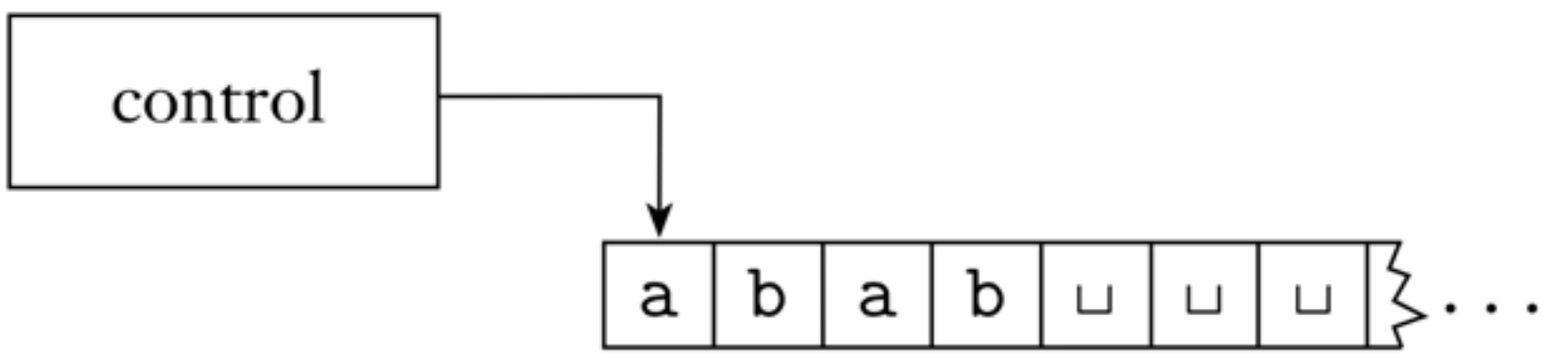
CS 321

Theory of Computation

Part III: Turing Machines and (Un-)Decidability

Sipser 3-4; Linz 9, 12

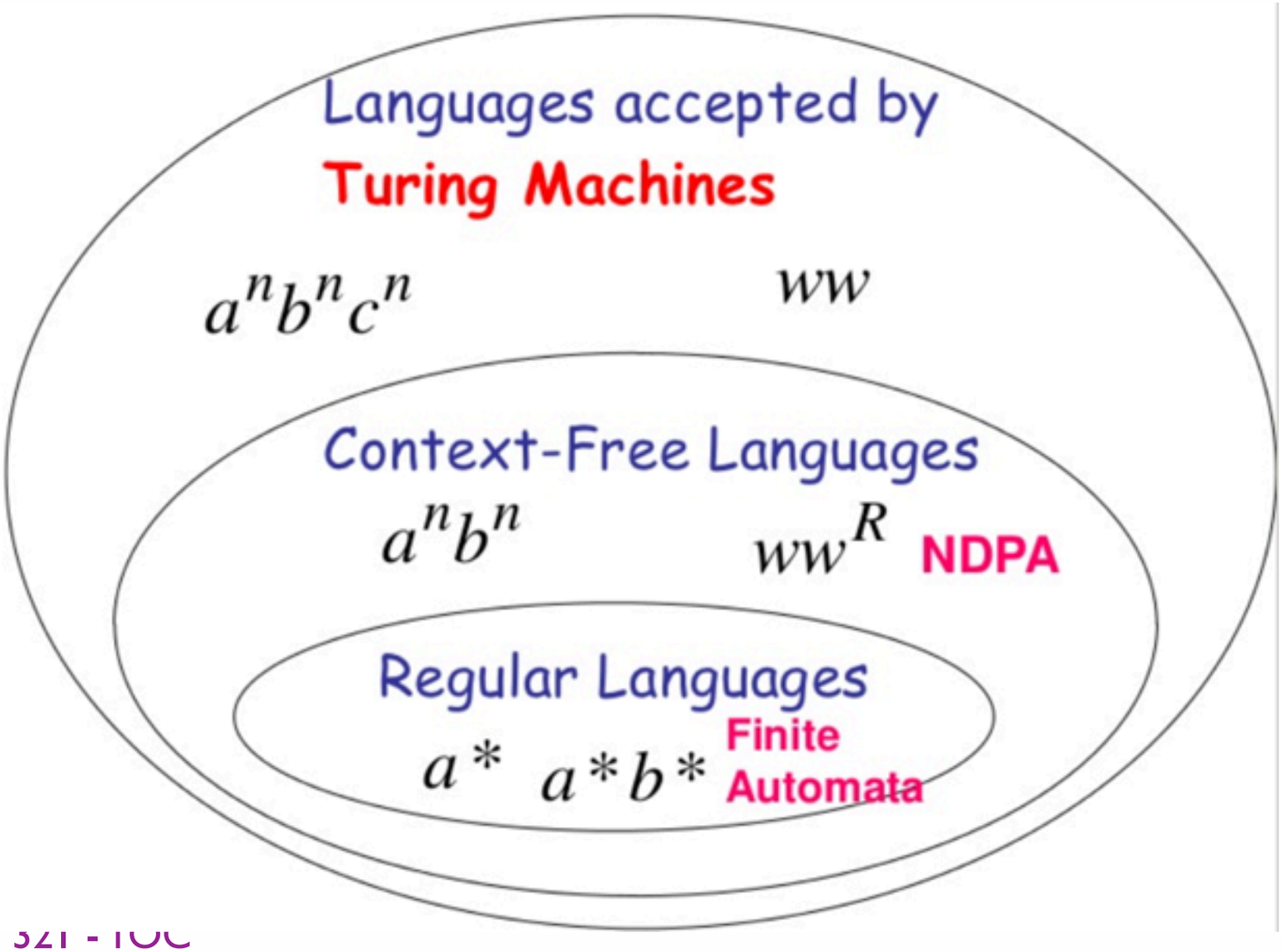


Instructor: Liang Huang

(some slides courtesy of H. Potter)

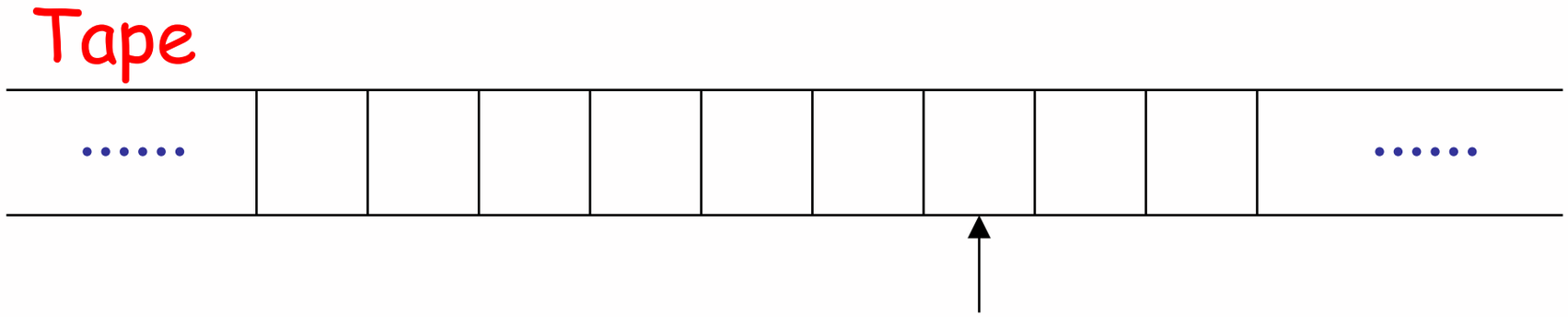
Turing Machines

- idealized model of real computers (Alan Turing, 1936)
- strictly more powerful than context-free grammars and PDAs
- can perform any calculation that can be performed by any computing machine



Turing Machines

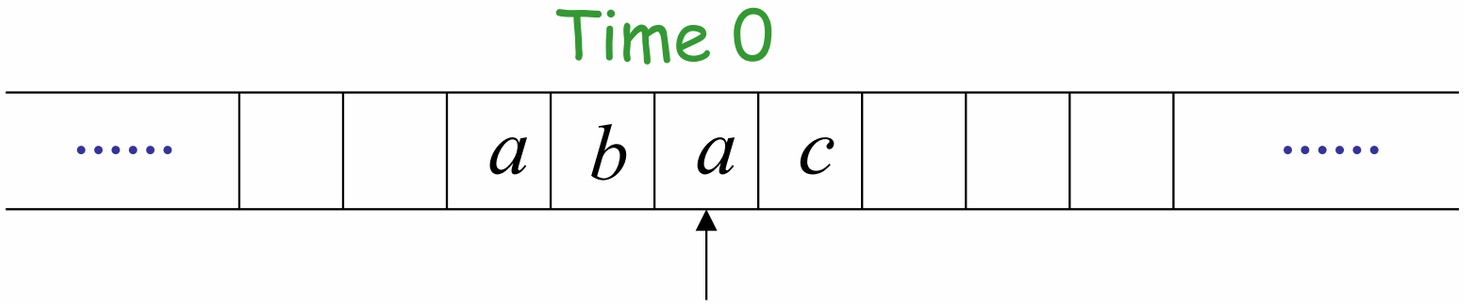
- the read/write head moves left or right on the infinite tape
- the control unit maintains state transition (like DFA)

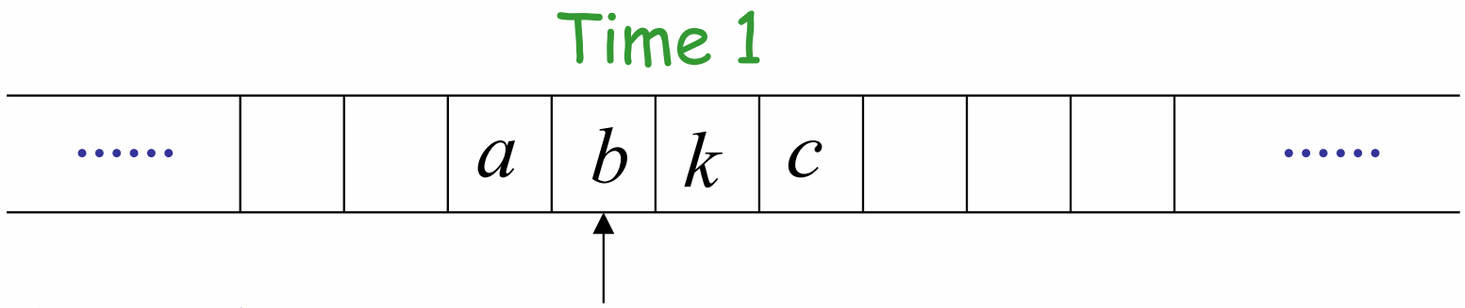


The head at each time step:

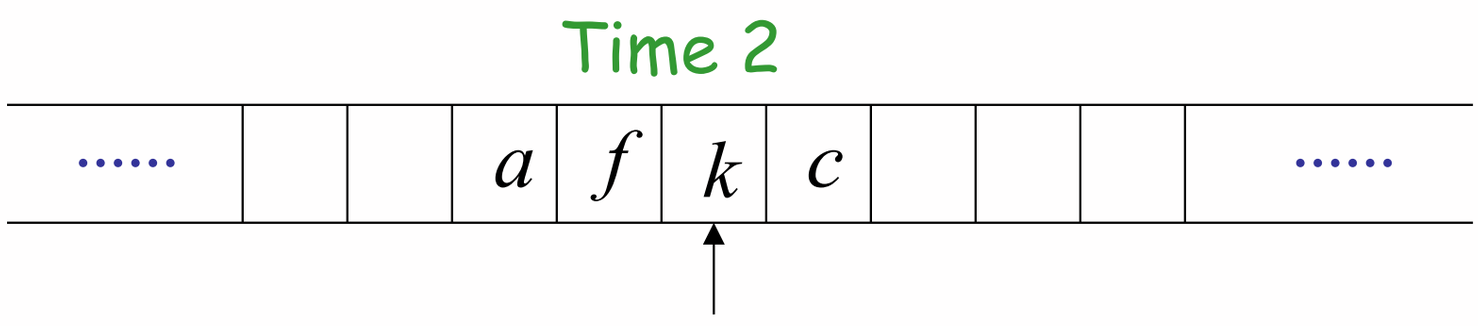
1. Reads a symbol
2. Writes a symbol
3. Moves Left or Right

Example



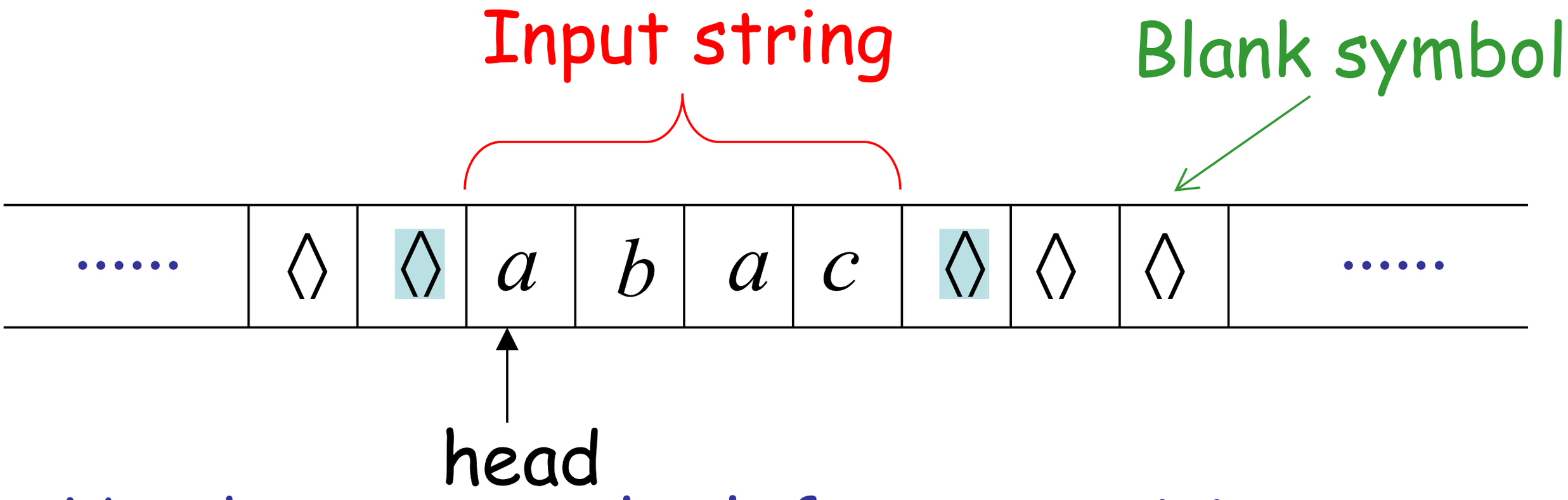


1. Reads a
2. Writes k
3. Moves Left



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

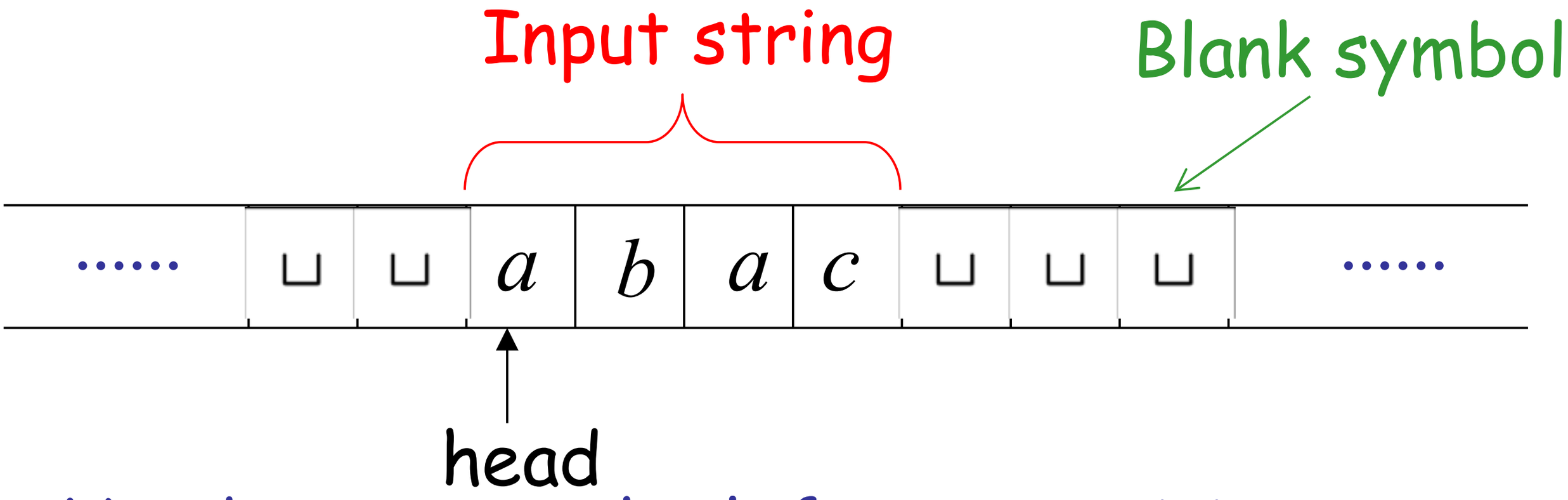
The Input String



Head starts at the leftmost position of the input string

- ⟨⟩ **Are treated as left and right brackets for the input written on the tape.** Sipser: ␣ Linz:□

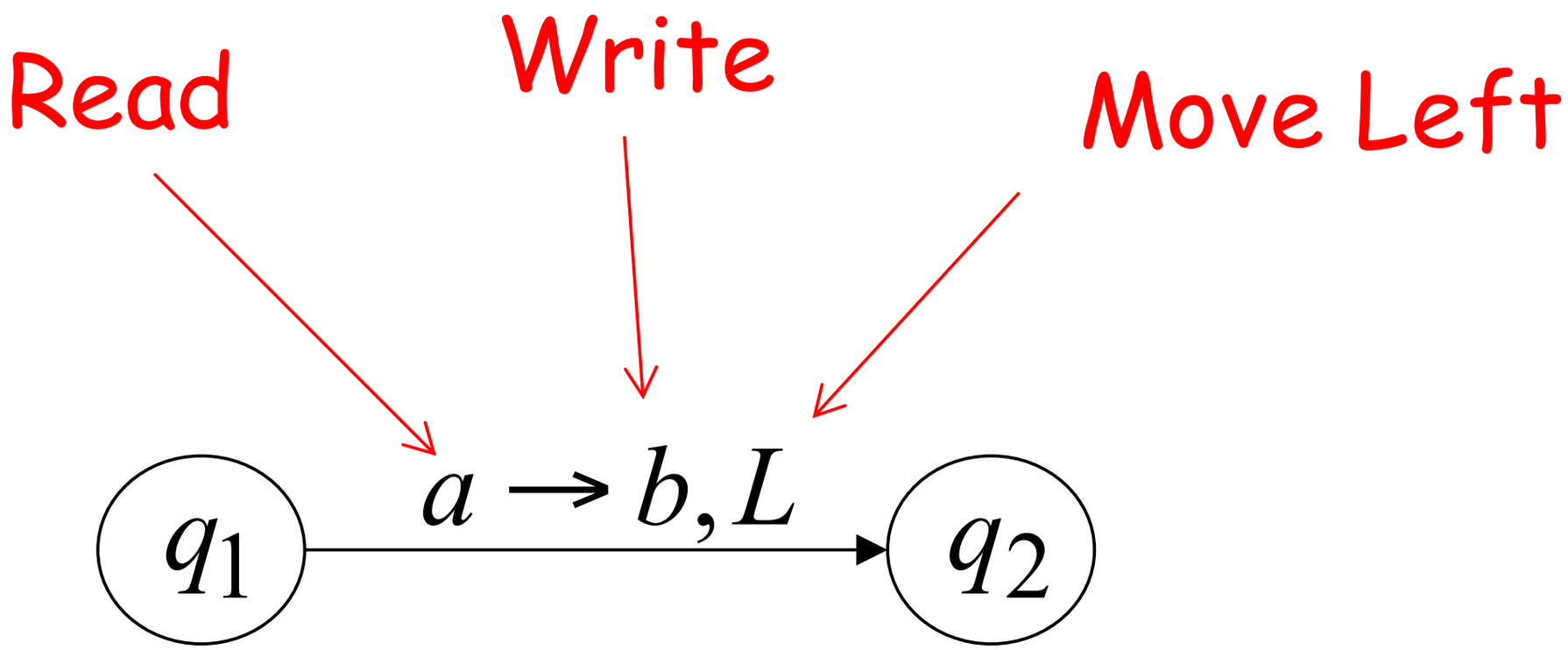
The Input String

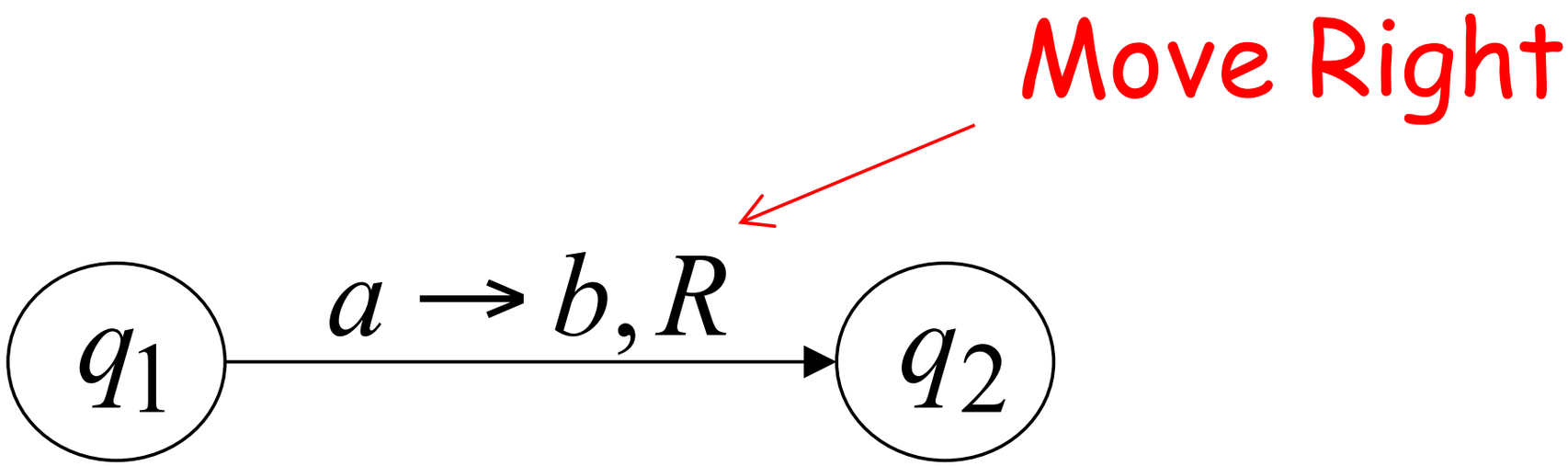


Head starts at the leftmost position of the input string

- ⟨⟩ **Are treated as left and right brackets for the input written on the tape.** Sipser: ⊔ Linz:□

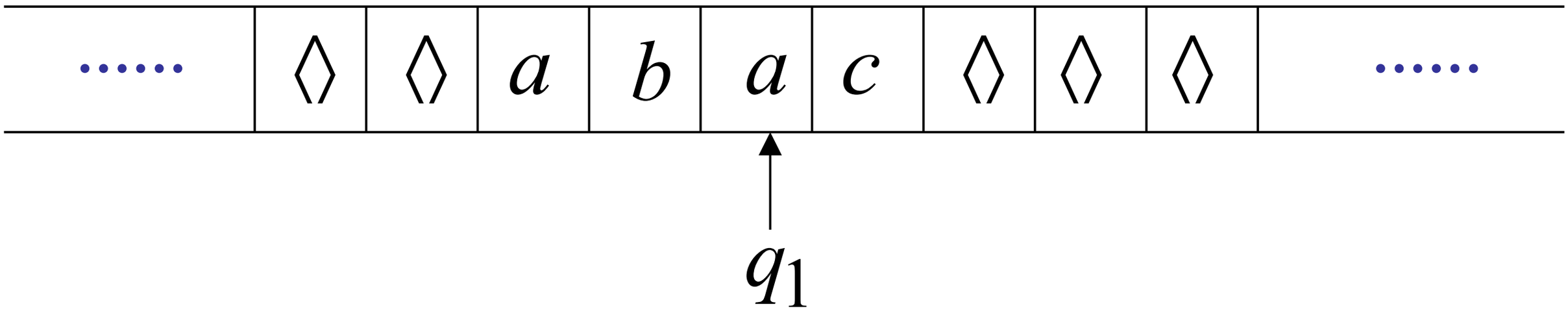
States & Transitions



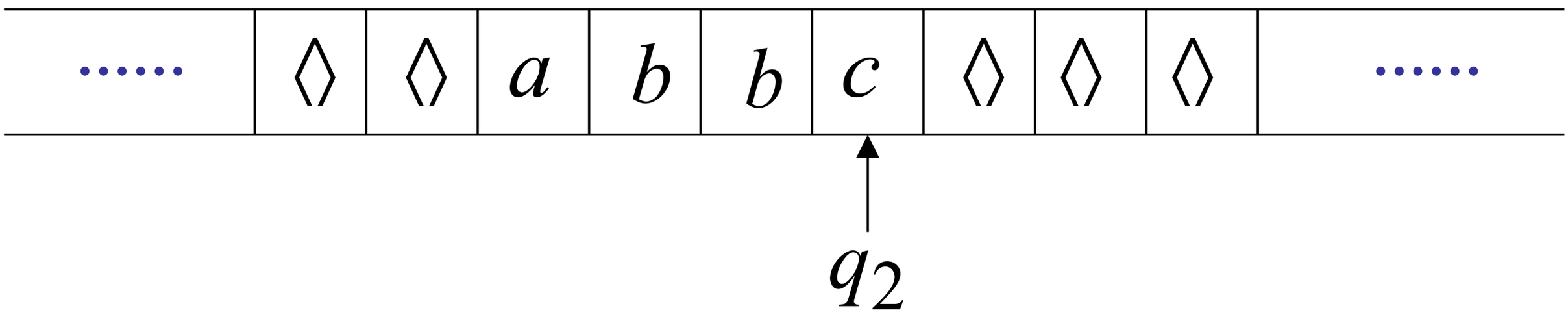


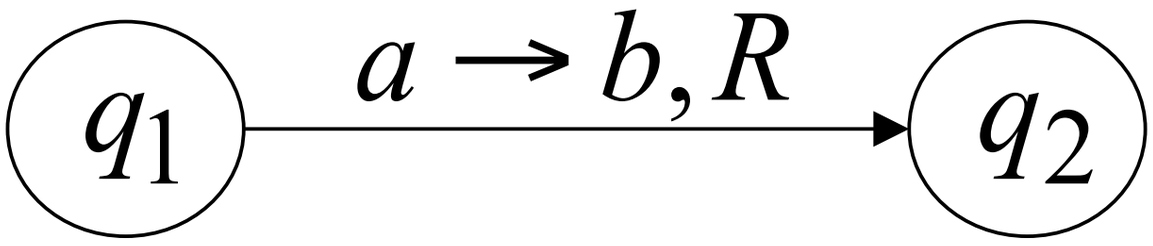
Example:

