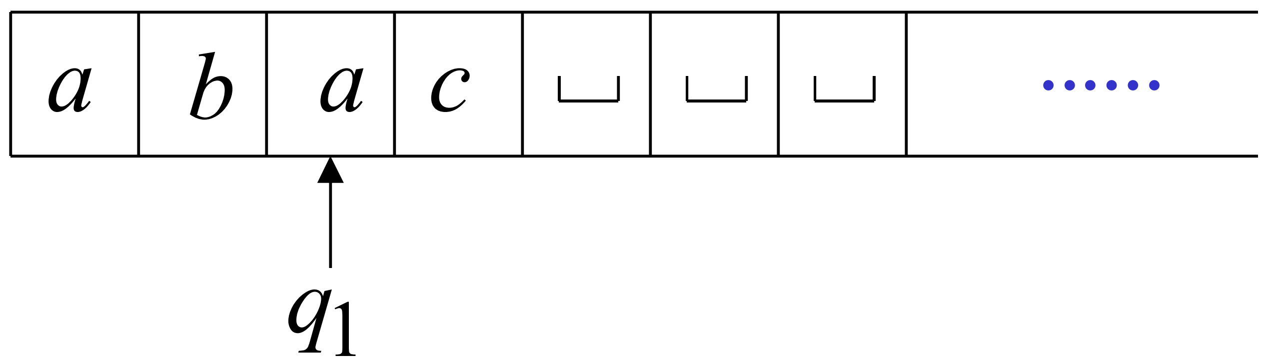
States & Transitions



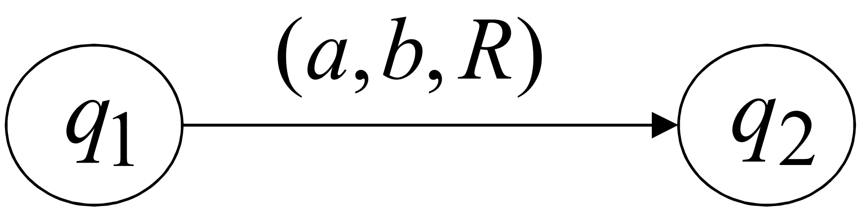


Example:

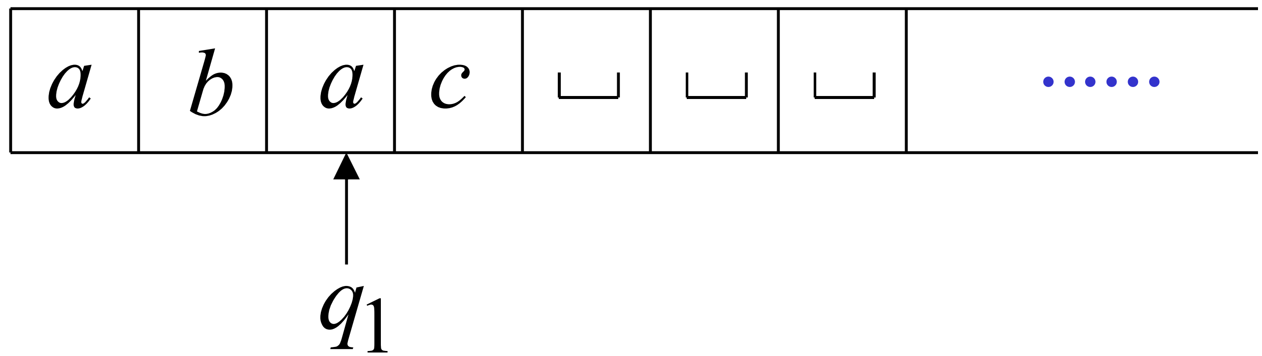
Time 1



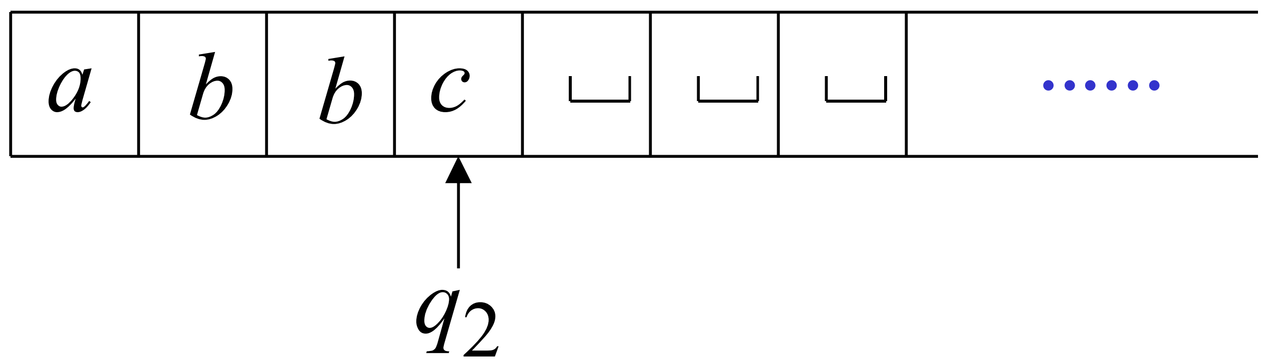
current state

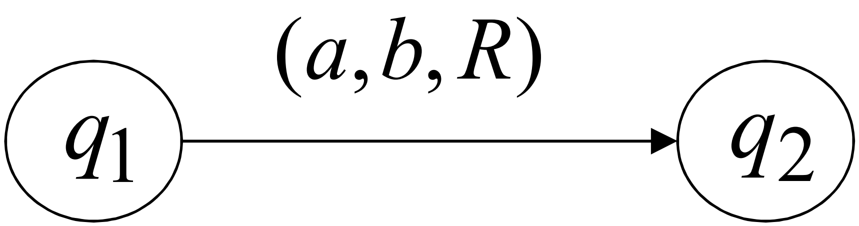


Time 1



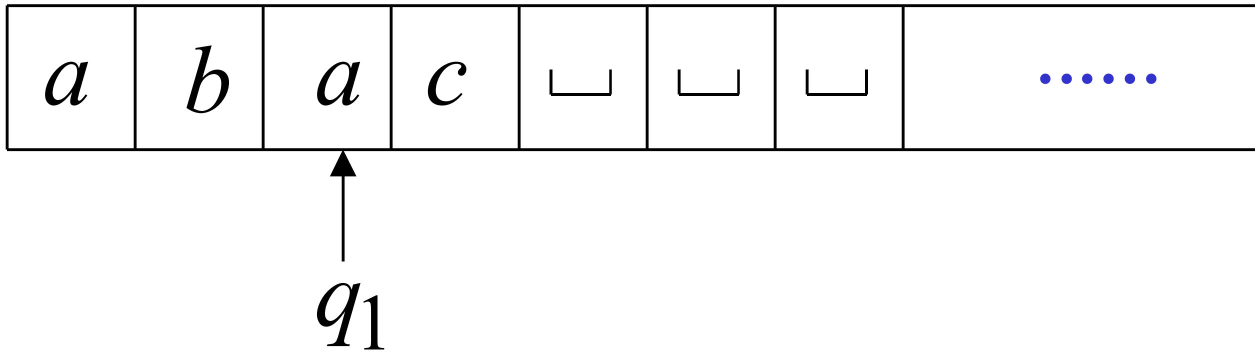
Time 2



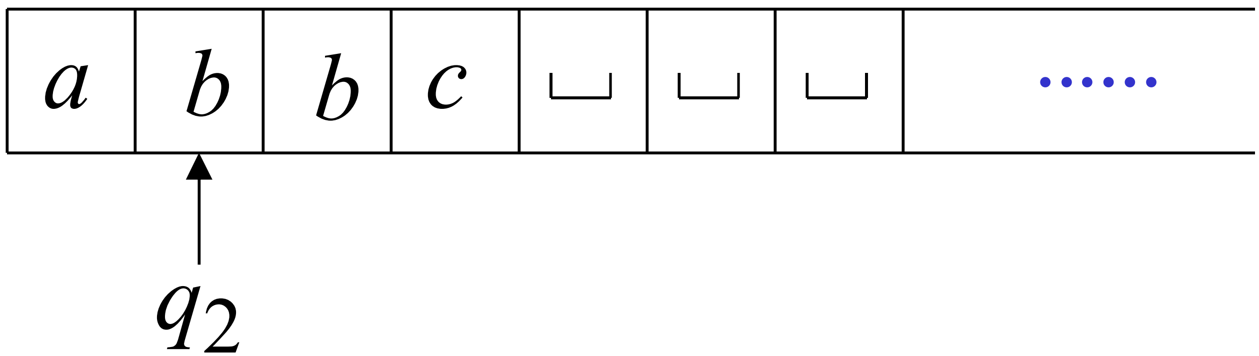


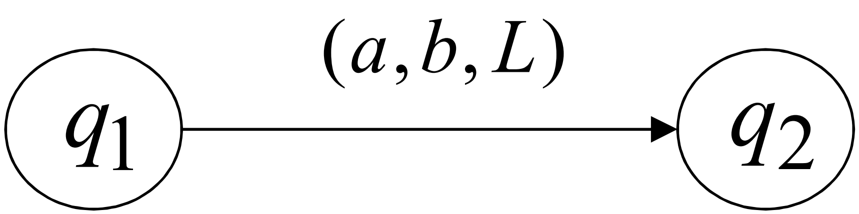
Example:

Time 1

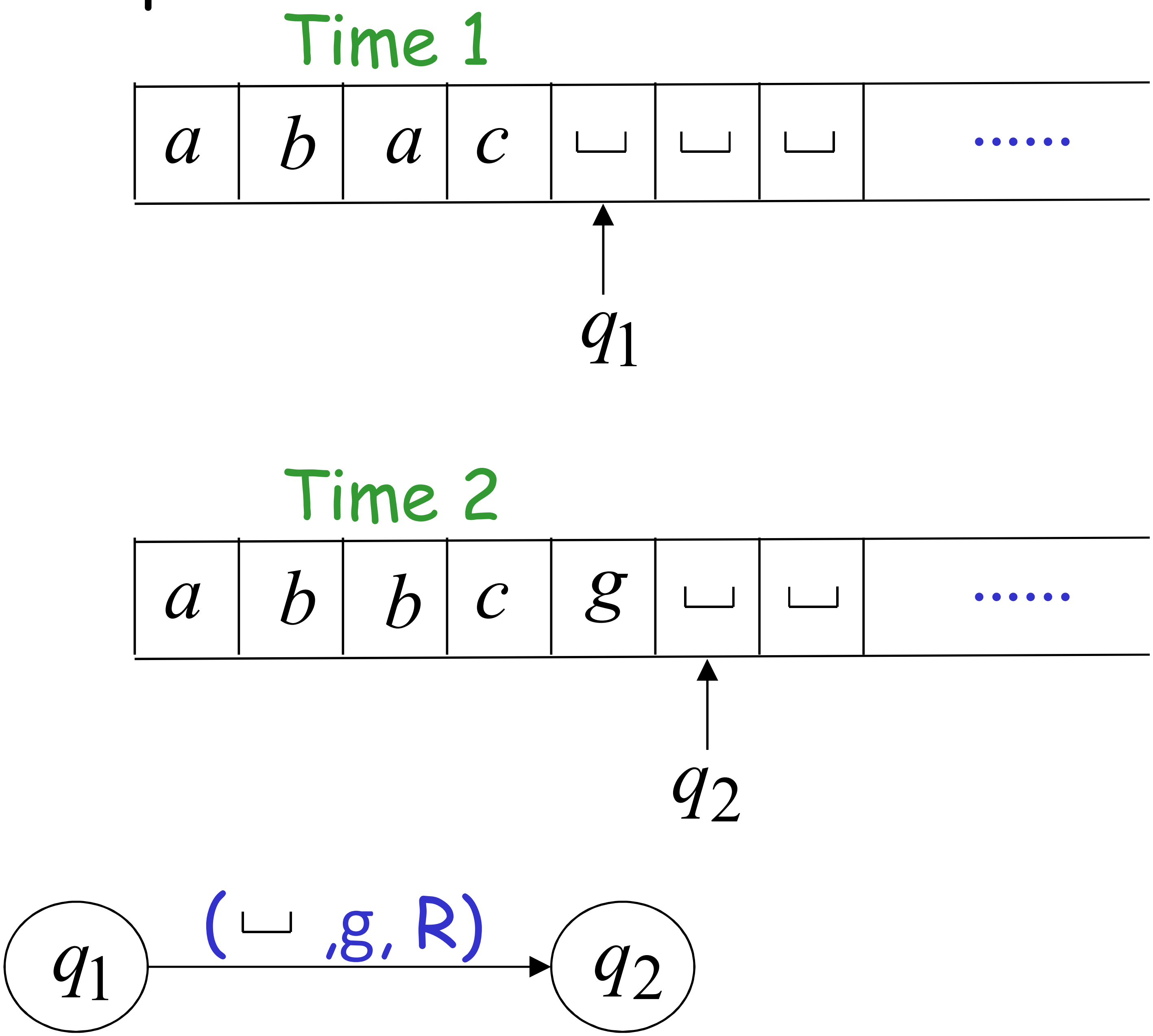


Time 2





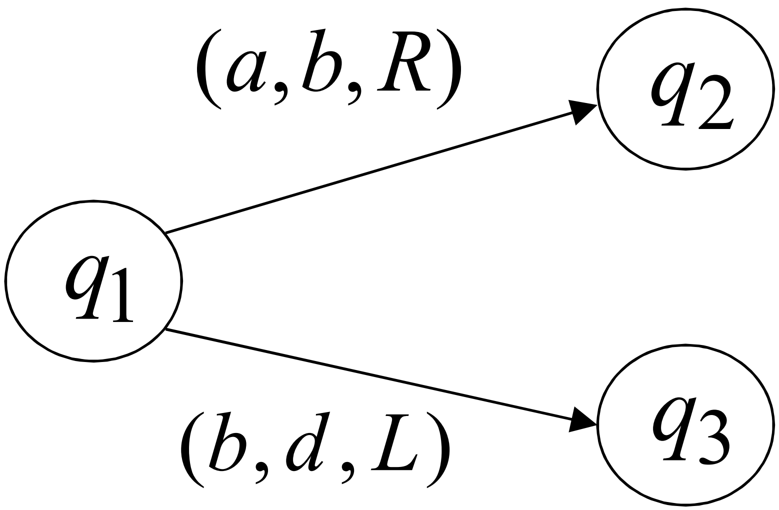
Example:



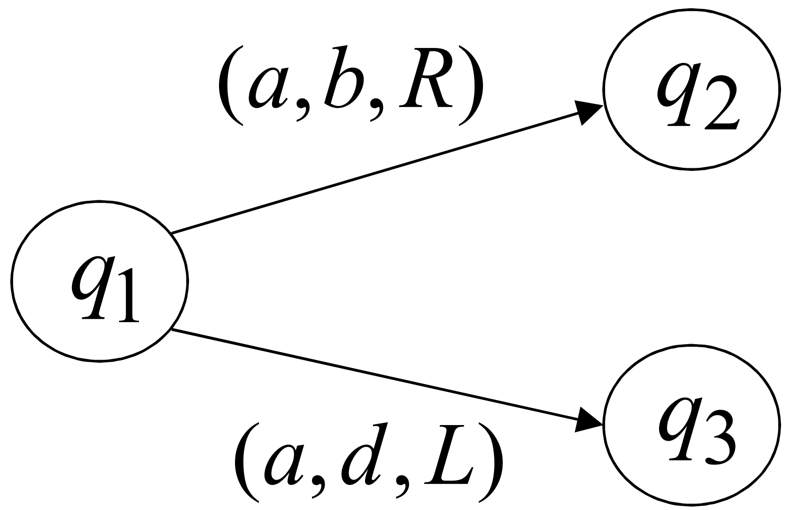
Determinism

Turing Machines are deterministic

<u>Allowed</u>

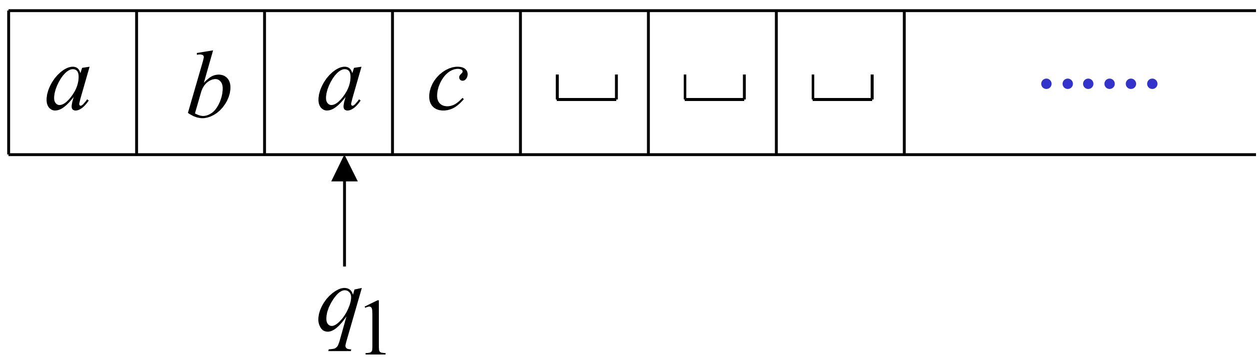


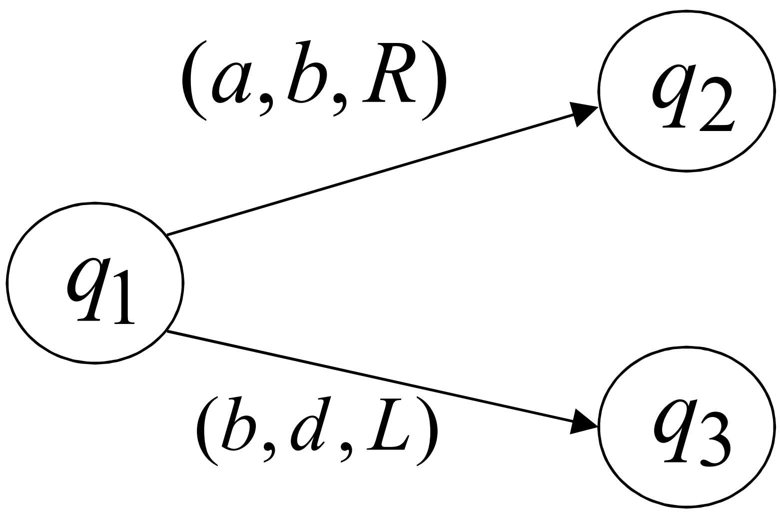
<u>**Not** Allowed</u>



Partial Transition Function

Example:





