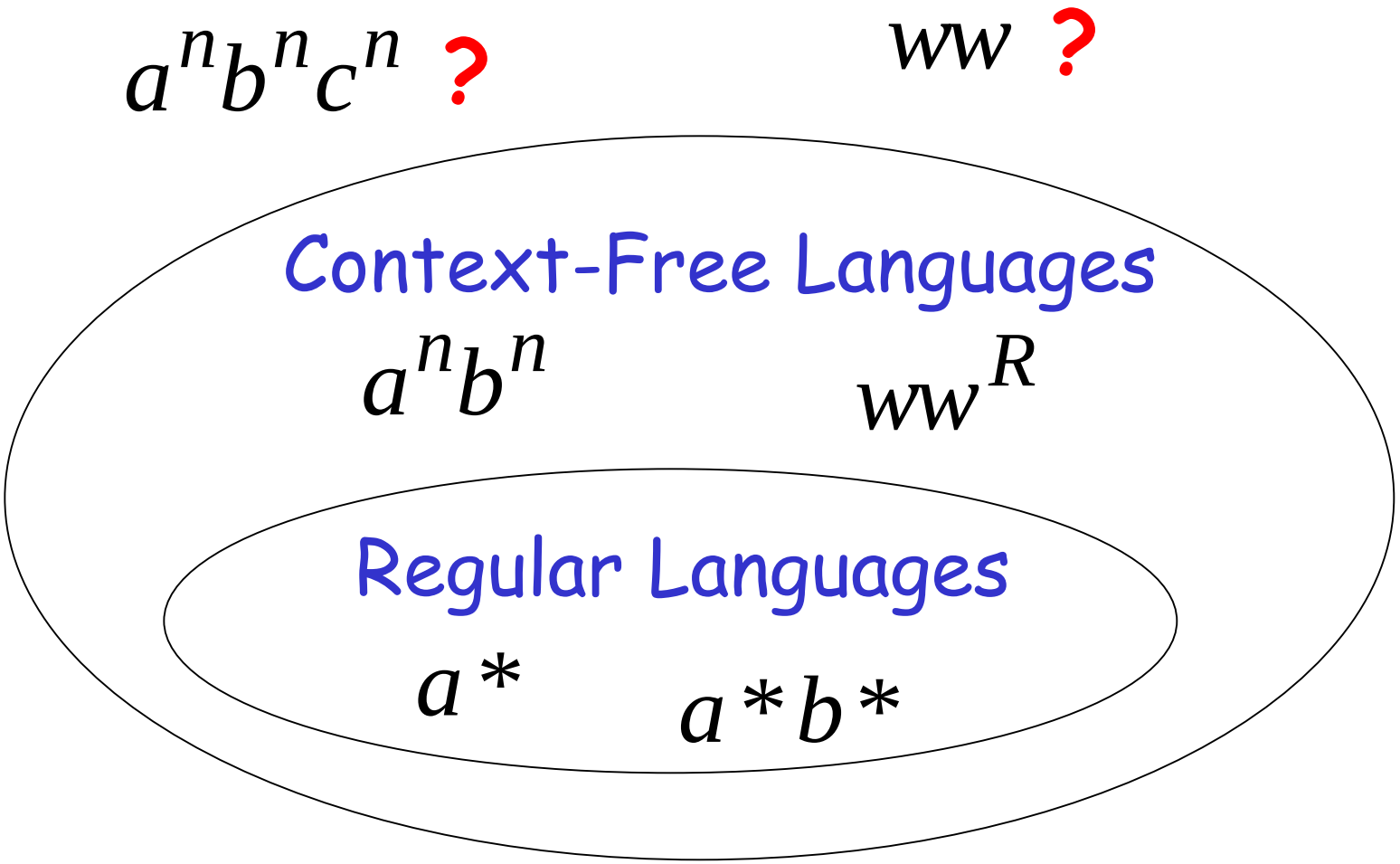
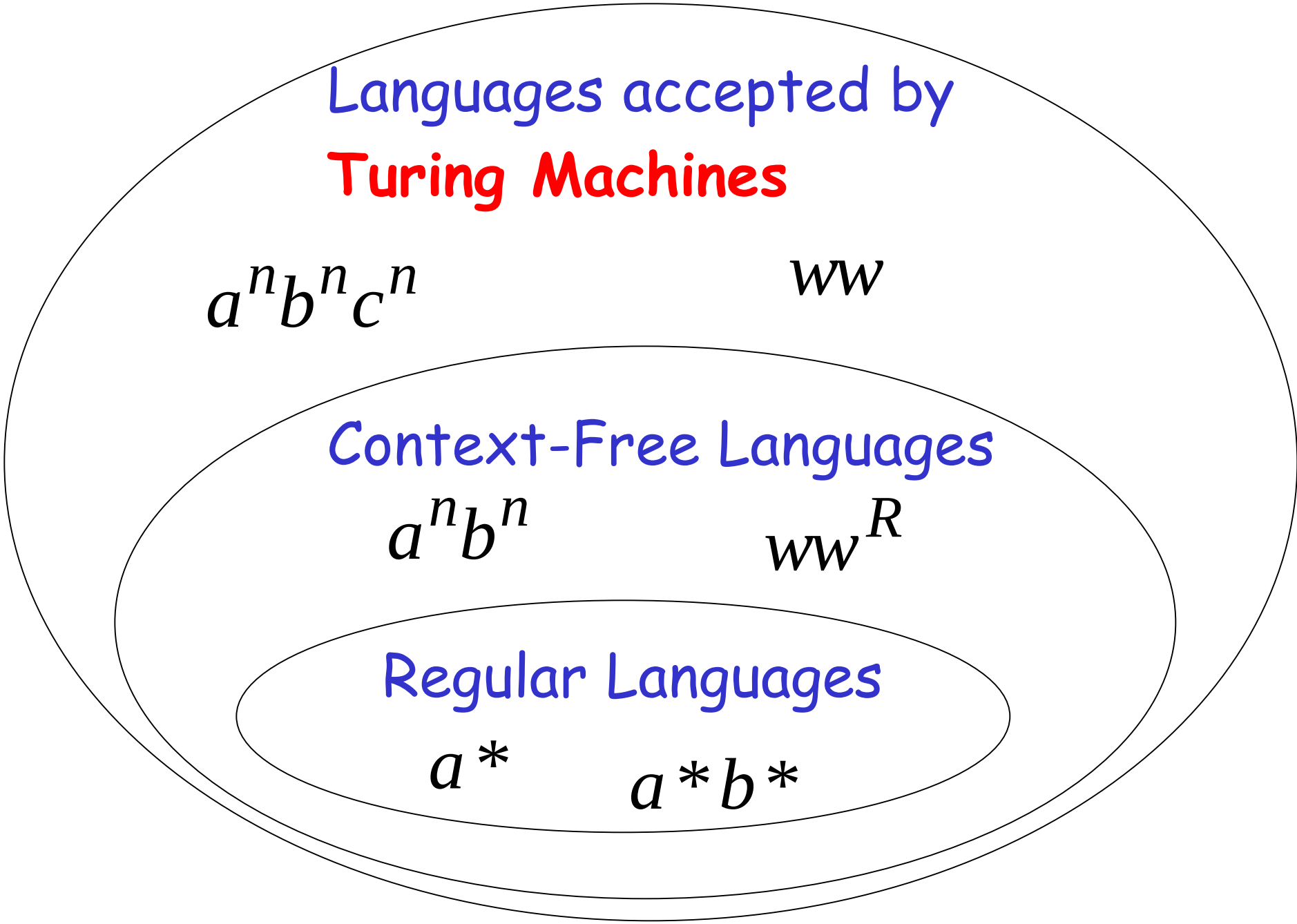
# Formal Languages

# Turing Machines

| Module-5 | |
|---|---|
| **TURING MACHINES AND OTHER MODELS OF TURING MACHINES:**<br>The Standard Turing Machine, Nondeterministic Turing Machines, A Universal Turing Machine.<br><br>**A HIERARCHY OF FORMAL LANGUAGES & AUTOMATA**<br>Recursive and Recursively Enumerable Languages, Unrestricted grammars, Context-Sensitive Grammars and Languages, The Chomsky Hierarchy. - Self Study part<br><br>**LIMITS OF ALGORITHMIC COMPUTATION**<br>Some Problems That Cannot Be Solved by Turing Machines, The Post Correspondence Problem<br><br>**Text Book 1:** Chapter 9: 9.1, Chapter 10:10.3-10.4, Chapter 11: 11.1-11.4 Chapter 12: 12.1, 12.3 | **10 Hours** |