Time 1



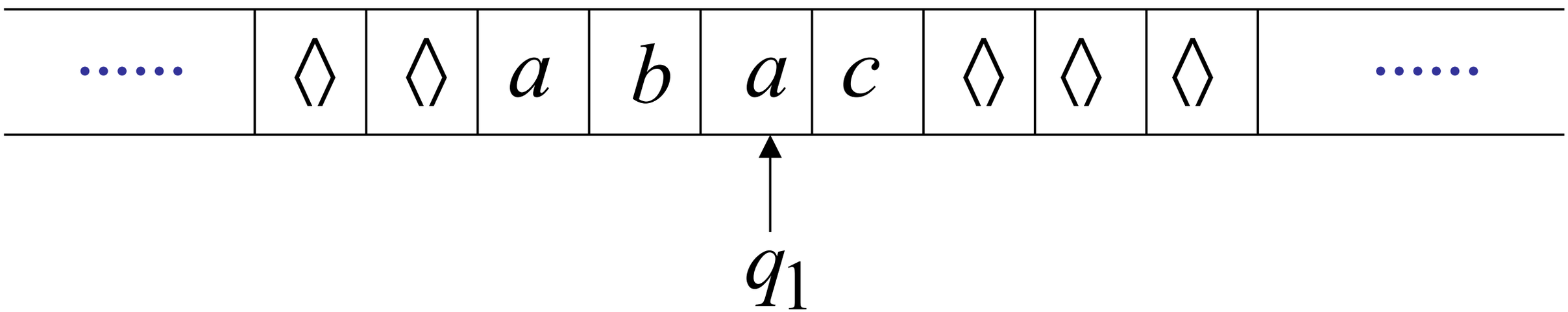
Time 2



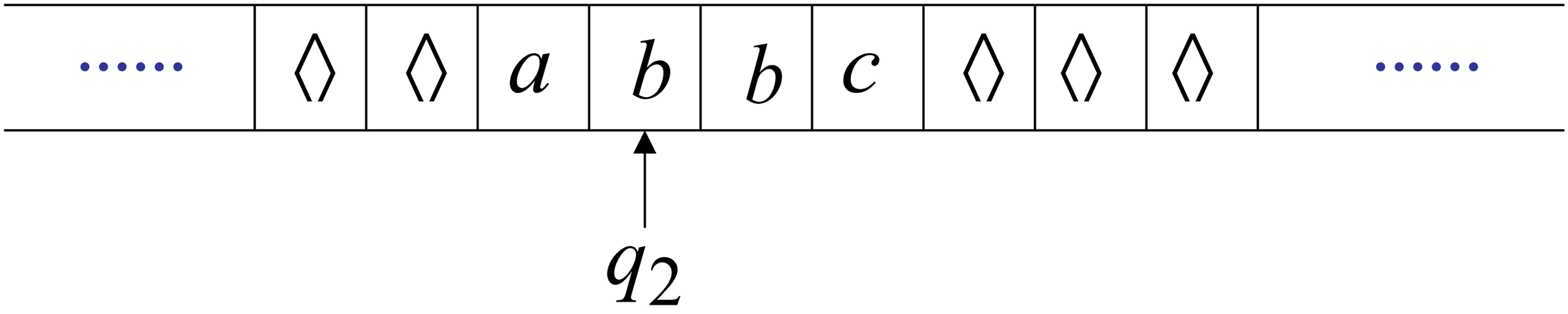


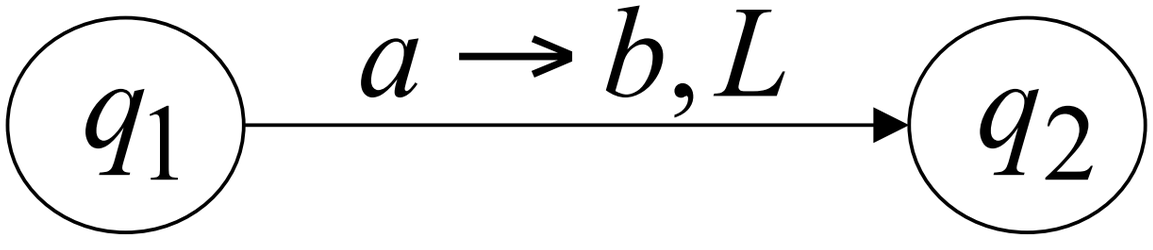
Example:

Time 1



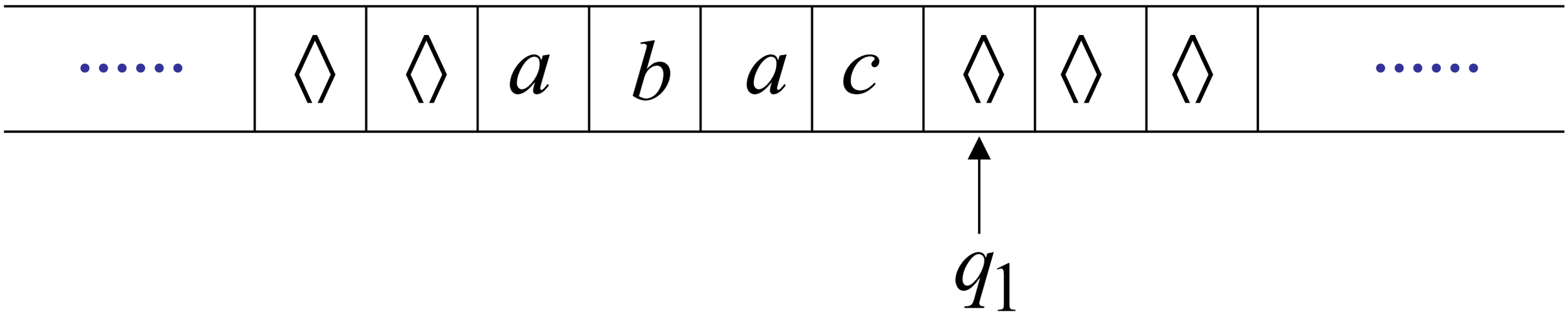
Time 2



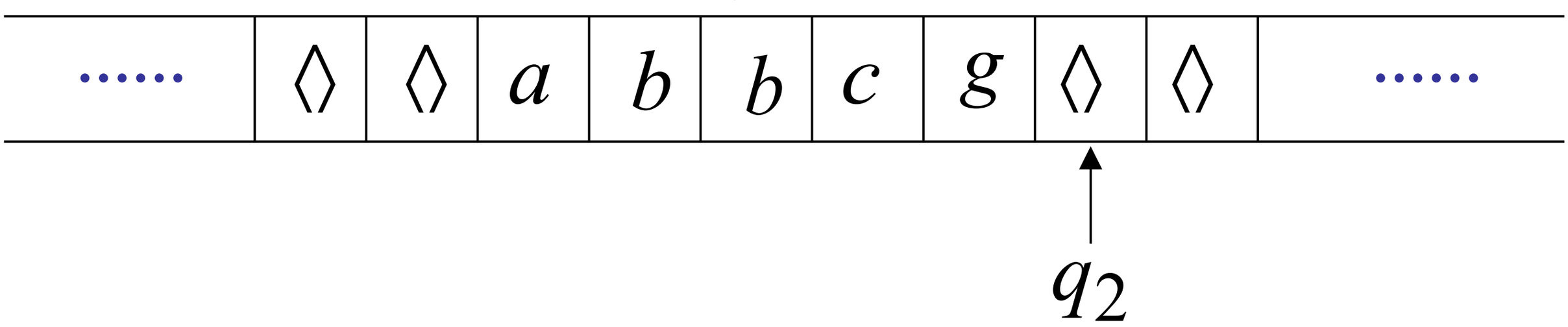


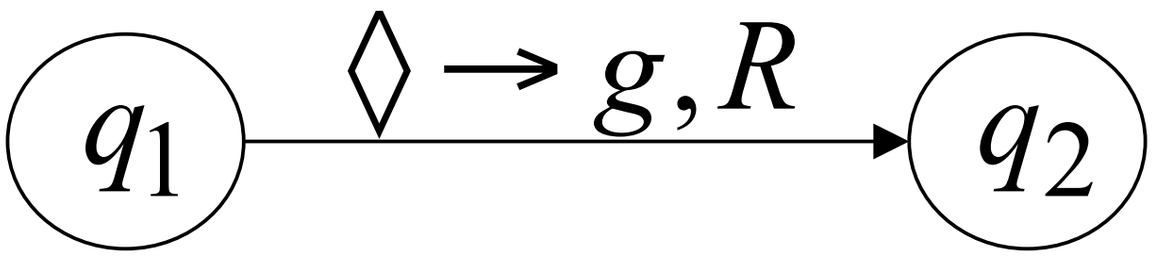
Example:

Time 1



Time 2

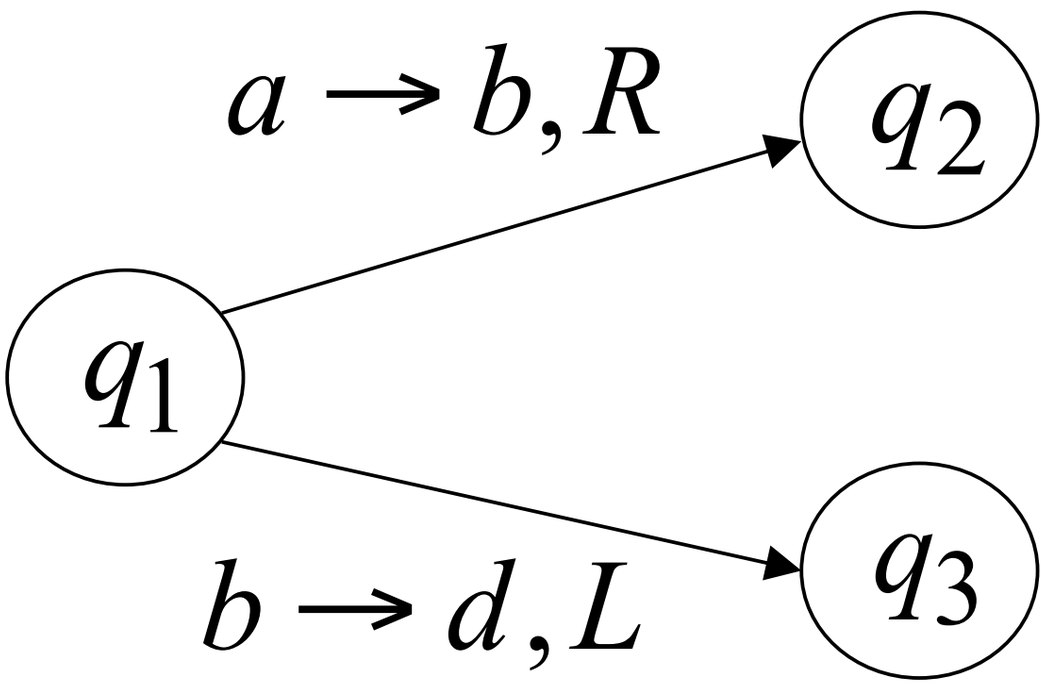




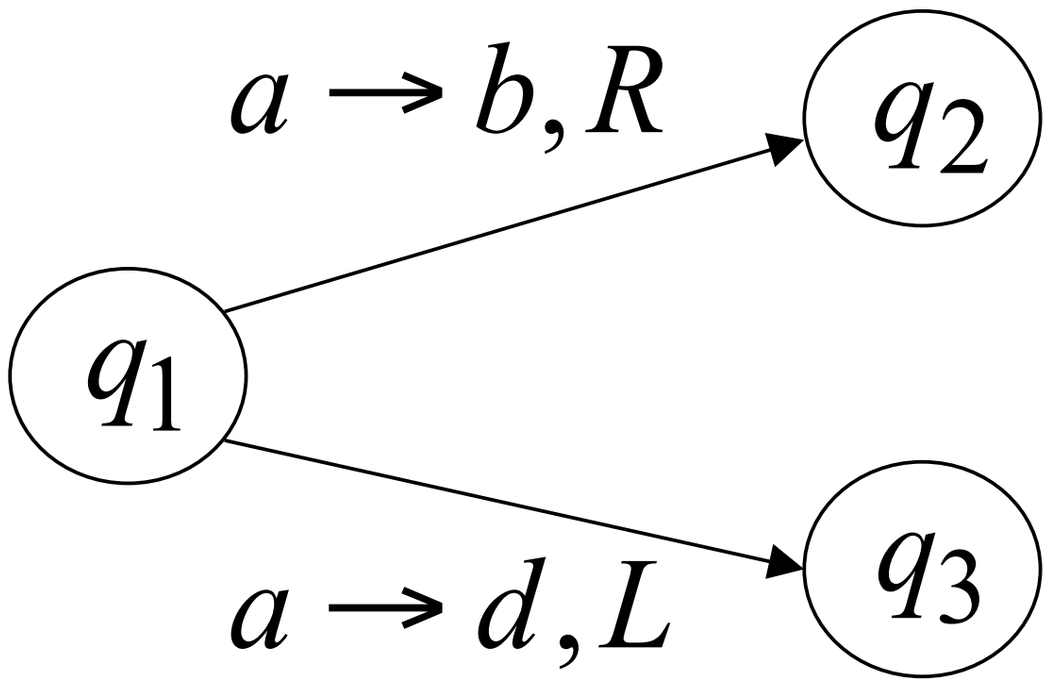
Determinism

Turing Machines are deterministic

Allowed



Not Allowed



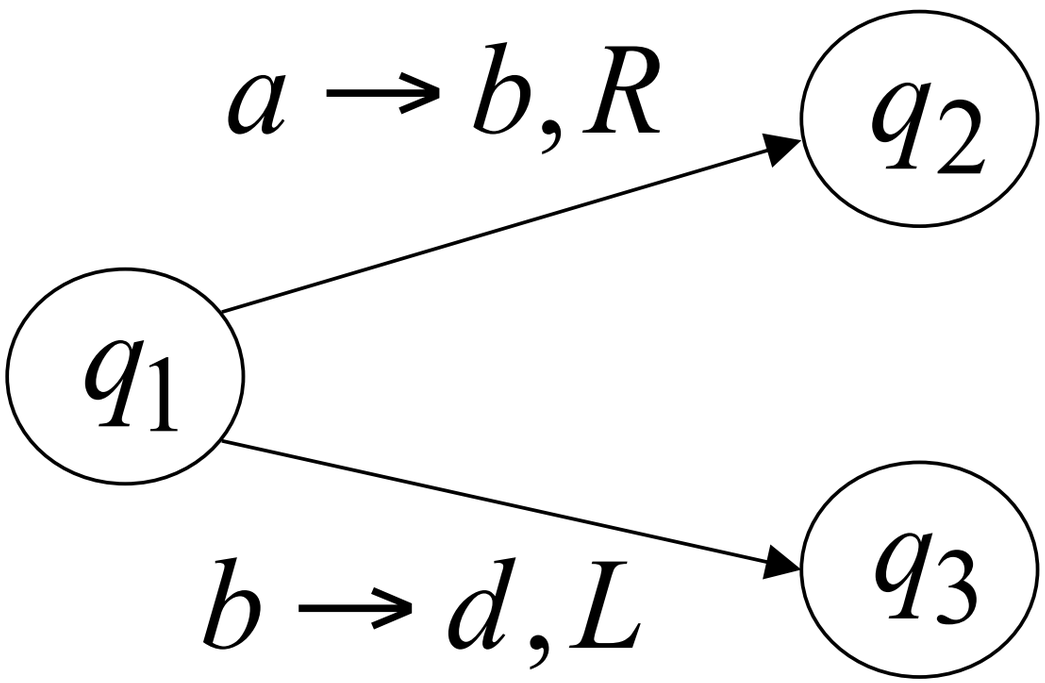
No epsilon transitions allowed

Determinism

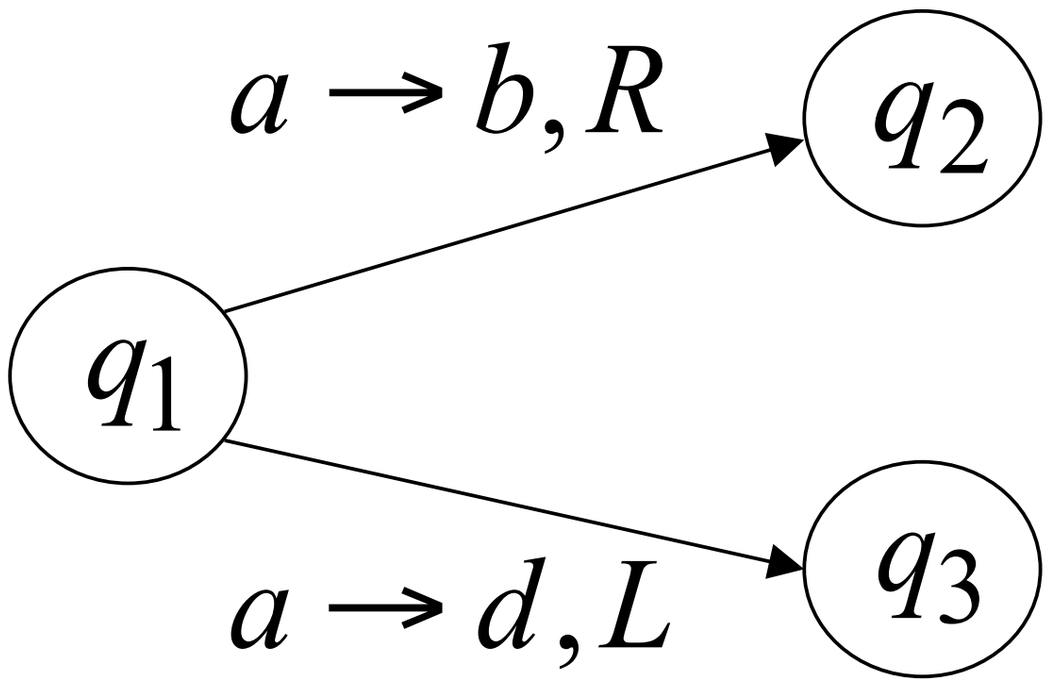
Turing Machines are deterministic

non-deterministic Turing machines are equally powerful!

Allowed



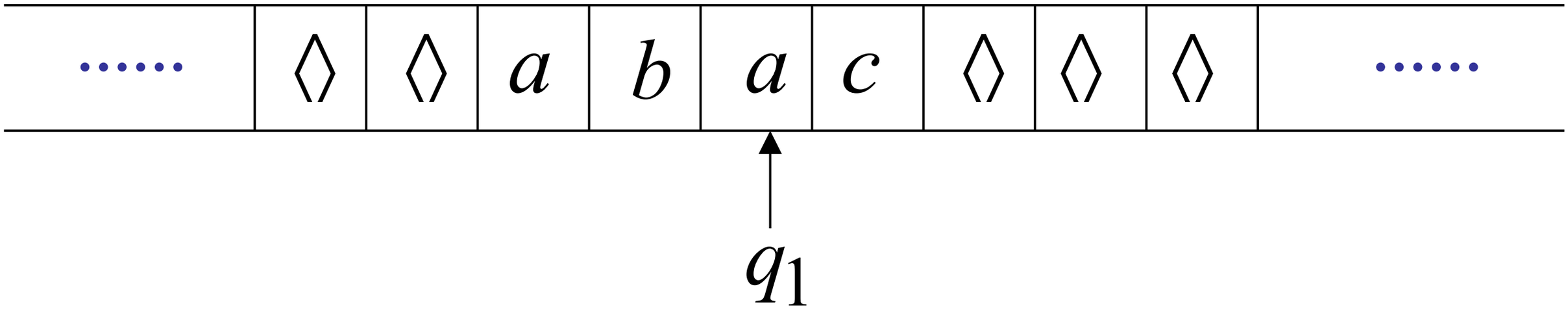
Not Allowed

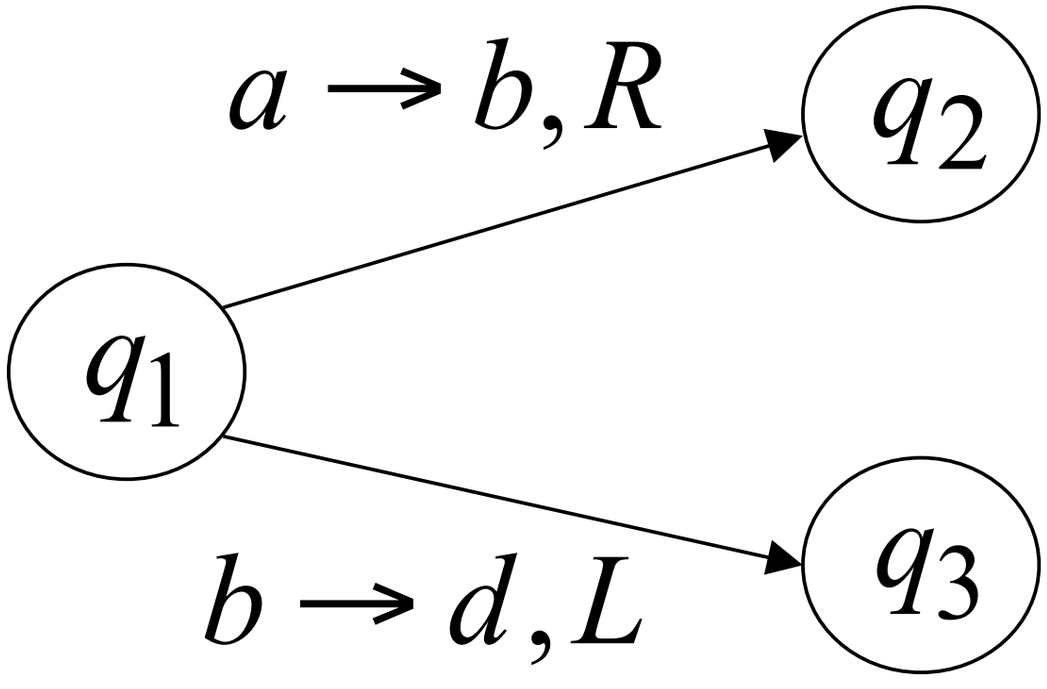


No epsilon transitions allowed

Partial Transition Function

Example:





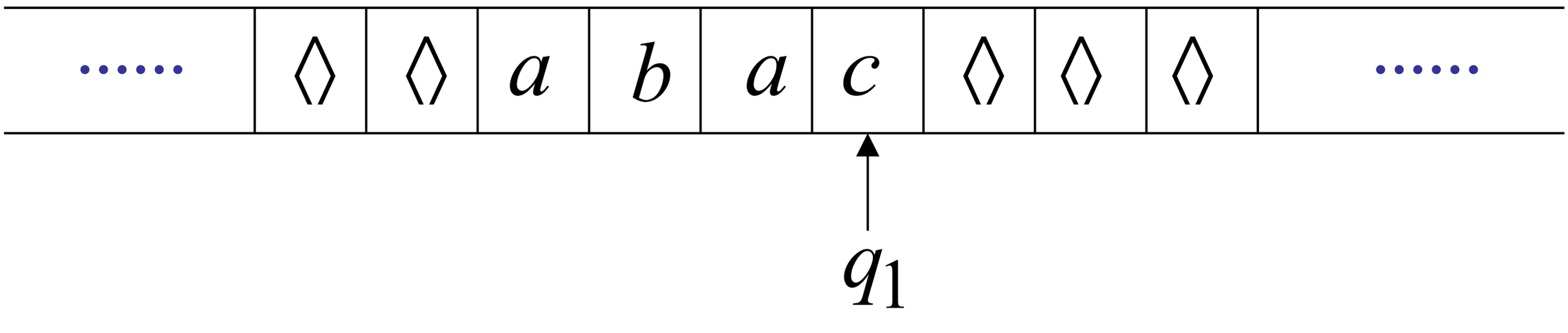
Allowed:

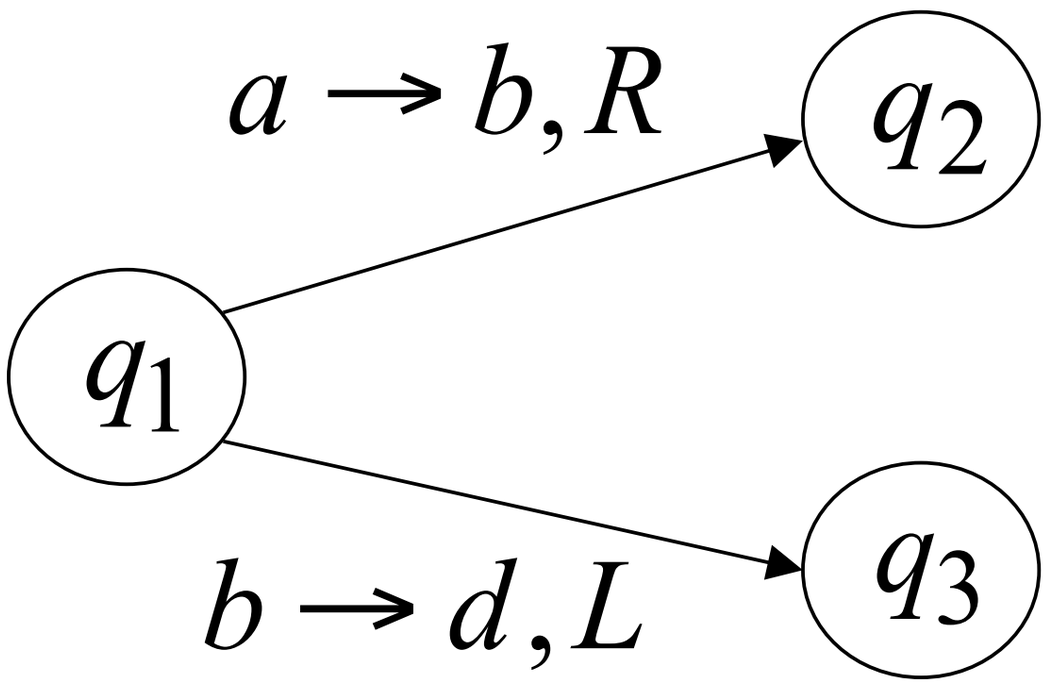
No transition
for input symbol c

Halting

The machine ***halts*** if there are no possible transitions to follow

Example:

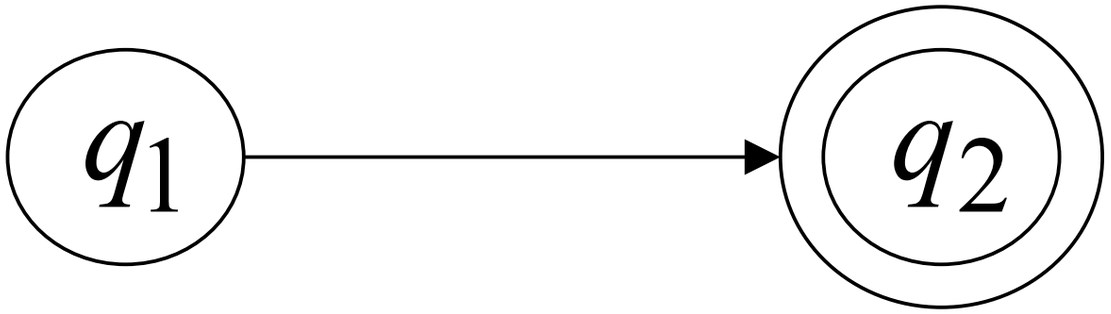




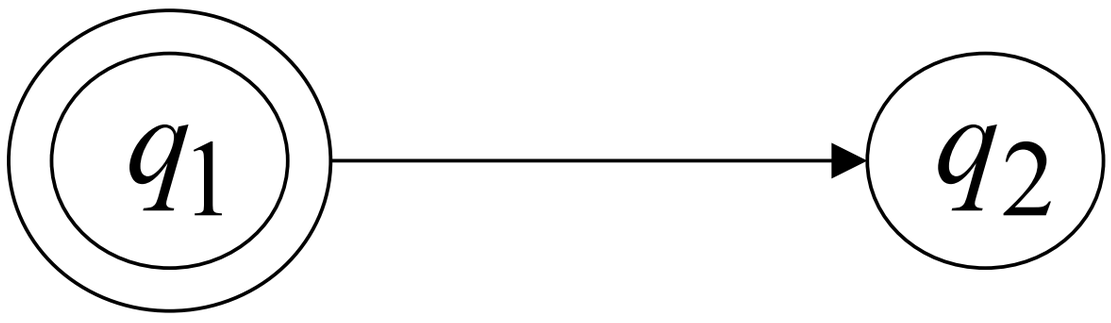
No possible transition

HALT!!!