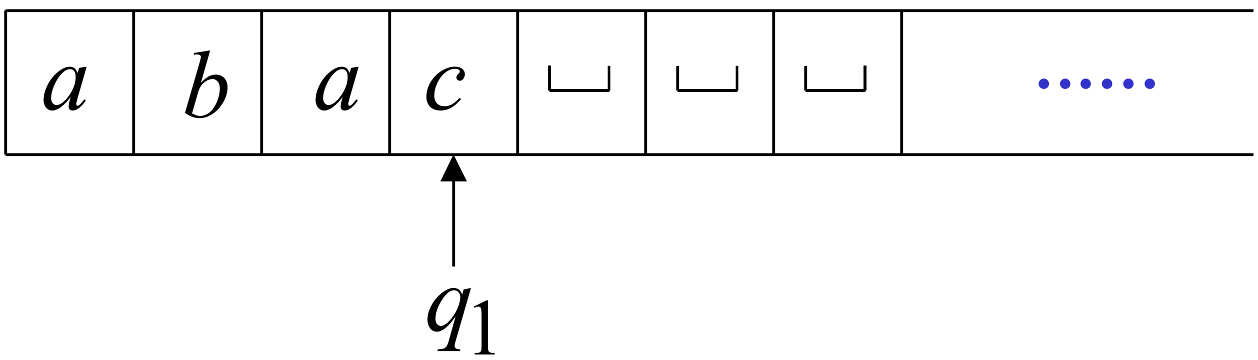
Allowed:

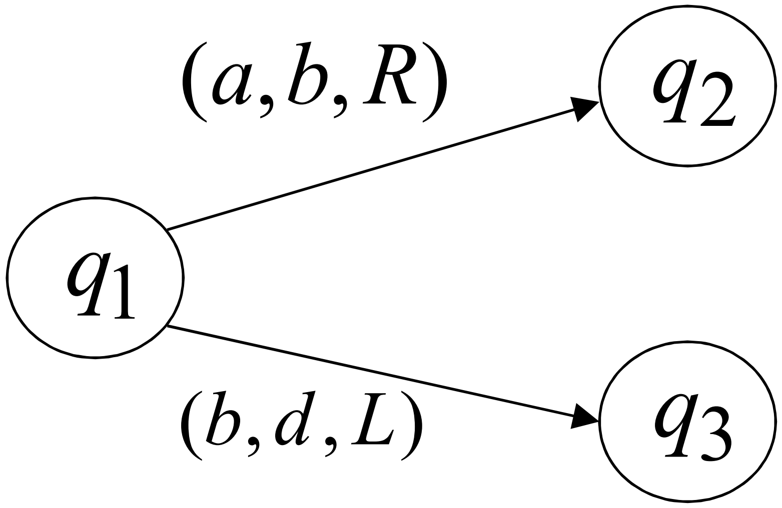
No transition for input symbol c

Halting

The machine ***halts*** if it arrives at the accepting states or if there are no possible transitions to follow.

Example:

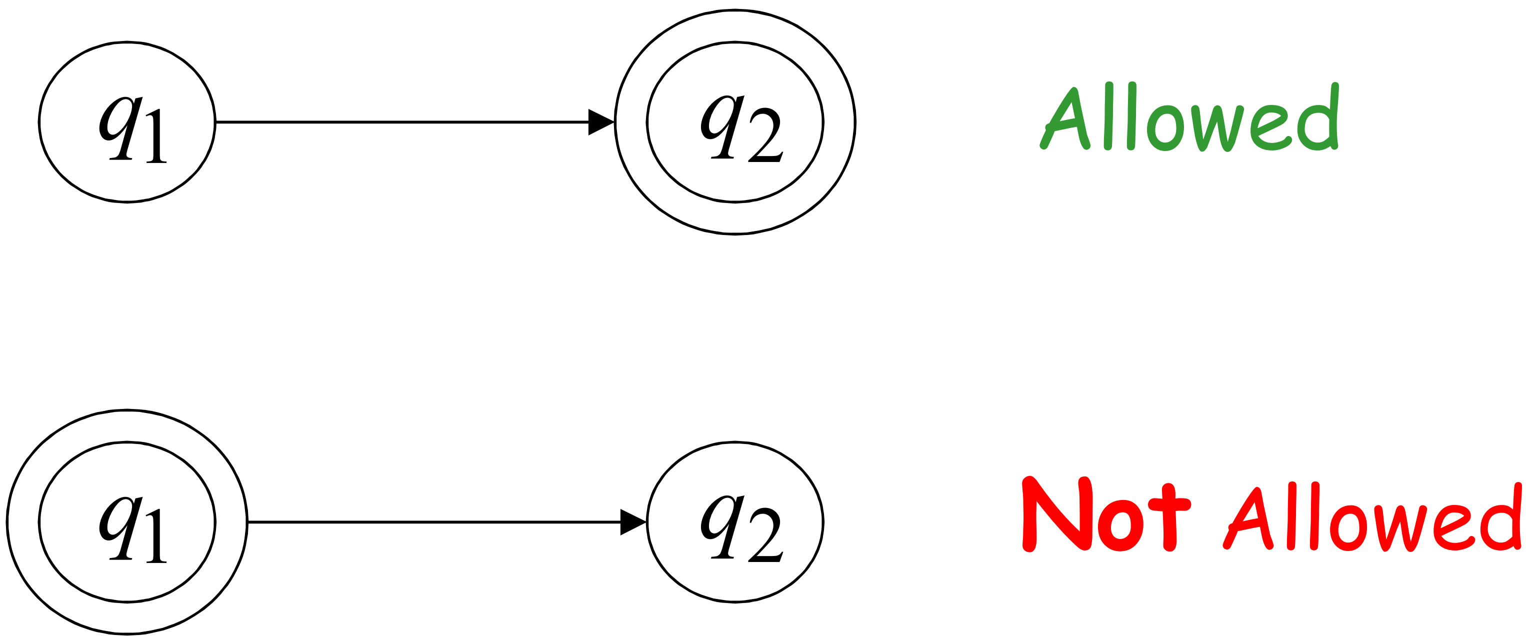




No possible transition

HALT!!!

Final States



- Final states have no outgoing transitions

- In a final state the machine halts

Acceptance

Accept Input → If machine halts in a final state

Reject Input → If machine halts in a non-final state

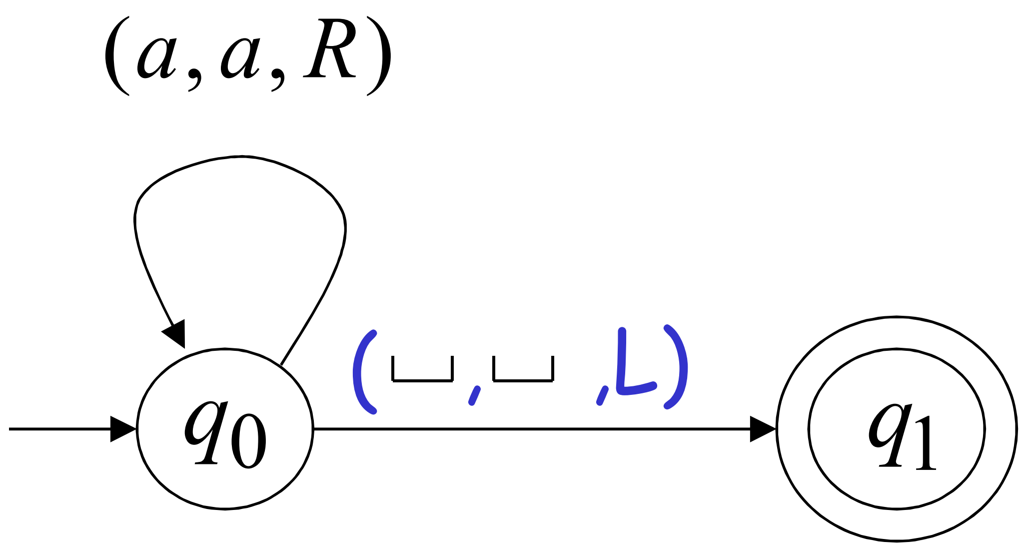
or

If machine enters an *infinite loop*

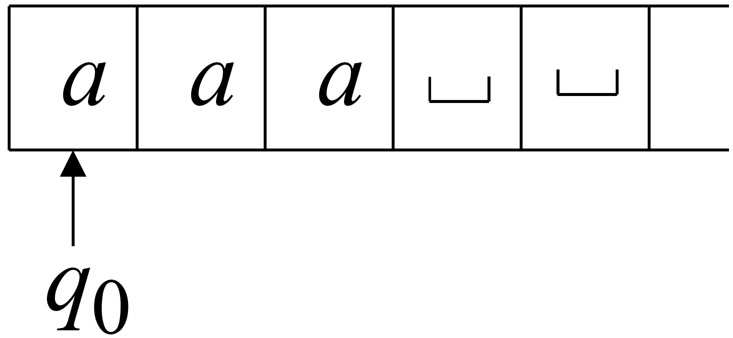
Turing Machine Example

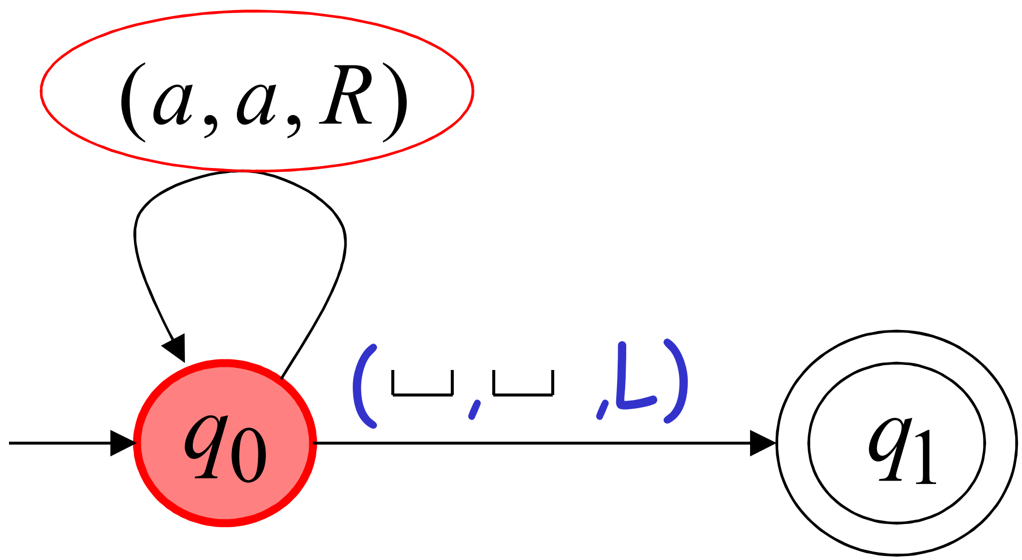
A Turing machine that accepts the language:

$$aa^*$$

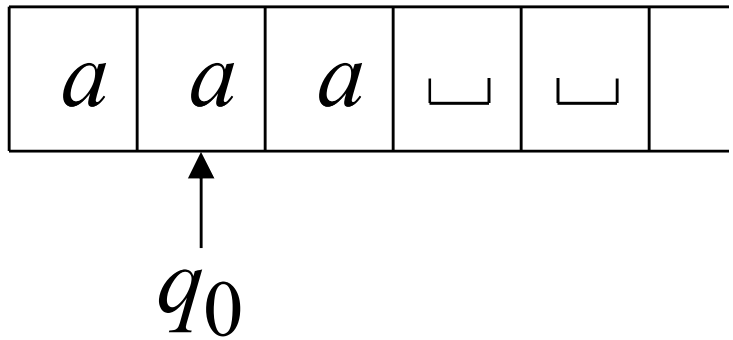


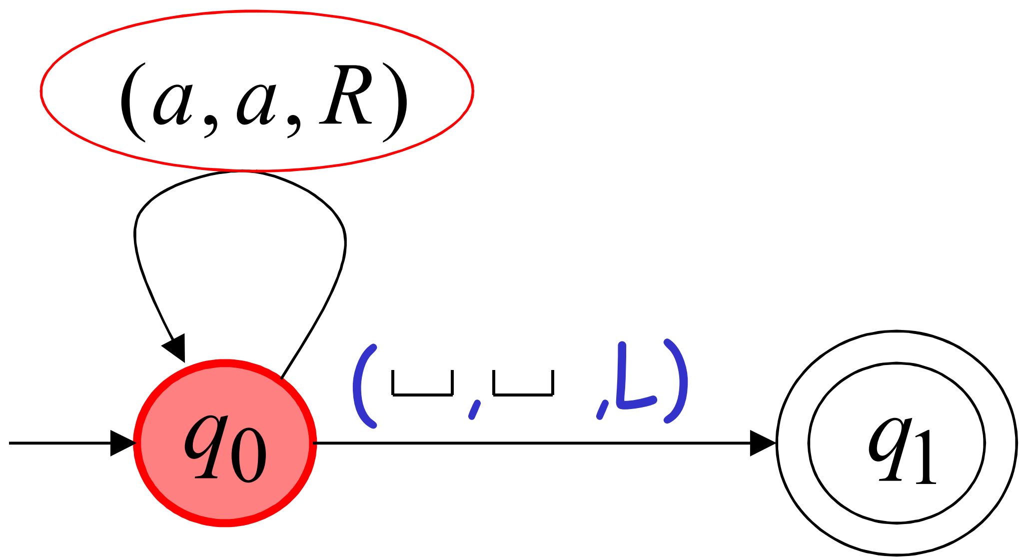
Time 0



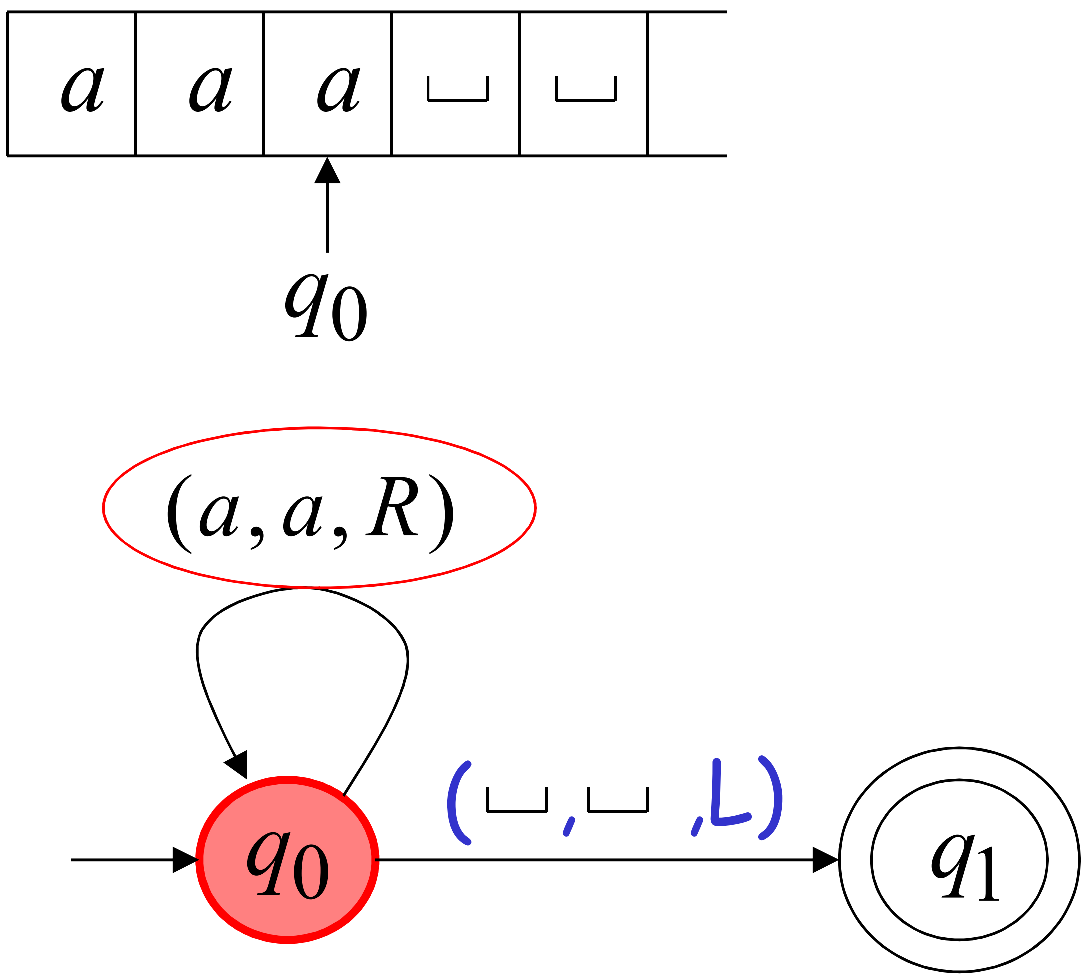


Time 1

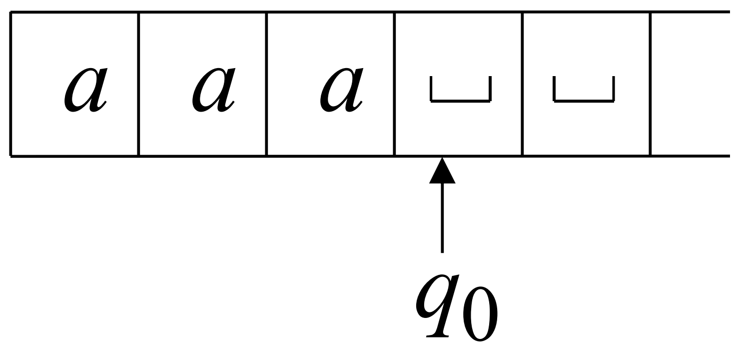


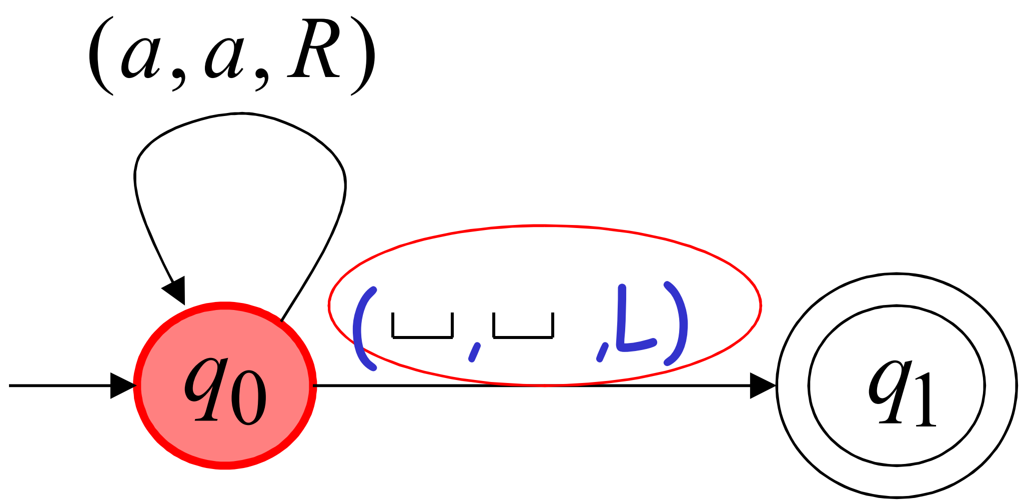


Time 2

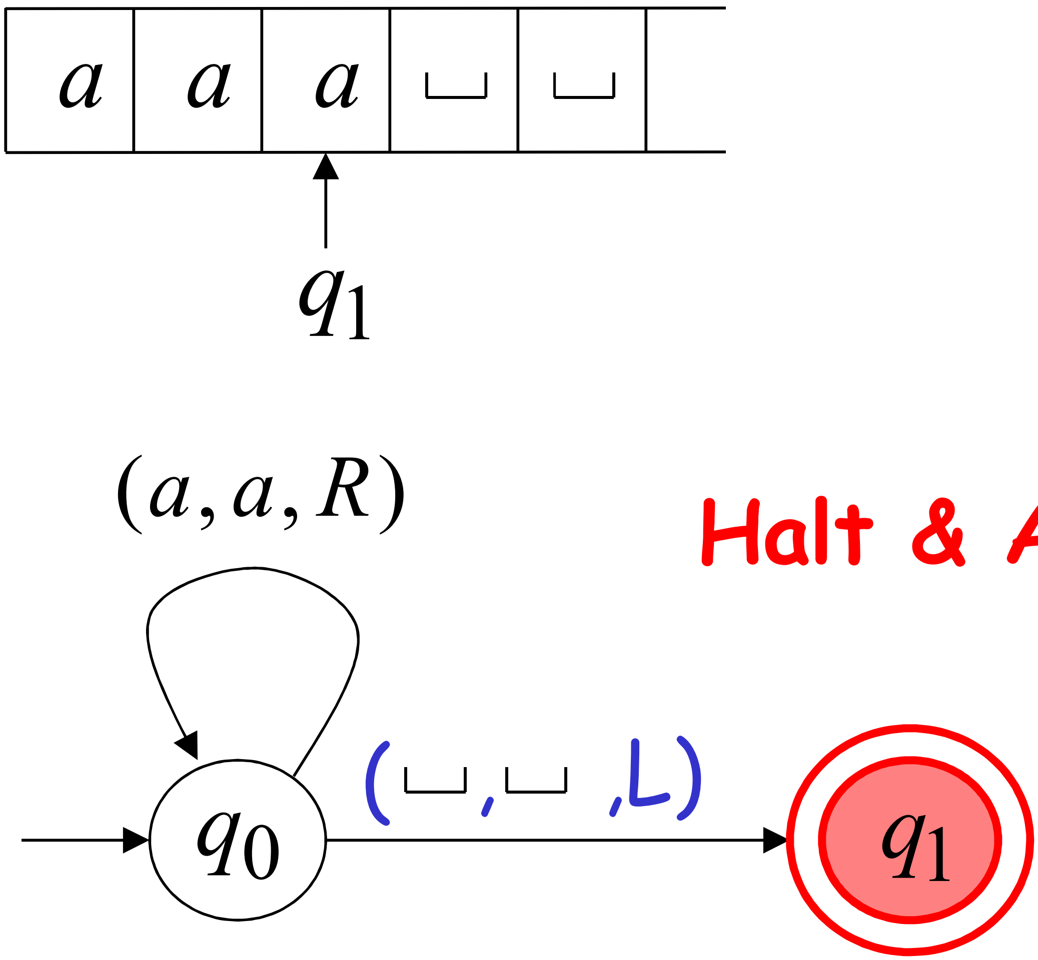


Time 3





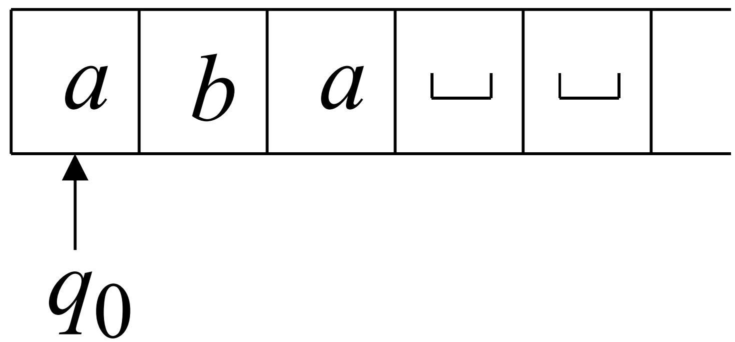
Time 4

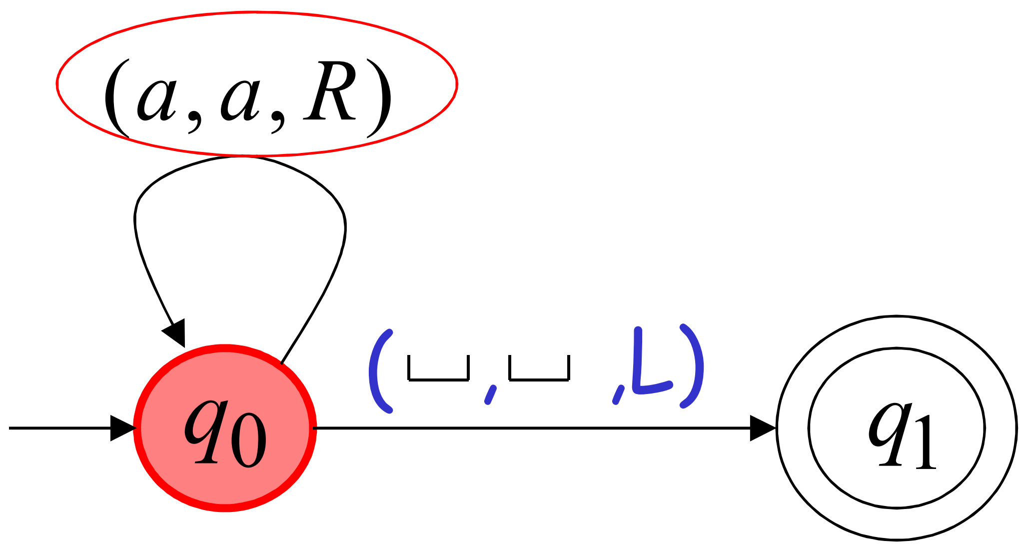


Halt & Accept

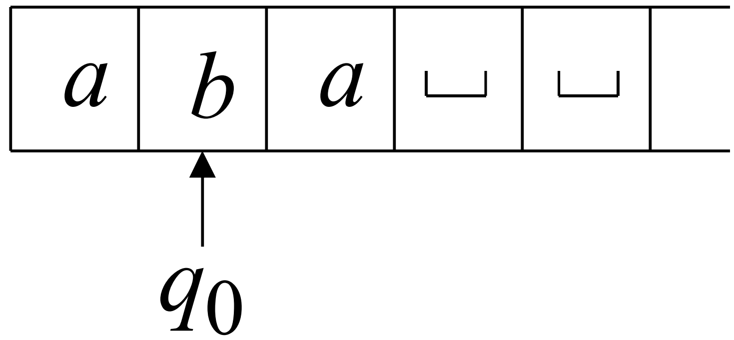
Rejection Example

Time 0



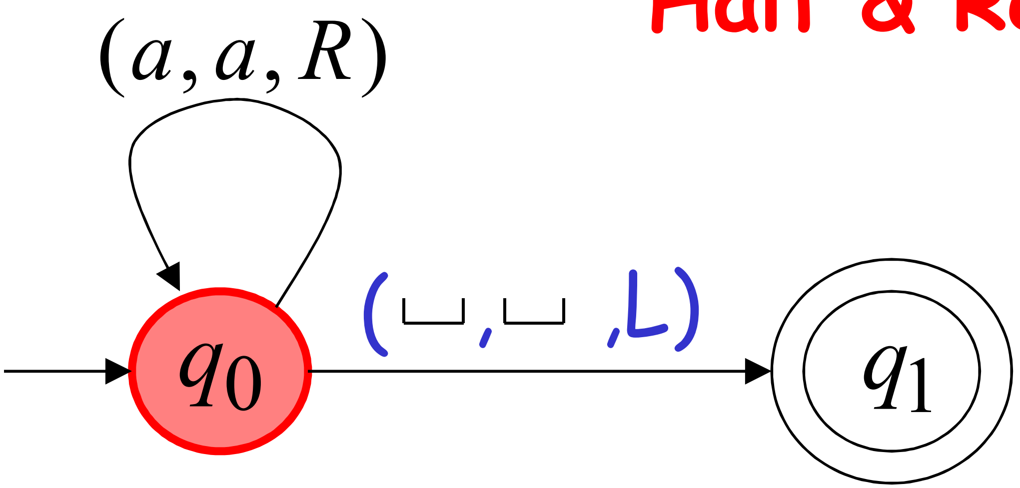


Time 1



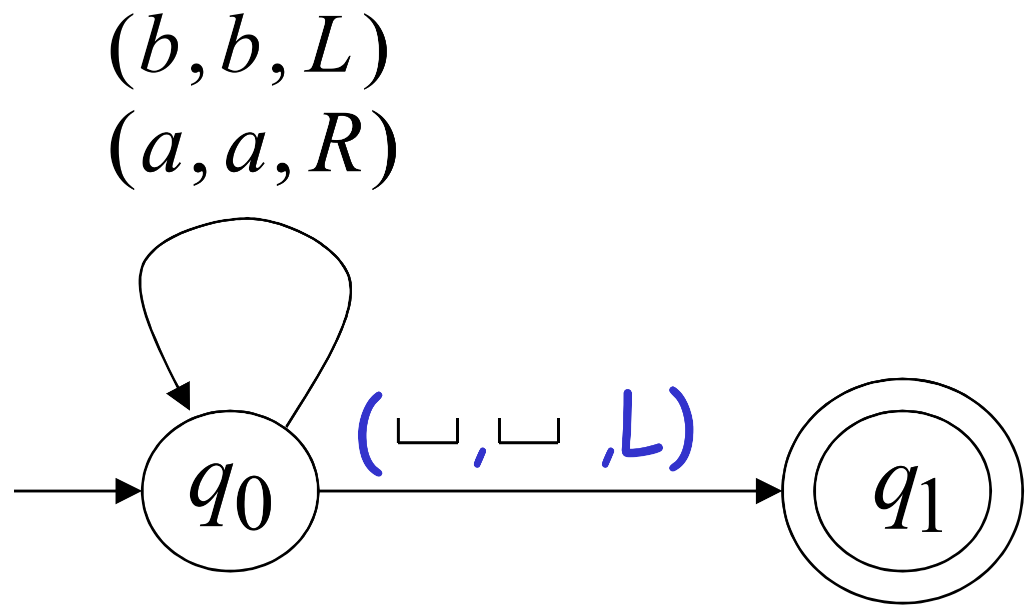
No possible Transition

Halt & Reject

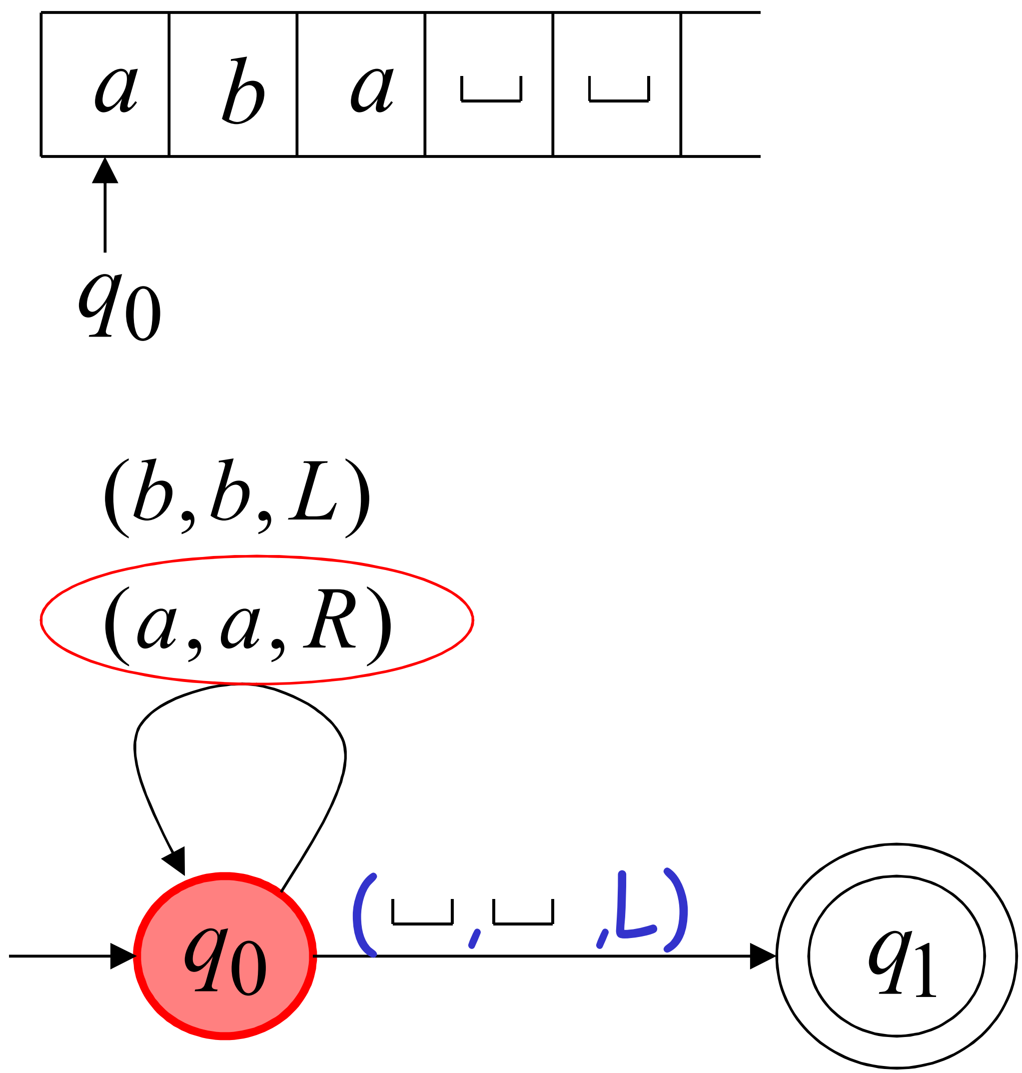


Infinite Loop Example

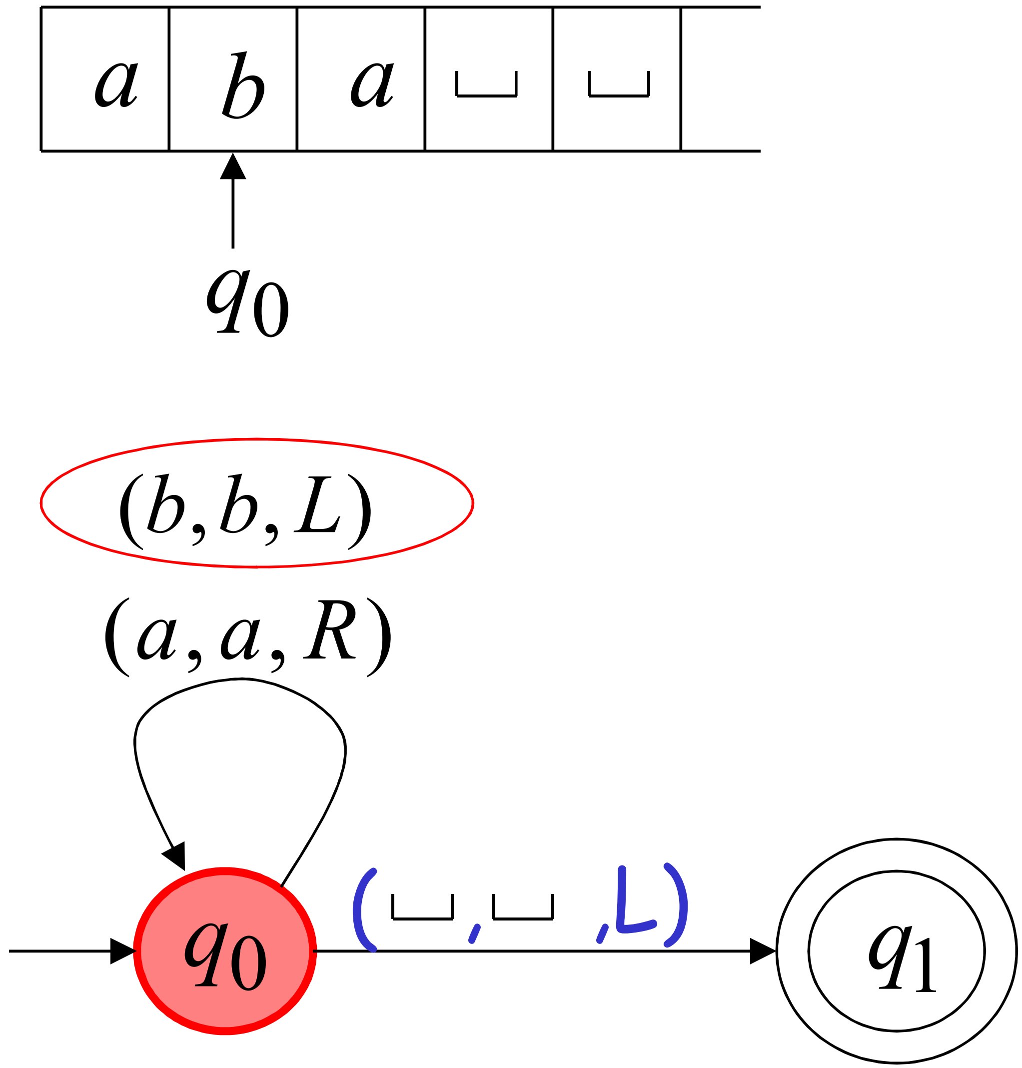
A Turing machine
for language $aa^*+b(a+b)^*$



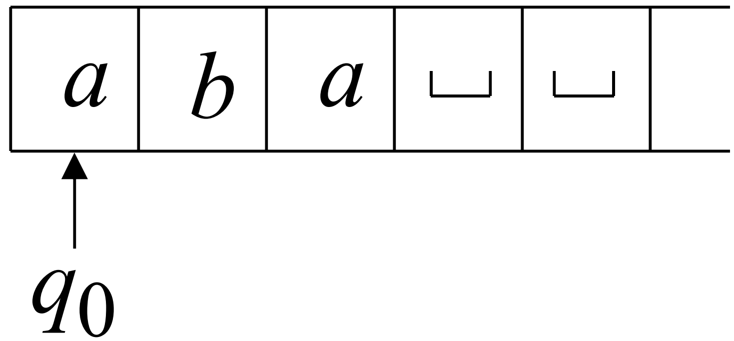
Time 0

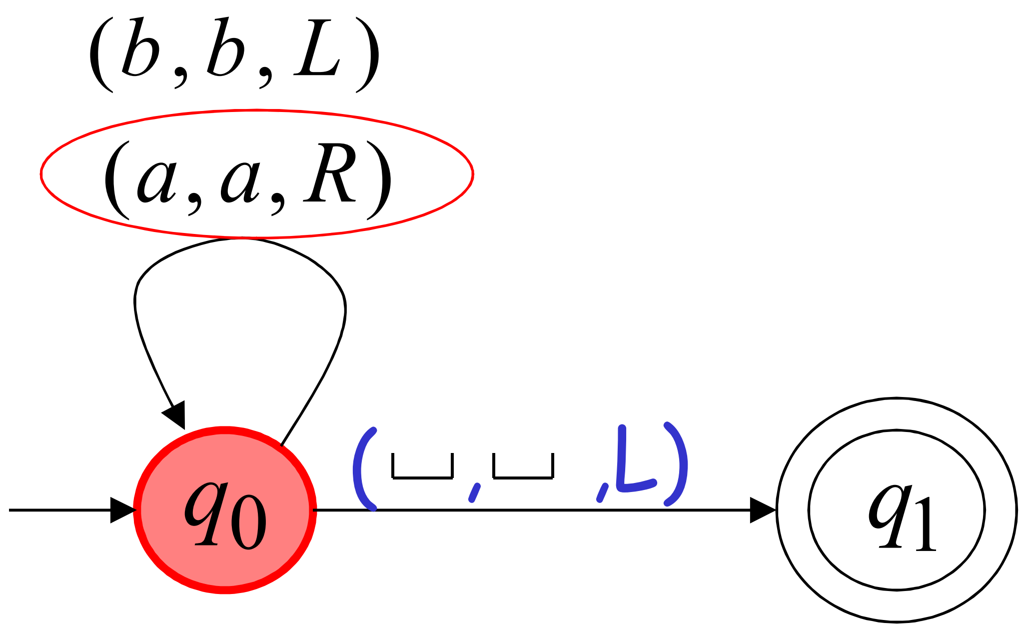


Time 1

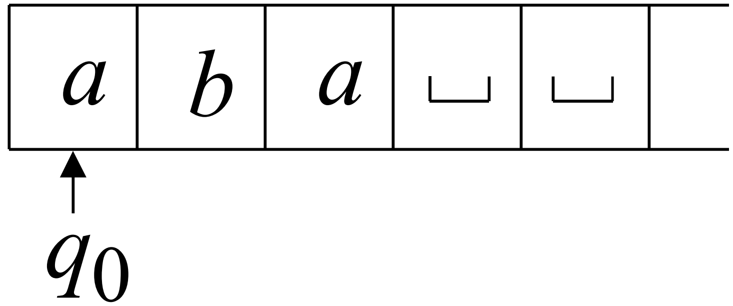


Time 2

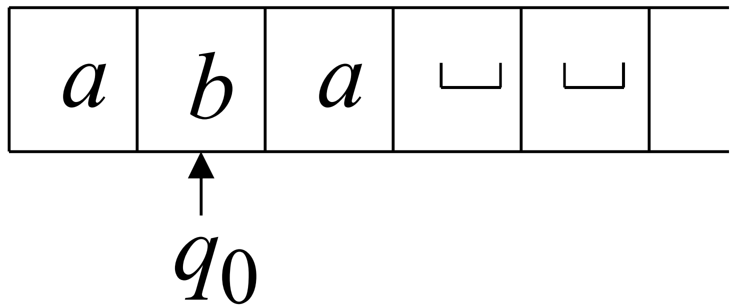