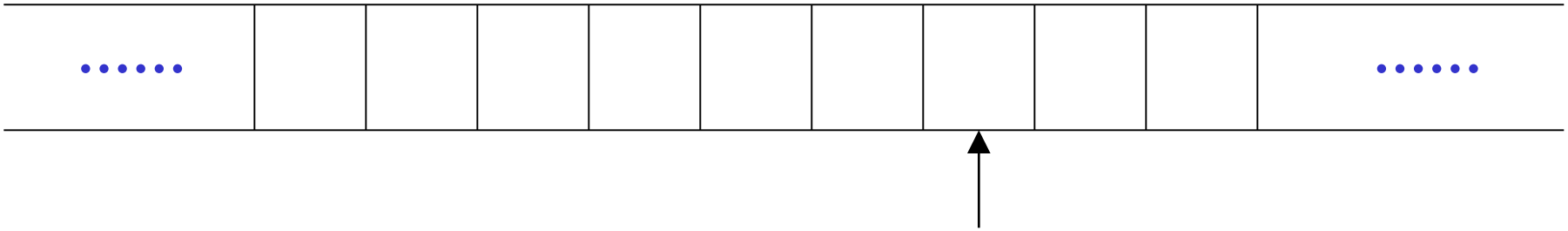
# The Language Hierarchy




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

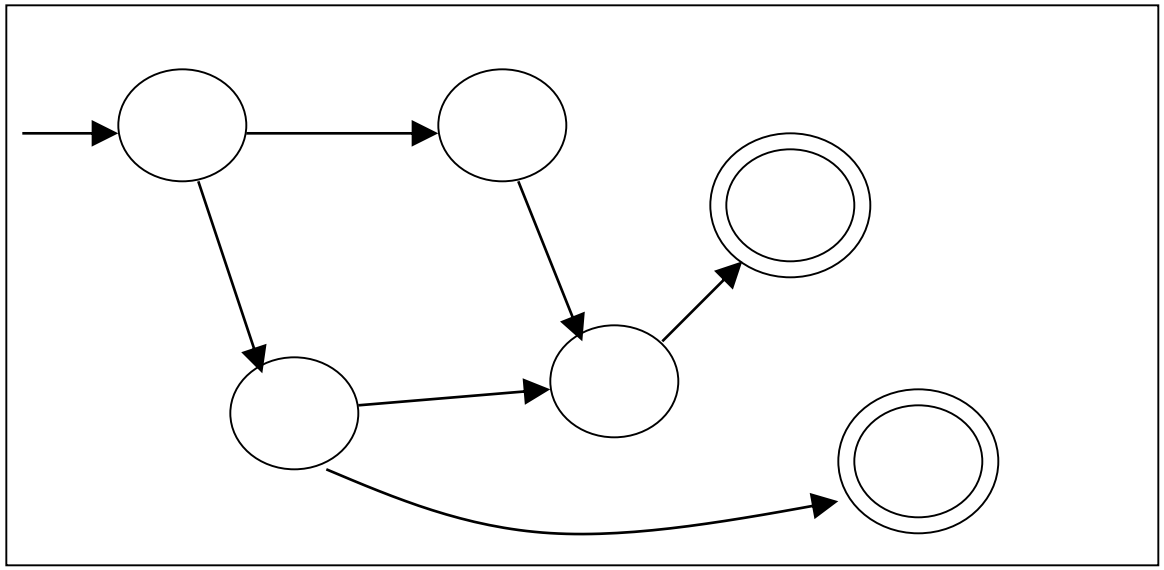
# A Turing Machine

Tape



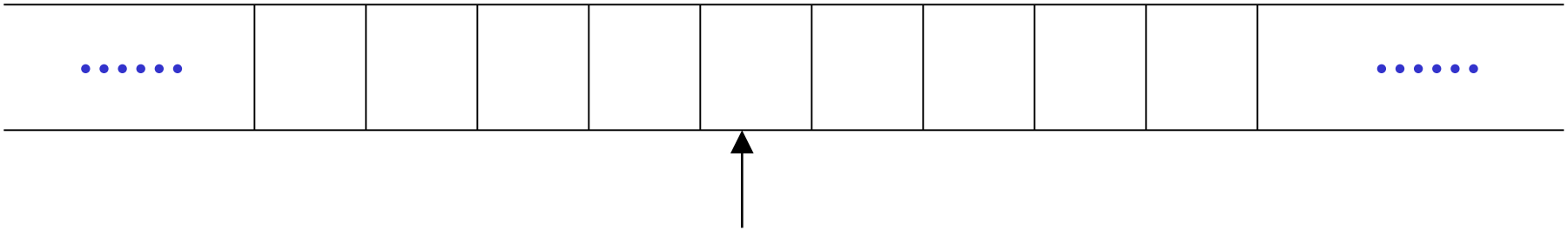

Read-Write head

Control Unit



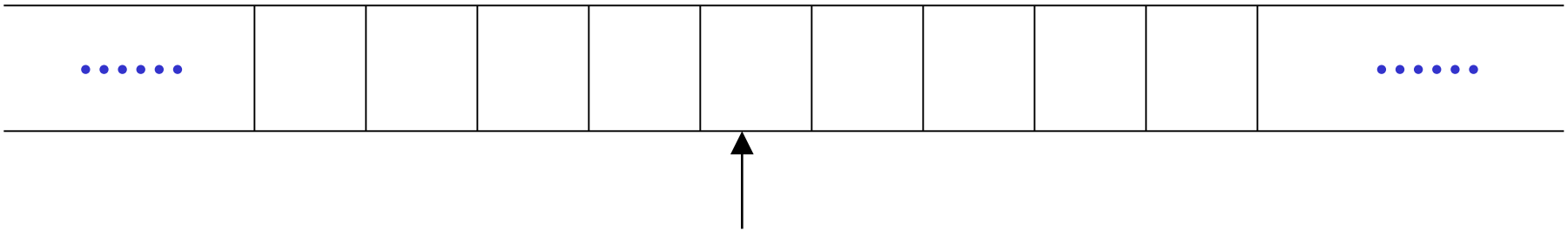
# The Tape

No boundaries -- infinite length




Read-Write head

The head moves Left or Right



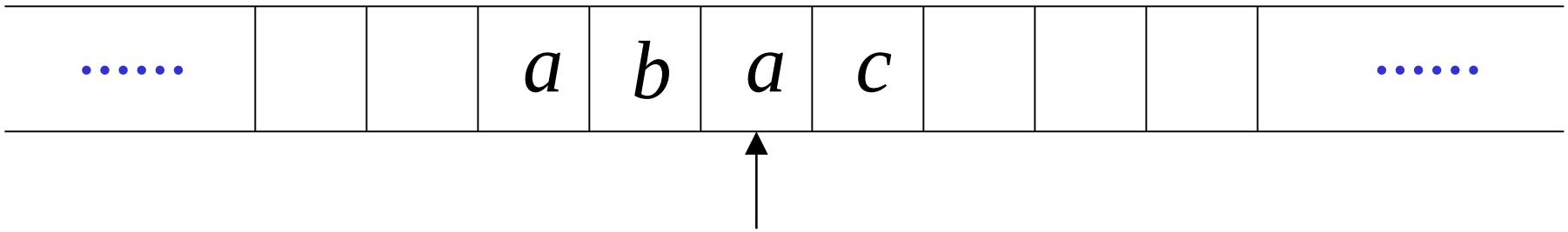

Read-Write head

The head at each time step:

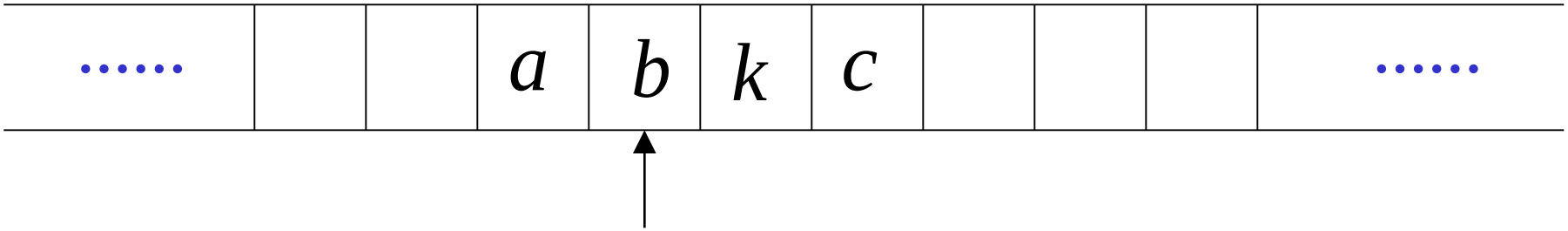
1. Reads a symbol
2. Writes a symbol
3. Moves Left or Right

Example:

Time 0

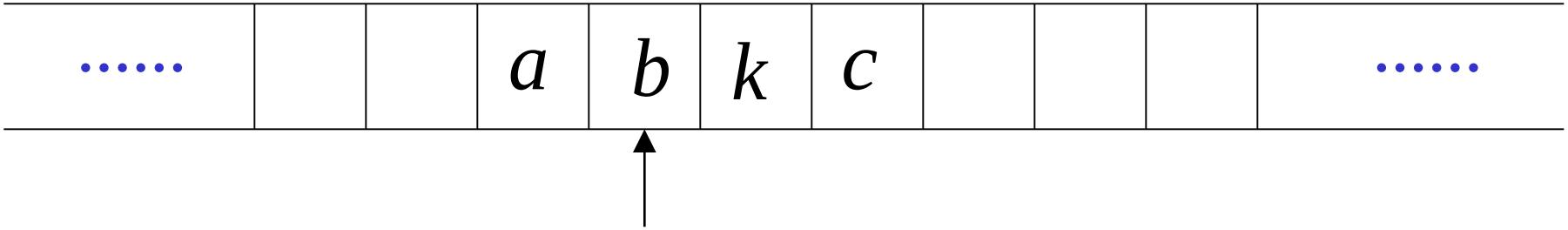



Time 1

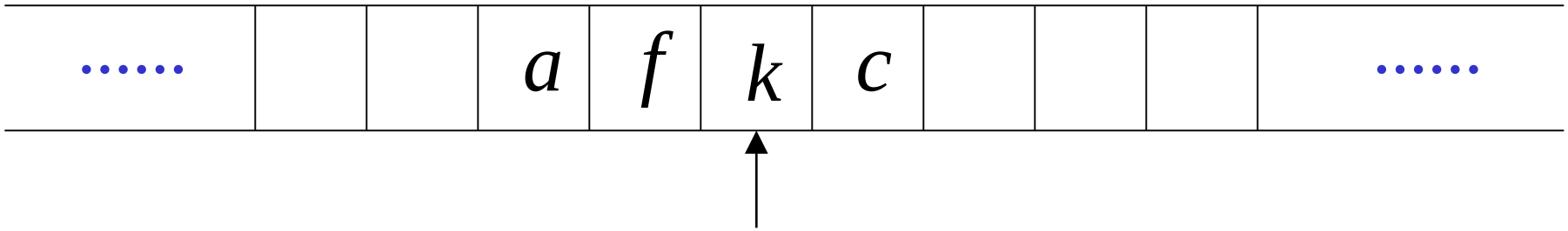



1. Reads $a$
2. Writes $k$
3. Moves Left

Time 1

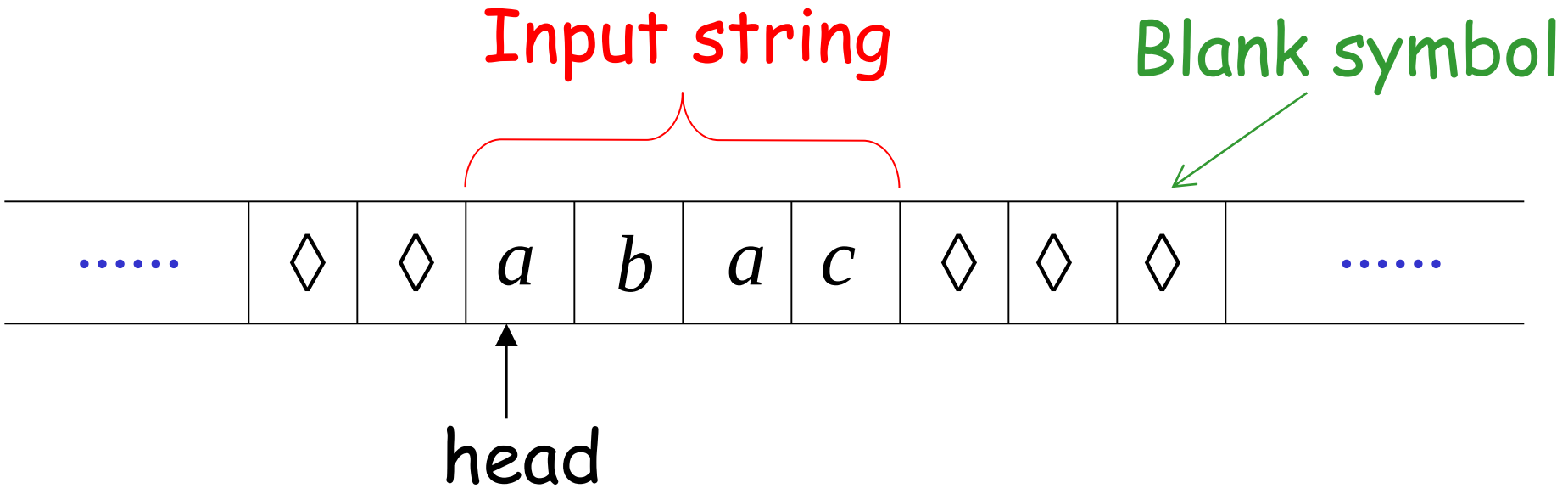



Time 2




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

# The Input String




Head starts at the leftmost position of the input string

**Turing thesis**, which maintains that any computational process, such as those carried out by present-day computers, can be done on a Turing machine.