Final States



Allowed

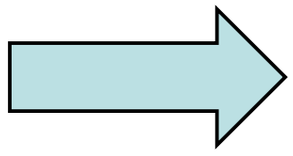


Not Allowed

- Final states have no outgoing transitions
- In a final state the machine halts

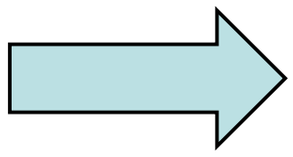
Acceptance

Accept Input



If machine halts in a final state

Reject Input



If machine halts in a non-final state

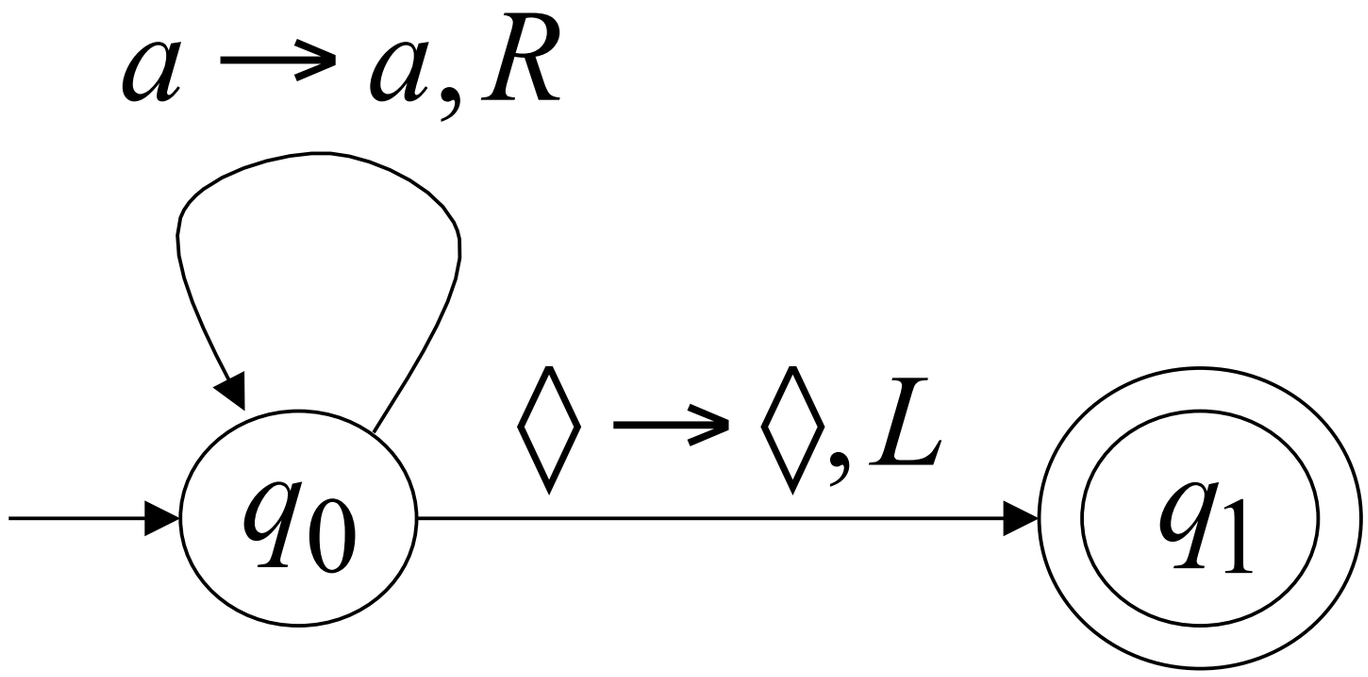
or

If machine enters an *infinite loop*

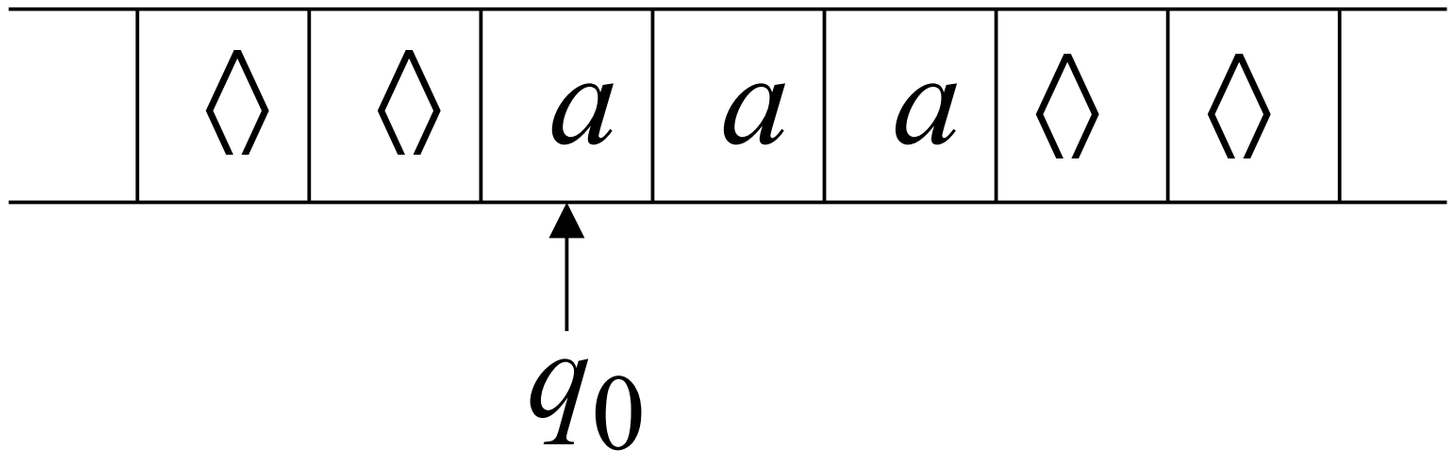
Turing Machine Example

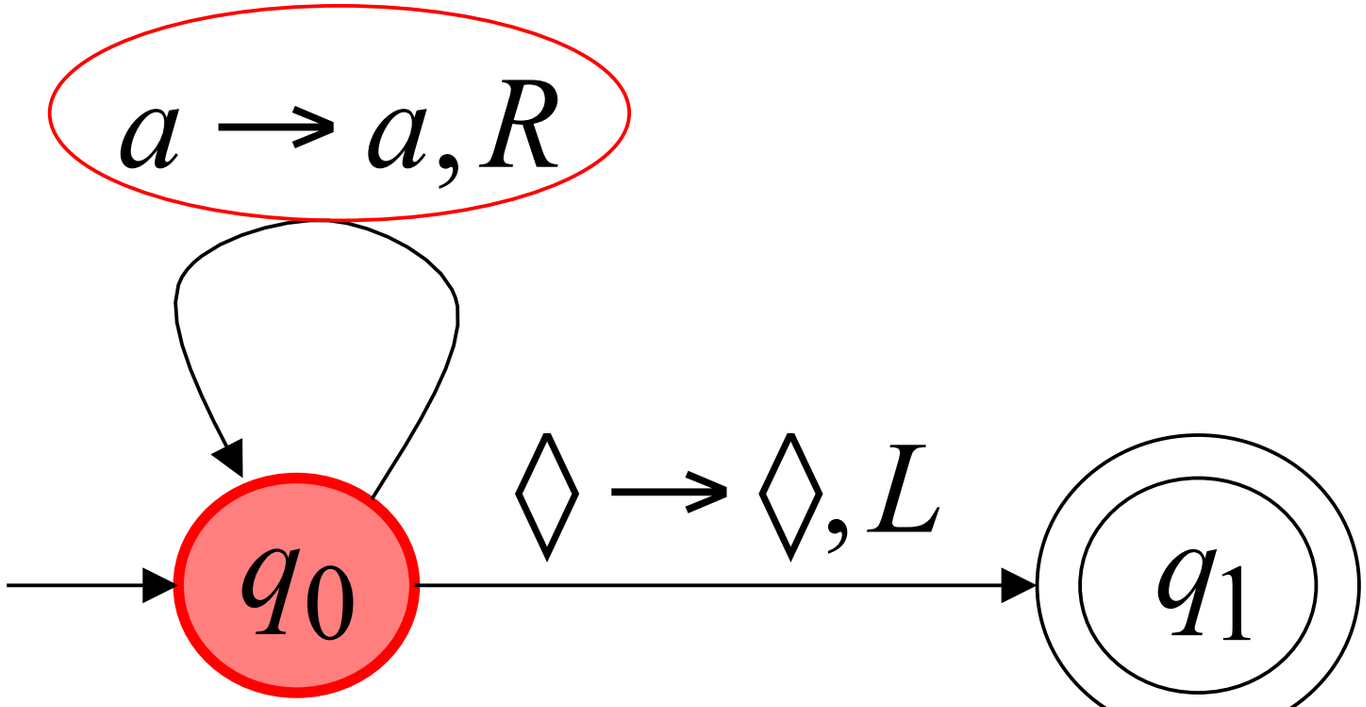
A Turing machine that accepts the language:

$$a^*$$

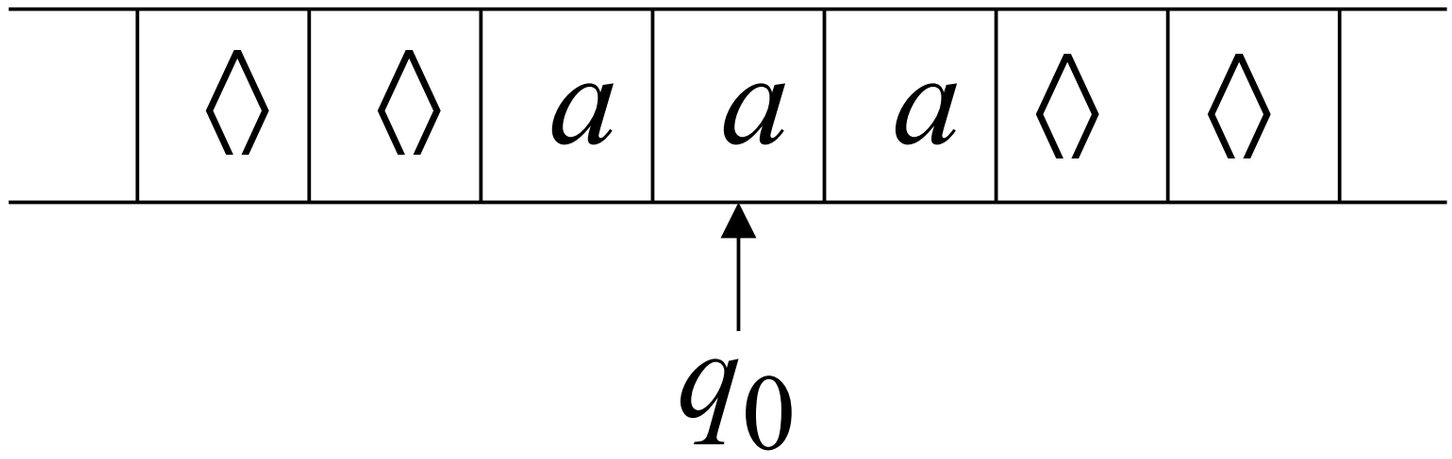


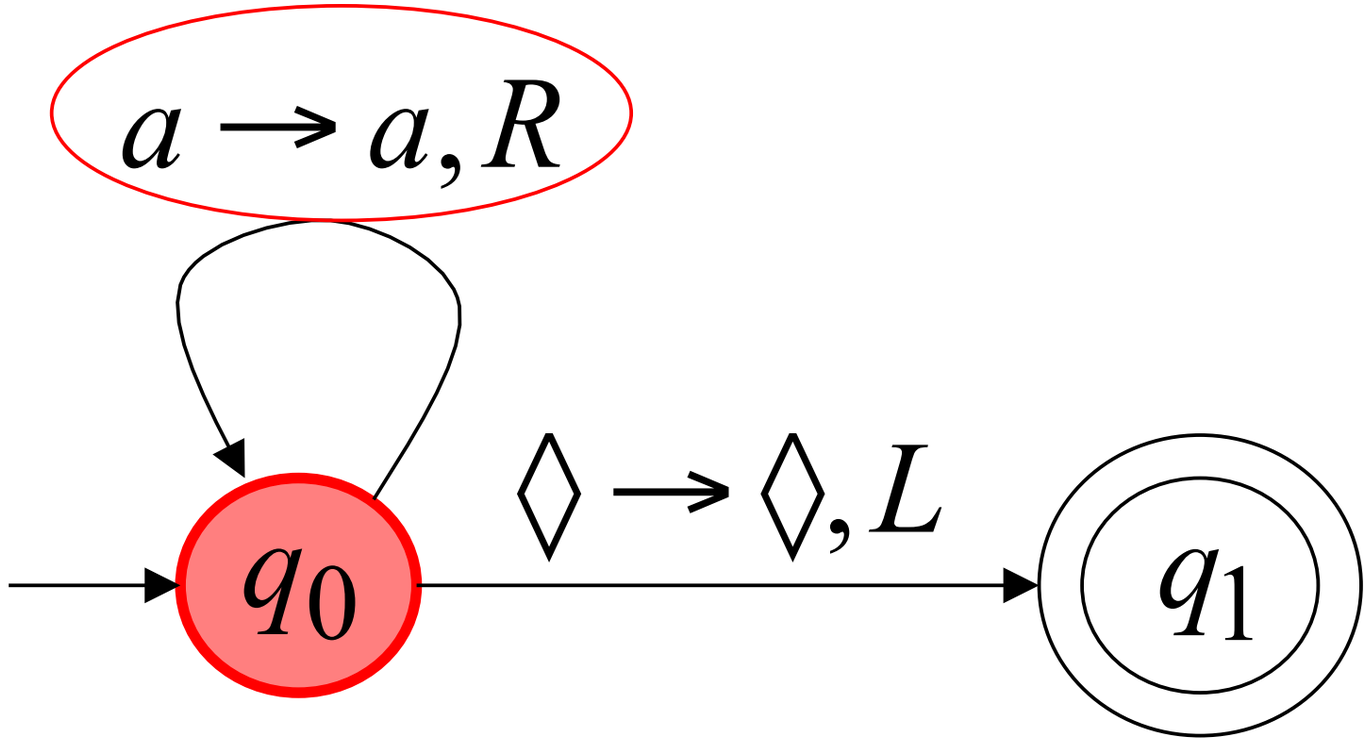
Time 0



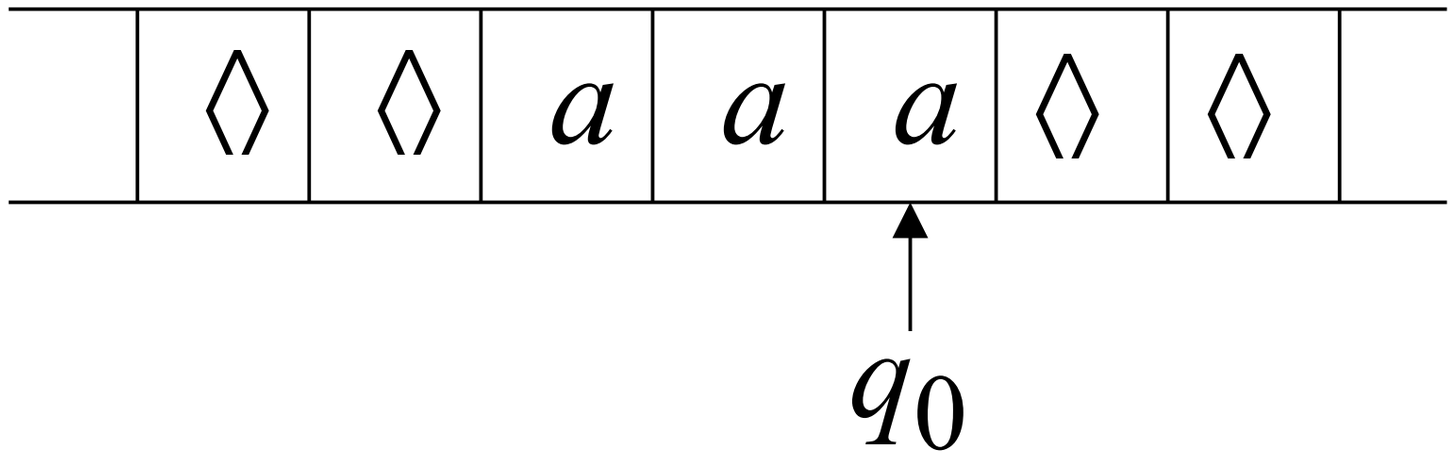


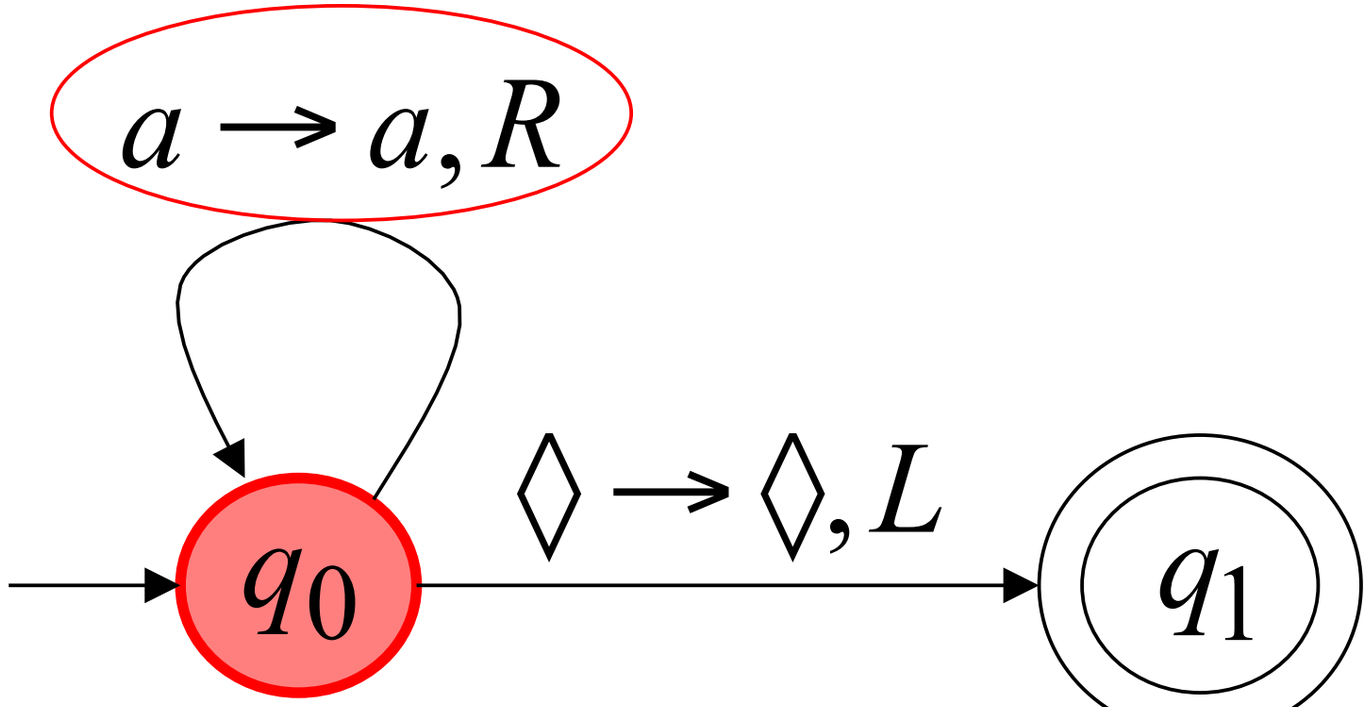
Time 1



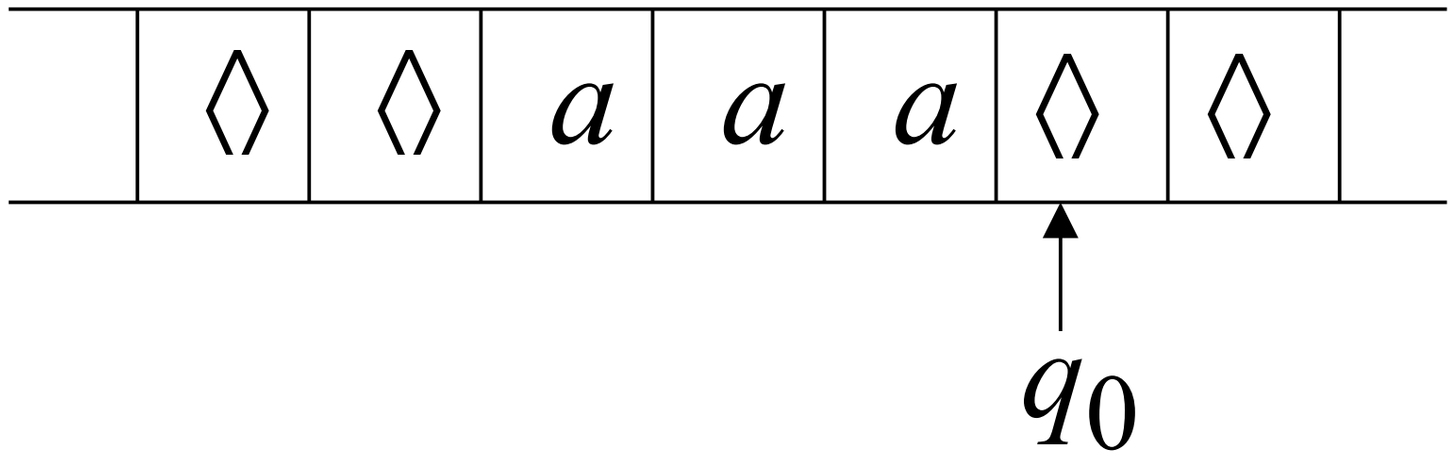


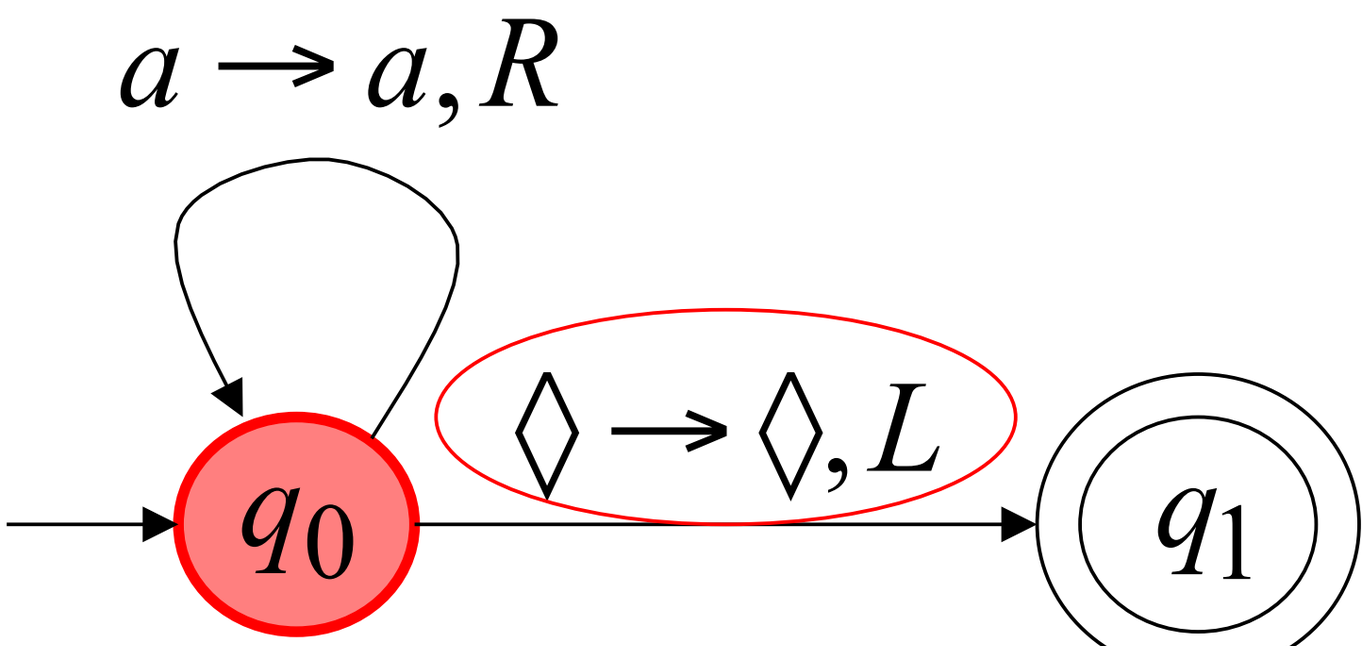
Time 2



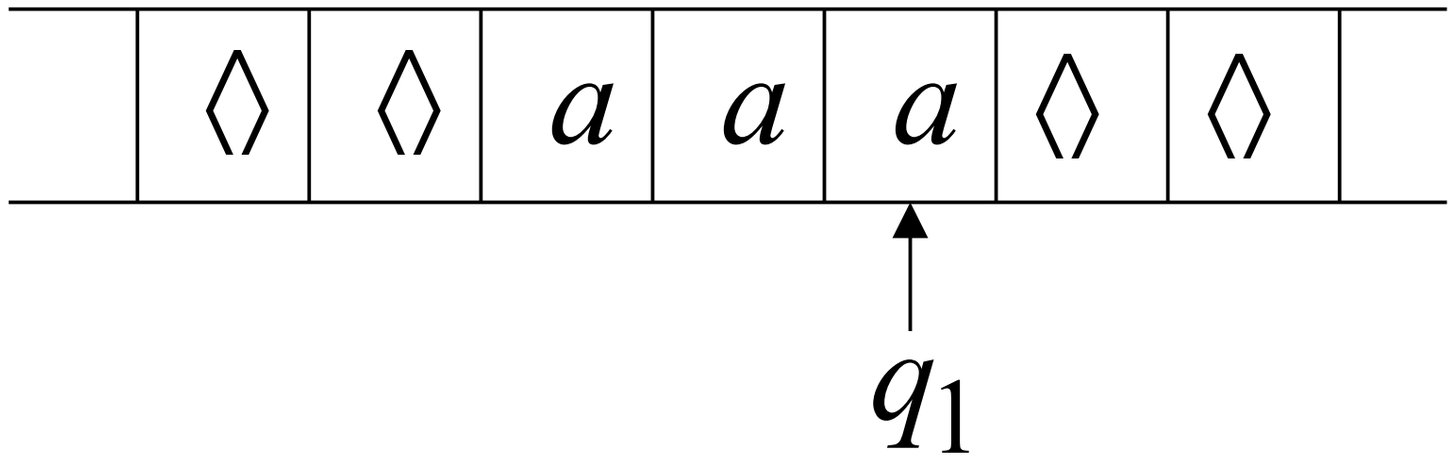


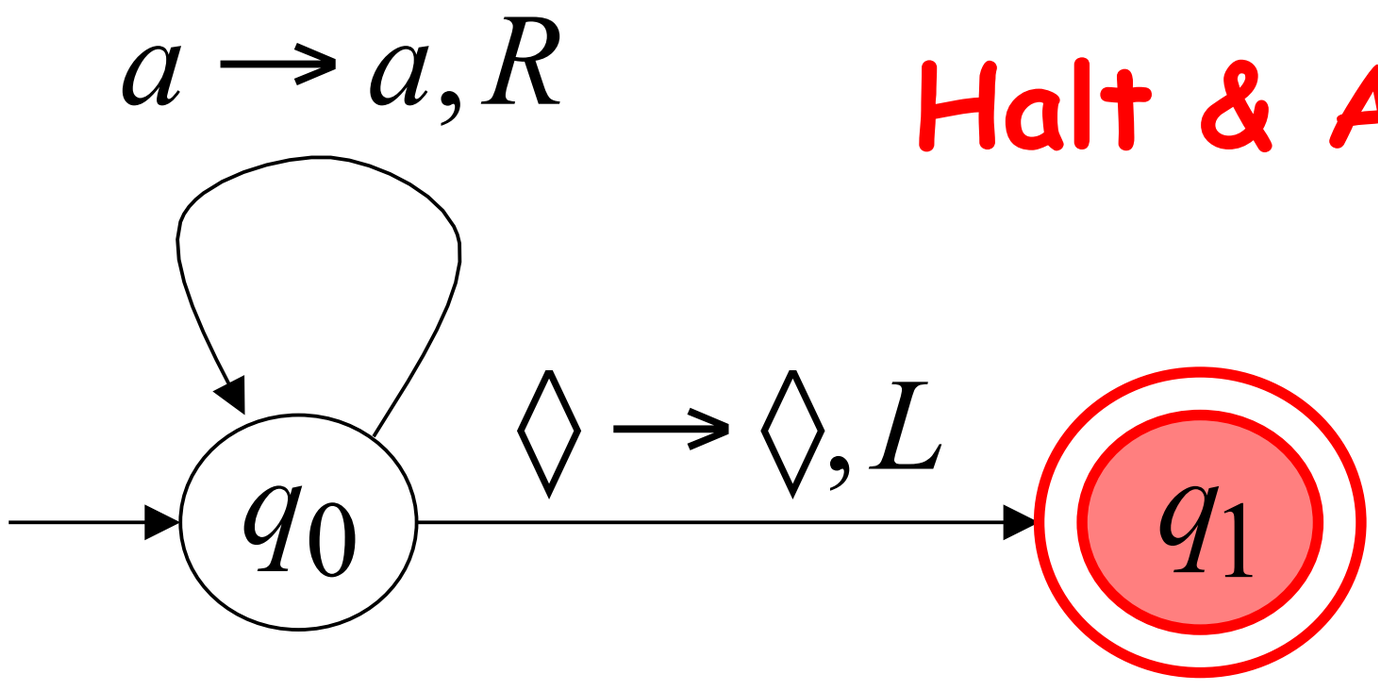
Time 3





Time 4

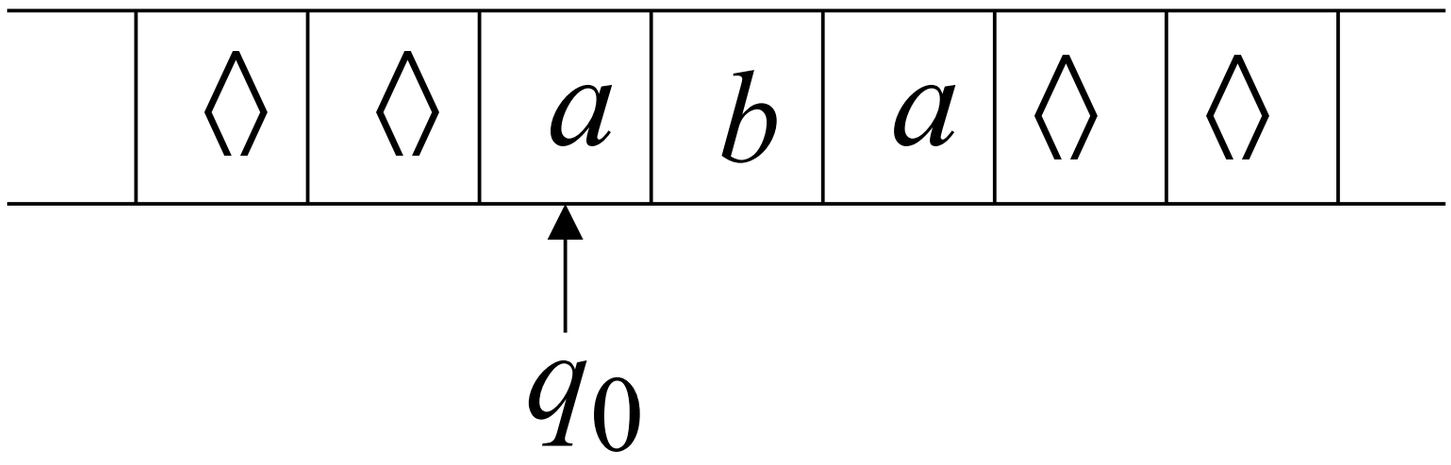


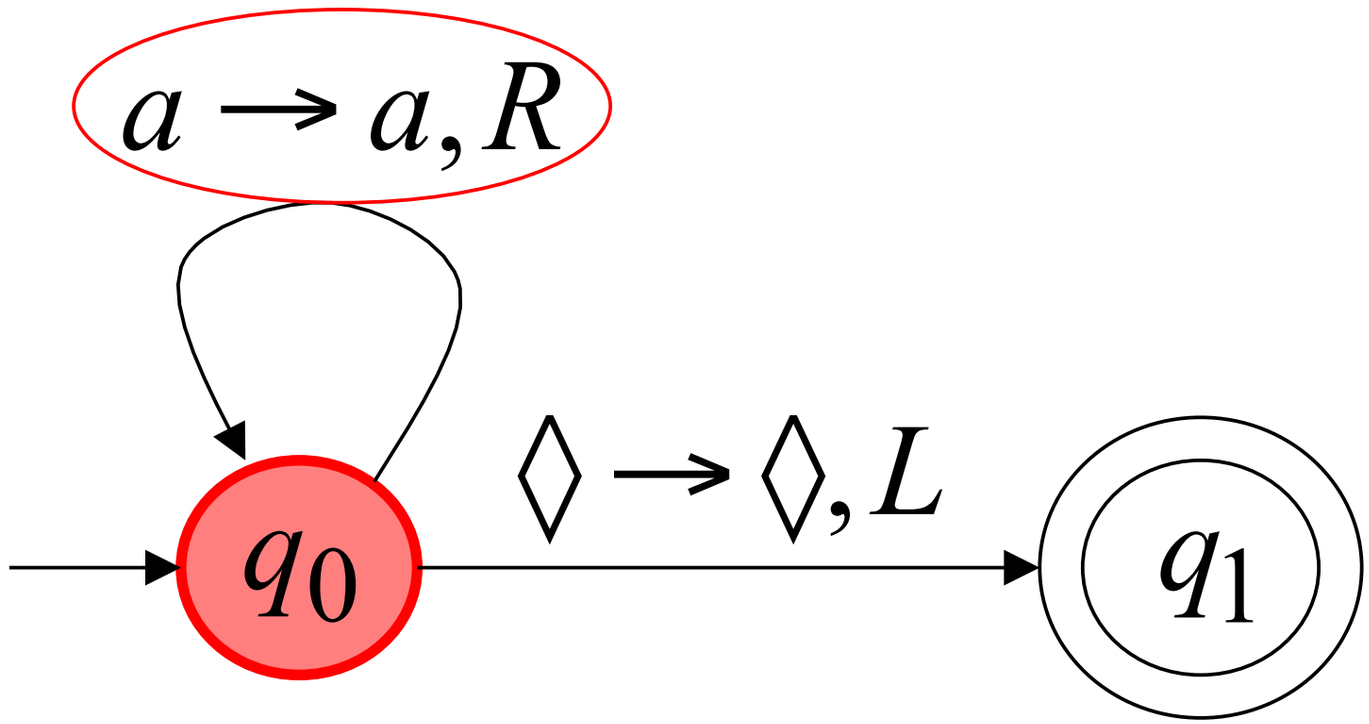


Halt & Accept

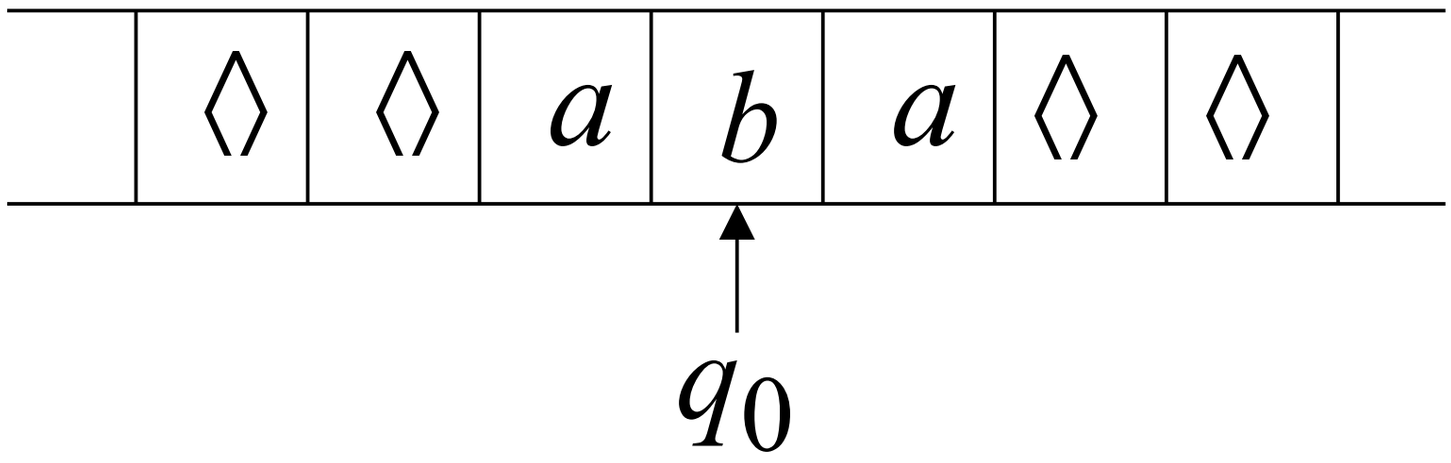
Rejection Example

Time 0



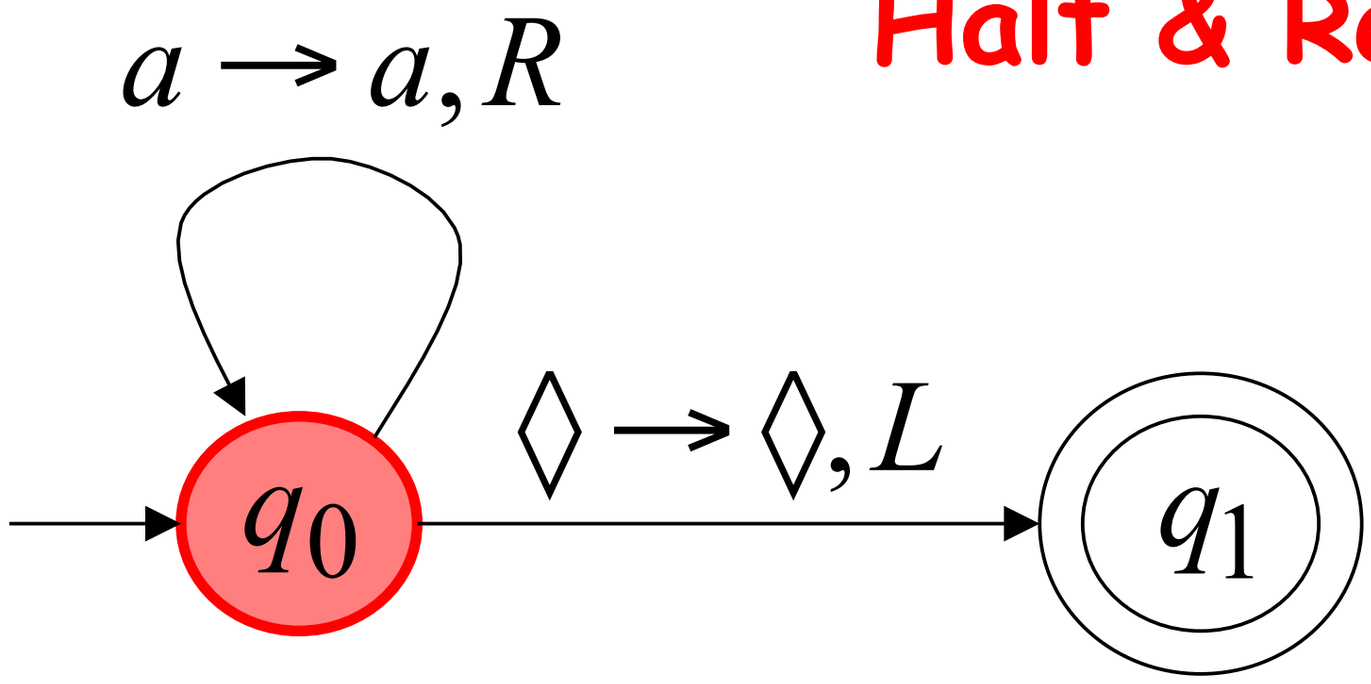


Time 1

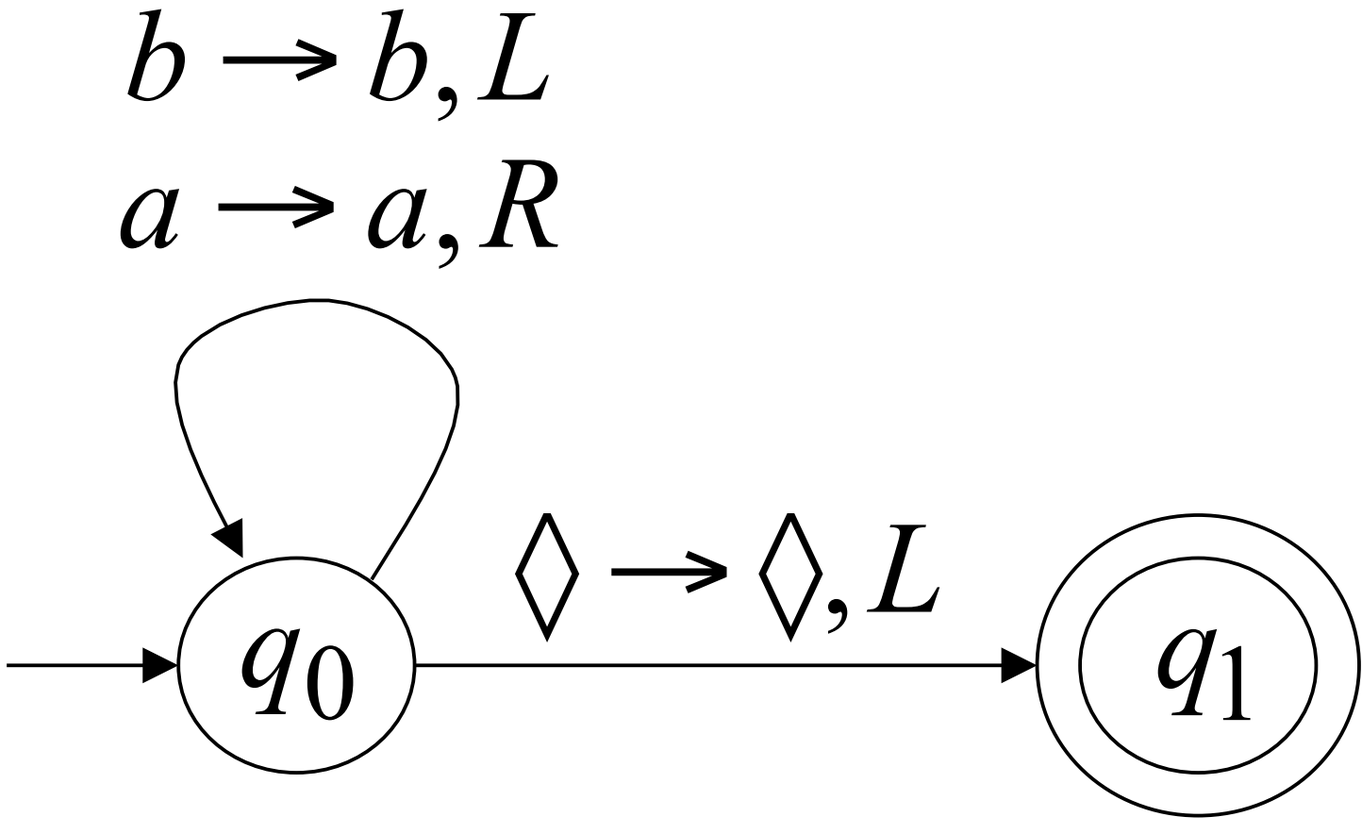


No possible Transition

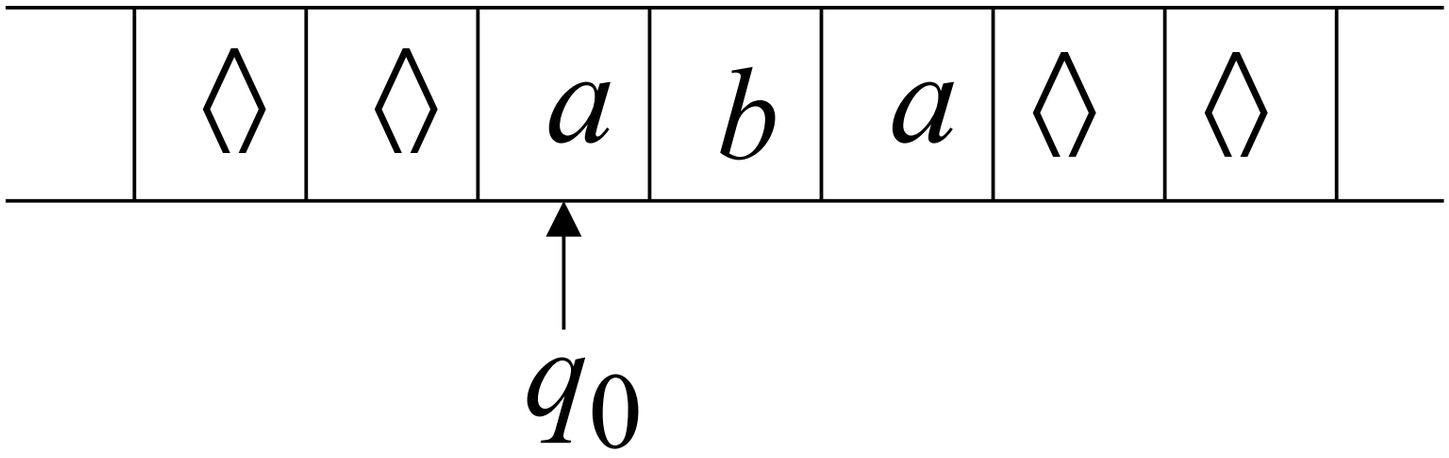
Halt & Reject

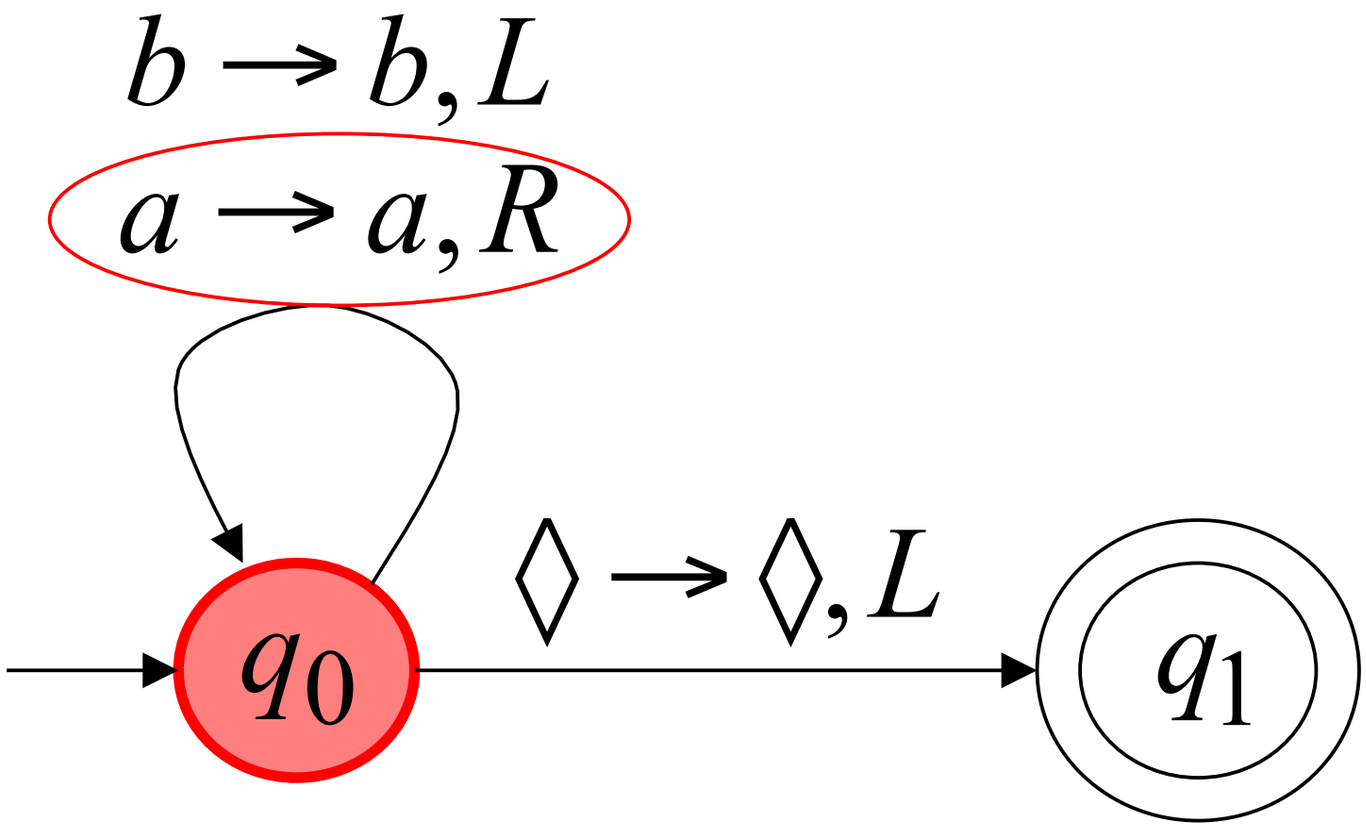


Infinite Loop Example

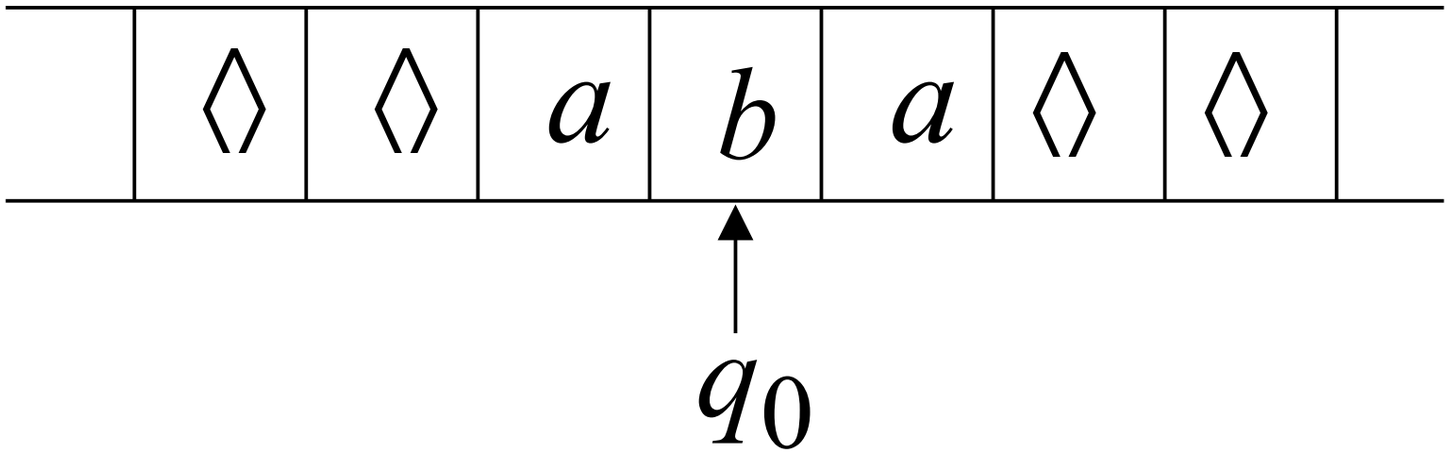


Time 0



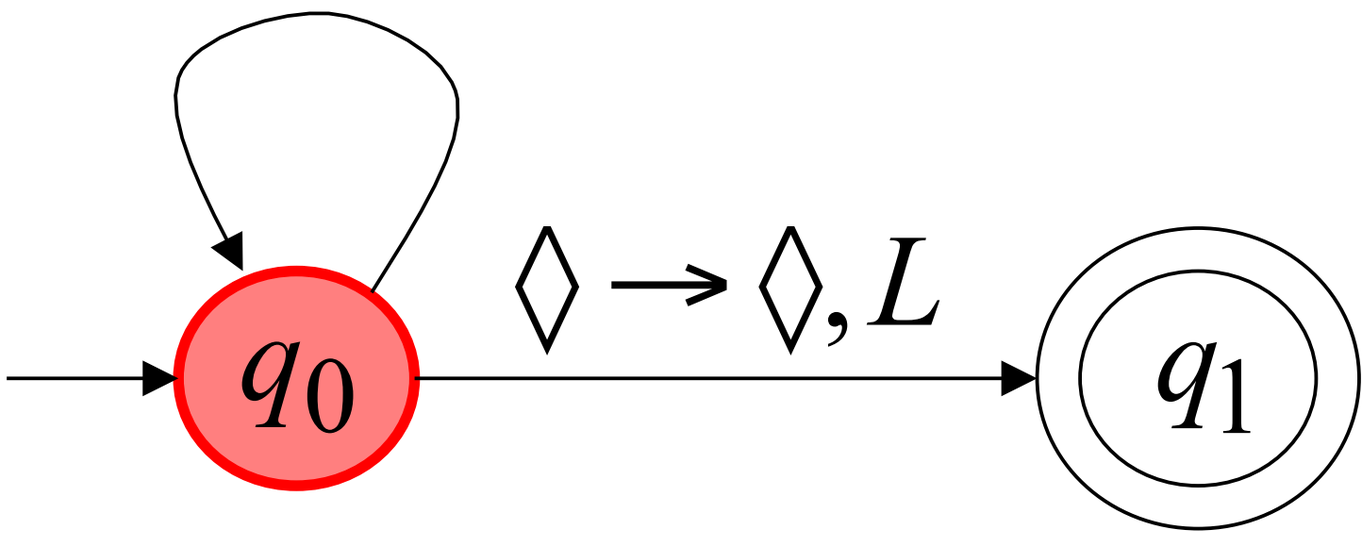


Time 1