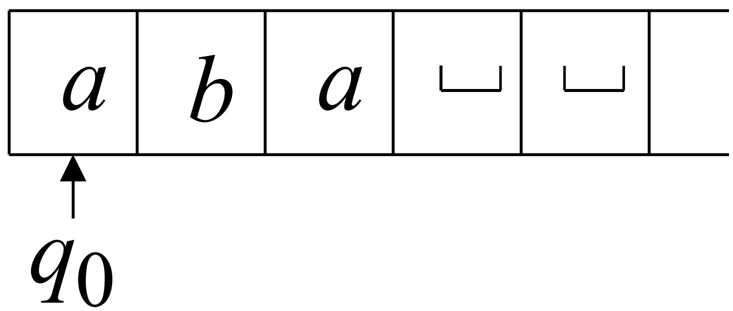
Time 2



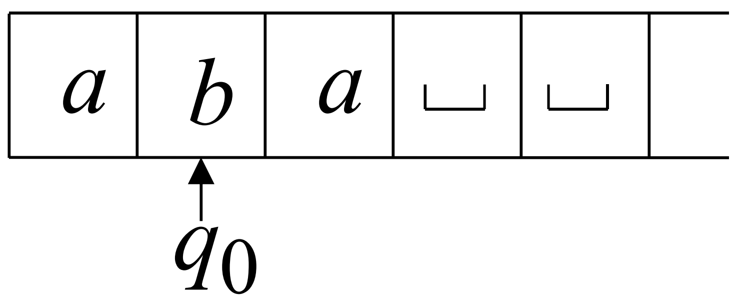
Time 3



Time 4



Time 5



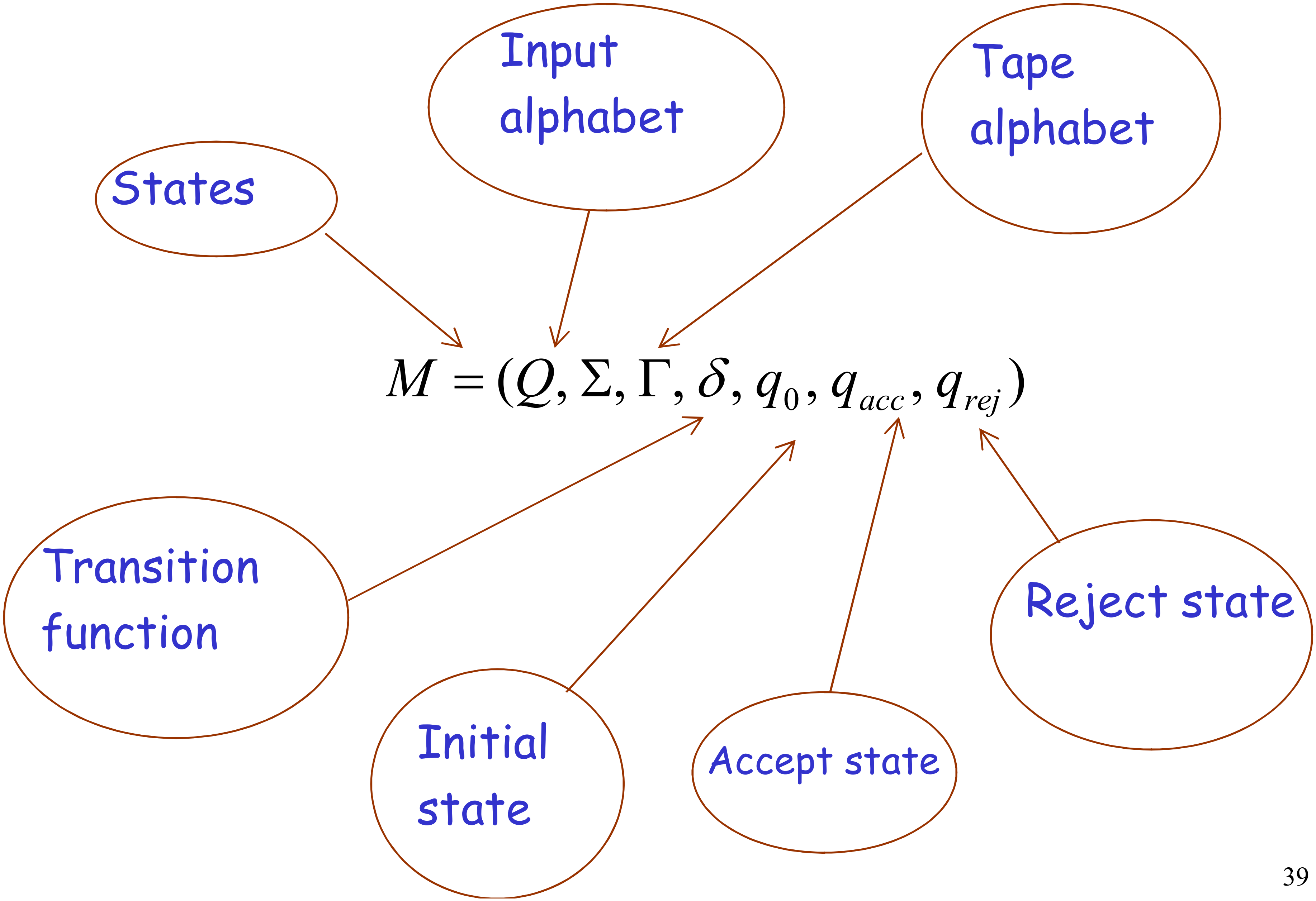
Infinite loop

Because of the infinite loop:

- The final state cannot be reached
- The machine never halts
- The input is not accepted

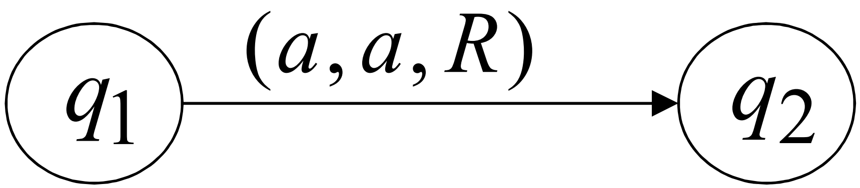
Formal Definitions
for
Turing Machines

Turing Machine:



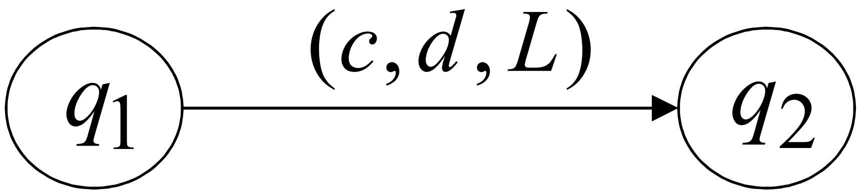
$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$$

Transition Function



$$\delta(q_1, a) = (q_2, b, R)$$

Transition Function



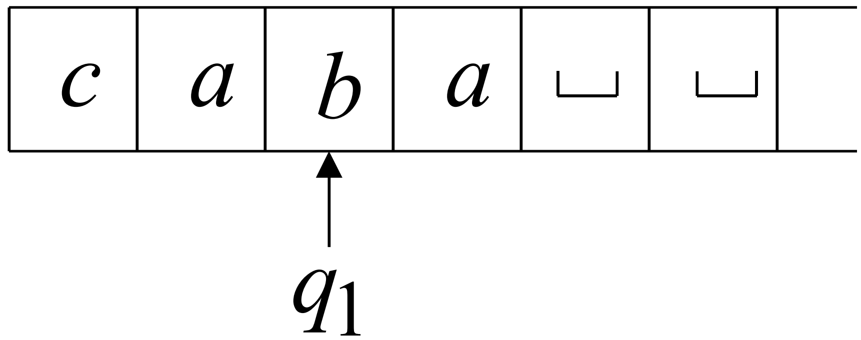
$$\delta(q_1, c) = (q_2, d, L)$$

Example : Consider a Turing machine with following transitions:

- $\delta(q_0, a) = (q_1, a, R)$
- $\delta(q_0, b) = (q_1, b, R)$
- $\delta(q_0, \sqcup) = (q_1, \sqcup, R)$
- $\delta(q_1, a) = (q_0, a, L)$
- $\delta(q_1, b) = (q_0, b, L)$
- $\delta(q_1, \sqcup) = (q_0, \sqcup, R)$

What does this Turing machine do?