## The Standard Turing Machine

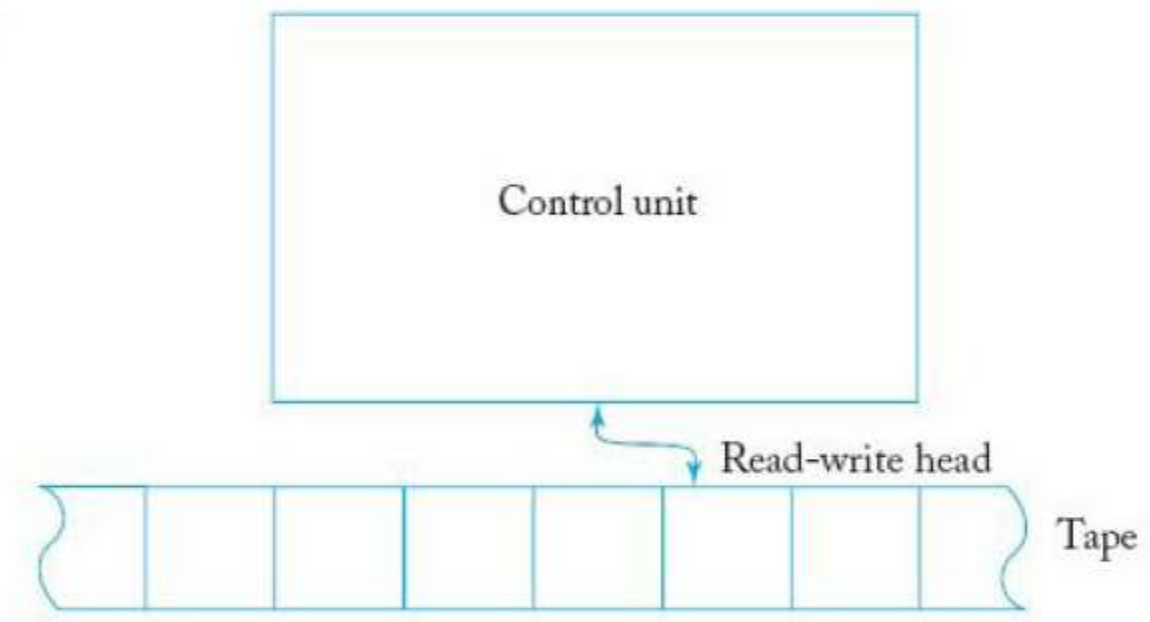
It can be visualized as

- → a single, one-dimensional array of cells, each of which can hold a single symbol.
- → This array extends indefinitely in both directions and is therefore capable of holding an unlimited amount of information.
- → The information can be read and changed in any order.
- → We will call such a storage device a **tape** because it is analogous to the magnetic tapes used in older computers.

# Definition of a Turing Machine

- A Turing machine is an automaton whose temporary storage is a tape.
- This tape is divided into cells, each of which is capable of holding one symbol.
- Associated with the tape is a **read-write head** that can travel right or left on the tape and that can read and write a single symbol on each move.
- The automaton that we use as a Turing machine will have neither an input file nor any special output mechanism.
- Whatever input and output is necessary will be done on the machine's tape.

# Turing Machine



A Turing machine $M$ is defined by

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

where

| | |
|---|---|
| $Q$ | is the set of internal states, |
| $\Sigma$ | is the input alphabet |
| $\Gamma$ | is the finite set of symbols called the **tape alphabet**, |
| $\delta$ | is the transition function, |
| $\square \in \Gamma$ | is a special symbol called the **blank**, |
| $q_0 \in Q$ | is the initial state, |
| $F \subseteq Q$ | is the set of final states. |

we assume that $\Sigma \subseteq \Gamma - \{\square\}$, that is, that the input alphabet is a subset of the tape alphabet, not including the blank.

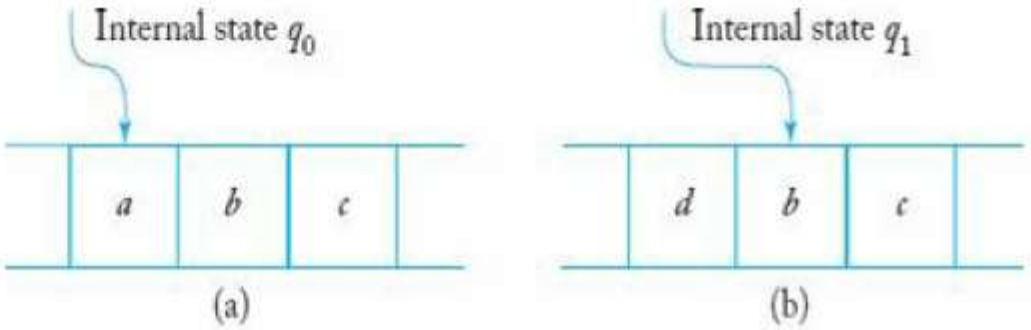
The transition function δ is defined as

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$$

Figure 9.2 shows the situation before and after the move

$$\delta\ (q_0, a) = (q_1, d, R).$$

**Figure 9.2**




The situation (a) before the move and (b) after the move.

- The whole process may terminate, which we achieve in a Turing machine by putting it into a **halt state.**
- A Turing machine is said to halt whenever it reaches a configuration for which δ is not defined.
- A Turing machine is said to halt whenever it reaches a configuration for which δ is defined or when string gets accepted.

Consider the Turing machine defined by

$$Q = \{q_0, q_1\},$$

$$\Sigma = \{a, b\},$$

$$\Gamma = \{a, b, \square\},$$

$$F = \{q_1\},$$

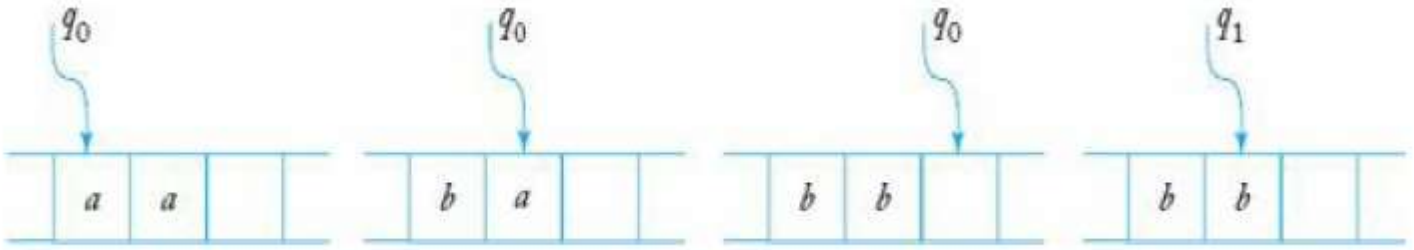
and

$$\delta(q_0, a) = (q_0, b, R),$$

$$\delta(q_0, b) = (q_0, b, R),$$

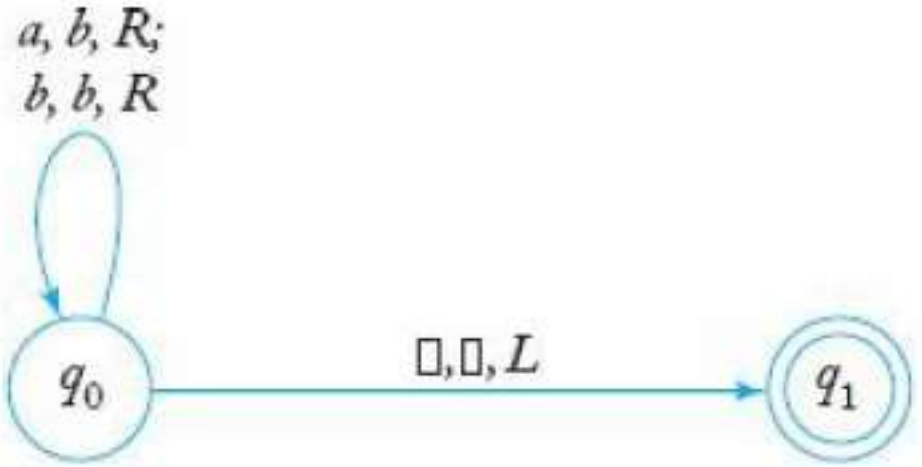
$$\delta(q_0, \square) = (q_1, \square, L),$$

- If this Turing machine is started in state $q_0$ with the symbol *a* under the read-write head, the applicable transition rule is $\delta(q_0,a) = (q_0,b,R)$.
- Therefore, the read-write head will replace the *a* with a 'b', then move right on the tape.
- The machine will remain in state $q_0$.
- Any subsequent *a* will also be replaced with a 'b', but b's will not be modified.
- When the machine encounters the first blank, it will move left one cell, then halt in final state $q_1$.

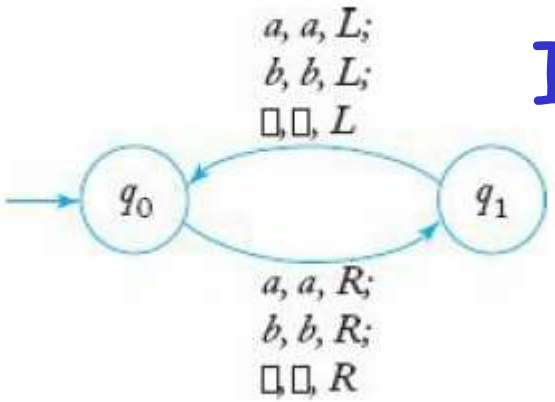



Sequence of moves

# Transition Graph




# Infinite Loop




In this TM does not halt

# Summarize the main features of our model

1. The Turing machine has a tape that is unbounded in both directions, allowing any number of left and right moves.

**2.** The Turing machine is deterministic in the sense that δ defines at most one move for each configuration.

**3.** There is no special input file. We assume that at the initial time the tape has some specified content. Some of this may be considered input.

4. Similarly, there is no special output device. Whenever the machine halts, some or all of the contents of the tape may be viewed as output.