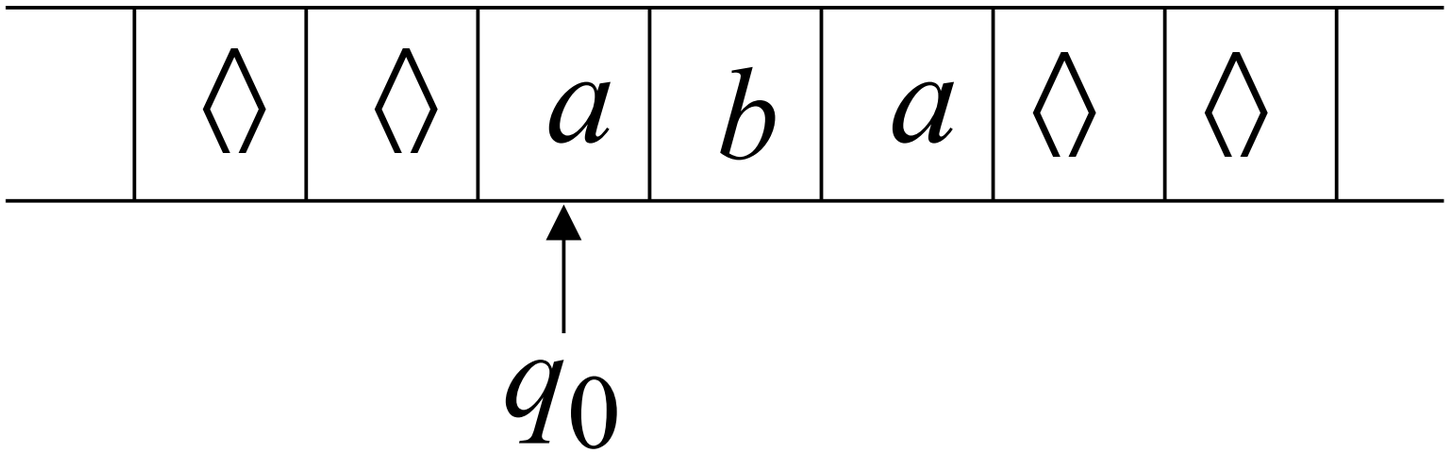


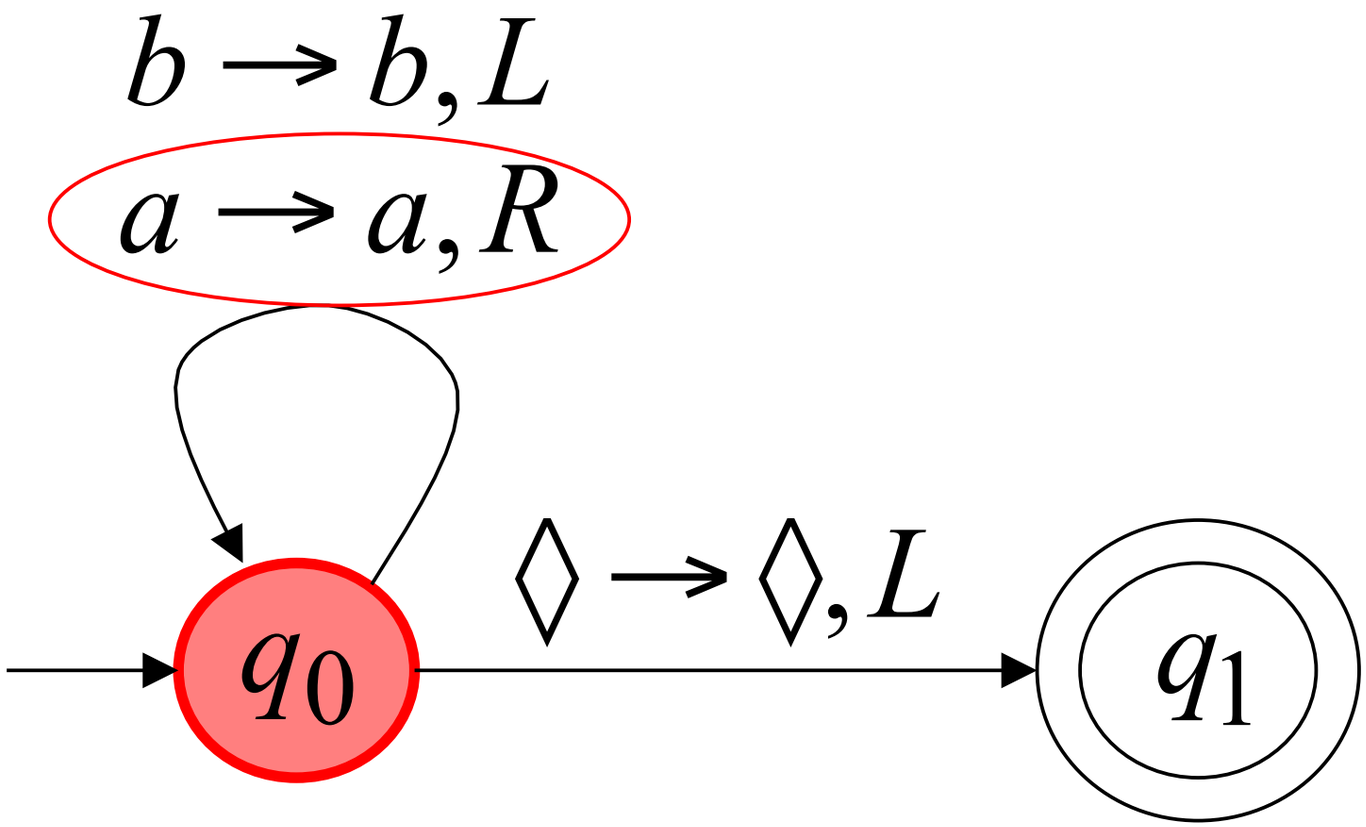
$b \rightarrow b, L$

$a \rightarrow a, R$

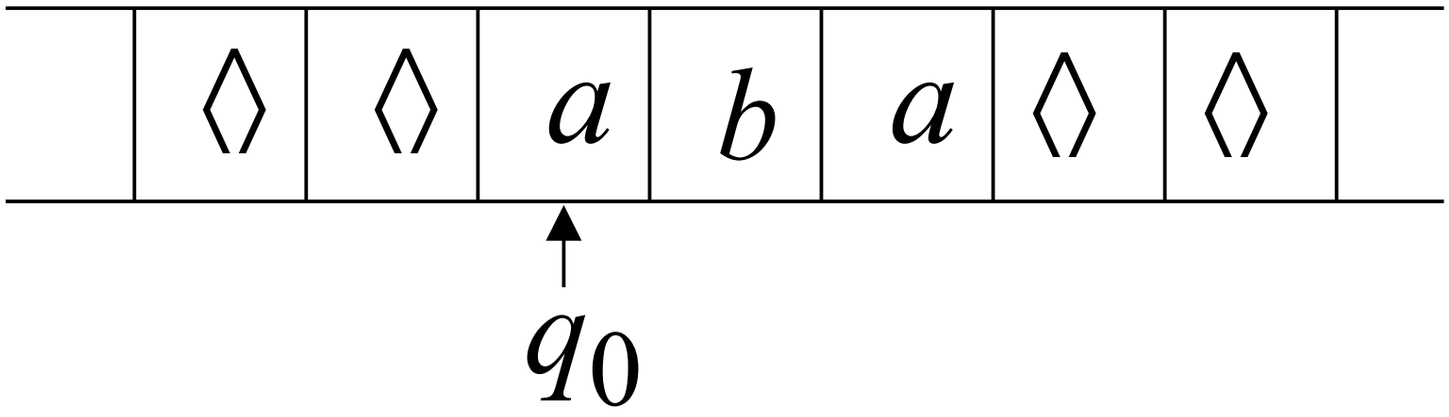


Time 2

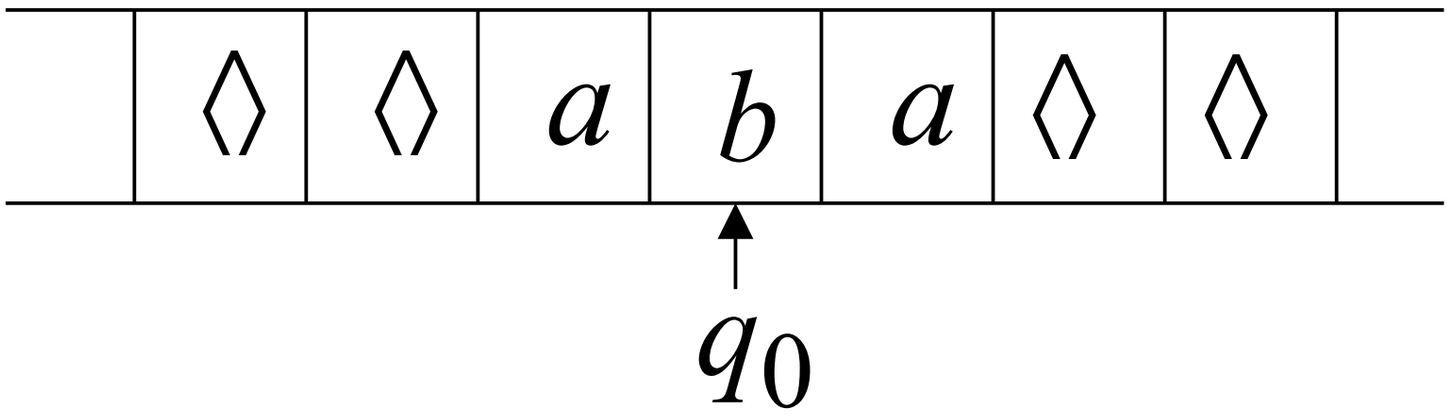




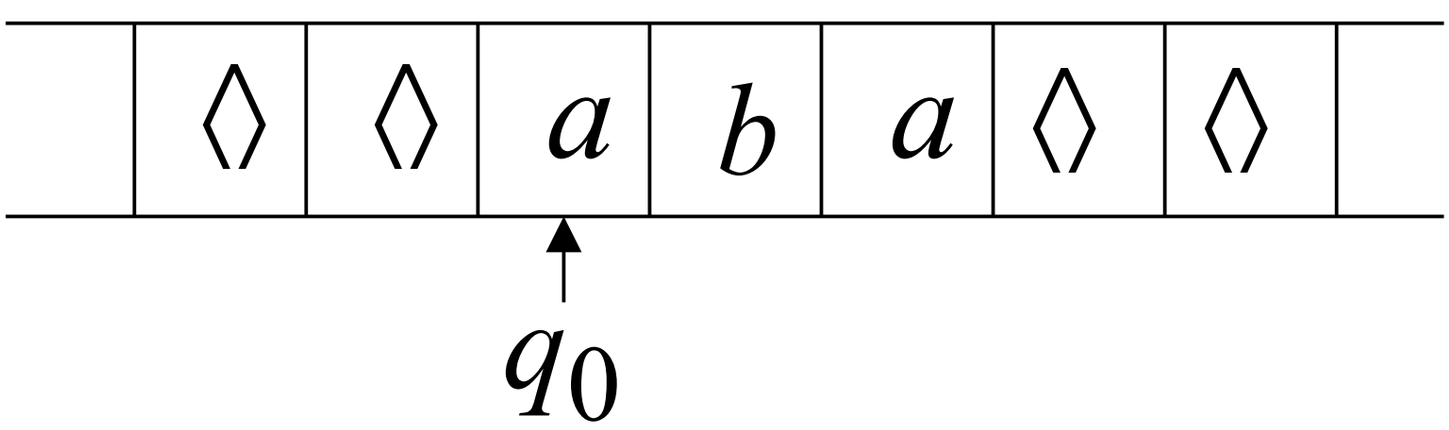
Time 2



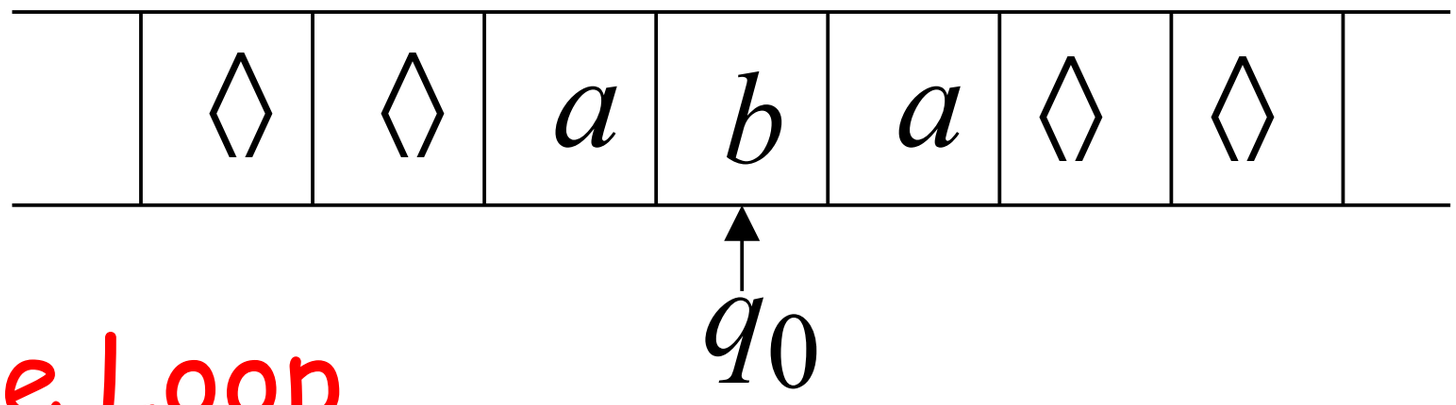
Time 3



Time 4



Time 5



… Infinite Loop

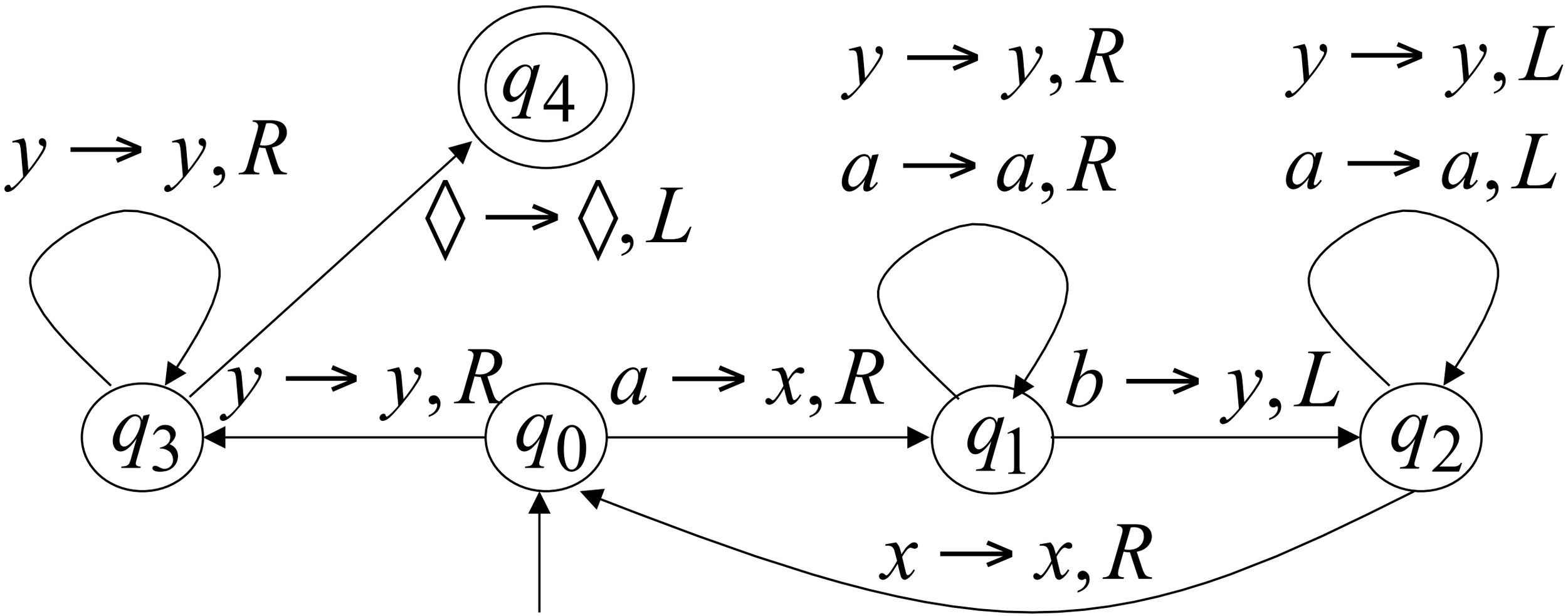
Because of the infinite loop:

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

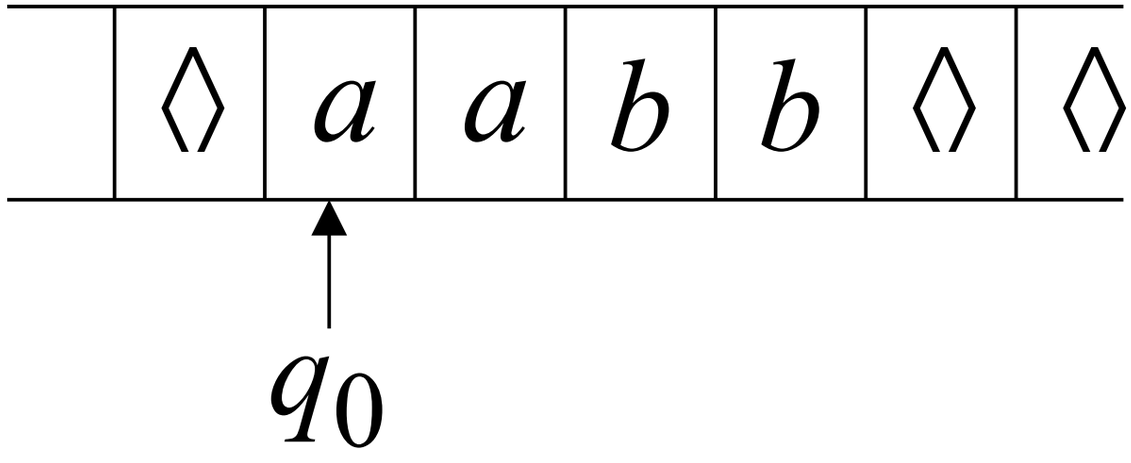
Another Turing Machine Example

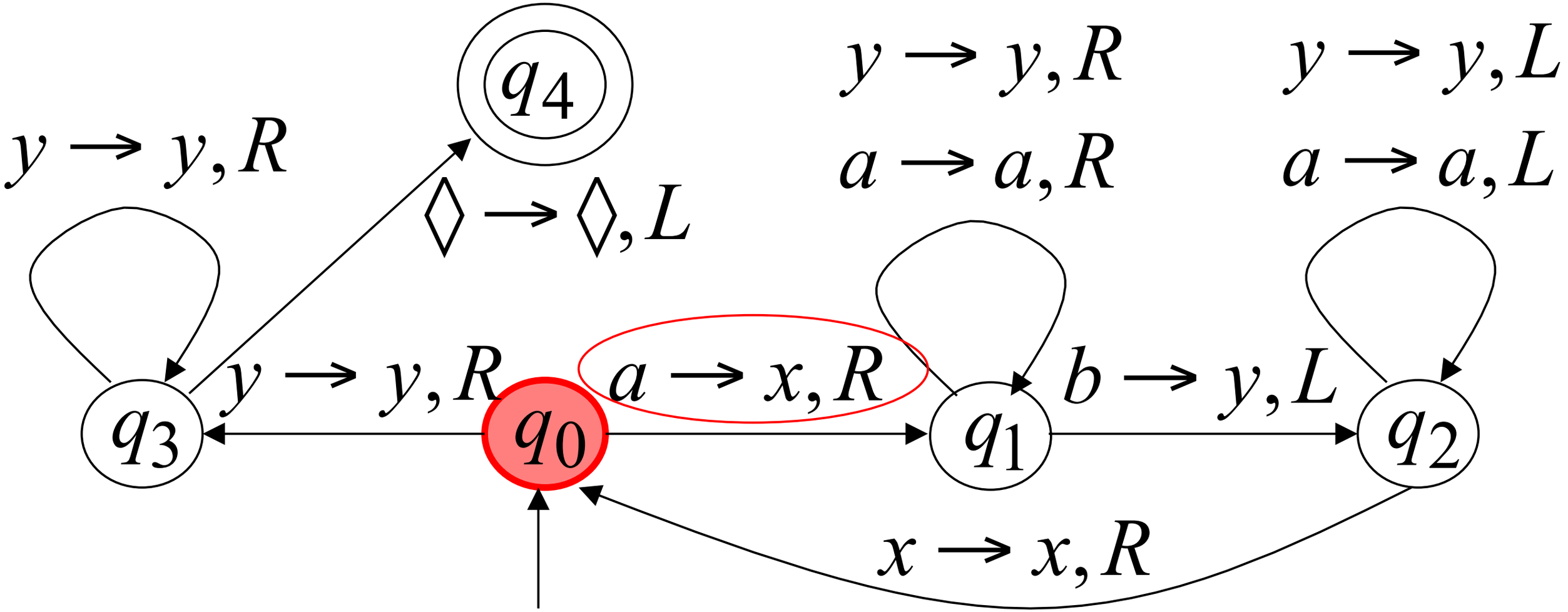
Turing machine for the language $\{a^n b^n\}$

(n >= 1)

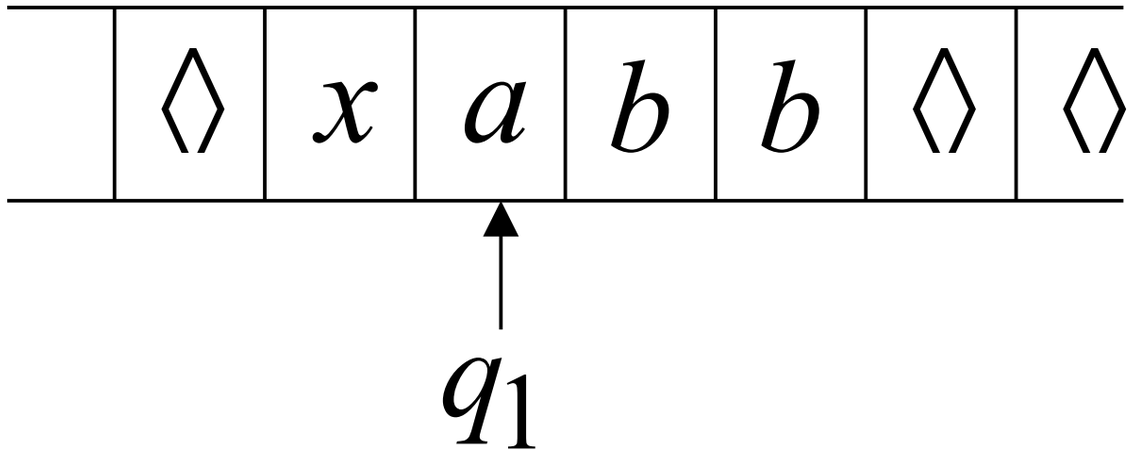


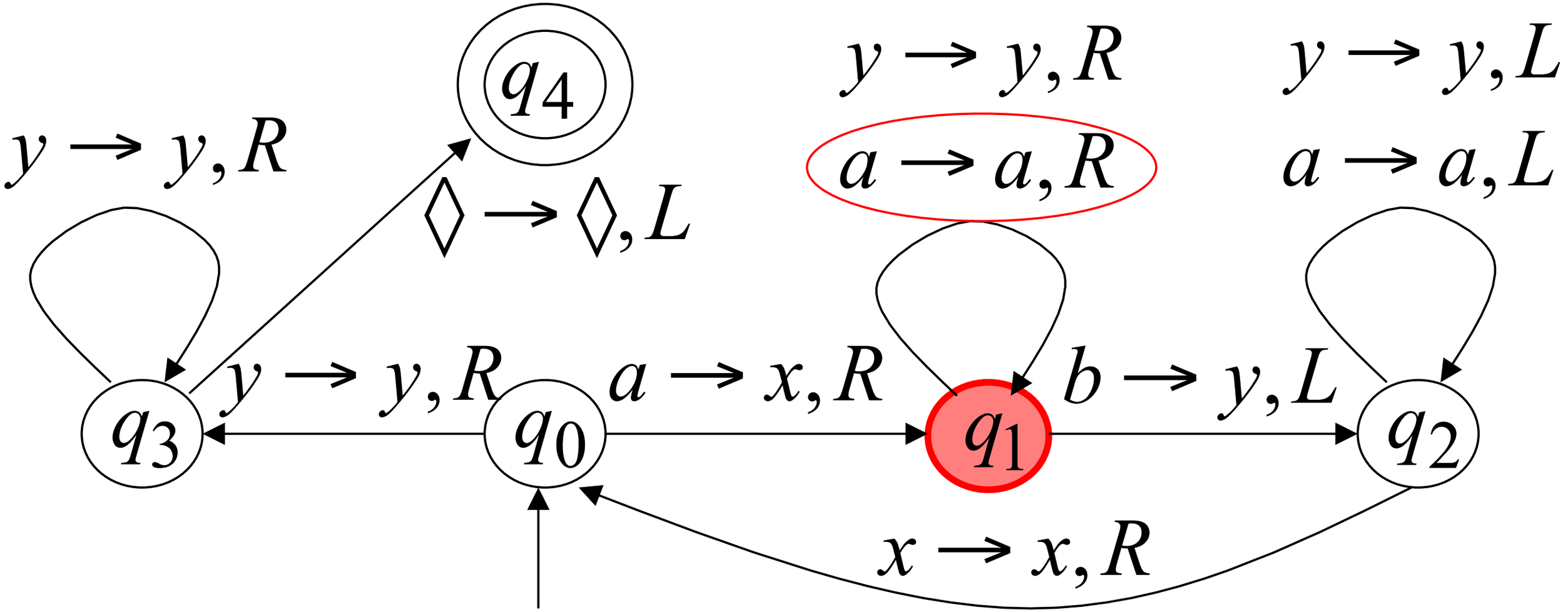
Time 0


⋋ a a b b ⋋ ⋋
q_0

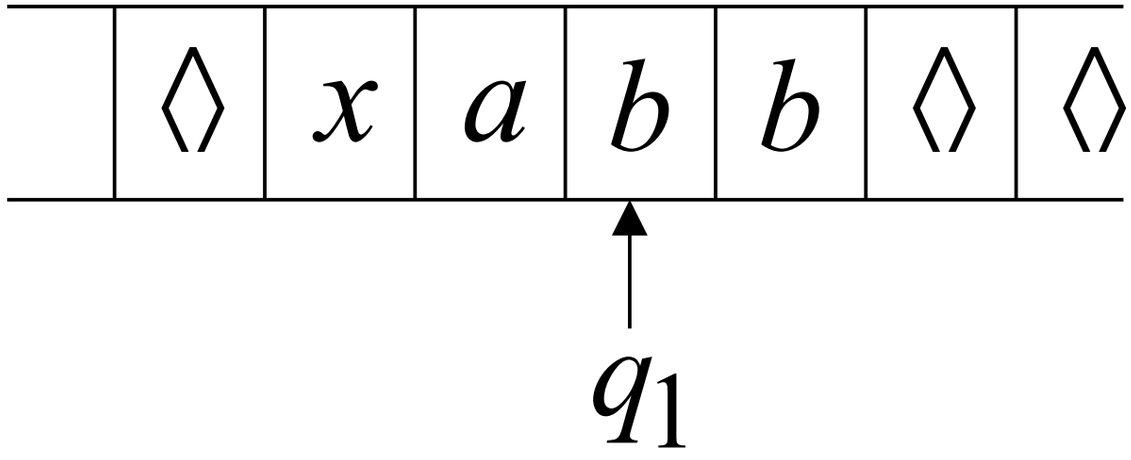

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
$a \to x, R$
q_0
q_1
$b \to y, L$
q_2
$x \to x, R$