Configuration



Instantaneous description: $ca\ q_1\ ba$

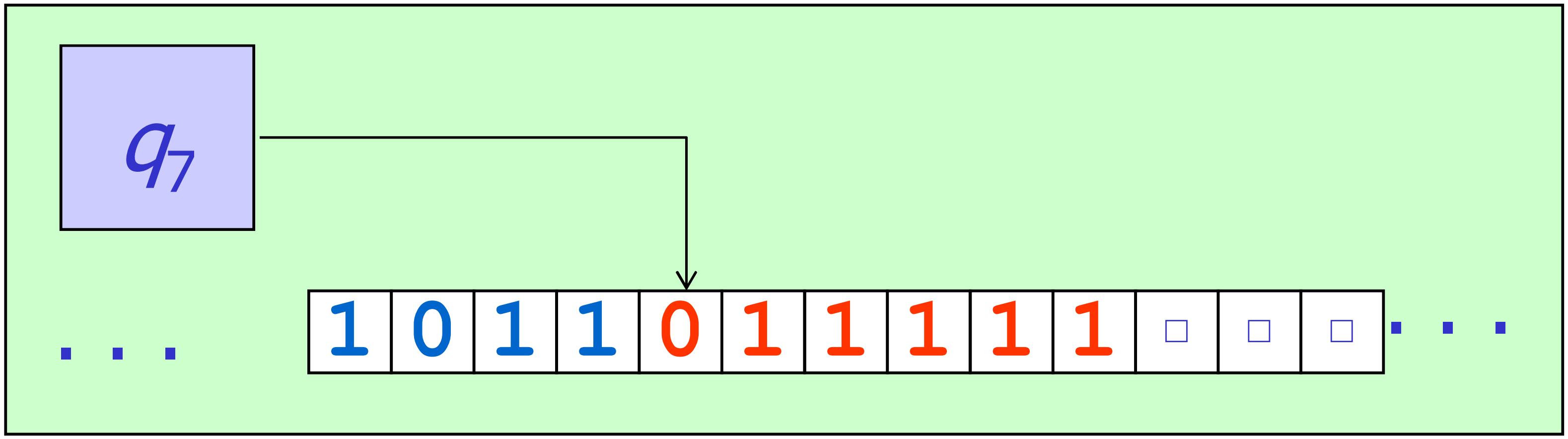
TM configurations

- The configuration of a Turing machine is the current setting i.e.
  - Current state
  - Current tape contents
  - Current head location

  These three items are a configuration of the TM
- Configurations are represented in a special way:
  - When the TM is in state *q*, and
  - The contents of the tape is two strings *uv*, and
  - The head is on the leftmost position of string *v*
  - Then we represent this configuration as "*u q v* "- this string is the Instantaneous Description (ID).

Configurations of TMs

- Example: $1011q_7011111$



Instantaneous Descriptions of a Turing Machine

- Initially, a TM has a tape consisting of a string of input symbols surrounded by an infinity of blanks in both directions.

- The TM is in the start state, and the head is at the leftmost input symbol.

Standard Turing Machine

The machine we described is the standard:

- Deterministic

- Infinite tape in one directions

- Tape is the input/output file

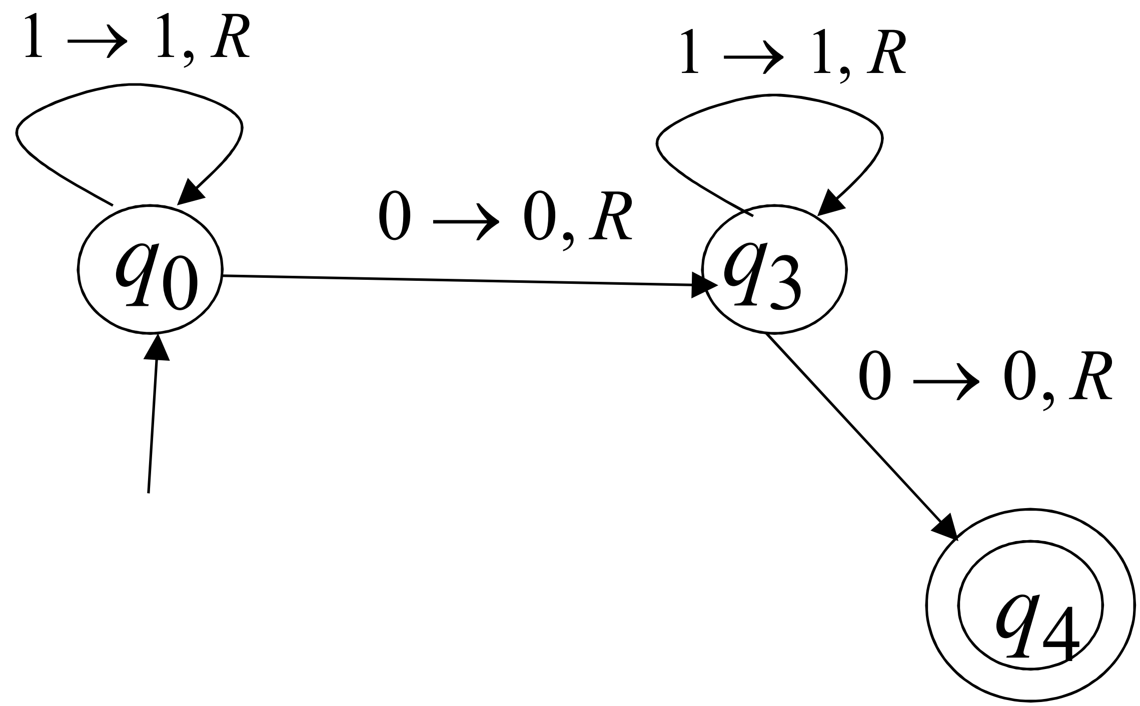
Construction of Turing Machines

Example I

- Given: w is a bitstring
- Construct TM that accepts the language

  L ={w: w contains at least two 0s}

Example I (Cont'd)



Example II

- Construct TM that accepts the language

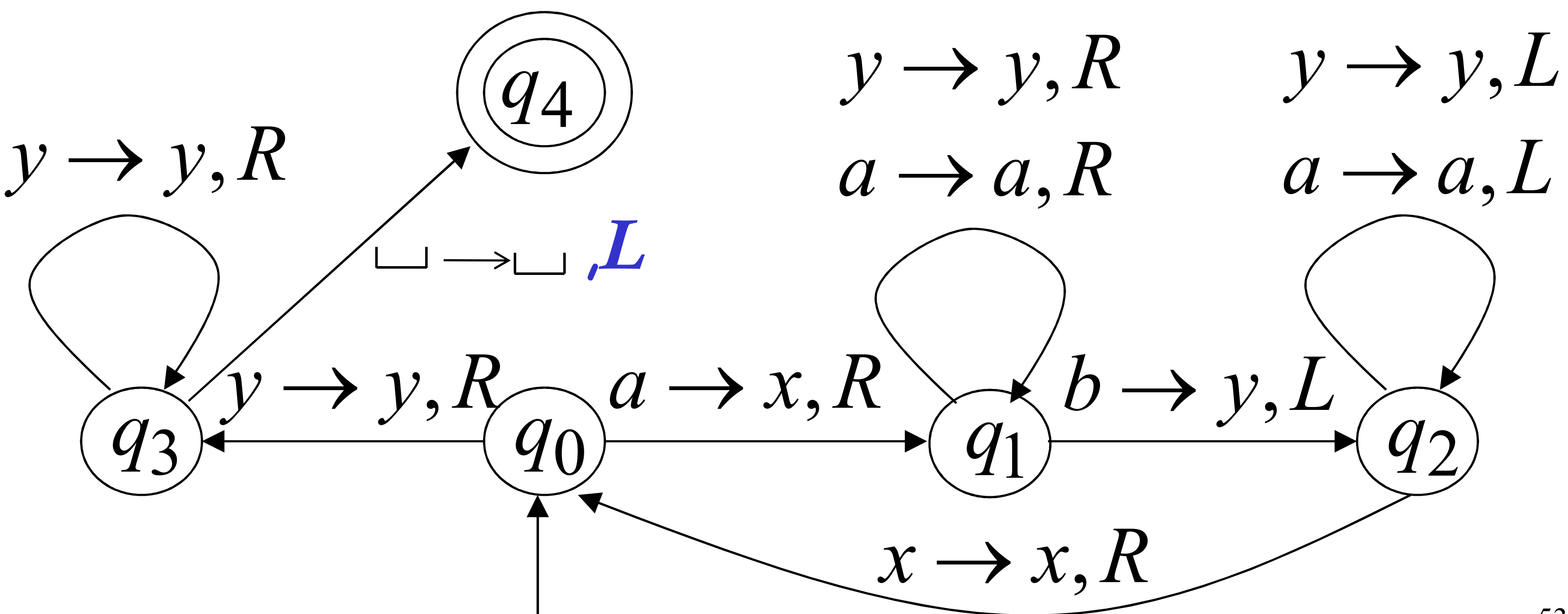
$$L = \{a^n b^n : n \geq 1\}$$

Design

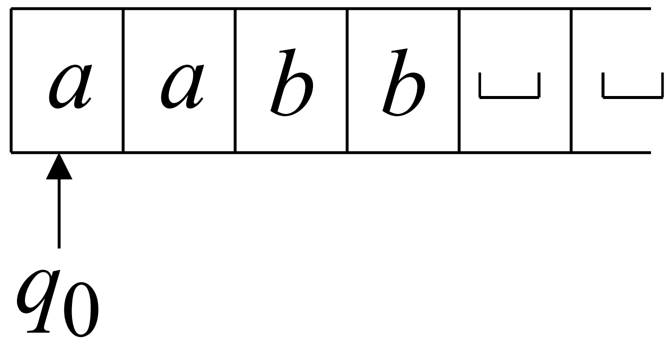
- Check first symbol is an a
  - If not, then reject
  - If so, replace with X (to mean counted) and begin recursion
- Move to the right all the way to the first unread b, and mark it with Y
- Move back (to the left) all the way to the last marked X, and then move one position to the right.
- If the next position is a, then go to step 2.
- Else move all the way to the right to ensure there are no excess bs. If not move right to the next blank symbol and stop & accept.

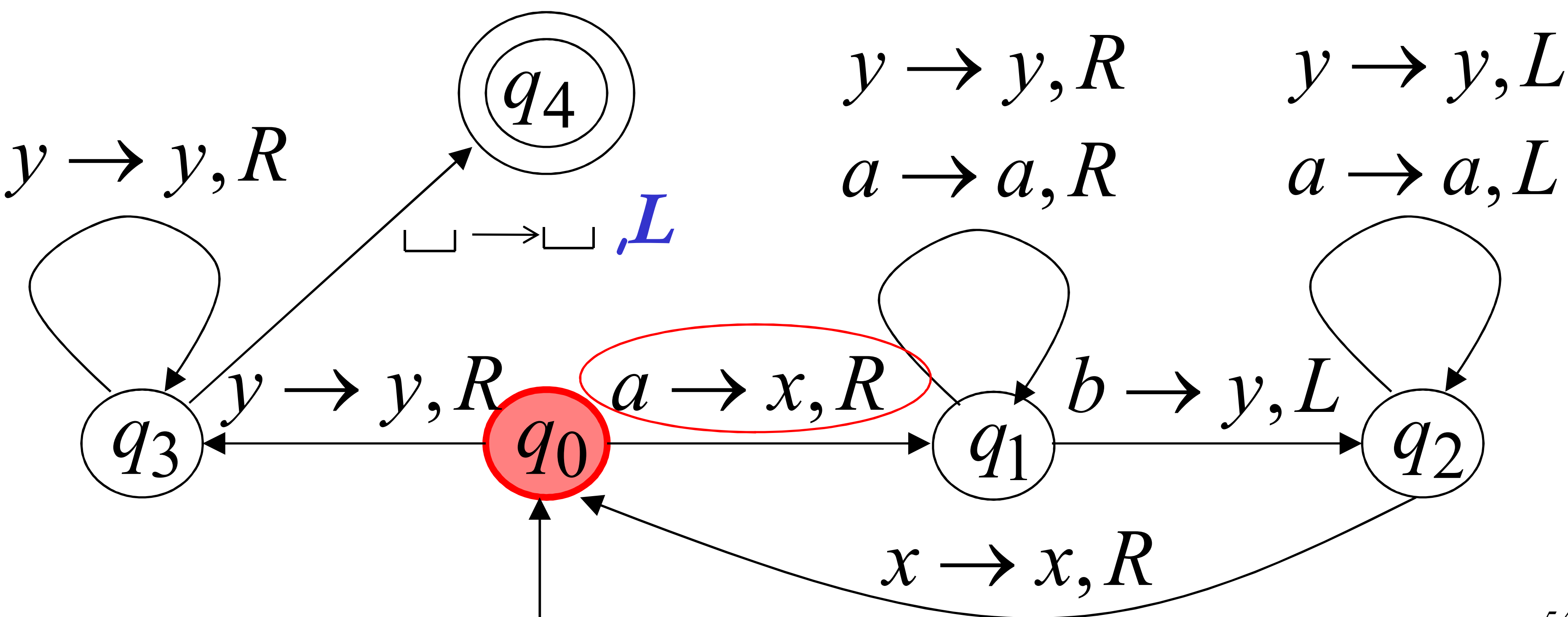
Example II

Turing machine for the language $\{a^n b^n\}$

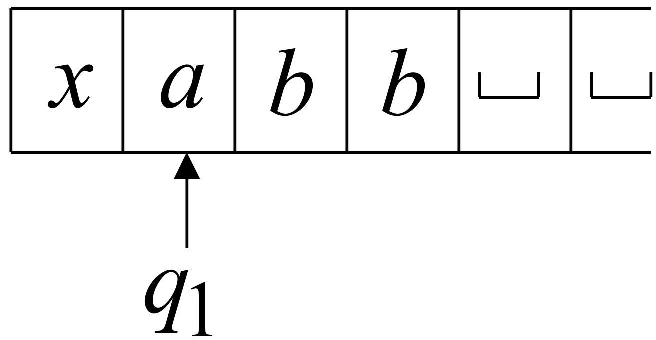


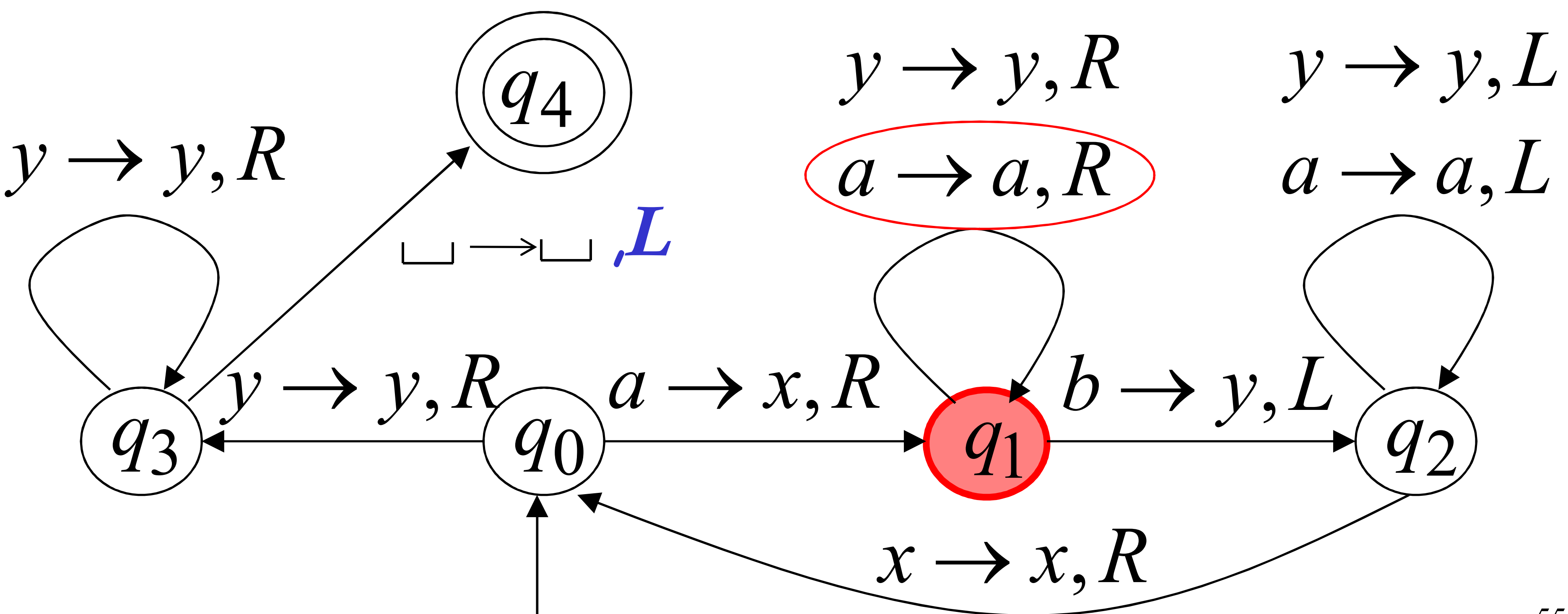
Time 0


a a b b ␣ ␣
q_0

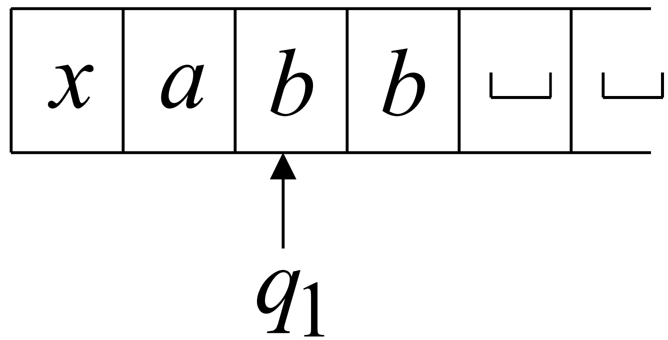

q_4
$y \rightarrow y, R$
$␣ \rightarrow ␣, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