The pictures drawn in Figure 9.3 correspond to the sequence of instantaneous descriptions $q_0aa$, $bq_0a$, $bbq_0\square$, $bq_1b$.

$$\delta(q_0, a) = (q_0, b, R),$$

$$\delta(q_0, b) = (q_0, b, R),$$

$$\delta(q_0, \square) = (q_1, \square, L),$$

q0aa, bq0a, bbq0 □ , bq1b


$$q_0aa \vdash bq_0a \vdash bbq_0\square \vdash bq_1b$$

or

$$q_0aa \overset{*}{\vdash} bq_1b.$$

A move from one configuration to another will be denoted by $\vdash$. Thus, if

$$\delta\,(q_1, c)=(q_2, e, R),$$

then the move

$$abq_1cd \vdash abeq_2d$$

The symbol $\overset{*}{\vdash}$ has the usual meaning of an arbitrary number of moves

The sequence of configurations leading to a halt state will be called a **computation.**

starting from the initial configuration $x_1qx_2$, the machine goes into a loop and never halts.

$$x_1qx_2 \overset{*}{\vdash} \infty,$$

# Turing Machines as Language Accepters

The machine is started in the initial state $q_0$ with the read write head positioned on the leftmost symbol of $\omega$. If, after a sequence of moves, the Turing machine enters a final state and halts, then $w$ is considered to be accepted.

Let $M = (Q, \Sigma, \Gamma, \delta; q_0, \square, F)$ be a Turing machine. Then the language accepted by $M$ is

$$L(M) = \left\{ w \in \Sigma^+ : q_0 w \overset{*}{\vdash} x_1 q_f x_2 \text{ for some } q_f \in F, x_1, x_2 \in \Gamma^* \right\}$$

The machine can halt in a non-final state or it can enter an infinite loop and never halt. Any string for which $M$ does not halt is by definition not in $L(M)$.

# Example 1:

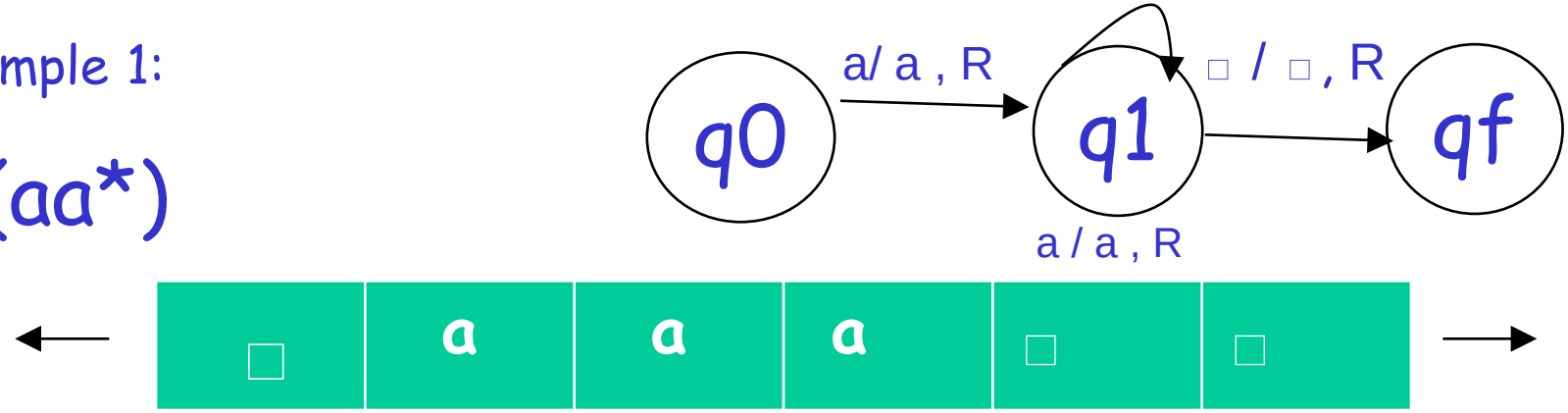
For Σ = {0,1}, design a Turing machine that accepts the language denoted by the regular expression $00^*$.

- Starting at the left end of the input, we read each symbol and check that it is a 0.
- If it is, we continue by moving right.
- If we reach a blank without encountering anything but 0, we terminate and accept the string
- If the input contains a 1 anywhere, the string is not in *L*(00*), and we halt in a non-final state.
- To keep track of the computation, two internal states *Q*= {*q*0,*q*1} and one final state *F*= {*q*1} can be assumed.

$$\delta(q_0, 0) = (q_0, 0, R),$$
$$\delta(q_0, \square) = (q_1, \square, R).$$

Example 1:

L(aa*)




M = (Q, Σ, Γ,δ, q0,, F)

Q= {q0, q1, qf}, Σ = {a}, Γ = {a, □}, F = {qf}

δ= Q x Γ → Q x Γ x {L, R}

δ( q0, a) = (q1, a, R)

δ( q1, a) = (q1, a, R)

δ( q1, □) = (qf, □, R)

ID: q0 a a a ⊢ a q1 a a ⊢ a a q1 a ⊢ a a a q1 □ ⊢ a a a □ qf □

Example 2:

# L(a( a+b)*) assume some string aba


M = (Q, Σ, Γ,δ, q0,, F), Q= {q0, q1, q2}, Σ = {a,b}, Γ = {a,b, □}, F = {q2} , δ= Q x Γ → Q x Γ x {L, R}

δ( q0, a) = (q1, a, R)

δ( q1, a) = (q1, a, R)

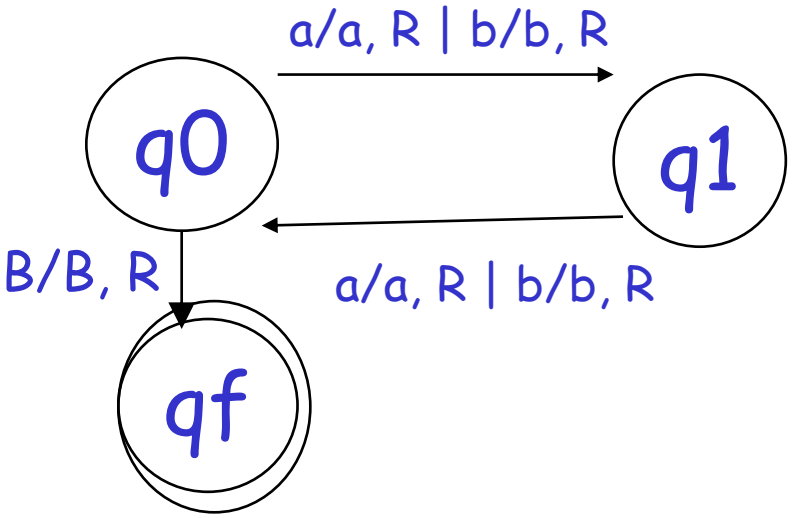
δ( q1, b) = (q1, b, R)

δ( q1, □) = (q2, □, R)

ID: q0 a b a ⊢ a q1 b a ⊢ a b q1 a ⊢ a b a q1 □ ⊢ a b a □ q2 □

Example 3:

L= {w | w is even and Σ = {a,b}, }



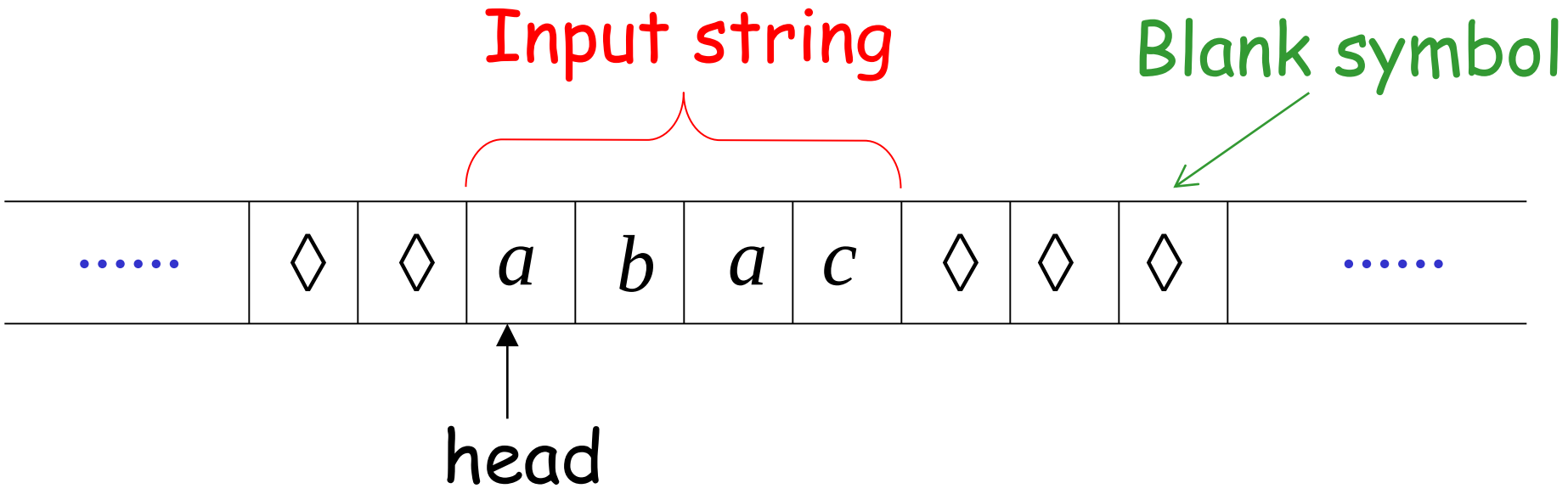

M = (Q, Σ, Γ,δ, q0,, F), Q= {q0, q1,q2}, Σ = {a,b}, Γ = {a,b, □}, F = {q2} , δ= Q x Γ → Q x Γ x {L, R}

δ( q0, a) = (q1, a, R) or δ( q0, b) = (q1, b, R)

δ( q1, a) = (qo, a, R) or δ( q1, b) = (q0, b, R)