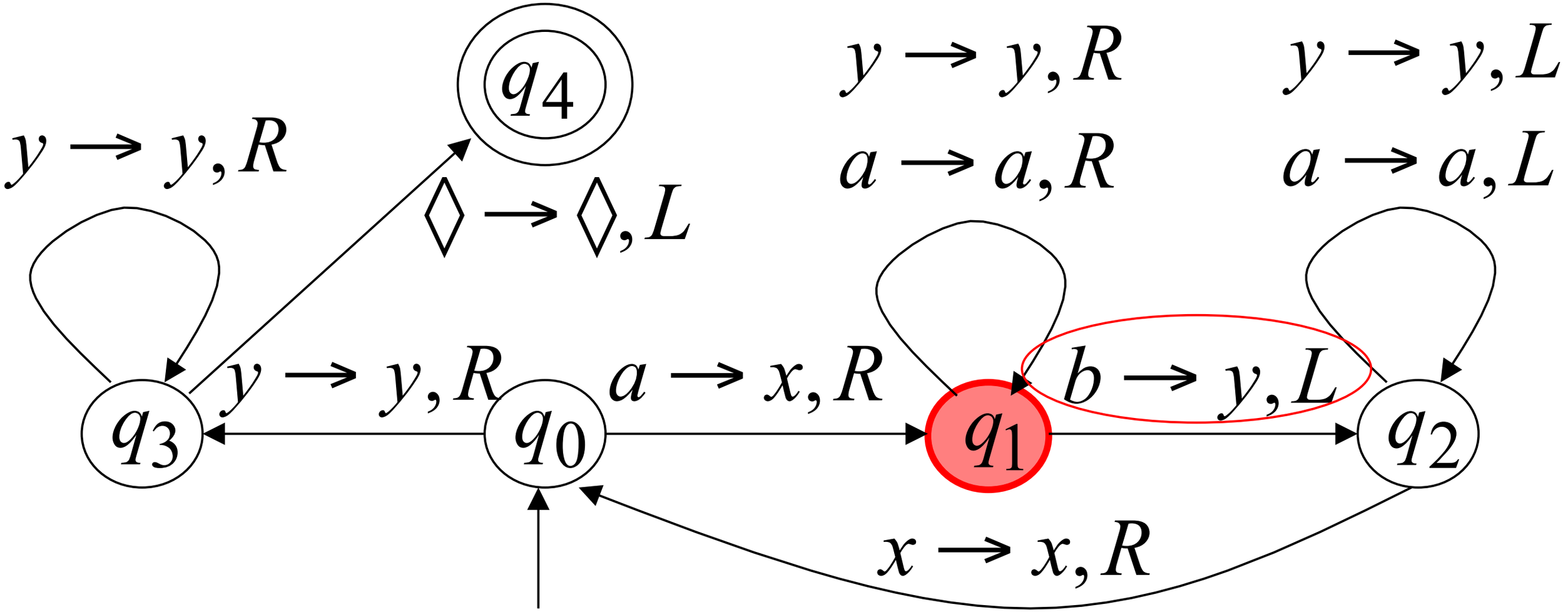
Time 1


◊ x a b b ◊ ◊
q_1

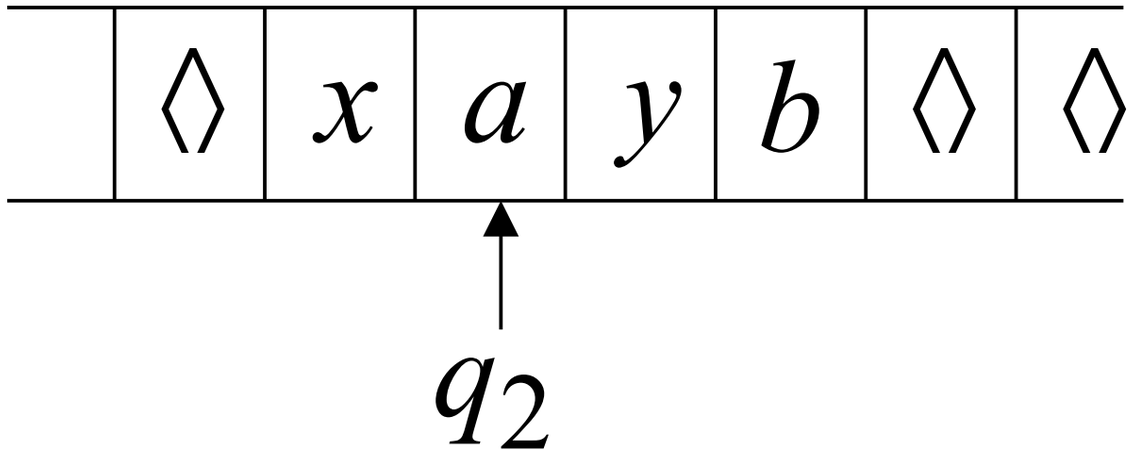

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

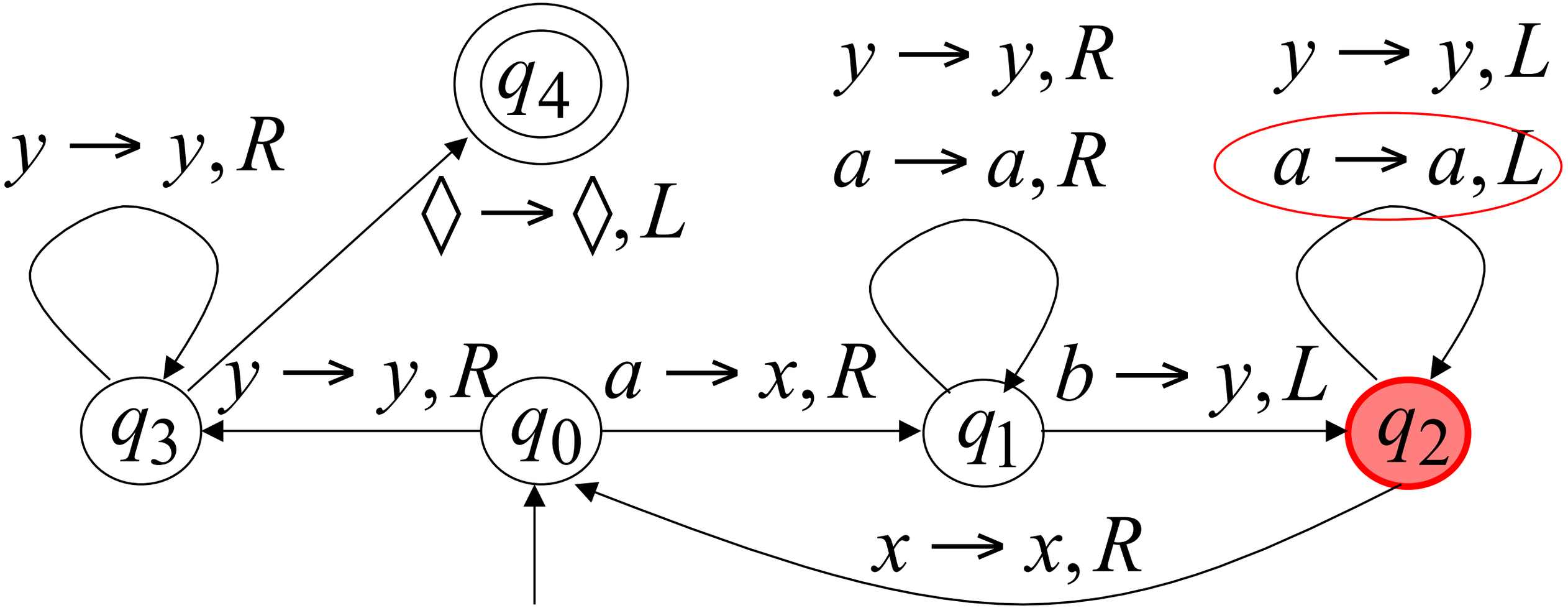
Time 2


◊ x a b b ◊ ◊
q_1

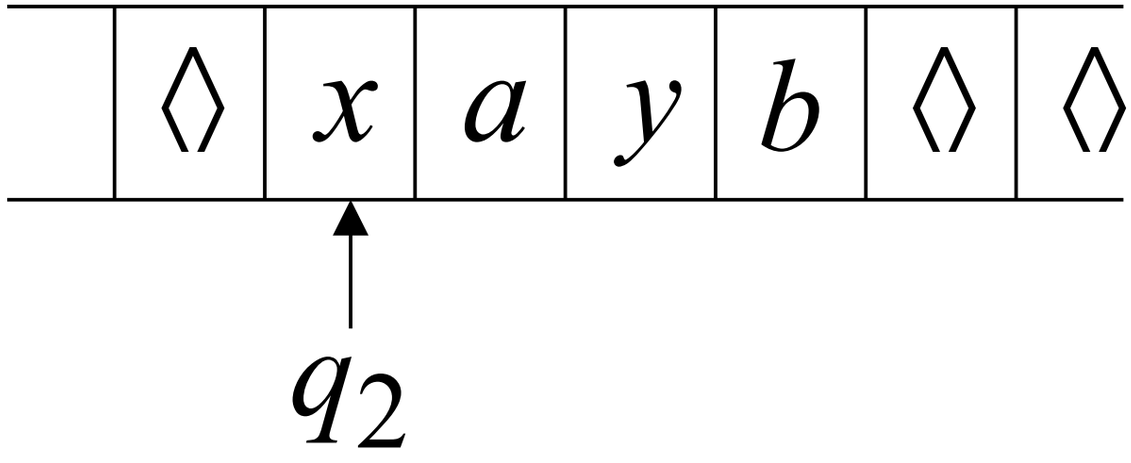

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

Time 3


◊
x
a
y
b
◊
◊
q_2

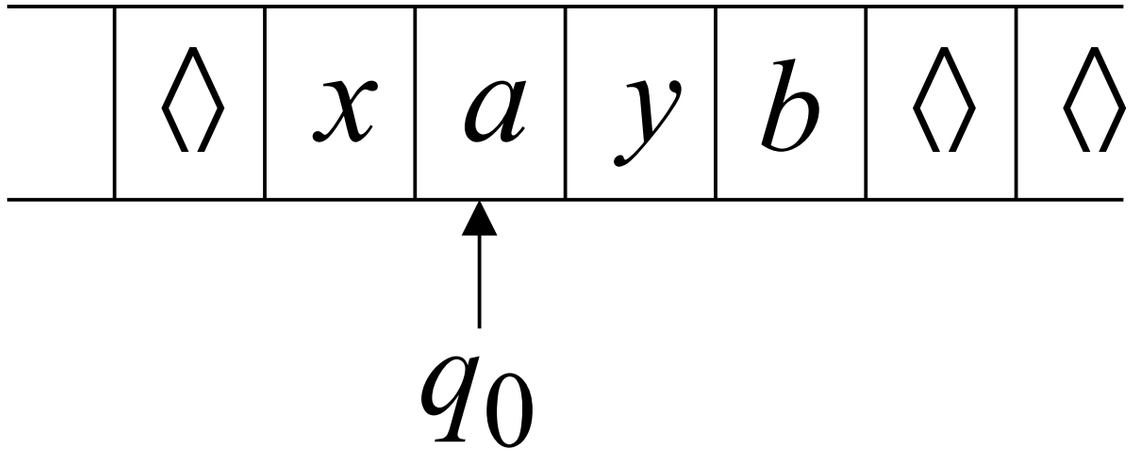

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

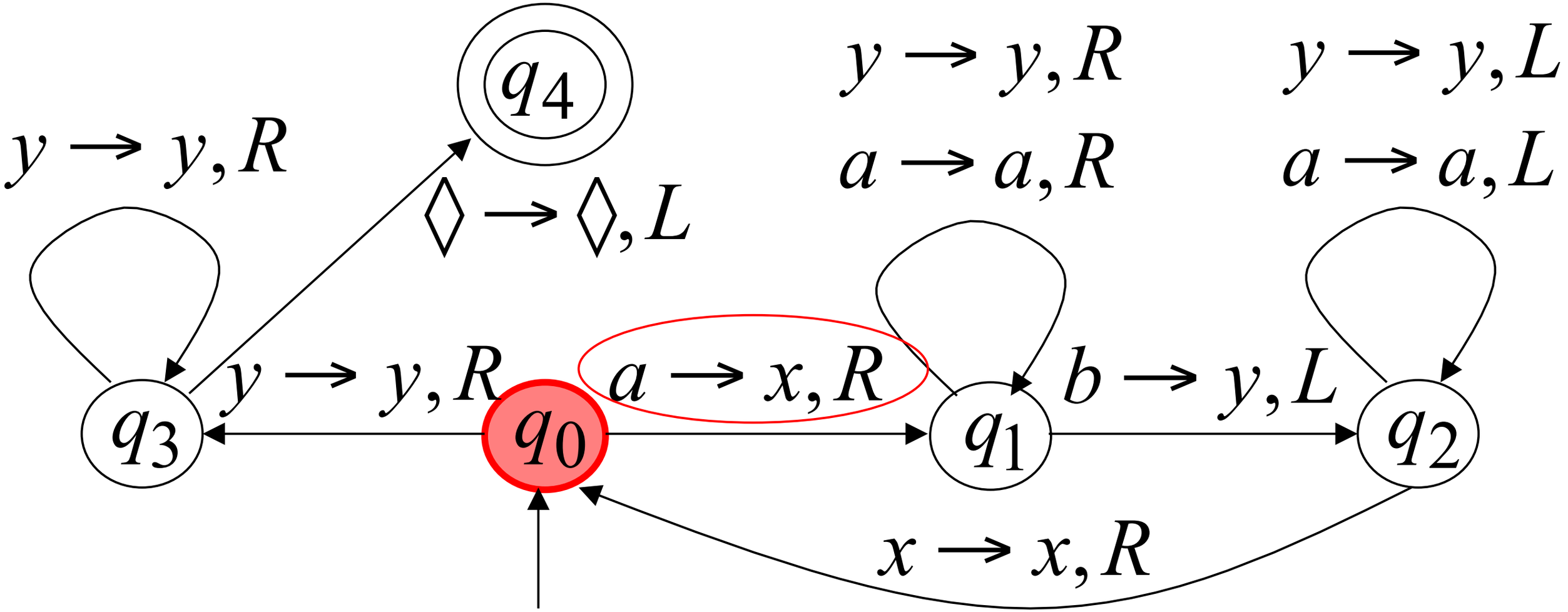
Time 4


◊
x
a
y
b
◊
◊
q_2

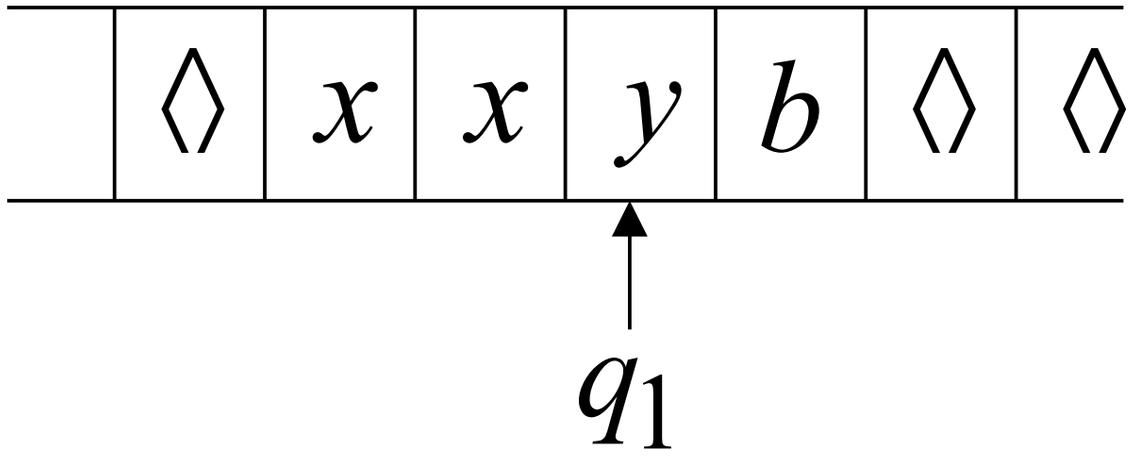
q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

Time 5


◊ x a y b ◊ ◊
q_0

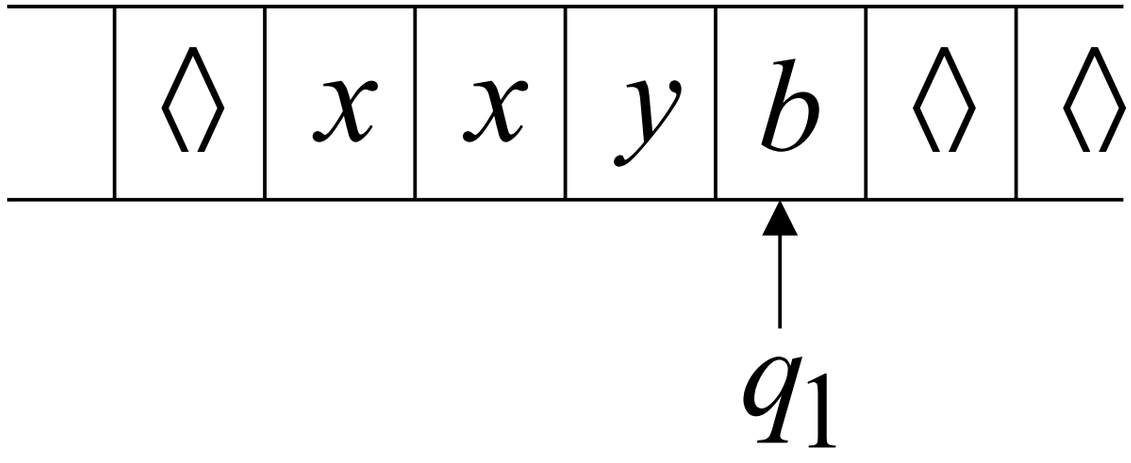

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

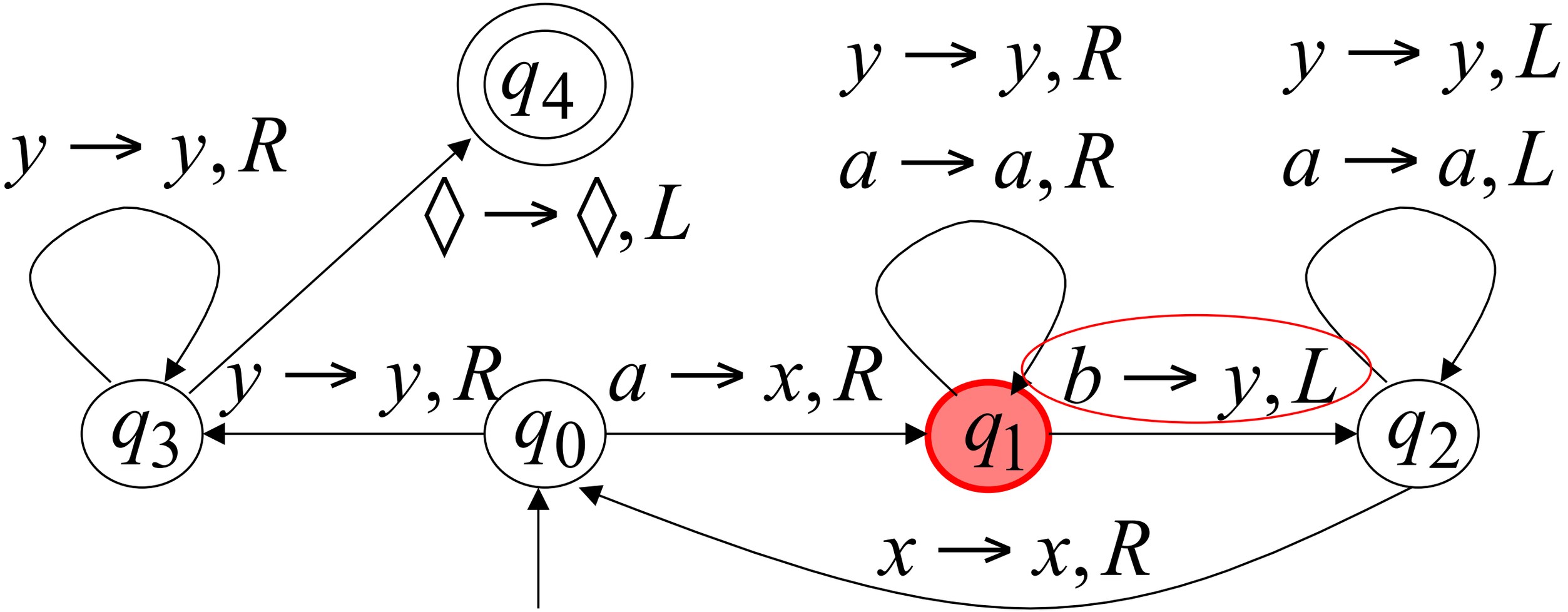
Time 6


◊ x x y b ◊ ◊
q_1

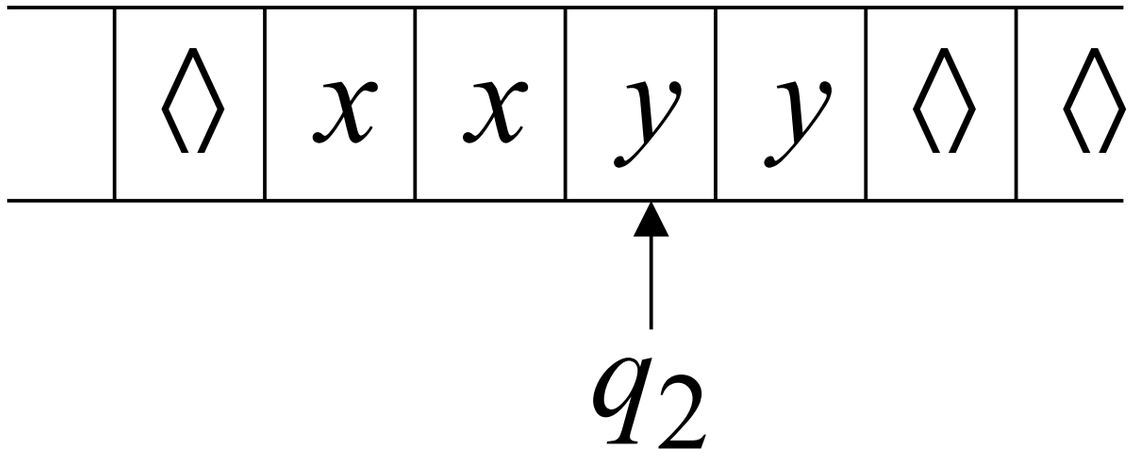
q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