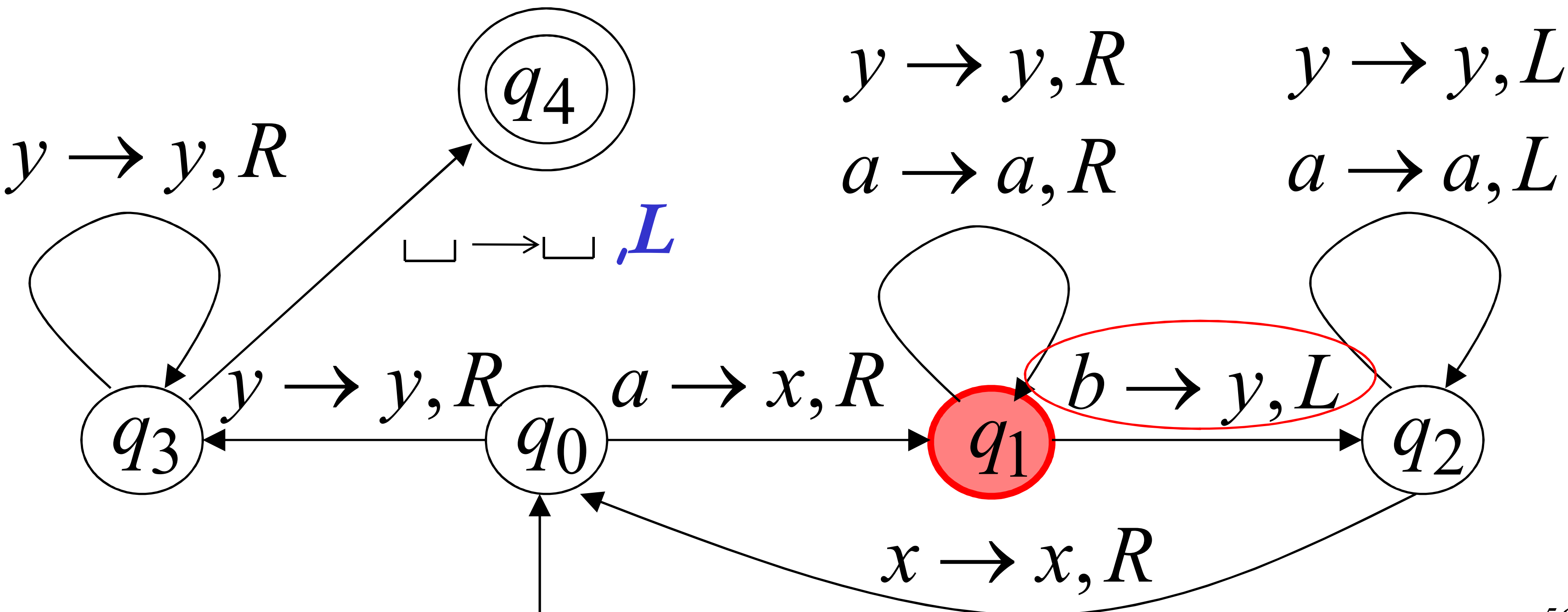
Time 1


x
a
b
b
␣
␣
q_1

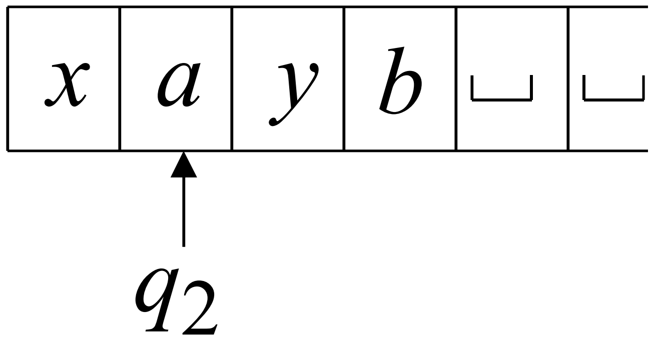

q_4
$y \rightarrow y, R$
$␣ \rightarrow ␣, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 2

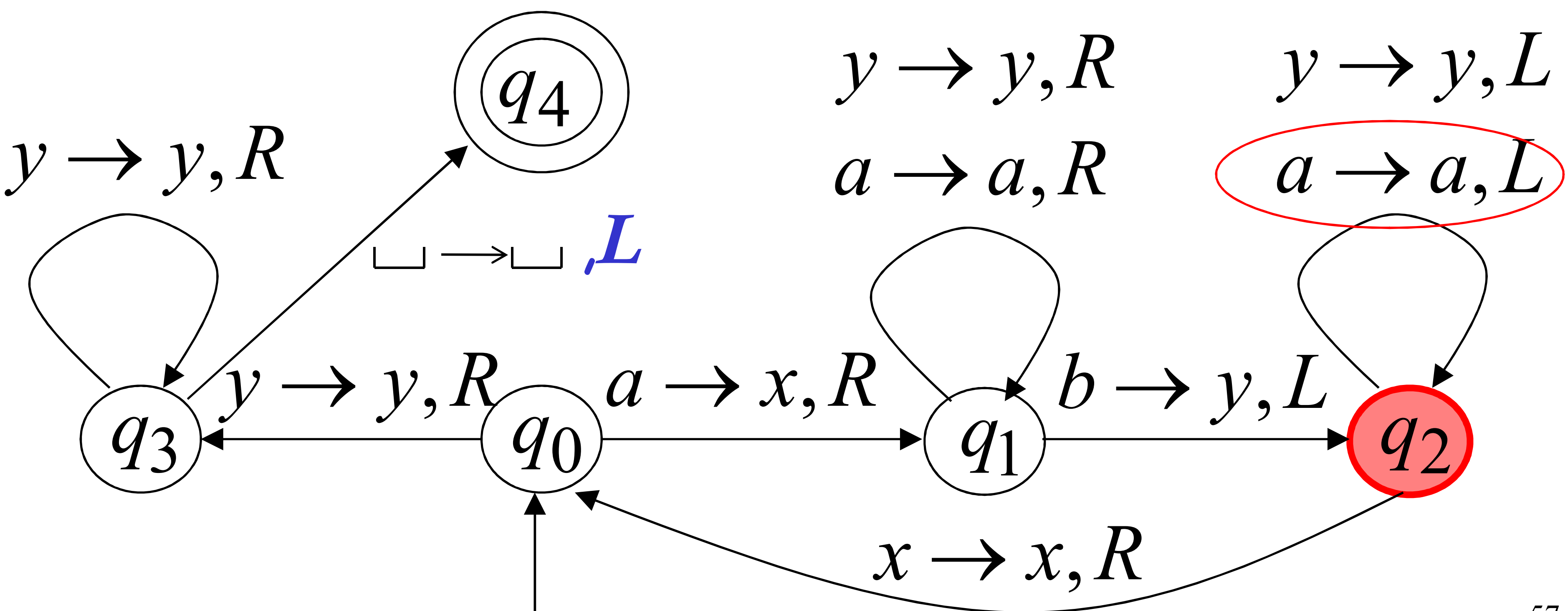




Time 3

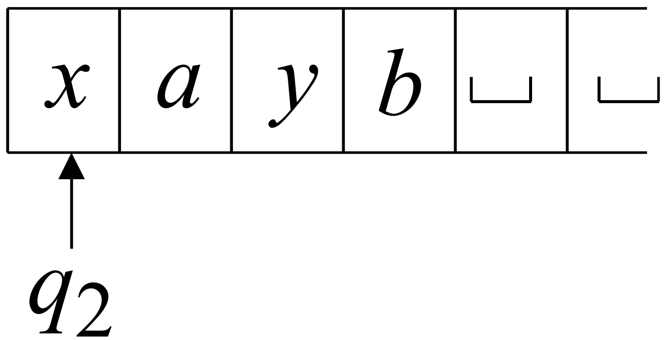


x
a
y
b
$\sqcup$
$\sqcup$
q_2

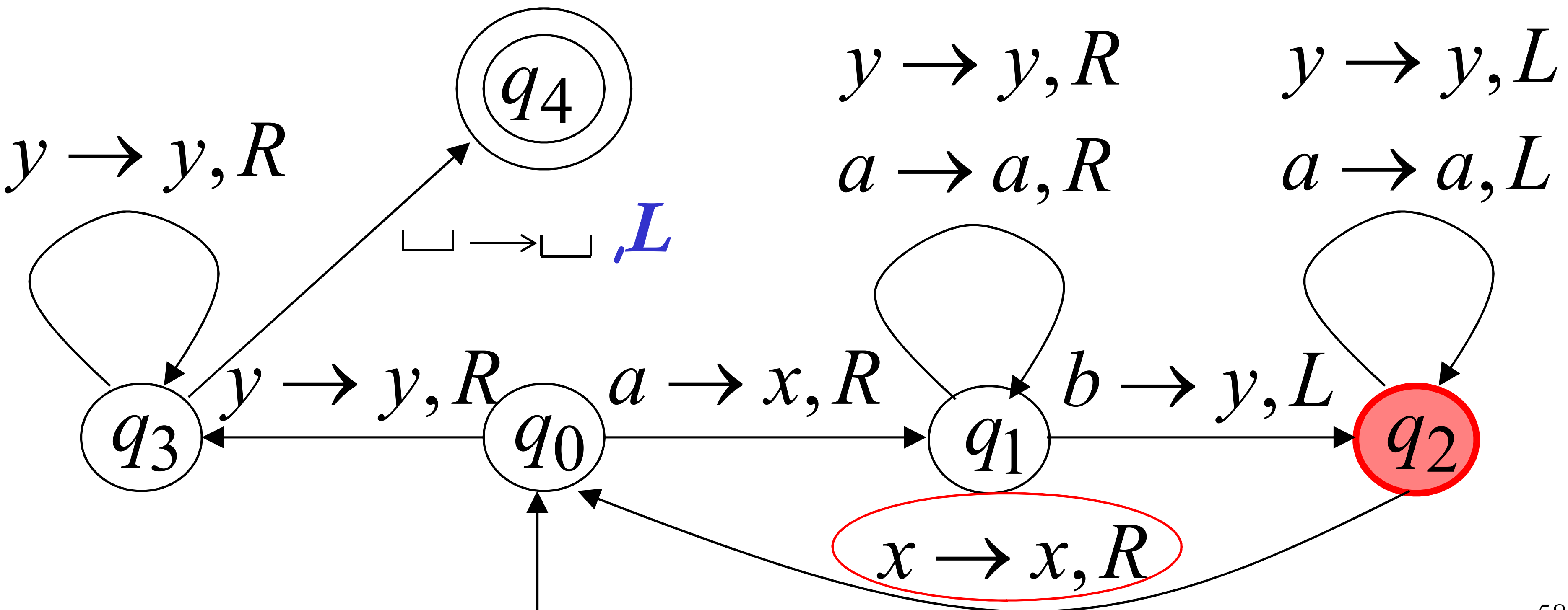


q_4
$y \rightarrow y, R$
$\sqcup \rightarrow \sqcup, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 4

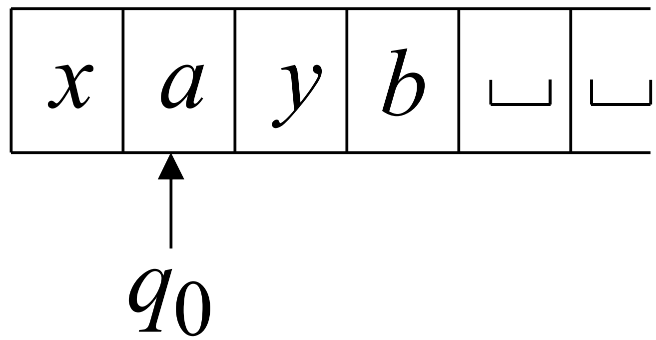


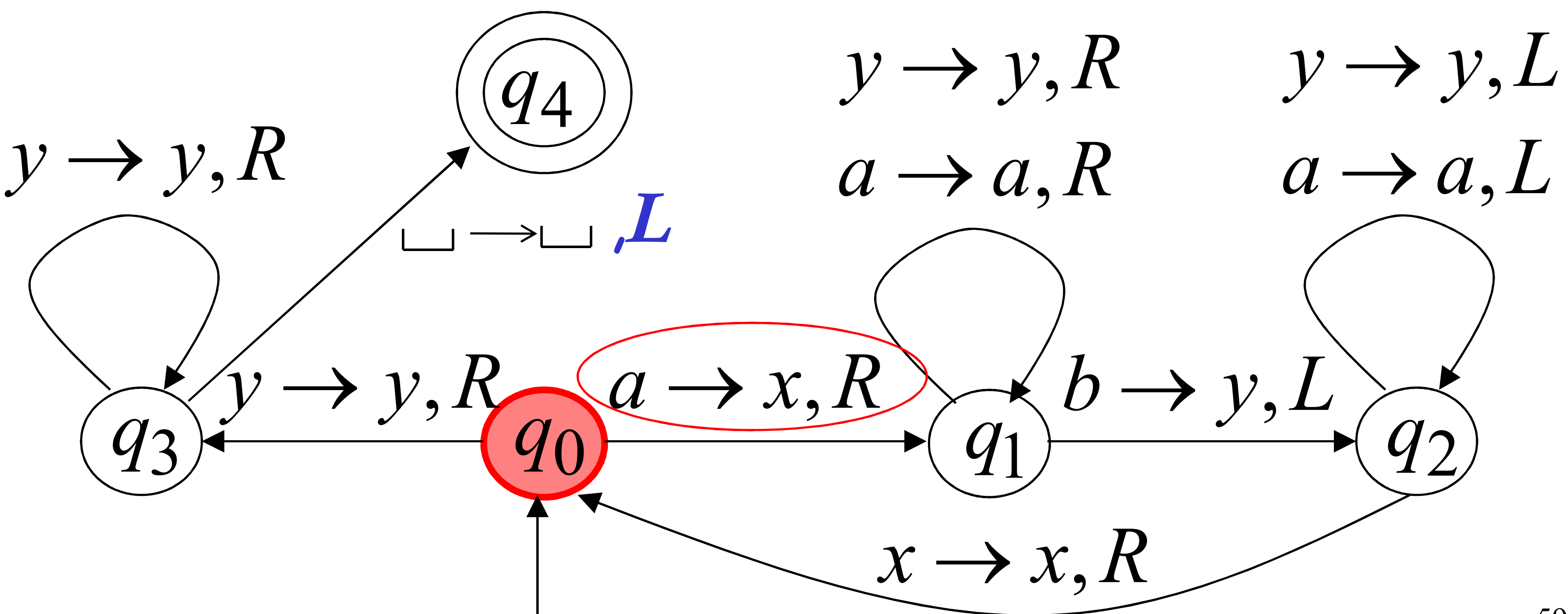
x
a
y
b
␣
␣
q_2



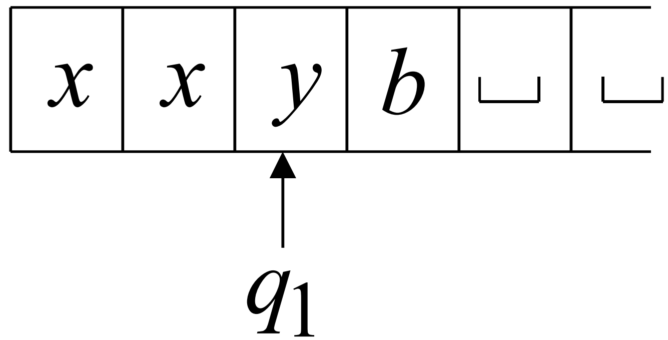
q_4
$y \rightarrow y, R$
␣ → ␣ ,L
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 5

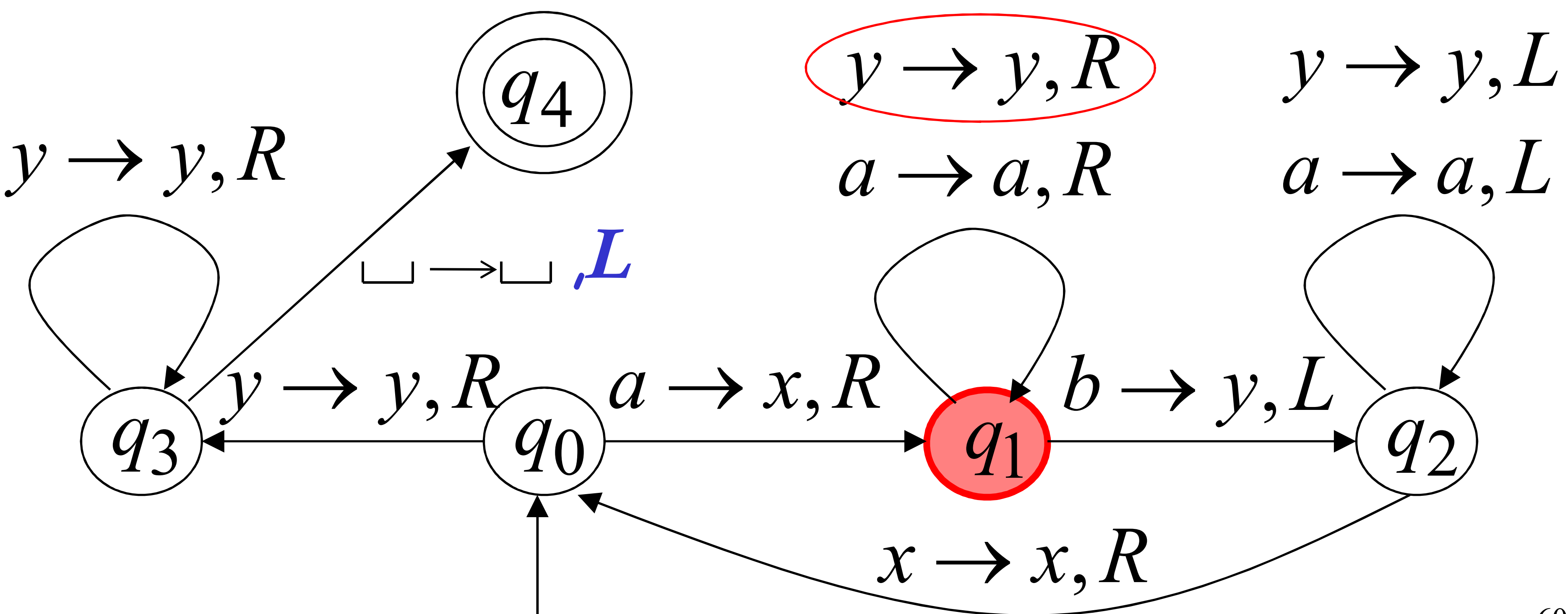

x
a
y
b
␣
␣
q_0


q_4
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$␣ \rightarrow ␣, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 6