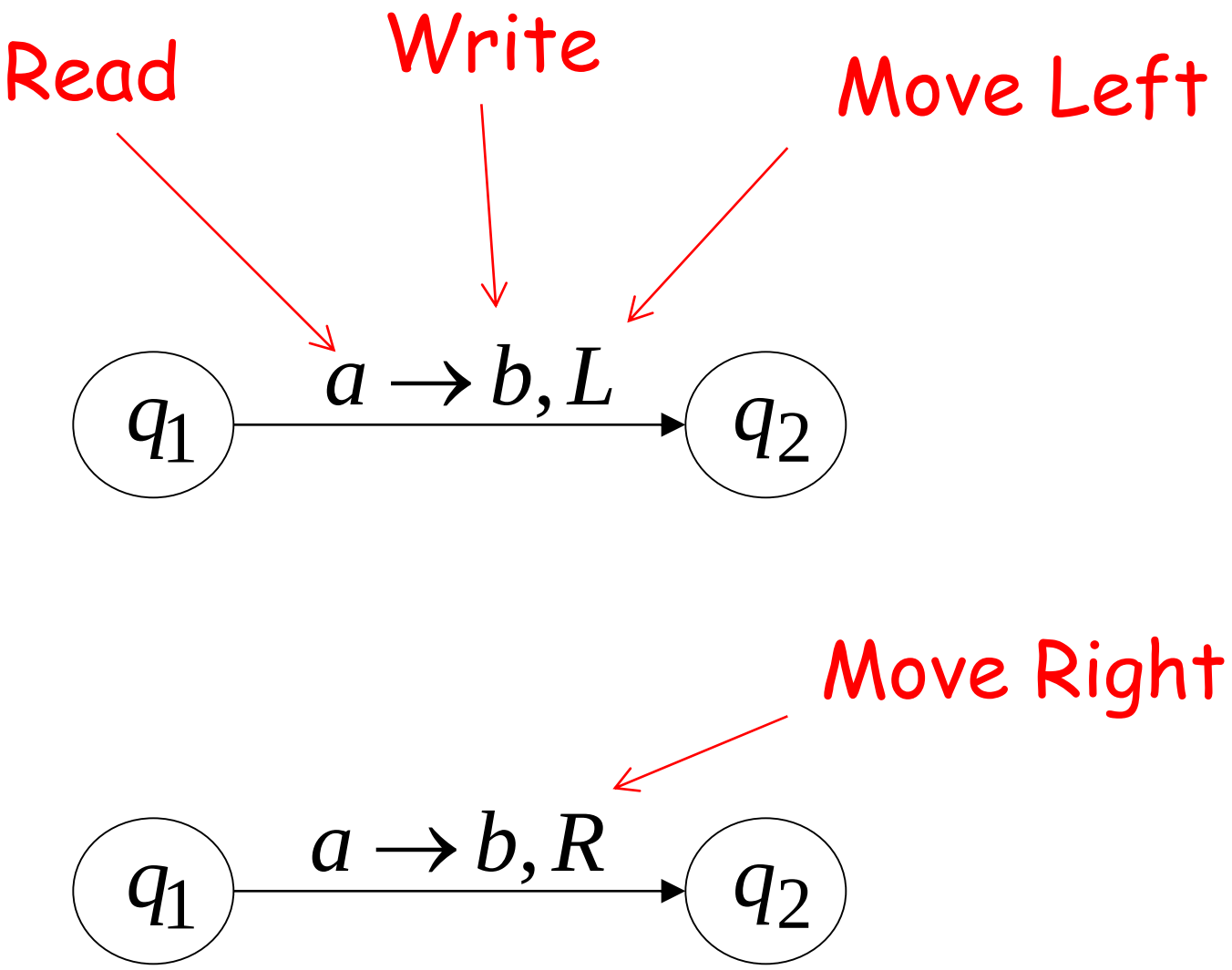
δ( q0, □) = (q2, □, R)

ID: q0 a b a b ⊢ a q1 b ab ⊢ a b q0 a b ⊢ a b a q1b □ ⊢ a b a b q0 □ ⊢ a b a b □ q2 □