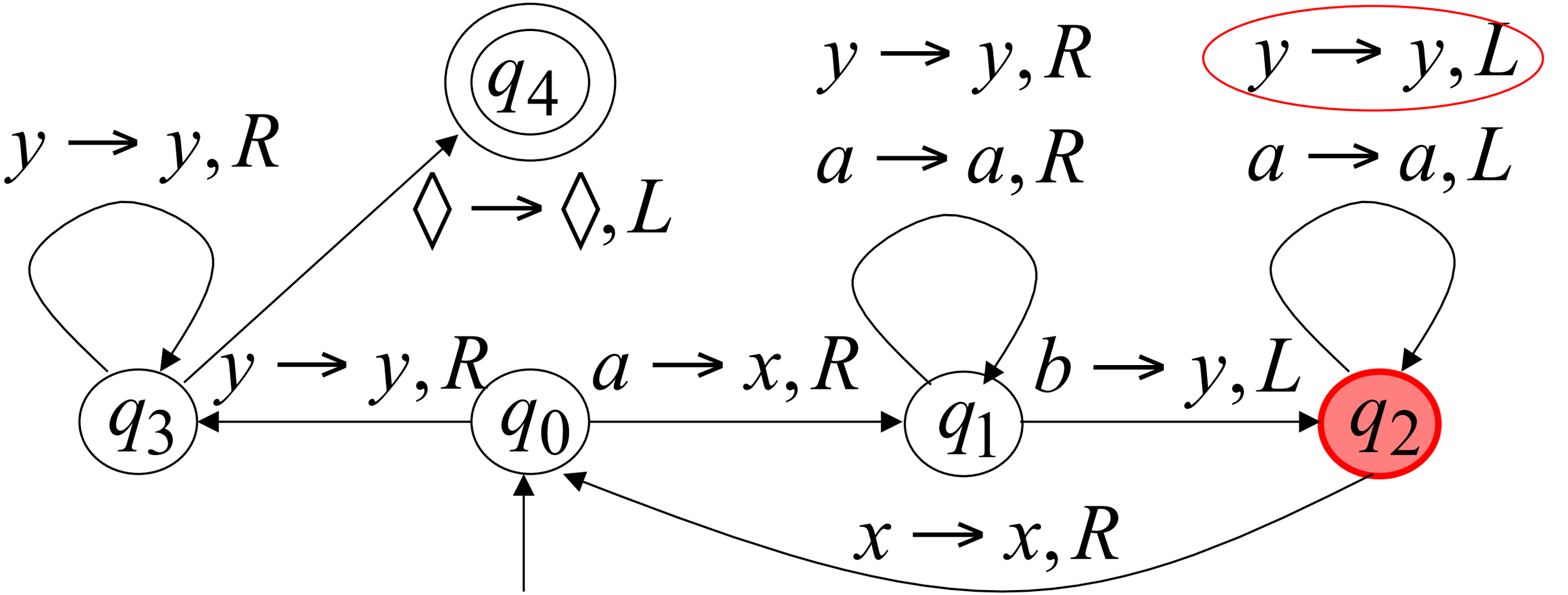
Time 7


◊ x x y b ◊ ◊
q_1

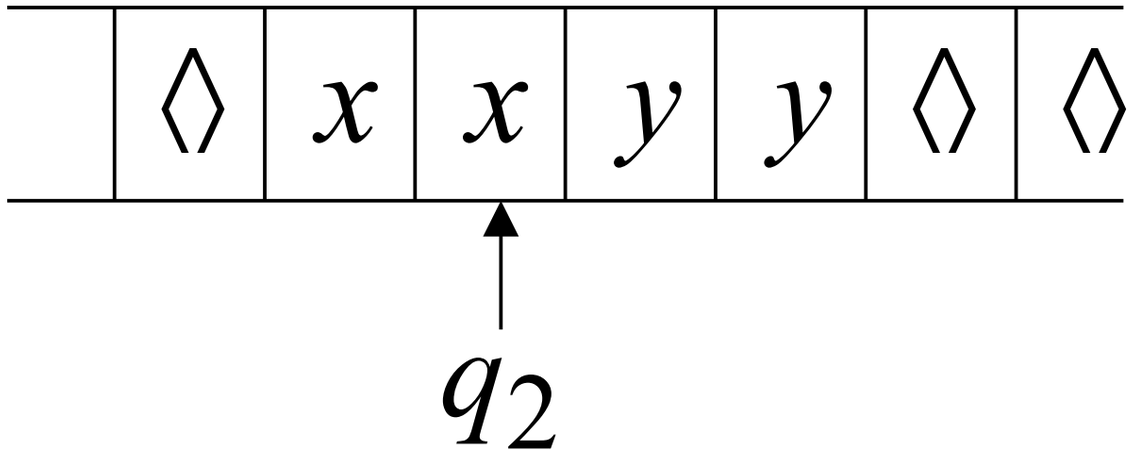

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

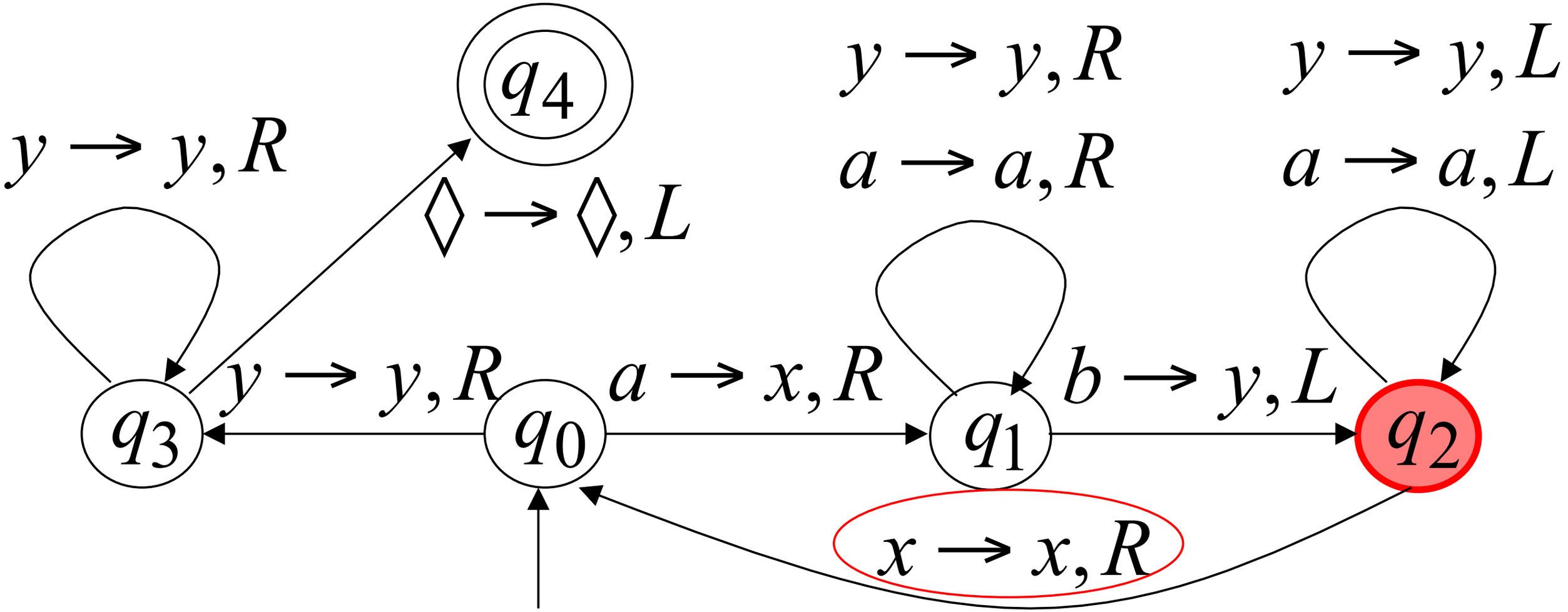
Time 8


◊ x x y y ◊ ◊
q_2

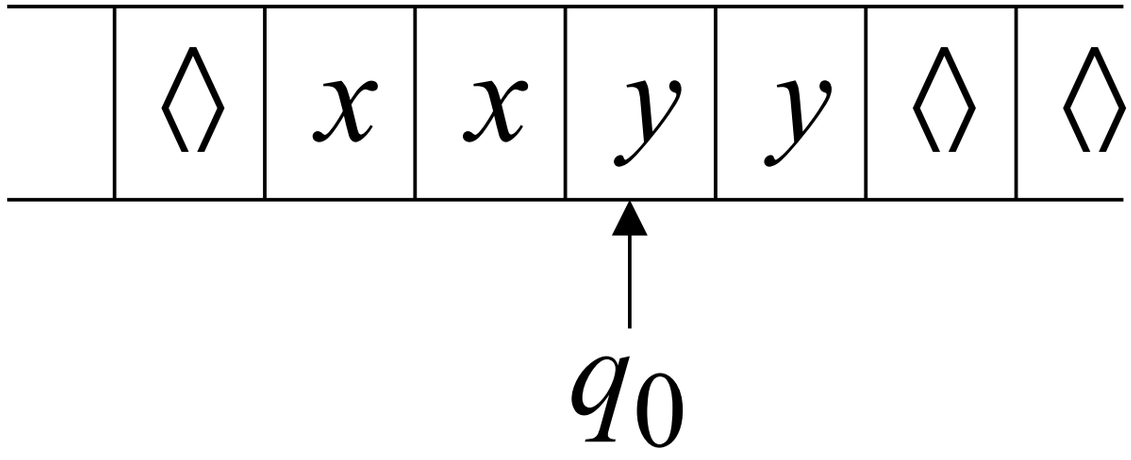

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

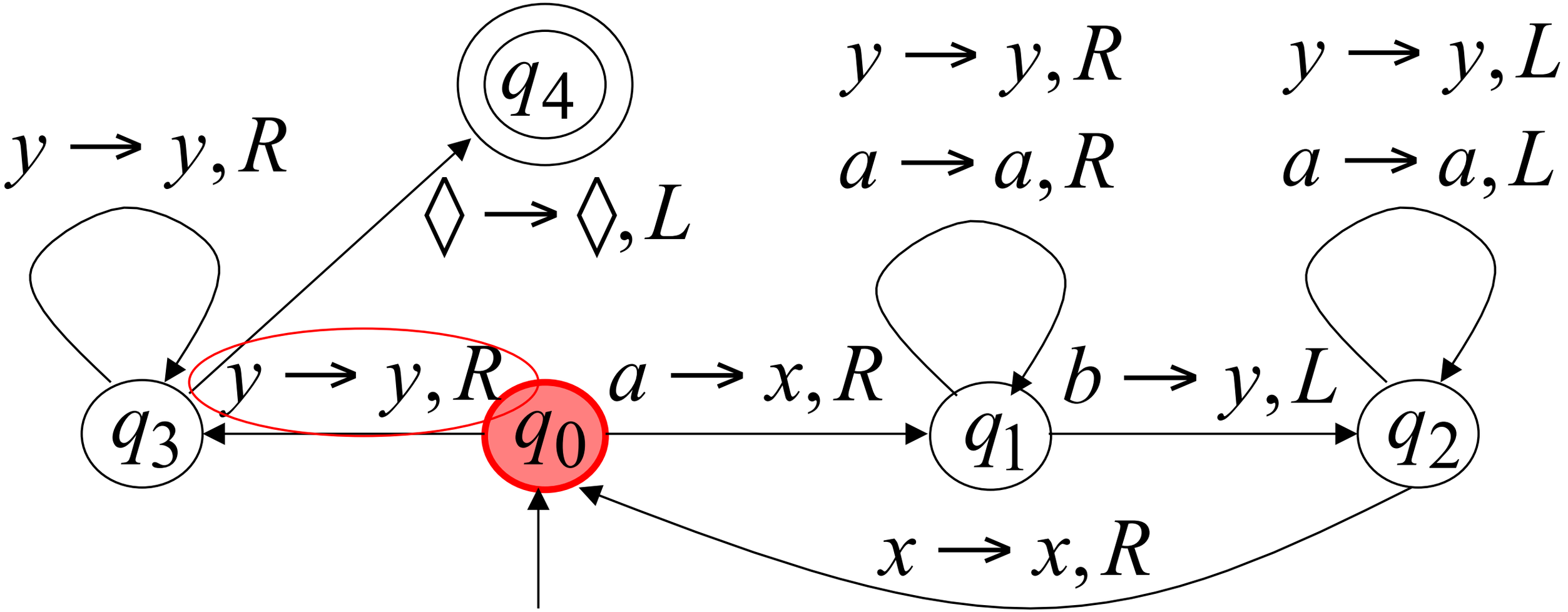
Time 9


◊ x x y y ◊ ◊
q_2

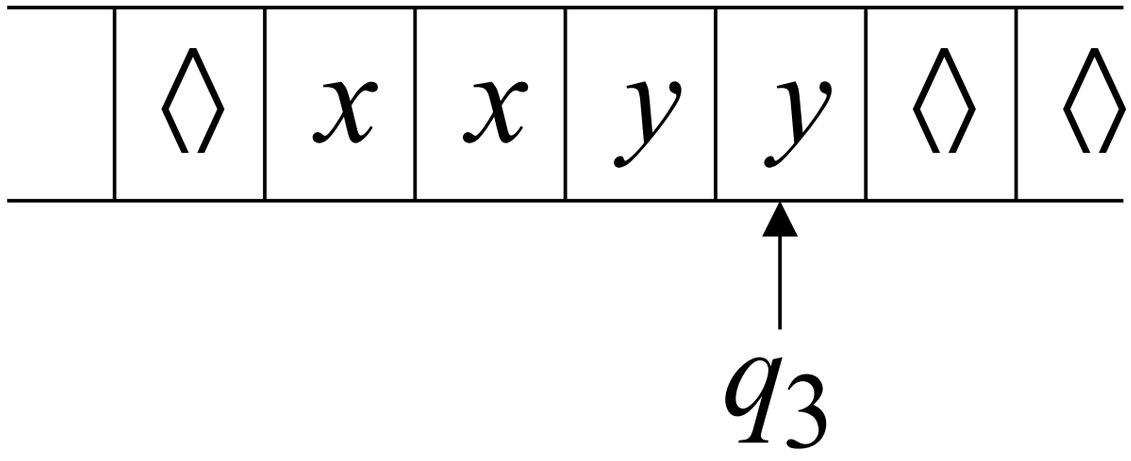

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

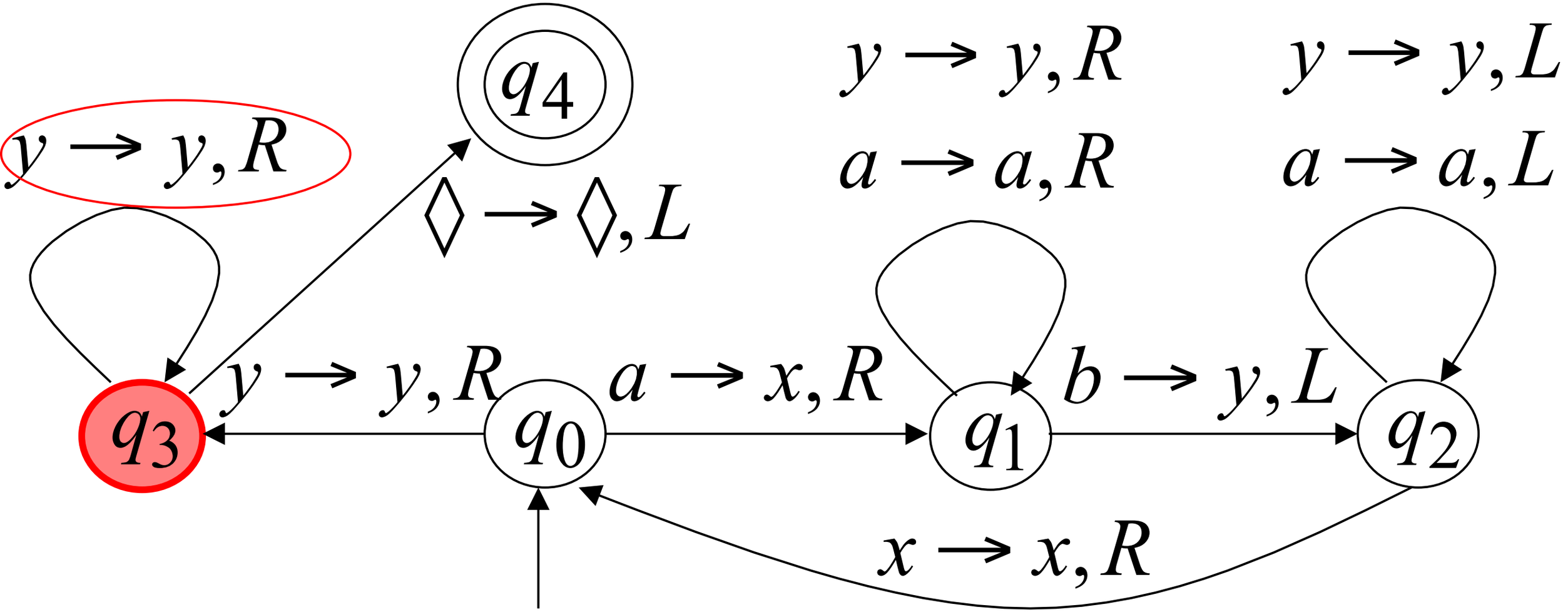
Time 10


◊ x x y y ◊ ◊
q_0

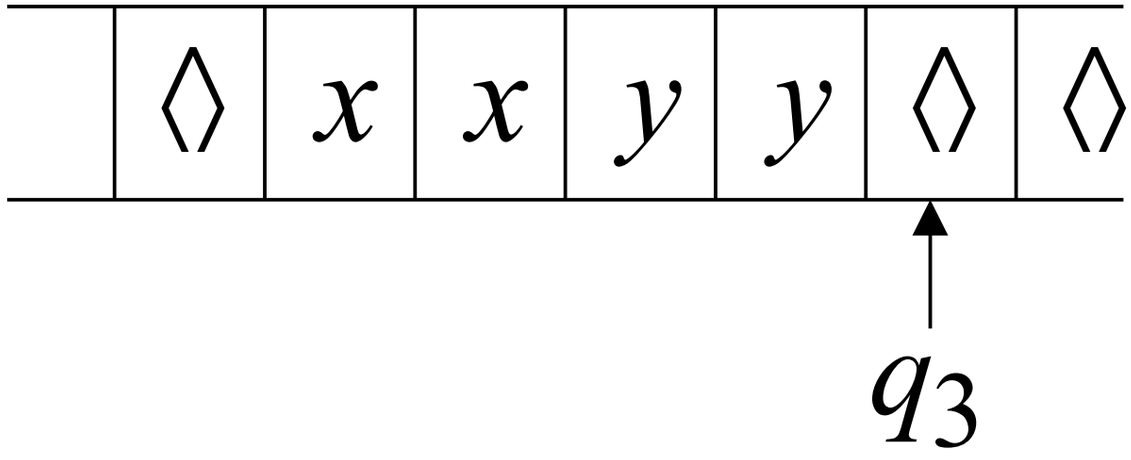

q_4
$y \to y, R$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$\Diamond \to \Diamond, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

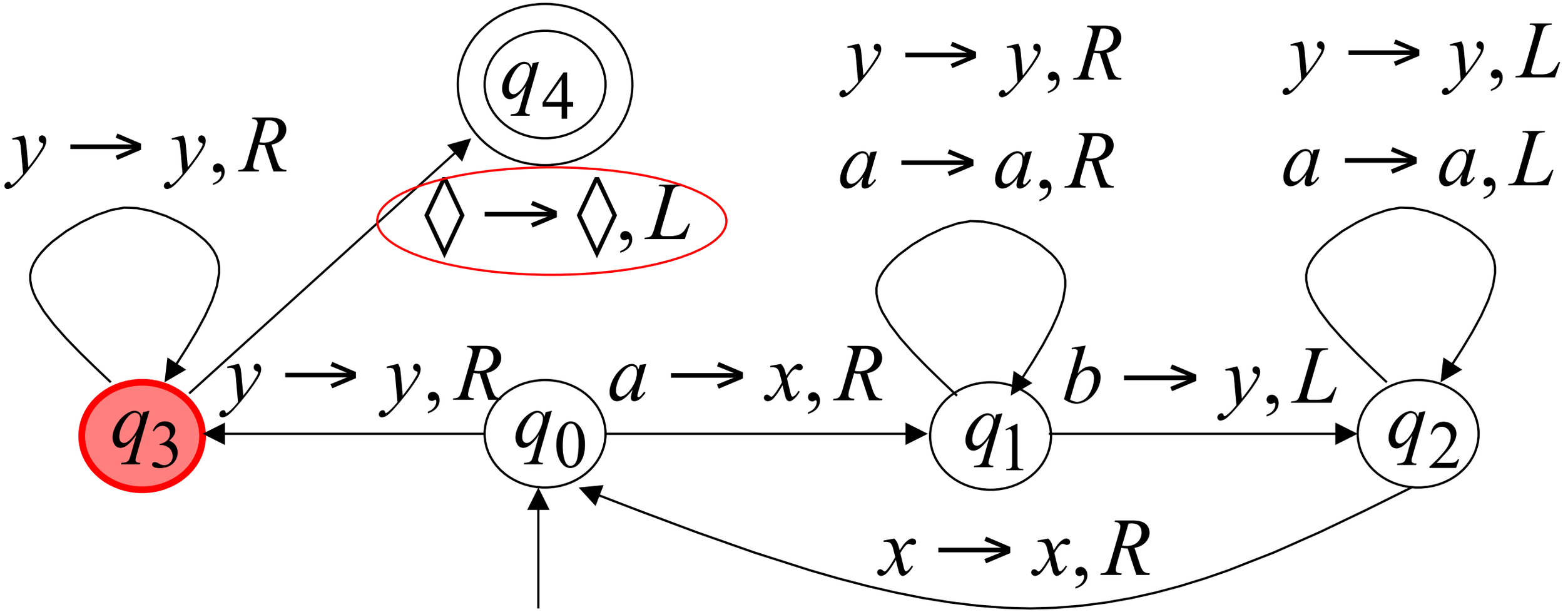
Time 11


◊ x x y y ◊ ◊
q_3

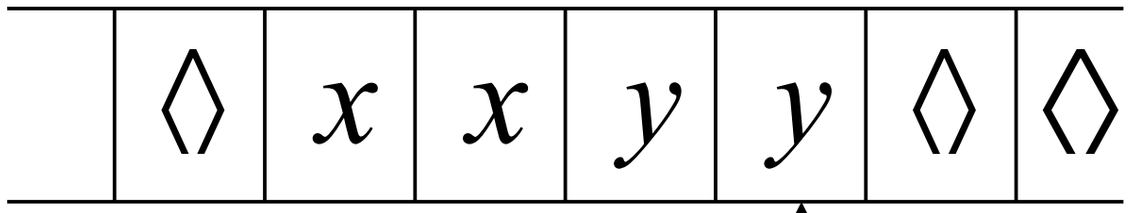

q_4
$y \to y, R$
$y \to y, R$
$y \to y, L$
$a \to a, R$
$a \to a, L$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
q_3
q_0
q_1
q_2
$x \to x, R$

Time 12


◊
x
x
y
y
◊
◊
q_3


q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

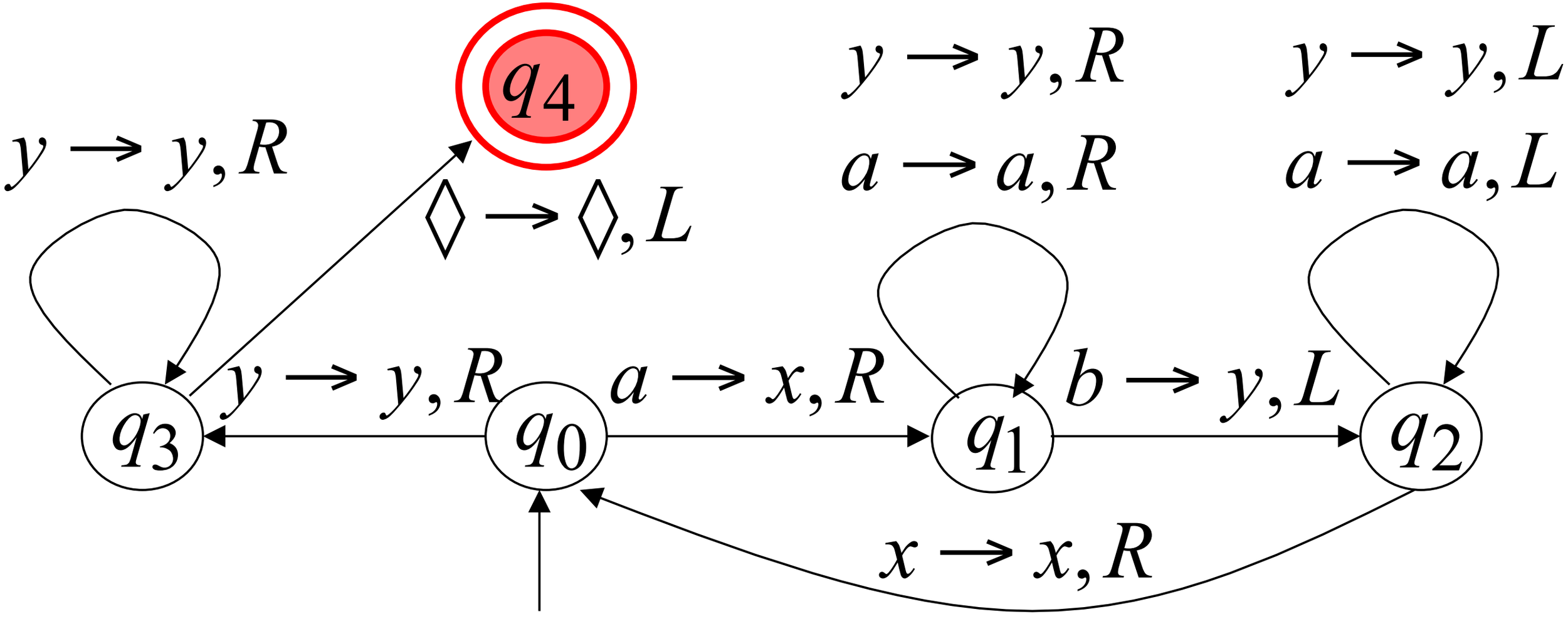
Time 13



Q1: which state makes sure there're no extra a's?

Q2: which state makes sure there're no extra b's? q_4

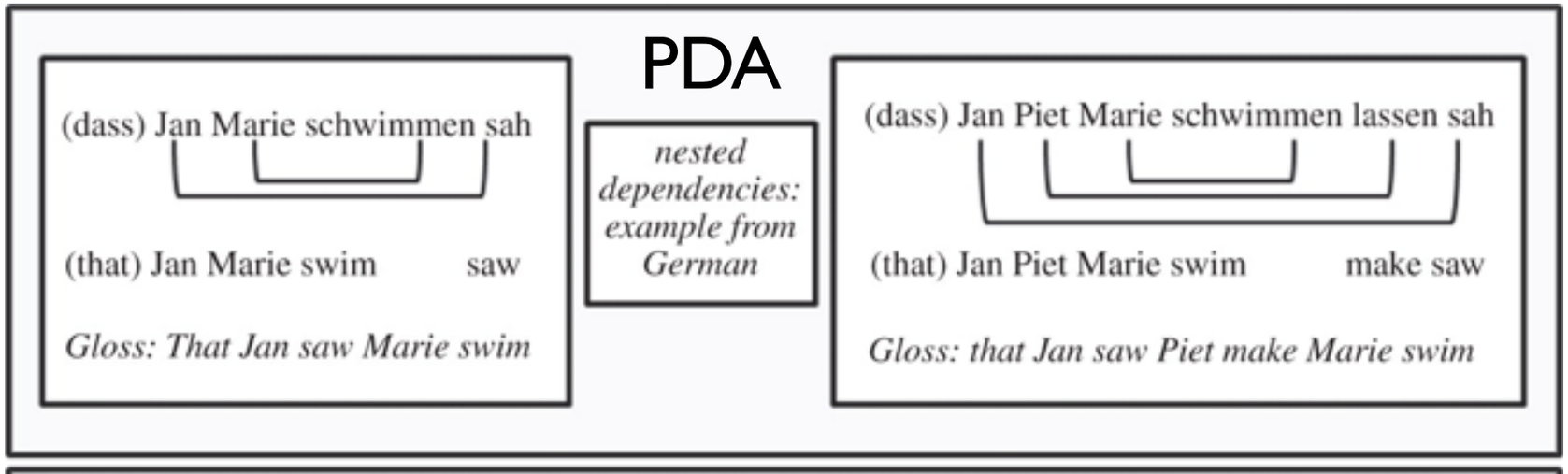
Halt & Accept

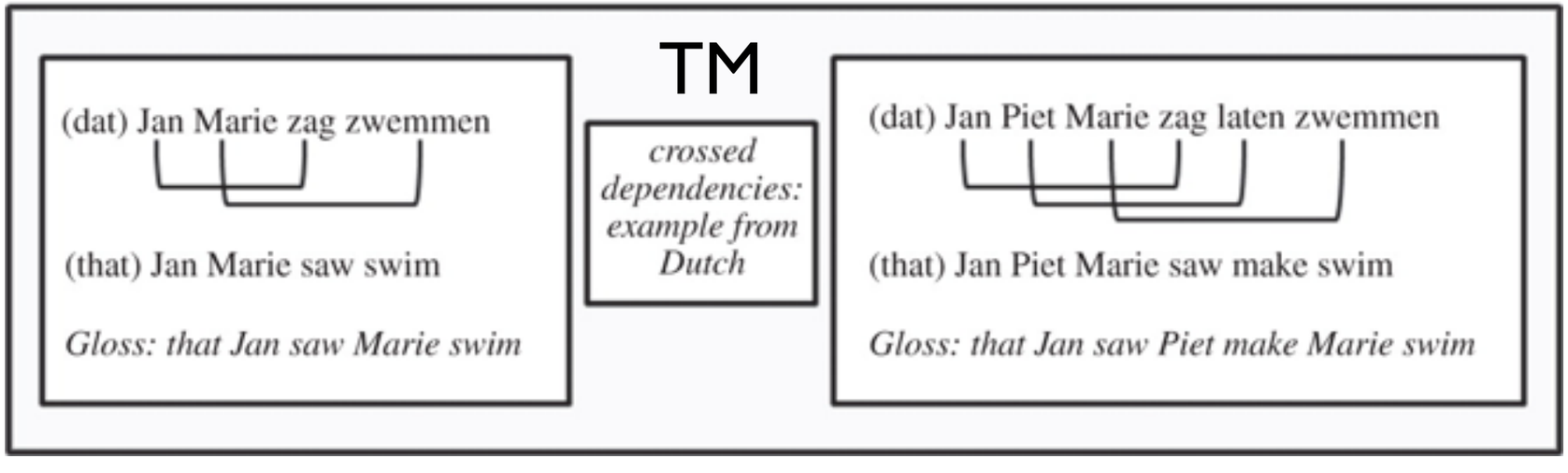


Observation:

```
On input string w
    while there are unmarked 0s, do
        Mark the left most 0
        Scan right till the leftmost unmarked 1;
            if there is no such 1 then crash
        Mark the leftmost 1
    done
    Check to see that there are no unmarked 1s;
        if there are then crash
    accept
```

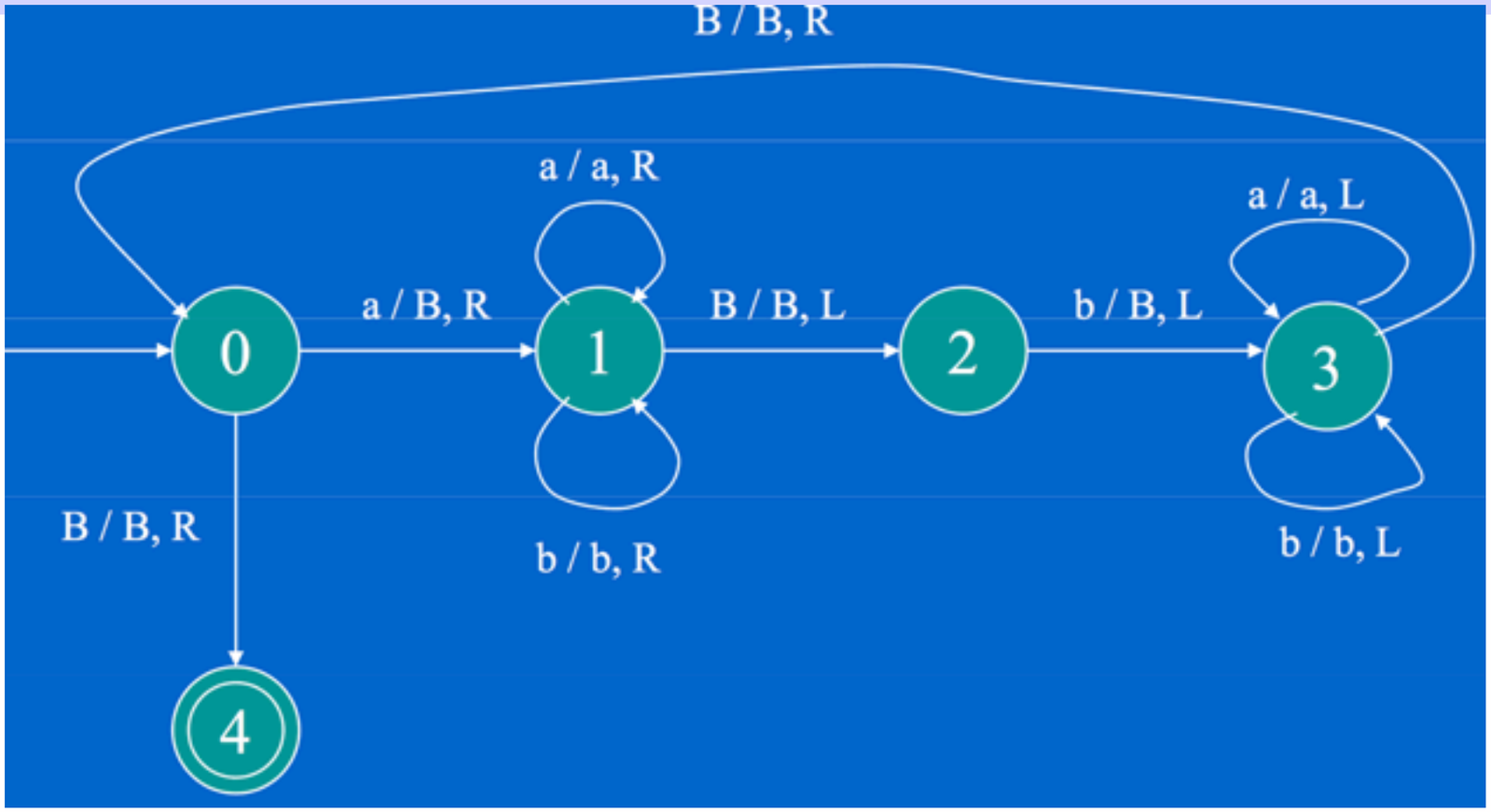
If we modify the machine for the language $\{a^n b^n\}$

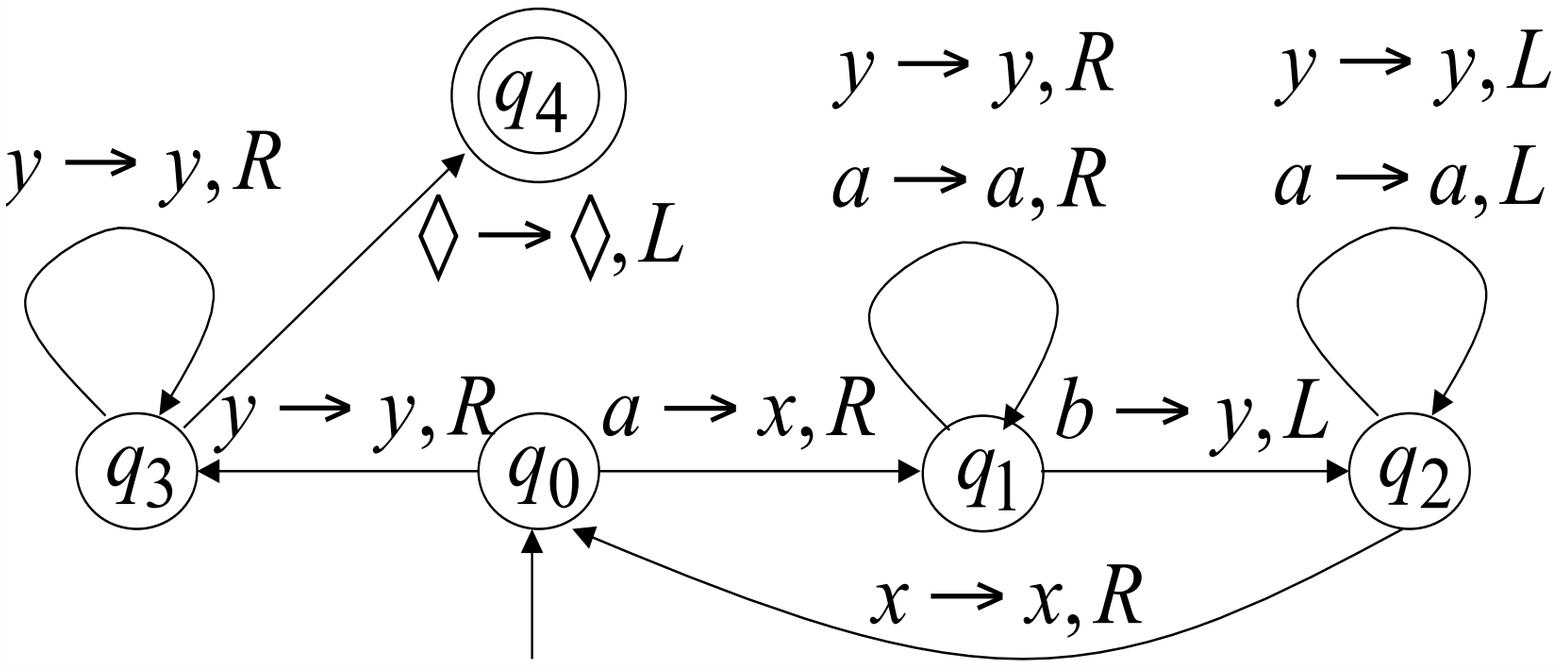




$\{a^n b^n c^n\}$

A Different TM for $a^n b^n$





(dass) Jan Piet Marie schwimmen lassen sah

(that) Jan Piet Marie swim make saw

Gloss: that Jan saw Piet make Marie swim

(dat) Jan Piet Marie zag laten zwemmen

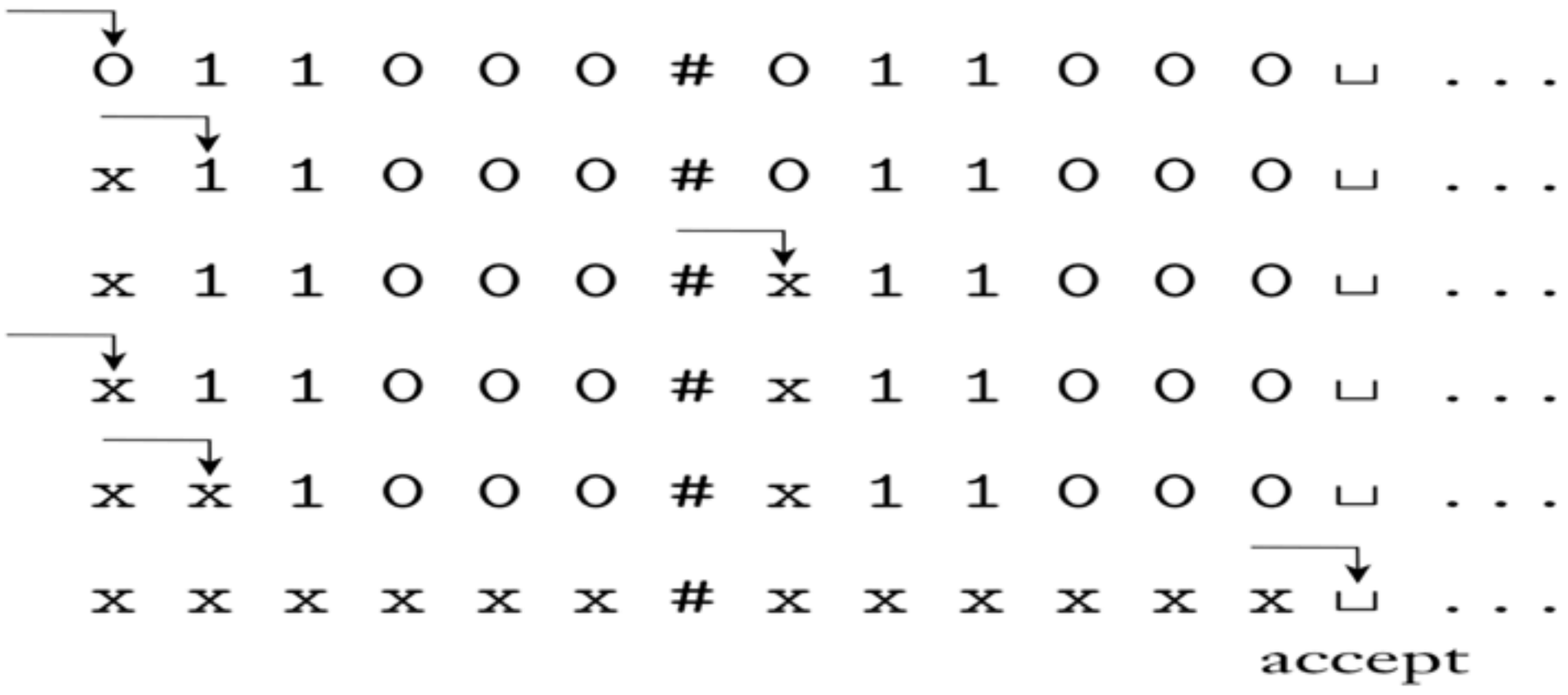
(that) Jan Piet Marie saw make swim

Gloss: that Jan saw Piet make Marie swim

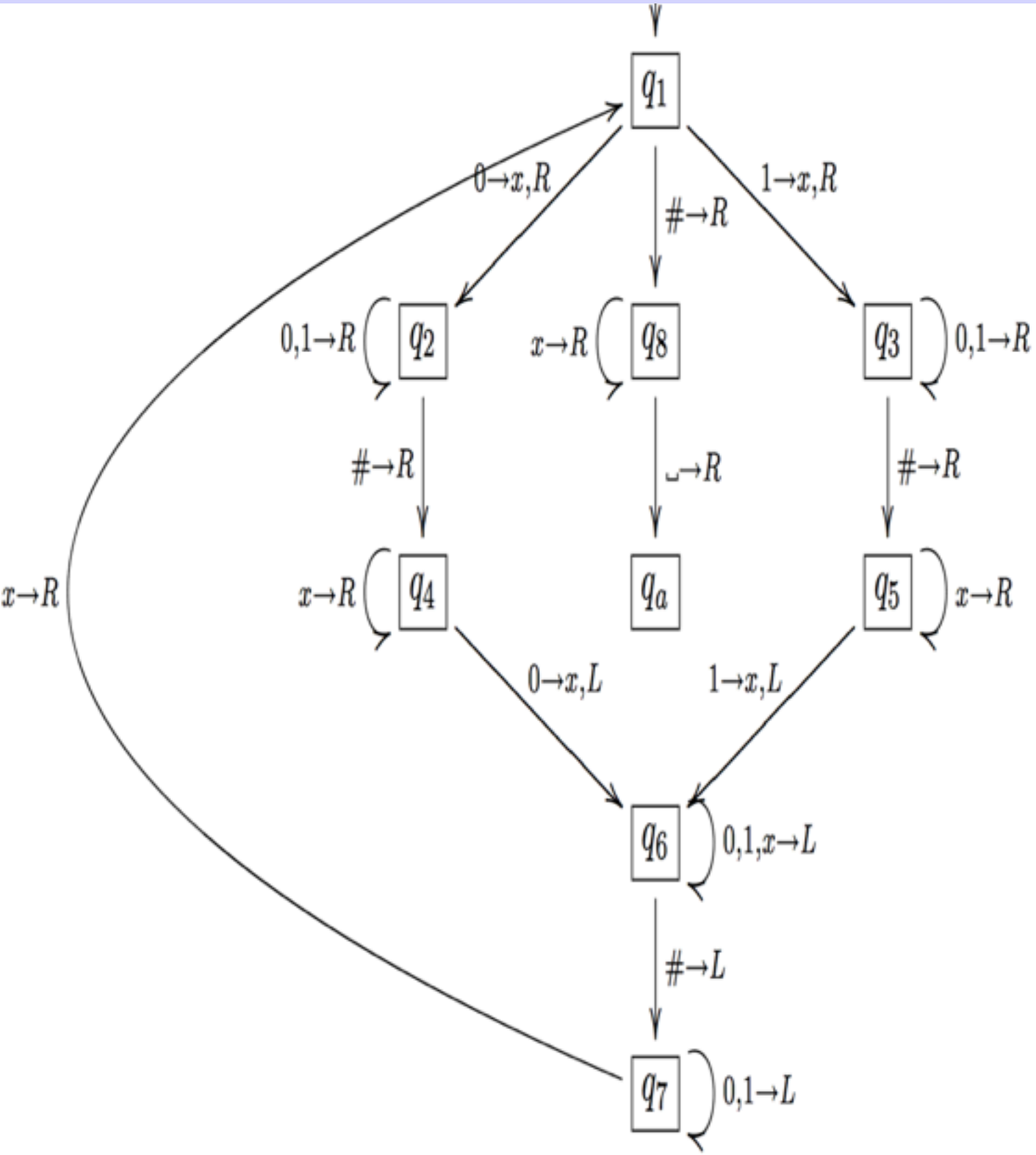
Turing Machine for w # w

M_1 = "On input string w:

1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, *reject*. Cross off symbols as they are checked to keep track of which symbols correspond.
2. When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right of the #. If any symbols remain, *reject*; otherwise, *accept*."