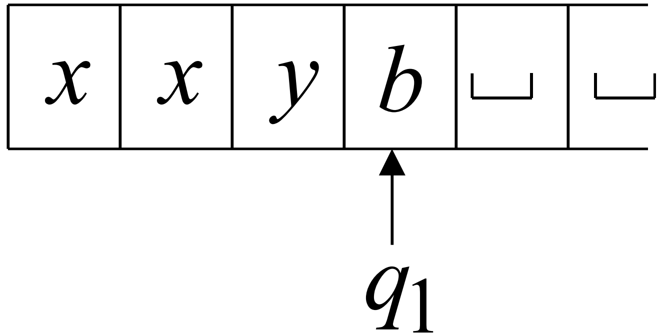


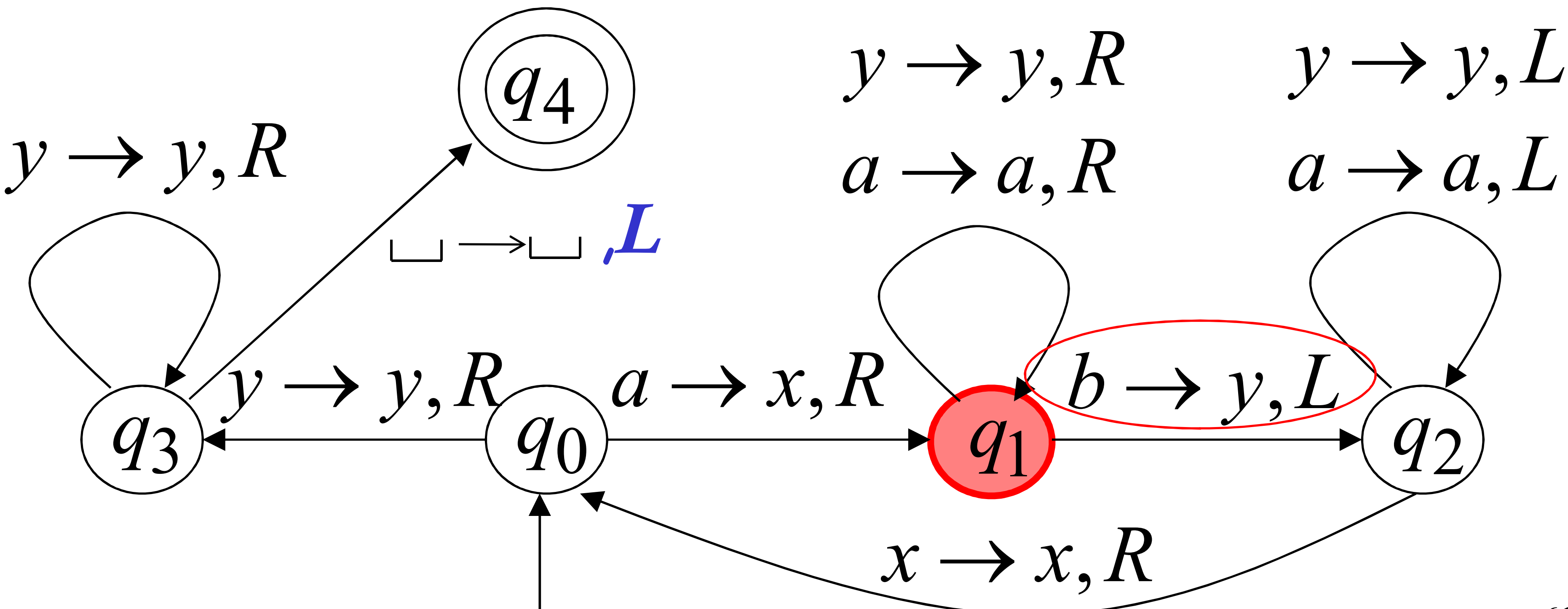
x x y b ⊔ ⊔
q_1



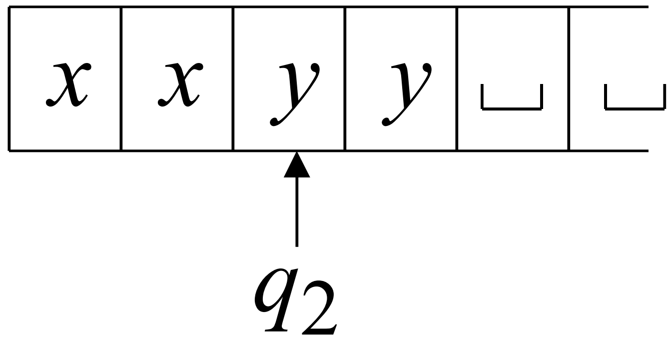
q_4
$y \rightarrow y, R$
$\sqcup \rightarrow \sqcup, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 7

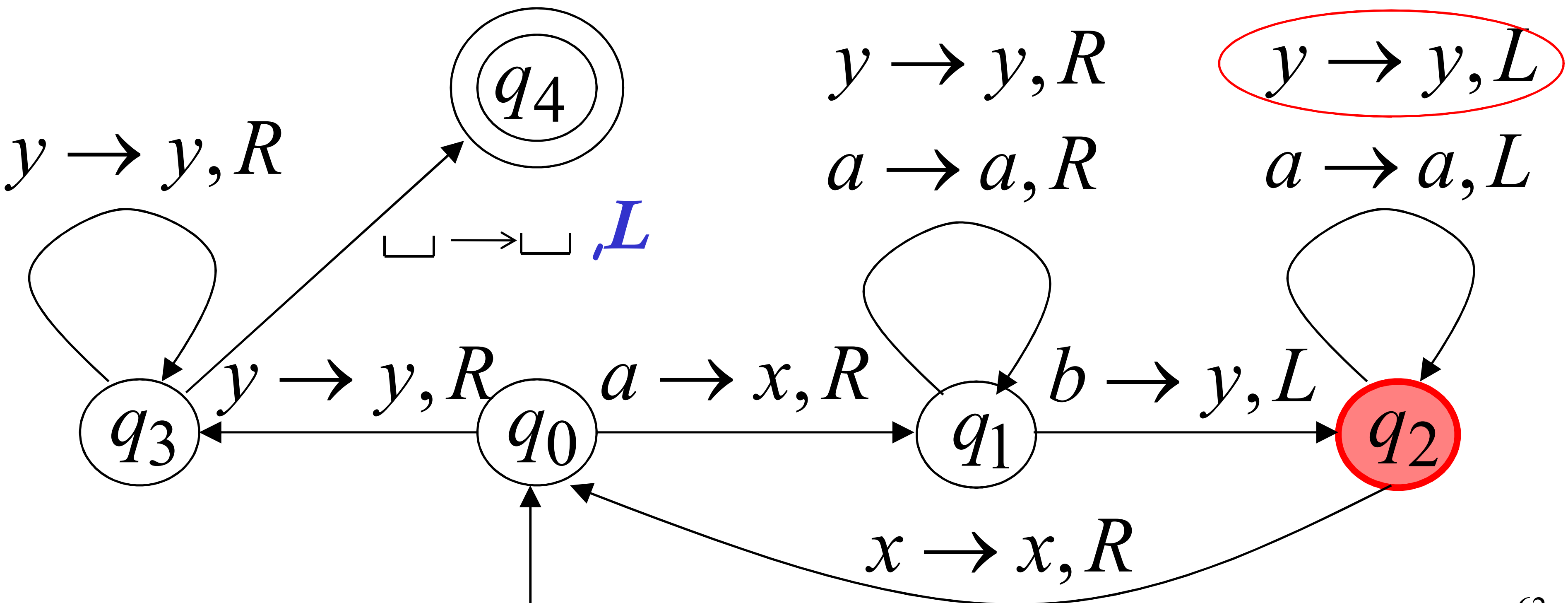




Time 8

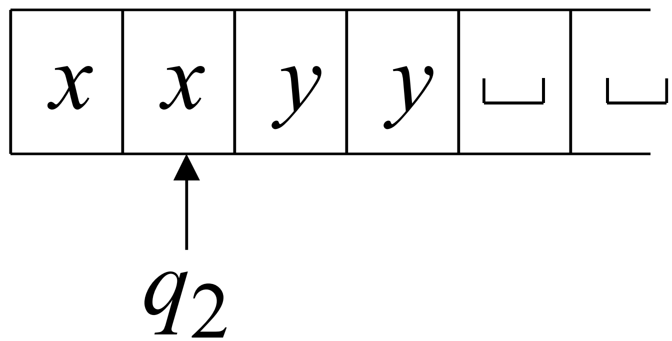


x
x
y
y
q_2

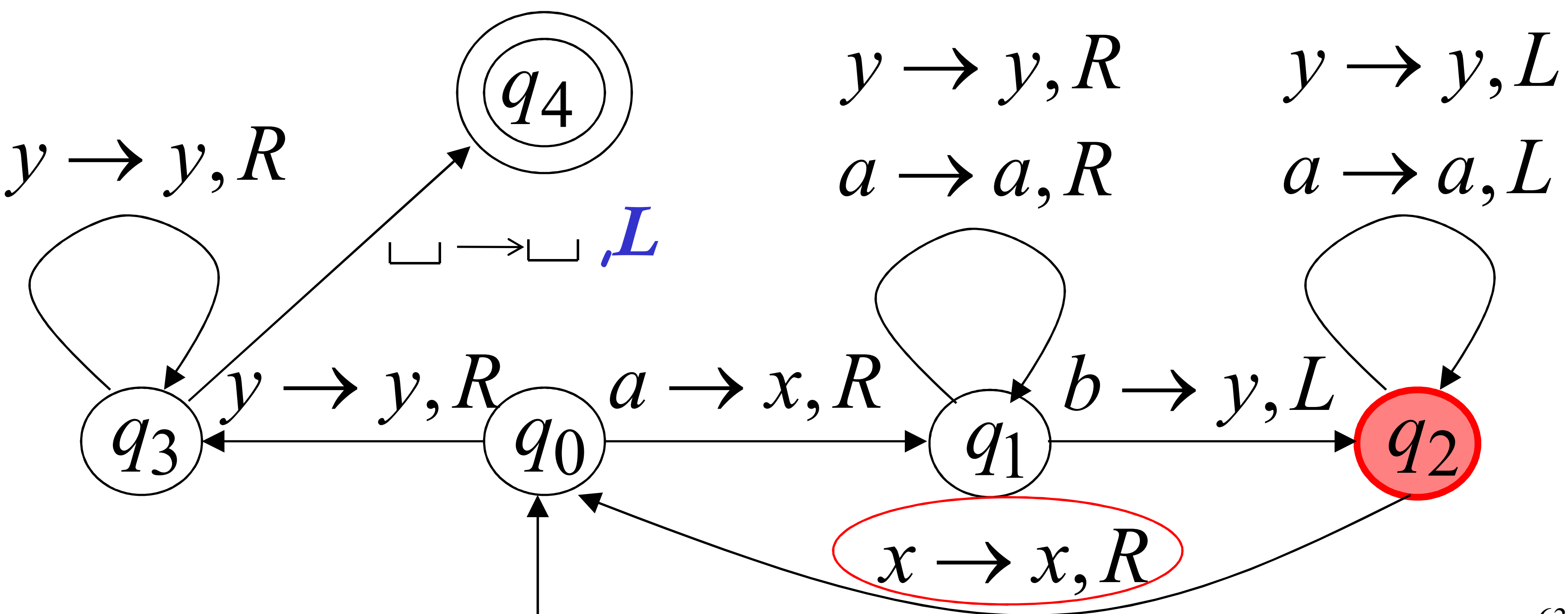


q_4
$y \rightarrow y, R$
$\sqcup \rightarrow \sqcup, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 9

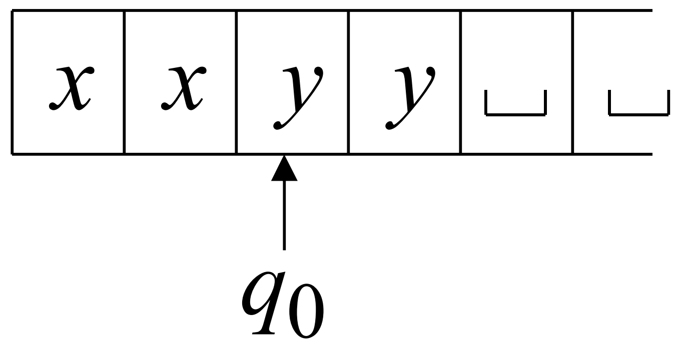


x
x
y
y
⊔
⊔
q_2

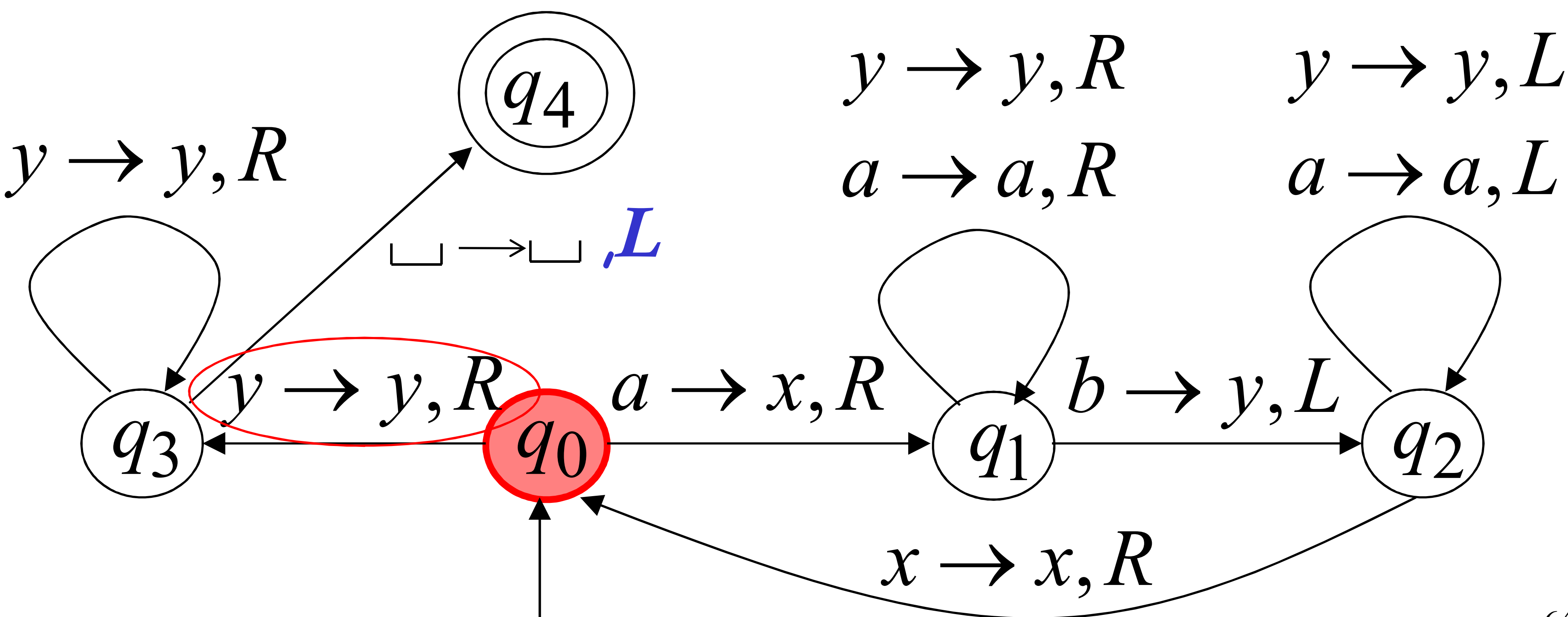


q_4
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$⊔ \rightarrow ⊔, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 10