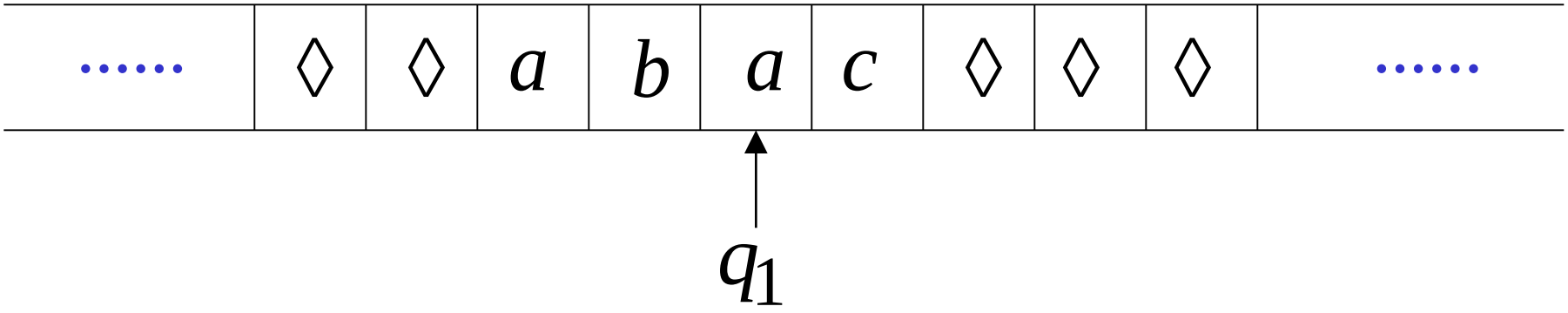

Remark: the input string is never empty

# States & Transitions

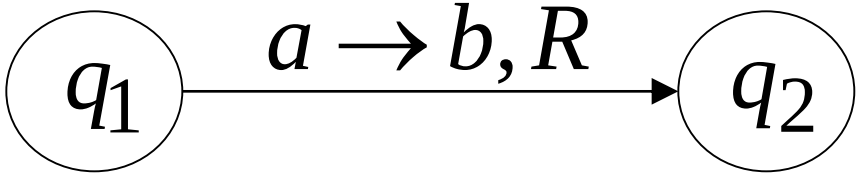


# Example:

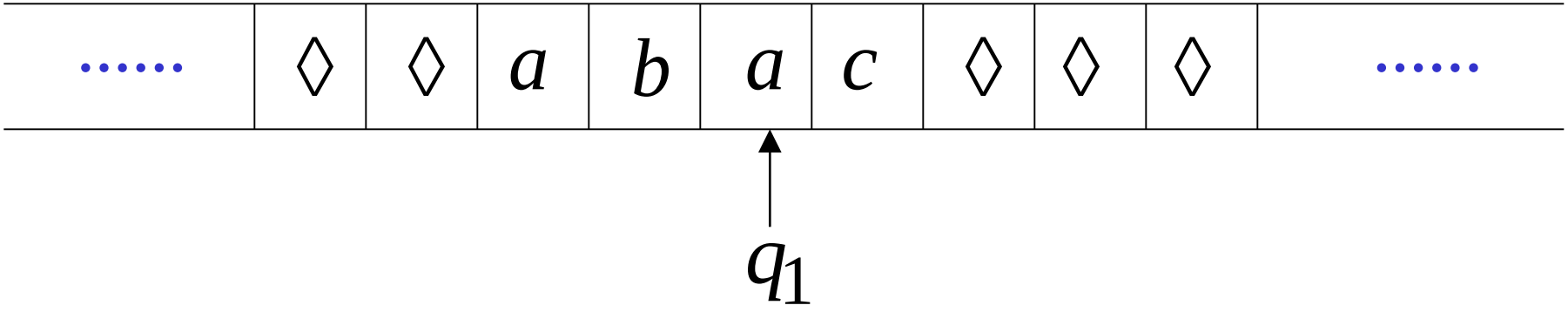
Time 1



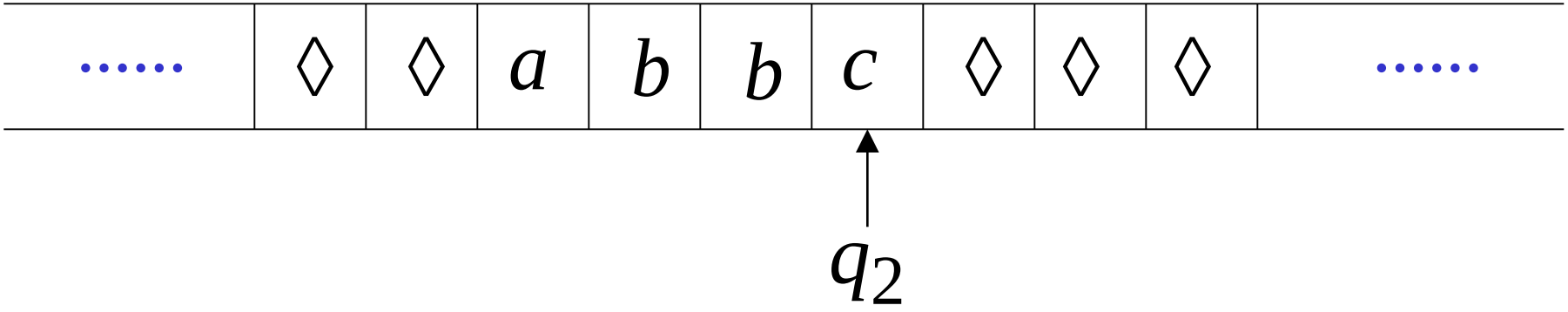

current state

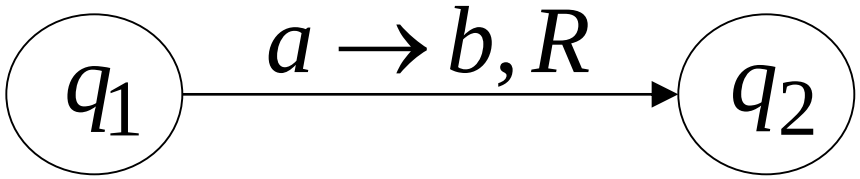


Time 1

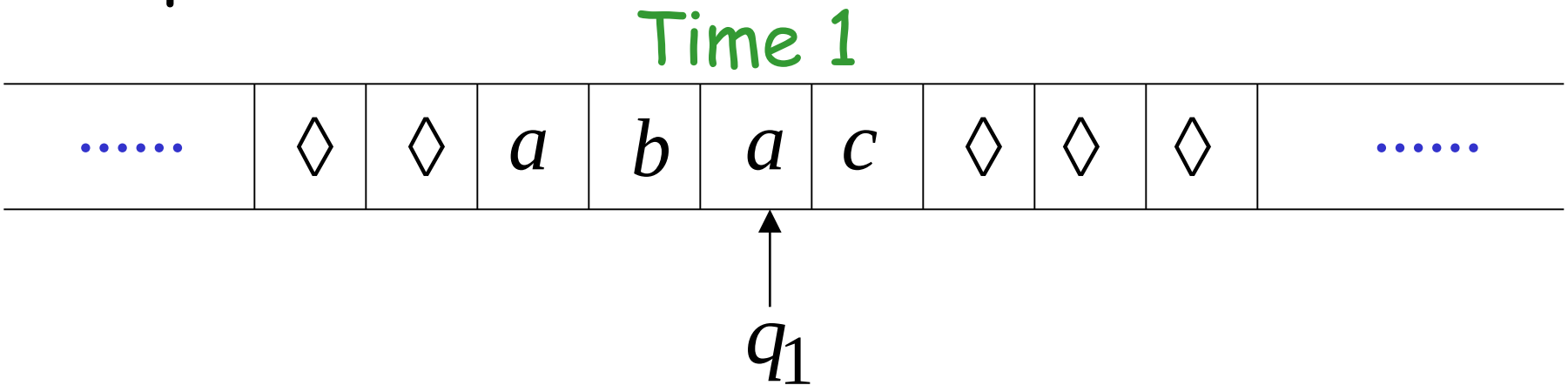


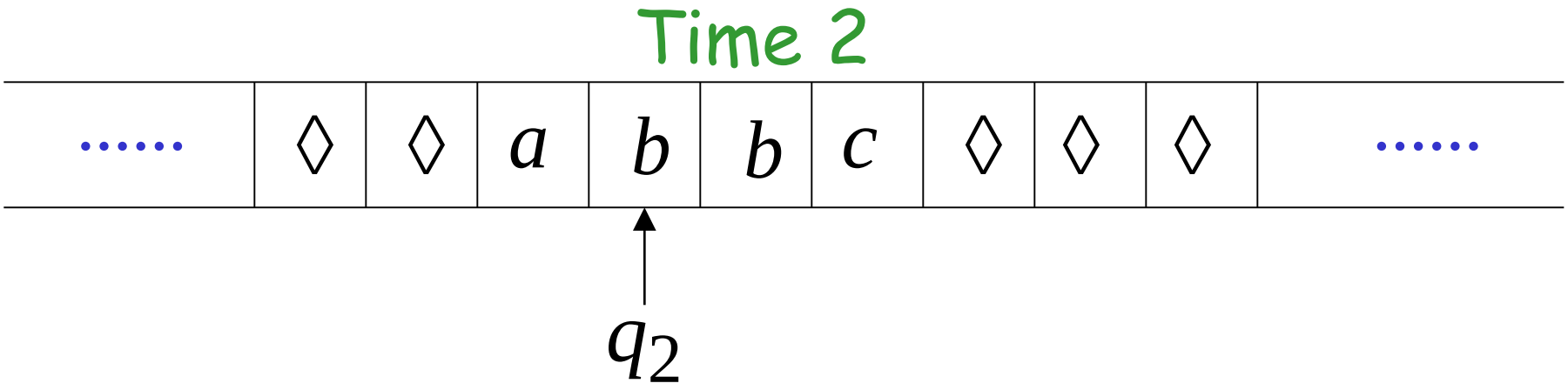

Time 2

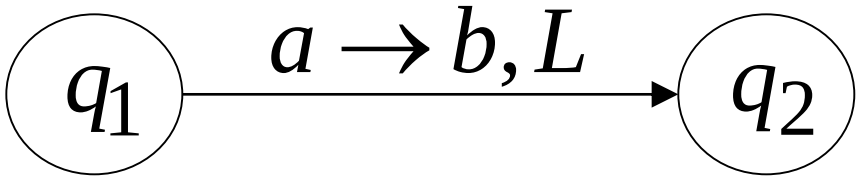


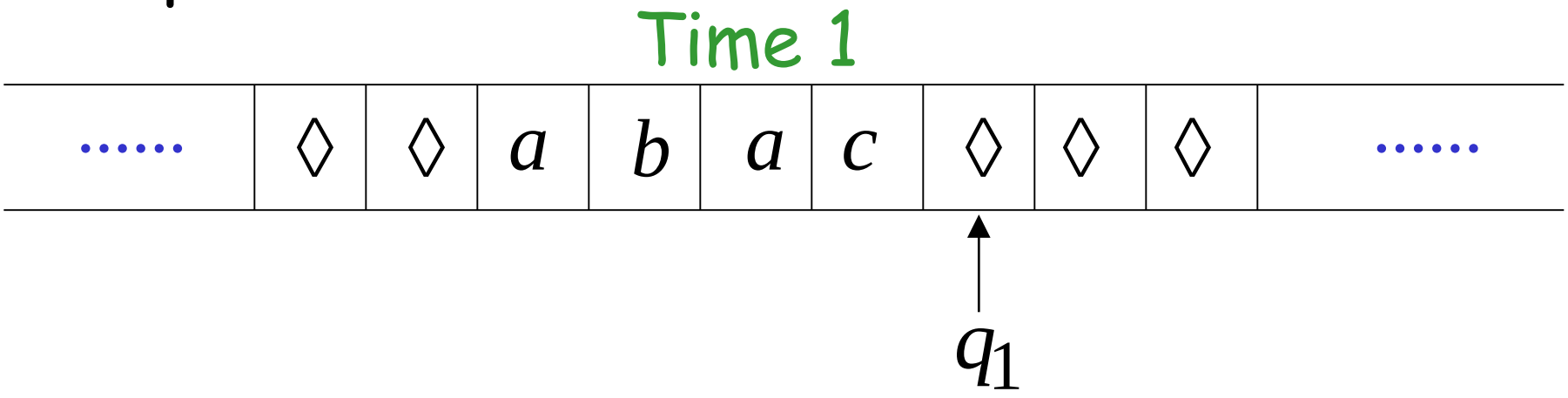
Example:

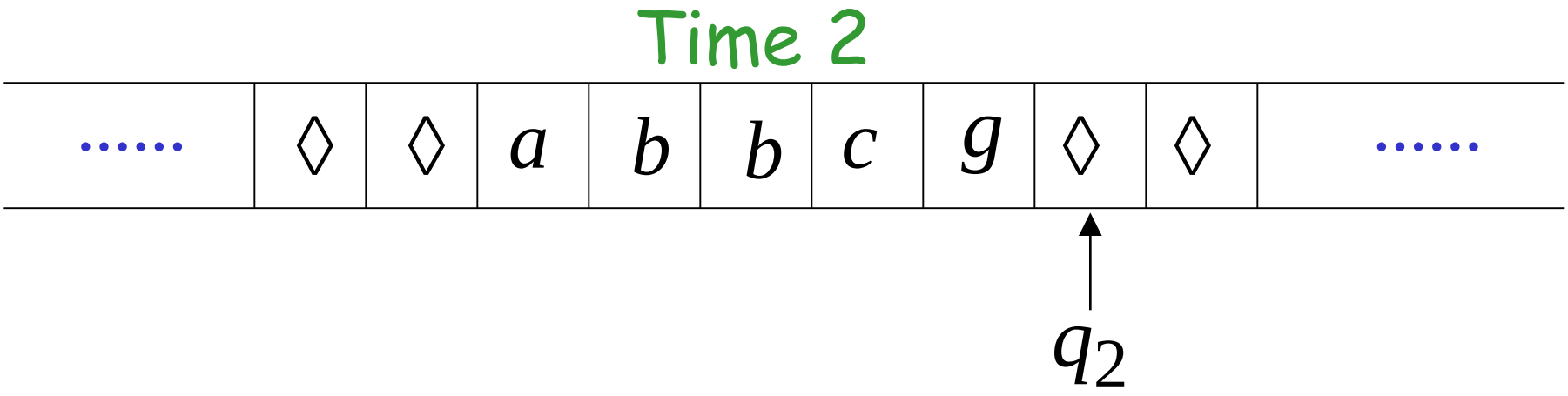


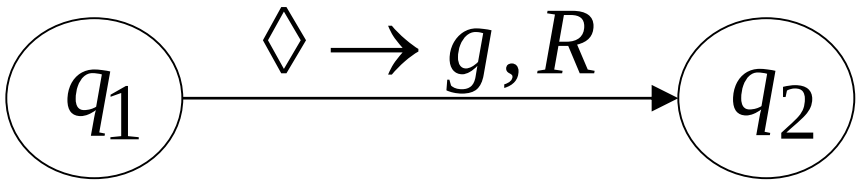



# Example:



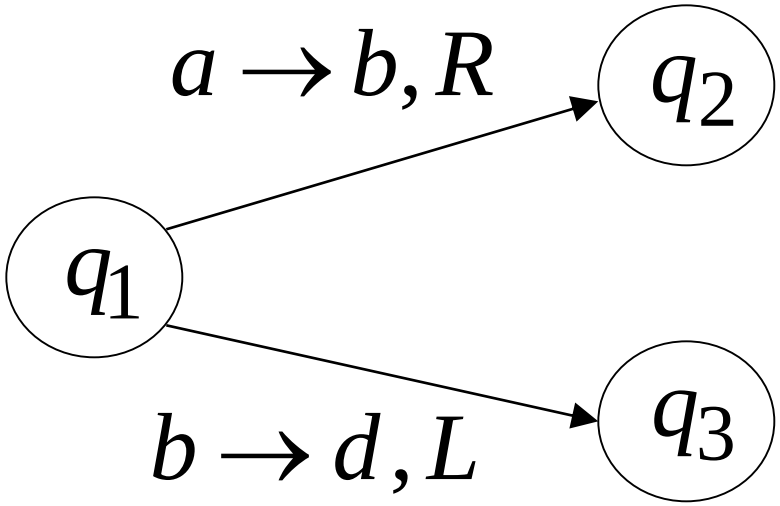



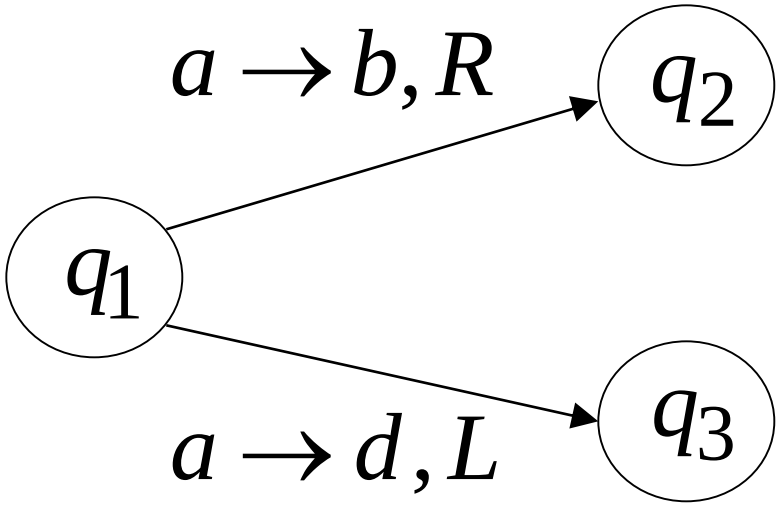
# Determinism

Turing Machines are deterministic

Allowed



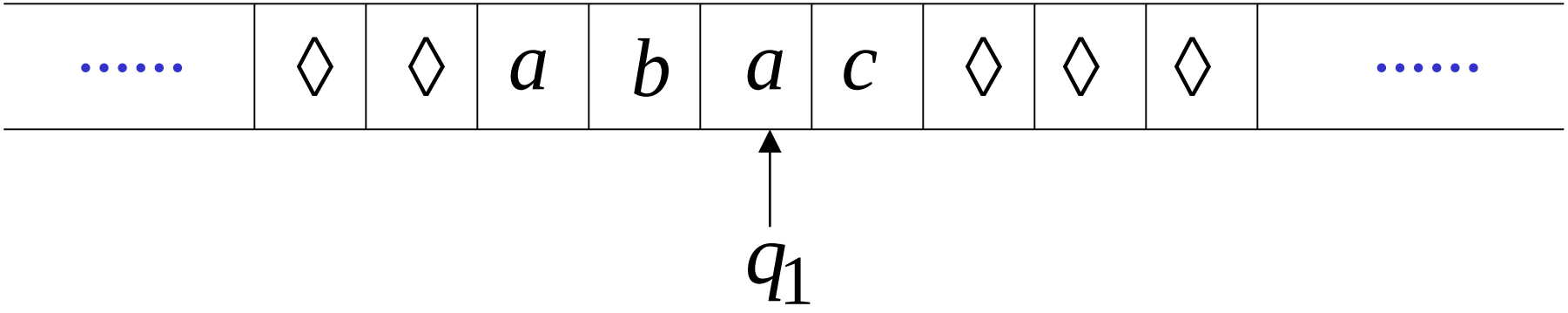

**Not** Allowed

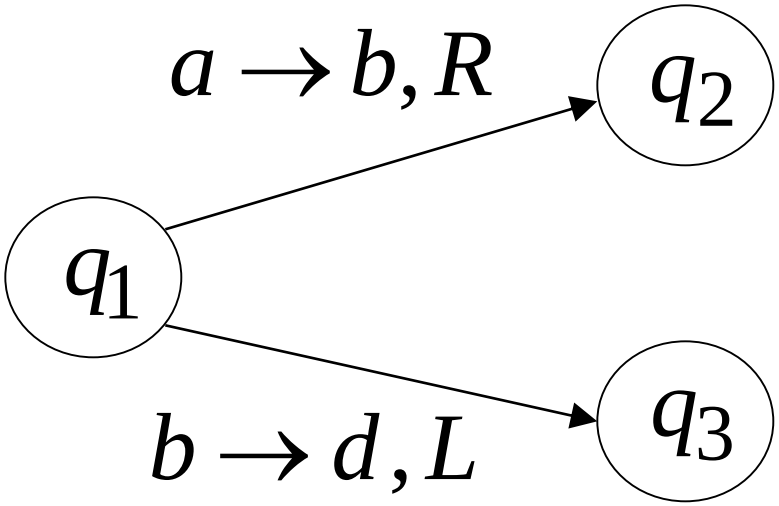



No lambda transitions allowed

# Partial Transition Function

Example:



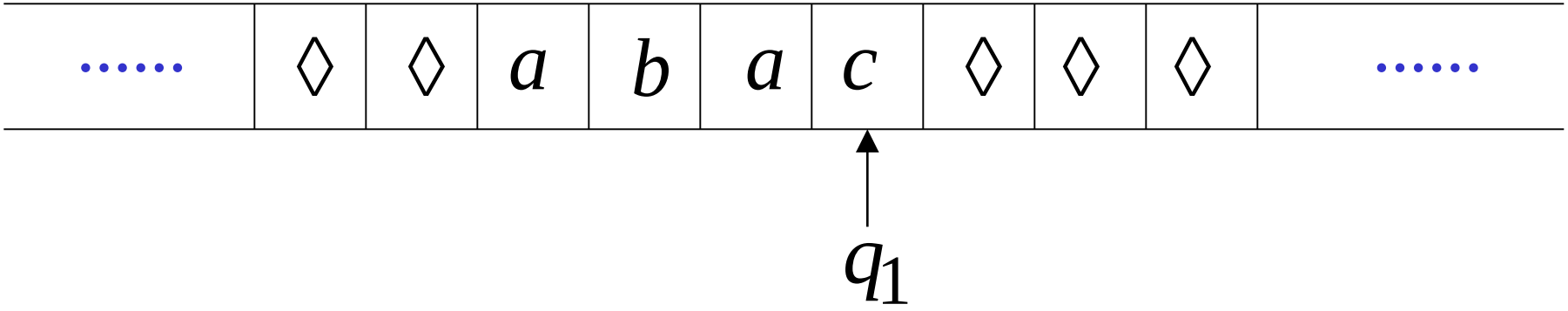

Allowed:

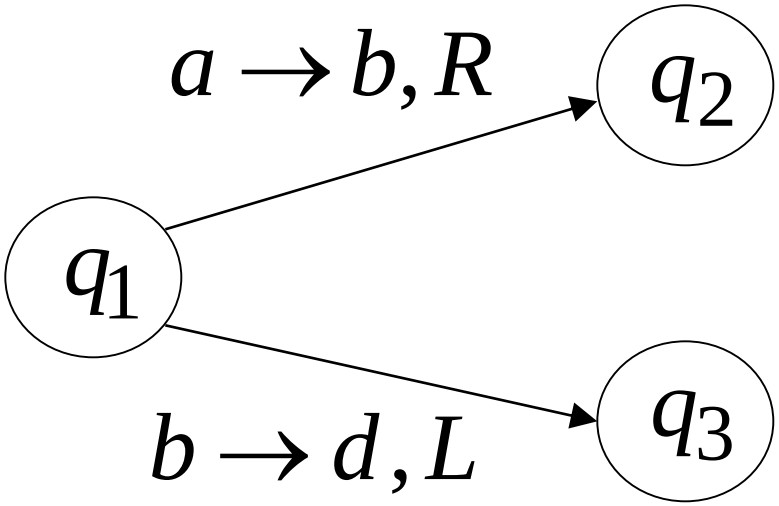
No transition
for input symbol $c$

# Halting

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

# Example:

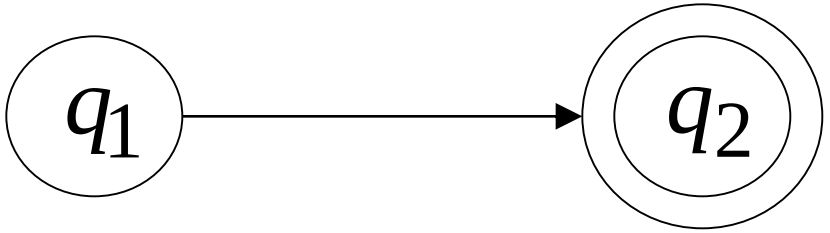


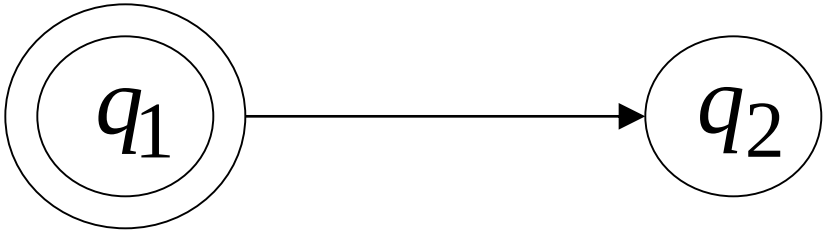

No possible transition

**HALT!!!**

# Final States



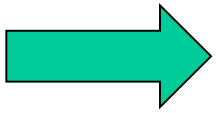

Allowed

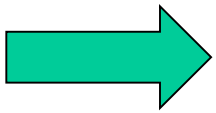



**Not** Allowed

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

# Acceptance

Accept Input  If machine halts in a final state

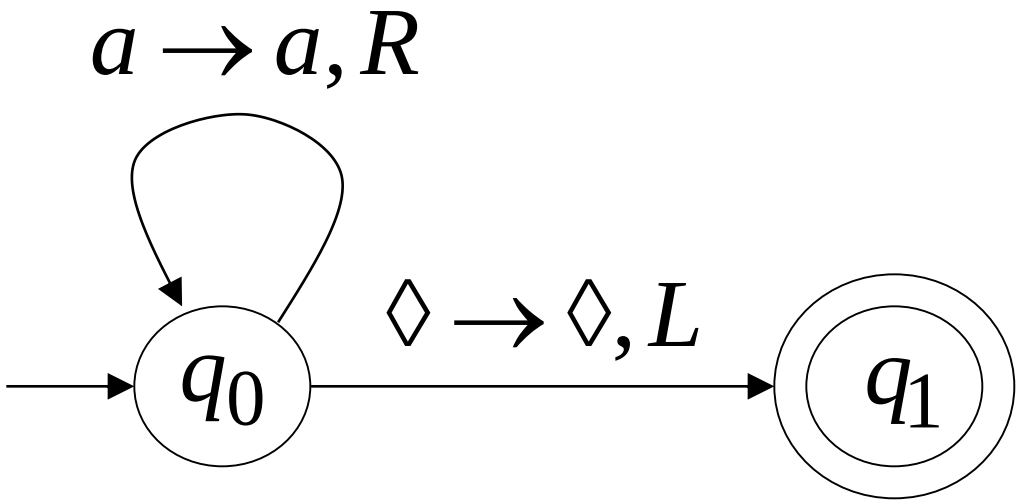
Reject Input  If machine halts in a non-final state

or

If machine enters an *infinite loop*

# Turing Machine Example

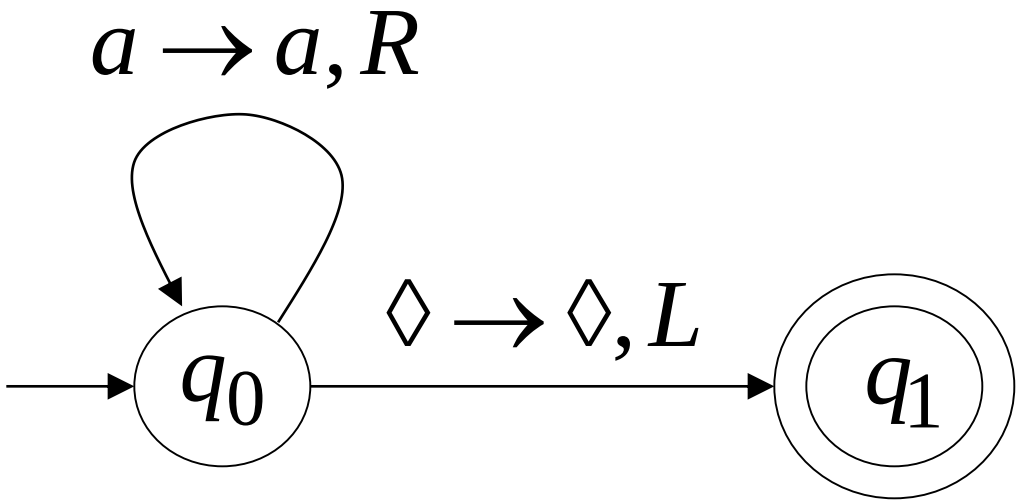
Language?