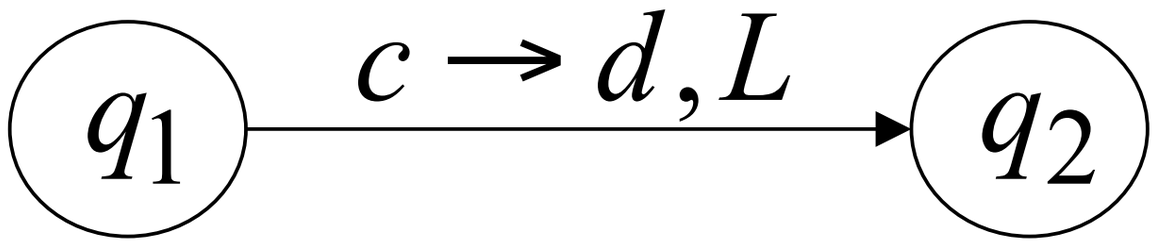
Turing Machine for w # w



$q_1001\#001\sqcup \Rightarrow xq_201\#001\sqcup \Rightarrow$
$x0q_21\#001\sqcup \Rightarrow x01\#q_4001\sqcup \Rightarrow$
$x01q_6\#x01\sqcup \Rightarrow x0q_71\#x01\sqcup \Rightarrow$
$xq_701\#x01\sqcup \Rightarrow q_7x01\#x01\sqcup \Rightarrow$
$xq_101\#x01\sqcup \Rightarrow xxq_21\#x01\sqcup \Rightarrow$
$xx1q_2\#x01\sqcup \Rightarrow xx1\#q_4x01\sqcup \Rightarrow$
$xx1\#xq_401\sqcup \Rightarrow xx1\#q_6xx1\sqcup \Rightarrow$
$xx1q_6\#xx1\sqcup \Rightarrow xxq_71\#xx1\sqcup \Rightarrow$
$xq_7x1\#xx1\sqcup \Rightarrow xxq_11\#xx1\sqcup \Rightarrow$
$xxxq_3\#xx1\sqcup \Rightarrow xxx\#q_5xx1\sqcup \Rightarrow$
$xxx\#xq_5x1\sqcup \Rightarrow xxx\#xxq_51\sqcup \Rightarrow$
$xxx\#xq_6xx \Rightarrow xxx\#q_6xxx \Rightarrow$
$xxxq_6\#xxx \Rightarrow xxq_7x\#xxx \Rightarrow$
$xxxq_1\#xxx \Rightarrow xxx\#q_8xxx \Rightarrow$
$xxx\#xq_8xx \Rightarrow xxx\#xxq_8x\sqcup \Rightarrow$
$xxx\#xxxq_8\sqcup \Rightarrow xxx\#xxxq_{accept}\sqcup \Rightarrow$

Formal Definitions for Turing Machines

Transition Function



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

Turing Machine:

States

Input alphabet

Tape alphabet

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \Diamond, F)$$

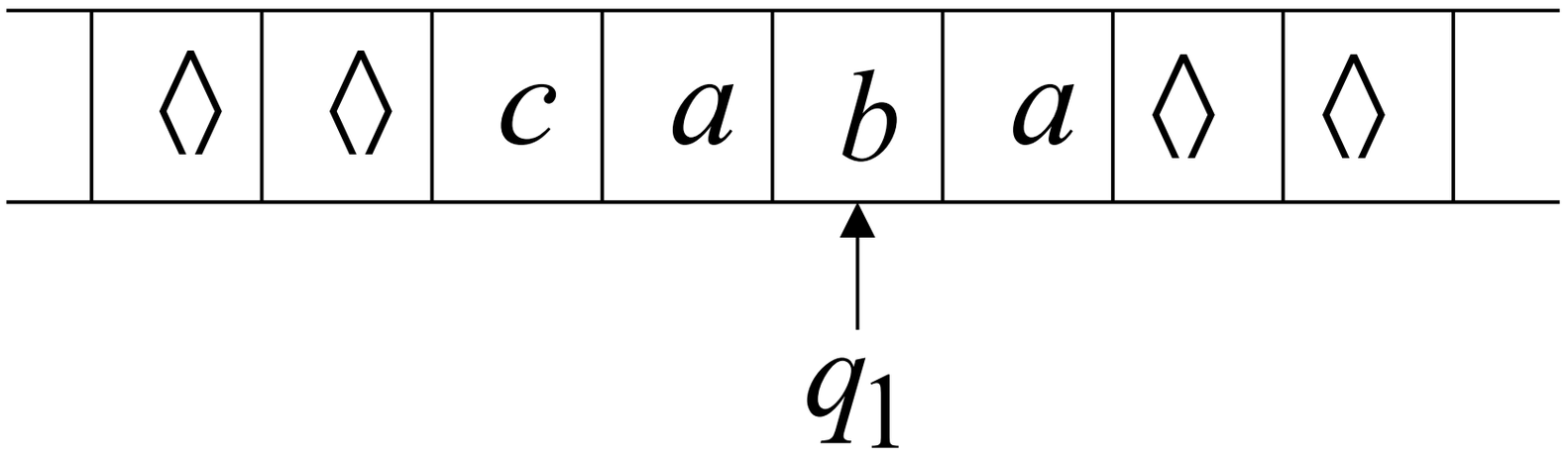
A partial Transition function

Initial state

Blank : a special symbol Of Γ

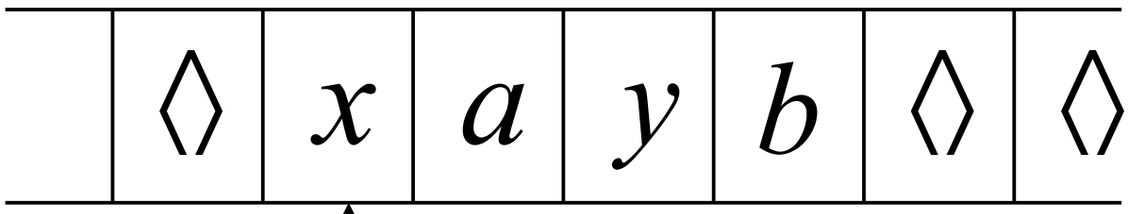
Final states

Configuration

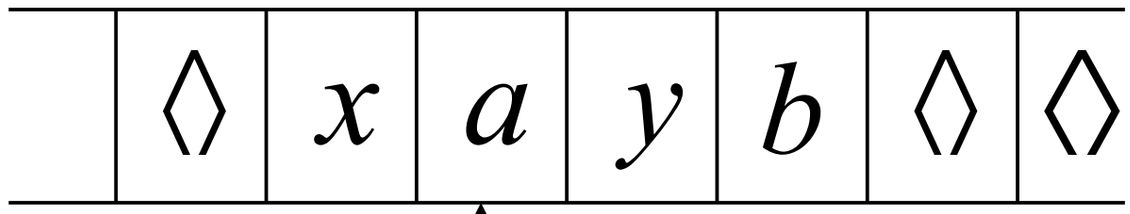


Instantaneous description: $ca\ q_1\ ba$

Time 4



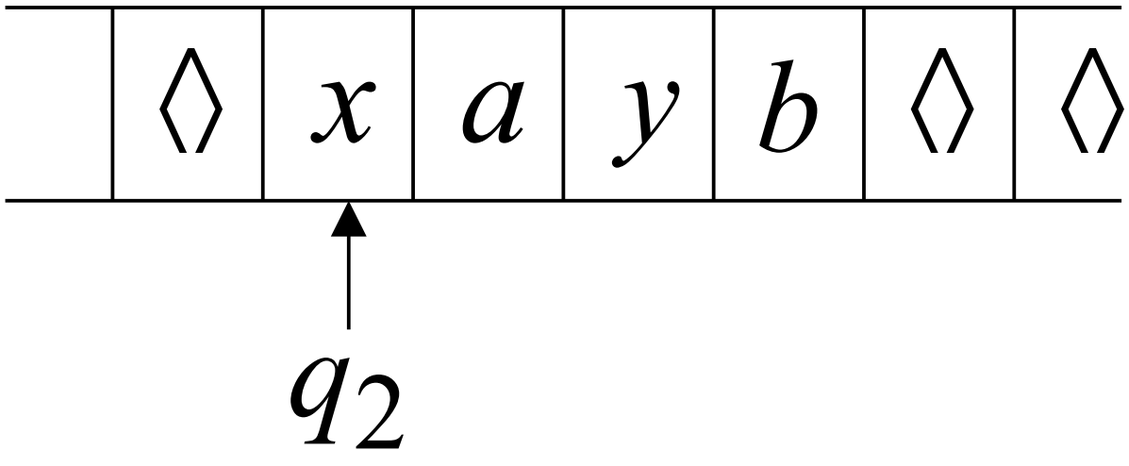
Time 5



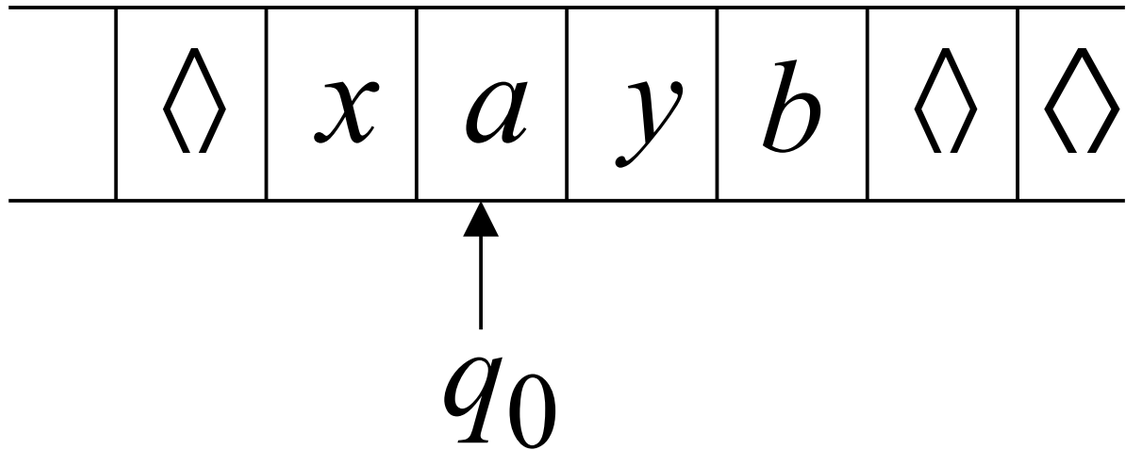
A Move:

$$q_2\ xayb \succ x\ q_0\ ayb$$

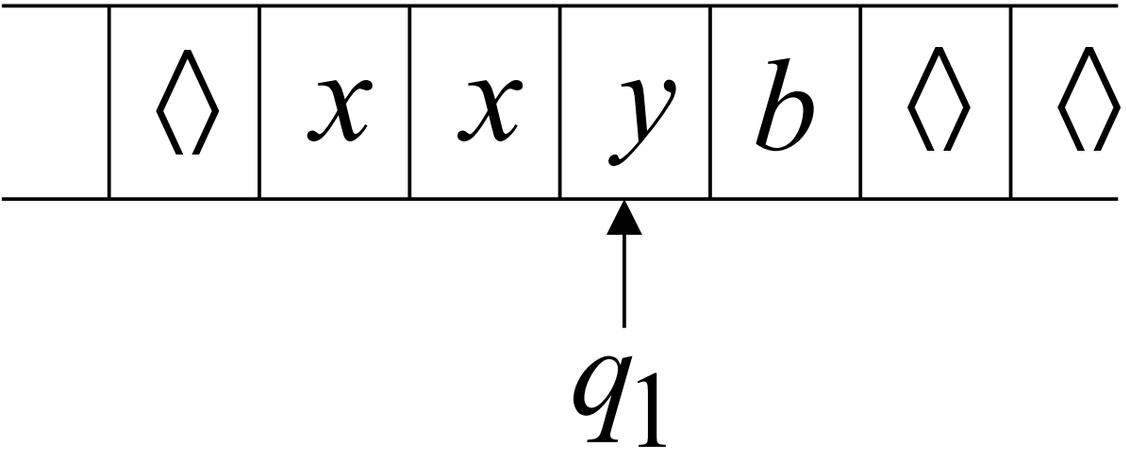
Time 4



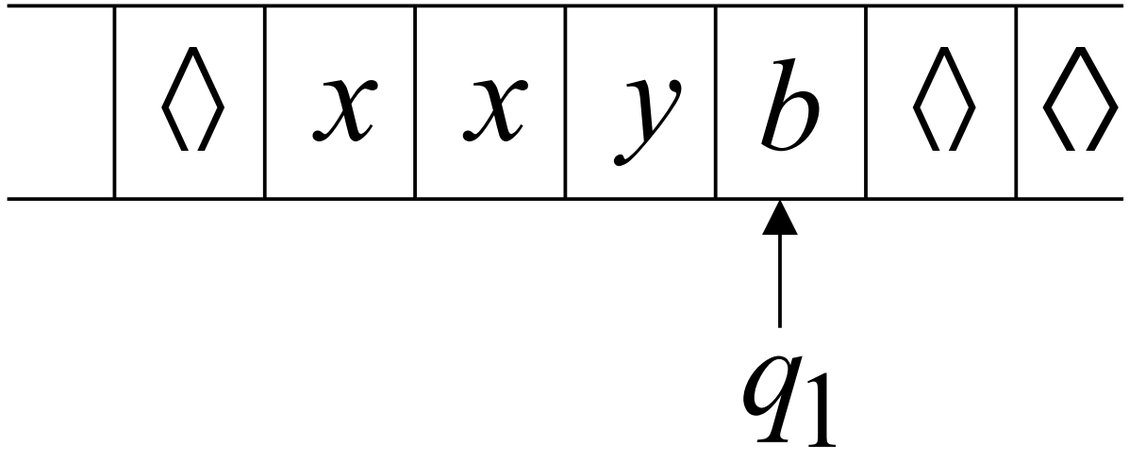
Time 5



Time 6



Time 7



$$q_2\ xayb \succ x\ q_0\ ayb \succ xx\ q_1\ yb \succ xxy\ q_1\ b$$

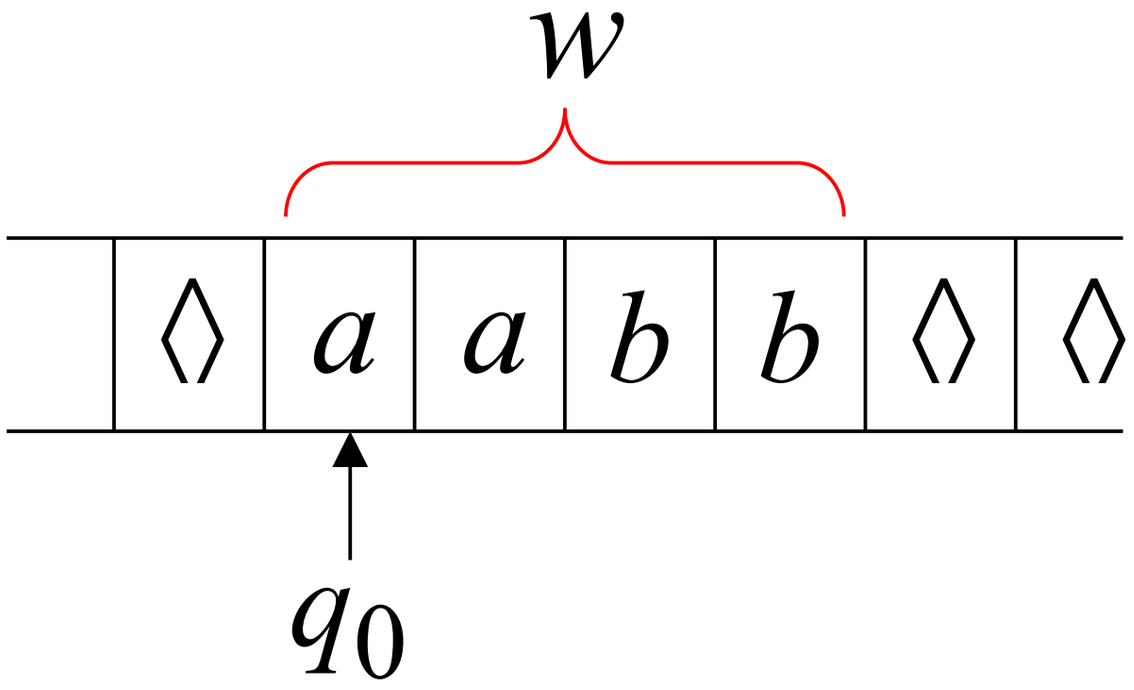
$$q_2\ xayb \ \succ\ x\ q_0\ ayb \ \succ\ xx\ q_1\ yb \ \succ\ xxy\ q_1\ b$$

*

Equivalent notation: $q_2\ xayb \ \succ\ xxy\ q_1\ b$

Initial configuration: $q_0\ w$

Input string



The Accepted Language

For any Turing Machine M

$$L(M) = \{w : \quad q_0\, w \overset{*}{\succ} x_1\, q_f\, x_2\}$$

Initial state

Final state

TM as Transducers/Computers

- TM as acceptor: $q_0\, w \vdash^* x\, q_f\, y$ accepts w
- TM as transducer: $q_0\, w \vdash^* q_f\, f(w)$ input w, output $f(w)$
  - example 1: $f(w) = ww$ for w in 1^+
    - replace every 1 by an x
    - find the rightmost x and replace it with 1
    - travel to the right end of nonblank region and create a 1
    - repeat the above 2 steps until there are no more x's

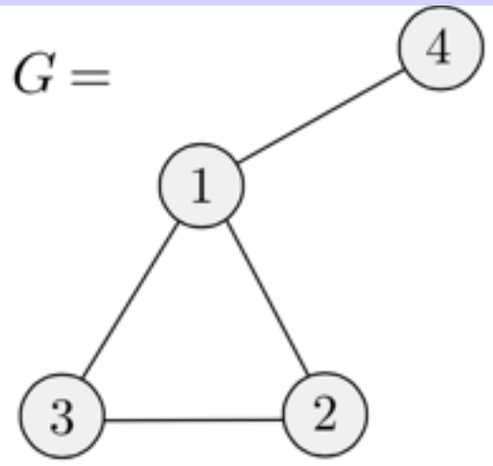
TM as Transducers/Computers

- TM as acceptor: $q_0\ w \vdash^* x\ q_f\ y$ accepts w
- TM as transducer: $q_0\ w \vdash^* q_f\ f(w)$ input w, output $f(w)$
  - example 2: $f(x) = x+1$ in unary for x in 1^+
  - example 3: $f(x) = x+y$ in unary for x, y in 1^+
  - example 4: $f(x) = x<y$ in unary for x, y in 1^+

Graph Example

EXAMPLE 3.23

Let A be the language consisting of all strings representing undirected graphs that are connected. Recall that a graph is ***connected*** if every node can be reached from every other node by traveling along the edges of the graph. We write

$G =$ 

$$A = \{\langle G \rangle \mid G \text{ is a connected undirected graph}\}.$$

$\langle G \rangle =$

```
(1,2,3,4)((1,2),(2,3),(3,1),(1,4))
```

The following is a high-level description of a TM M that decides A.

$M =$ "On input $\langle G \rangle$, the encoding of a graph G:

1. Select the first node of G and mark it.
2. Repeat the following stage until no new nodes are marked:
3. For each node in G, mark it if it is attached by an edge to a node that is already marked.
4. Scan all the nodes of G to determine whether they all are marked. If they are, *accept*; otherwise, *reject*."

Computability

DEFINITION 5.17

A function $f: \Sigma^* \longrightarrow \Sigma^*$ is a ***computable function*** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.

Recursively Enumerable and Recursive Languages

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

Definition:

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

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

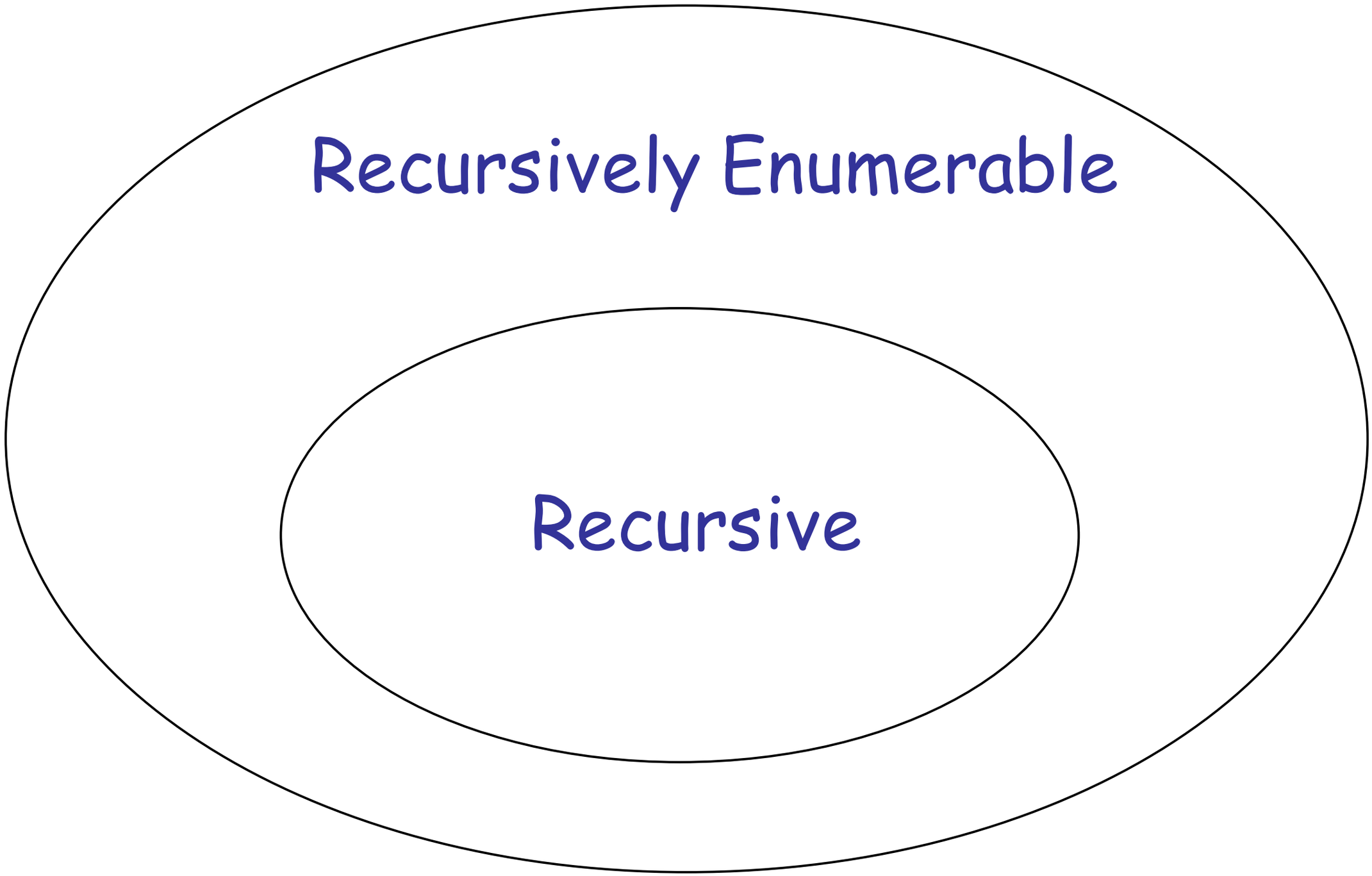
if $w \notin L$ then M halts in a non-final state

We will prove:

1. There is a specific language
   which is not recursively enumerable
   (not accepted by any Turing Machine)

2. There is a specific language
   which is recursively enumerable
   but not recursive

Non Recursively Enumerable



We will first prove:

- If a language is recursive then
  there is an enumeration procedure for it

- A language is recursively enumerable
  if and only if
  there is an enumeration procedure for it

The Chomsky Hierarchy

Unrestricted Grammars:

Productions

$$u \rightarrow v$$

String of variables and terminals

String of variables and terminals

Example unrestricted grammar:

$$S \rightarrow aBc$$

$$aB \rightarrow cA$$

$$Ac \rightarrow d$$

Theorem:

A language L is recursively enumerable if and only if L is generated by an unrestricted grammar

Context-Sensitive Grammars:

Productions

$$u \rightarrow v$$

String of variables and terminals

String of variables and terminals

and: $|u| \leq |v|$

The language $\{a^n b^n c^n\}$

is context-sensitive:

$$S \rightarrow abc \mid aAbc$$

$$Ab \rightarrow bA$$

$$Ac \rightarrow Bbcc$$

$$bB \rightarrow Bb$$

$$aB \rightarrow aa \mid aaA$$

The language $\{a^n b^n c^n\}$

is context-sensitive:

$$S \rightarrow abc \mid aAbc$$

$$Ab \rightarrow bA$$

$$Ac \rightarrow Bbcc$$

$$bB \rightarrow Bb$$

$$aB \rightarrow aa \mid aaA$$

Theorem:

A language L is context sensistive

if and only if

L is accepted by a Linear-Bounded automaton

Observation:

There is a language which is context-sensitive but not recursive

The Chomsky Hierarchy