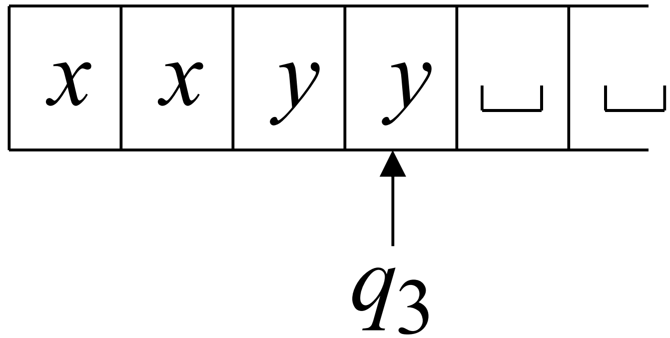


x x y y ␣ ␣
q_0

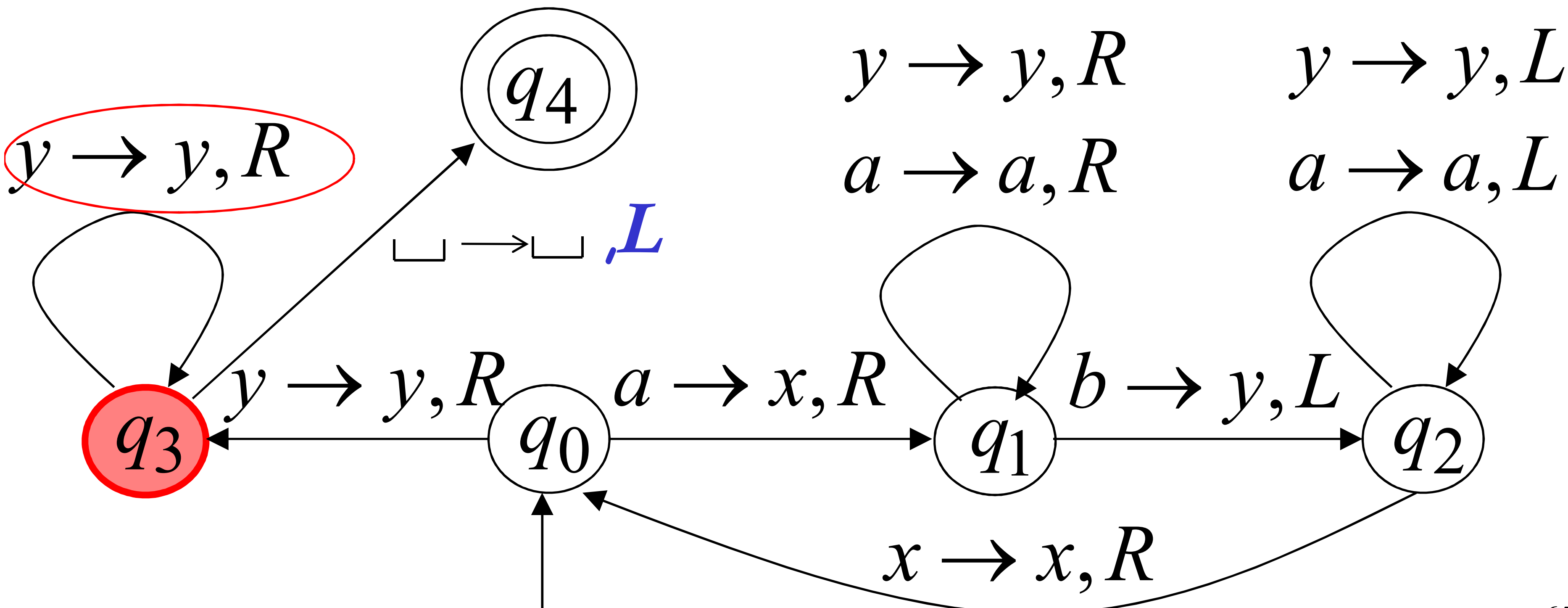


q_4
$y \rightarrow y, R$
$\sqcup \rightarrow \sqcup, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 11

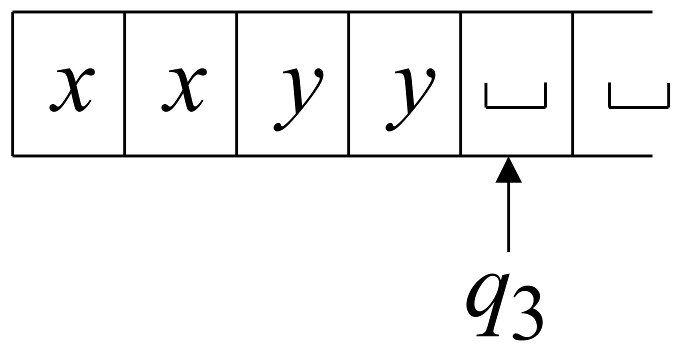


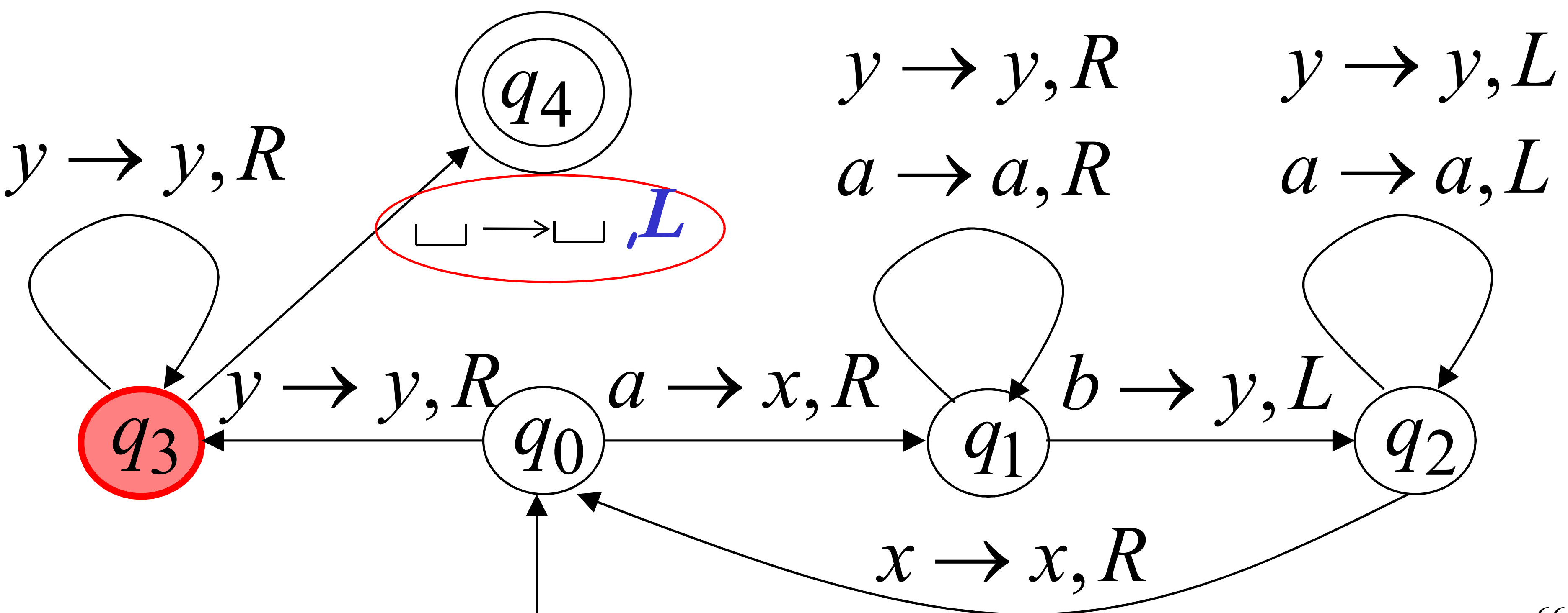
x x y y ␣ ␣
q_3



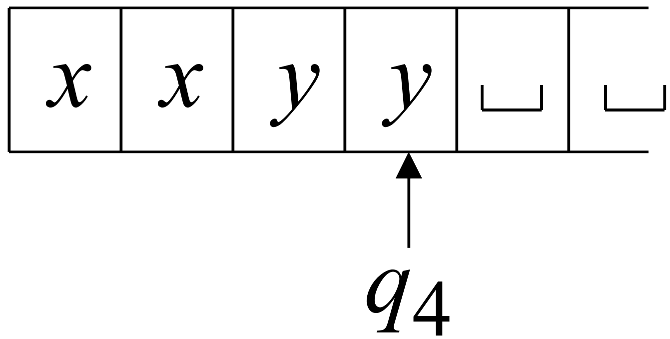
$y \rightarrow y, R$
q_4
␣ → ␣ ,L
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 12

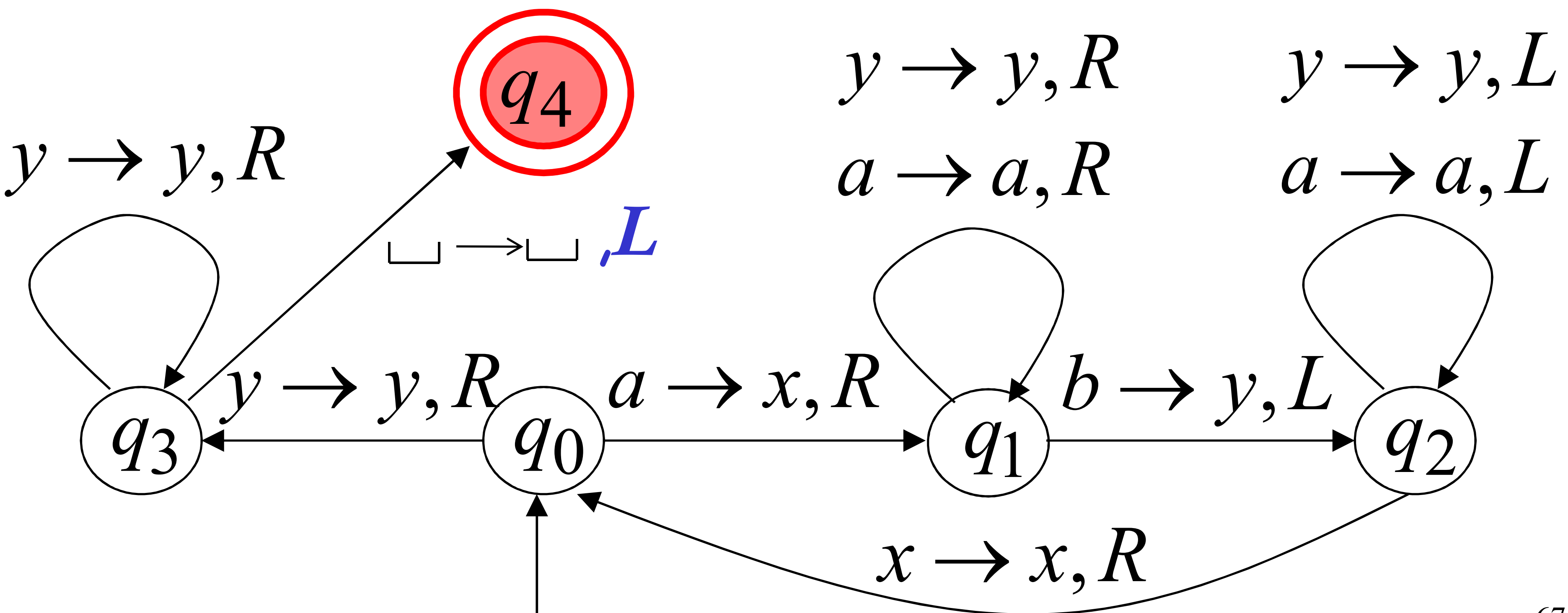




Time 13



Halt & Accept



Observation:

If we modify the
machine for the language $\{a^n b^n\}$

we can easily construct
a machine for the language $\{a^n b^n c^n\}$

Turing Languages

Recursively Enumerable and Recursive Languages

Definition: Let M = $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ be a TM, and let w be a string in Σ^*. Then w is *accepted by M* iff

$$q_0 w \;|\!\!-^* \alpha_1 q_f \alpha_2$$

where q_f is in F and α_1 and α_2 are in Γ^*

Definition: Let M = $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ be a TM. The *language accepted by M*, denoted L(M), is the set

$$\{w \mid w \text{ is in } \Sigma^* \text{ and } w \text{ is accepted by } M\}$$

Notes:

- If x is not in L(M) then M may enter an *infinite loop*, or halt in a *non-final state.*

- Some TMs halt on all inputs, while others may not. In either case the language defined by TM is still well defined.

Definition:

A language is **recursively enumerable** if some Turing machine accepts it

Let L be a recursively enumerable language

and M the Turing Machine that accepts it

For string w :

if $w \in L$ then M halts in a final state

if $w \notin L$ then M halts in a non-final state

or loops forever

This definition implies only that there exists a TM M, such that for every $w \in L$,

$$q_0 \, w \ |\!-^* \ x_1 \, q_f \, x_2$$

The definition says nothing about what happens for $w \notin L$;

…it may be that the machine halts in a non- final sate or it never halts and goes into an infinite loop.

Definition:

A language is **recursive**
if some Turing machine accepts it
and halts on ***any*** input string

In other words:

A language is recursive if there is
a membership algorithm for it

Let L be a recursive language

and M the Turing Machine that accepts it

For string w:

if $w \in L$ then M halts in a final state

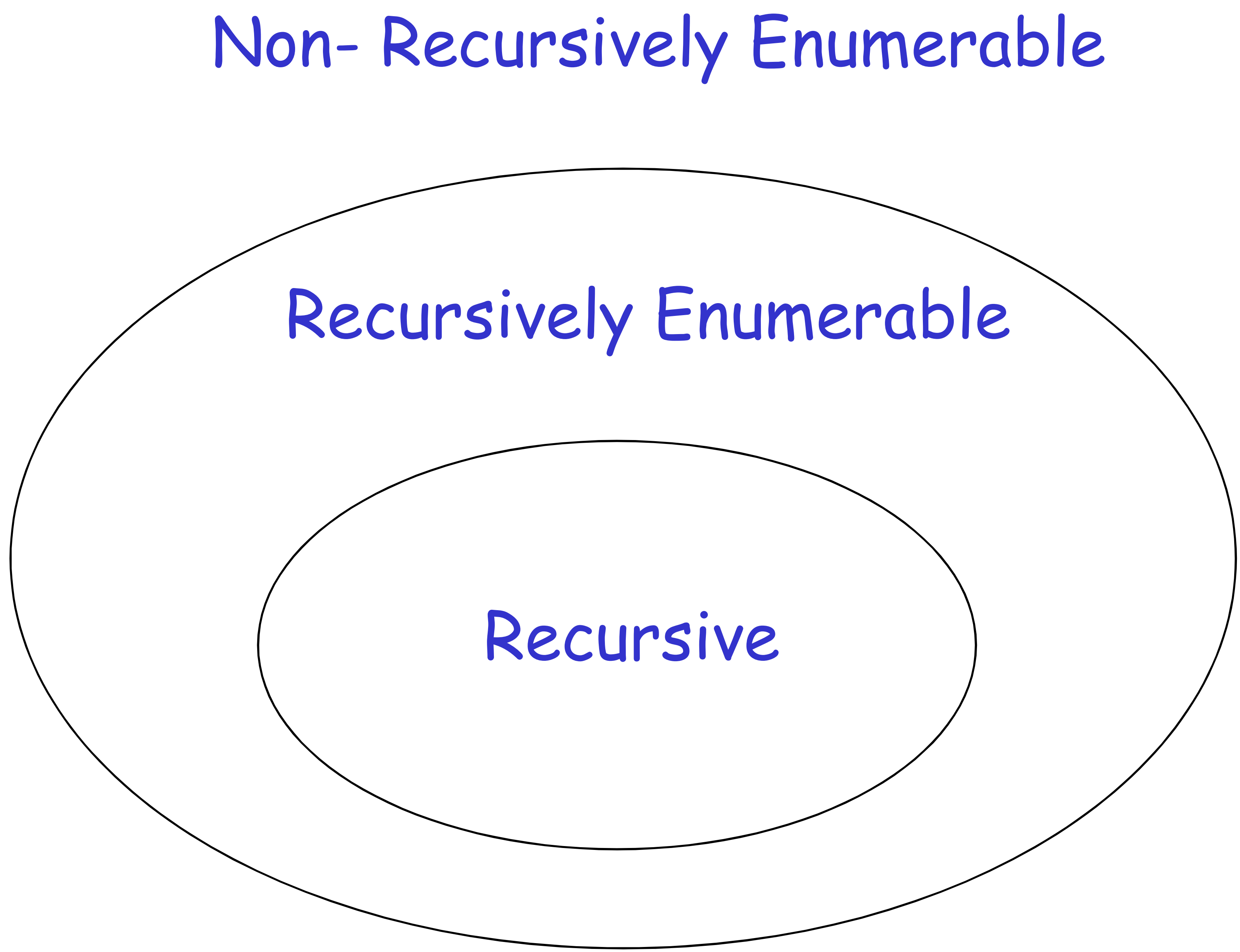
if $w \notin L$ then M halts in a non-final state

What do you think is the advantage of being recursive language over recursively enumerable language?

Assuming you are presented a string w, and you would like to know whether w is in the language.

Examples of Recursively Enumerable Languages

- $L = \{w \in \{a, b\}^*$: w contains at least one **a**}
- $L = \{w \in \{a, b\}^*$: w contains a double **a**}


Non- Recursively Enumerable
Recursively Enumerable
Recursive

Notes:

The set of all recursive languages is a subset of the set of all recursively enumerable languages

A TM is not recursive or recursively enumerable, rather a *language* is recursive or recursively enumerable.