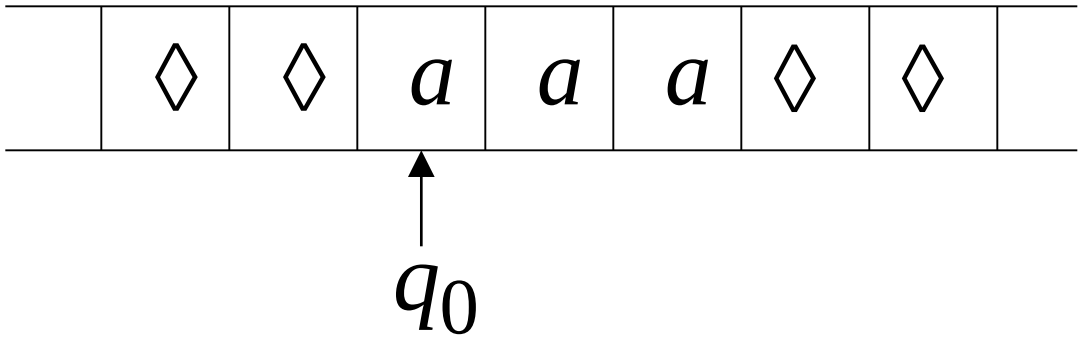
# Turing Machine Example

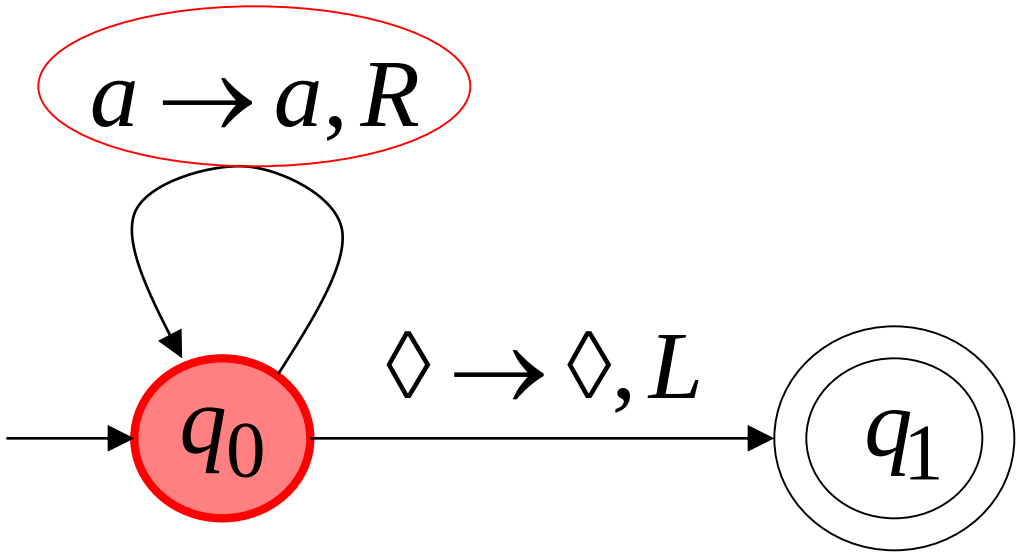
A Turing machine that accepts the language:

$$a^*$$

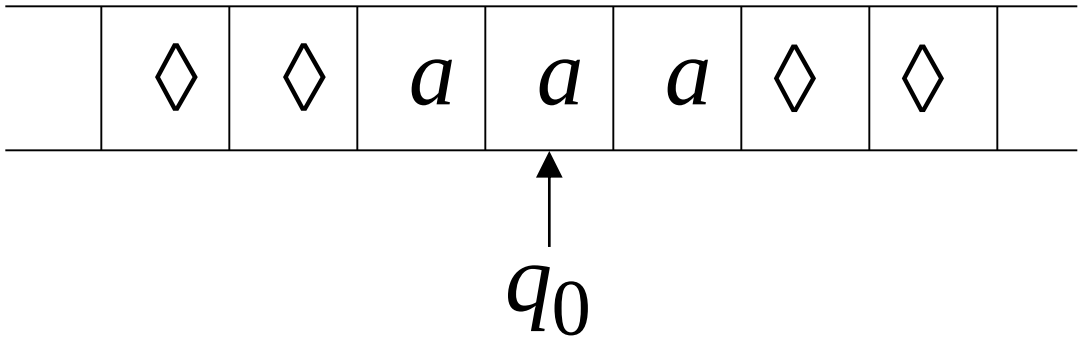


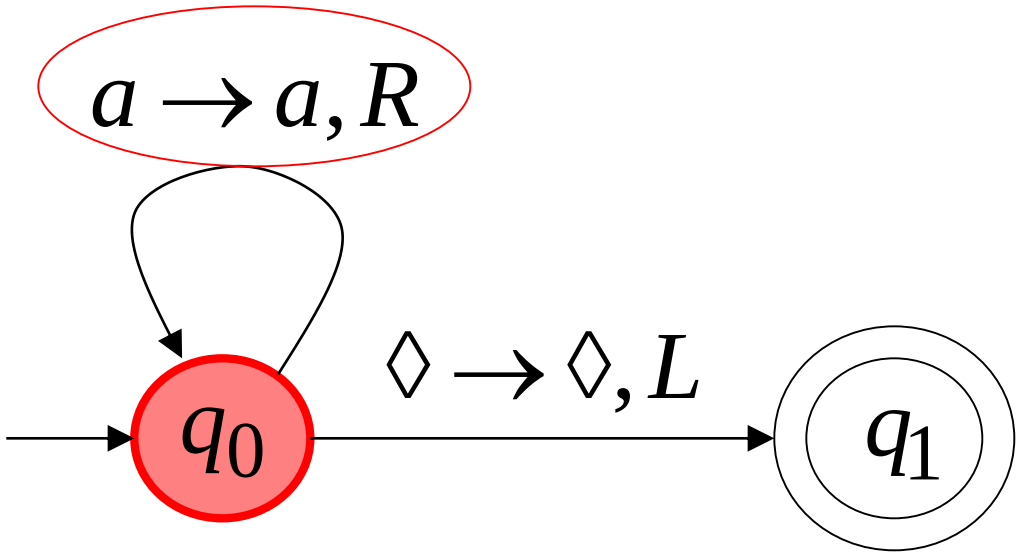
Time 0



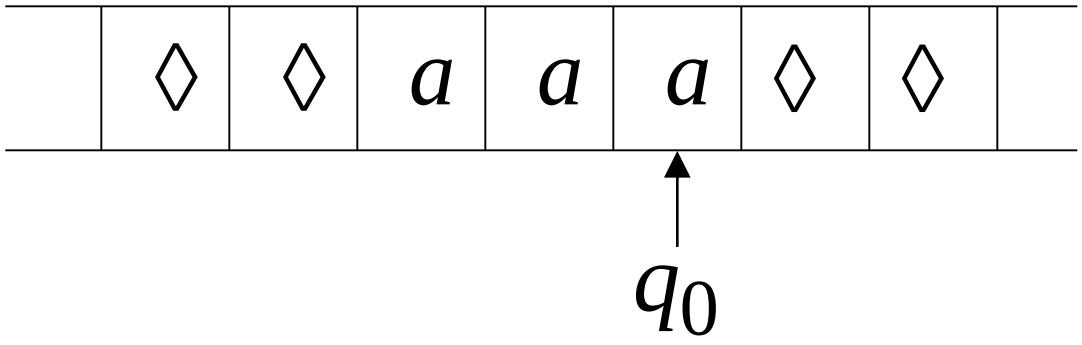



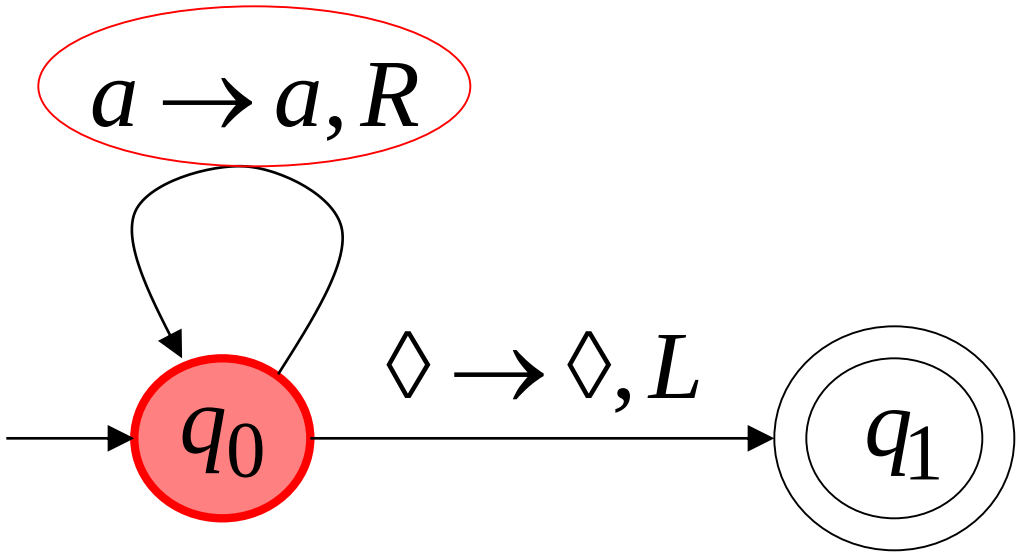
Time 1



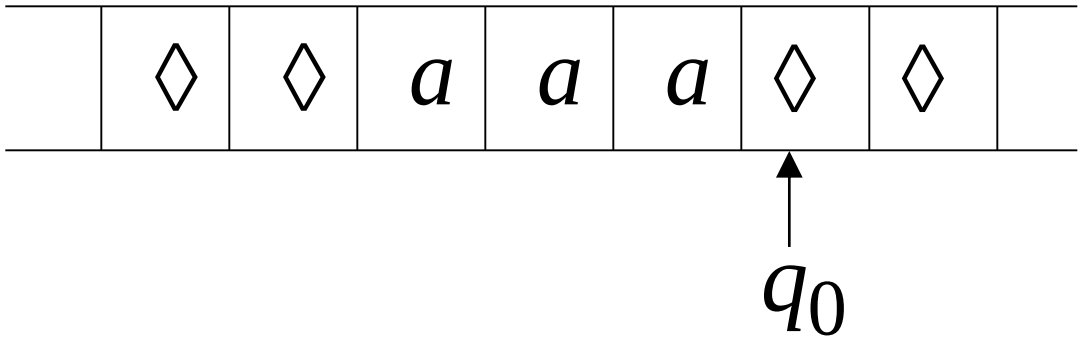



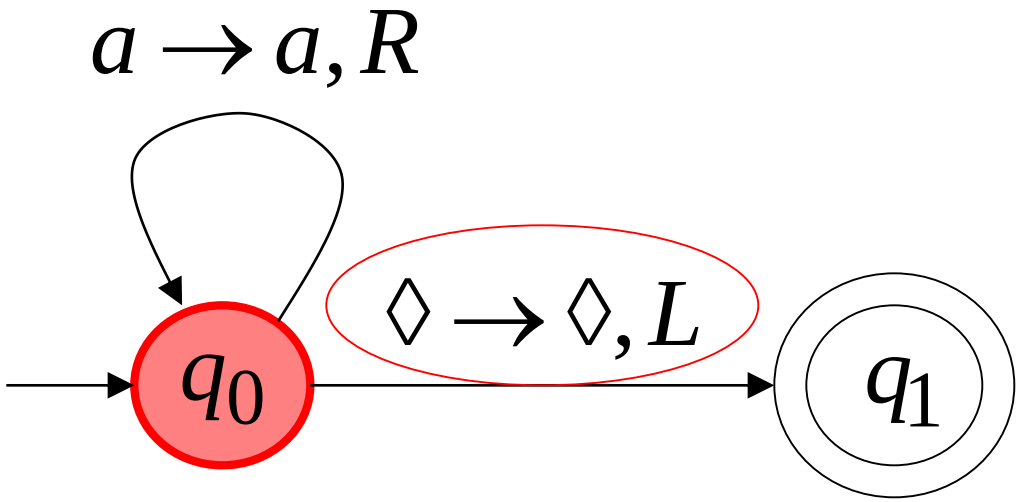
Time 2



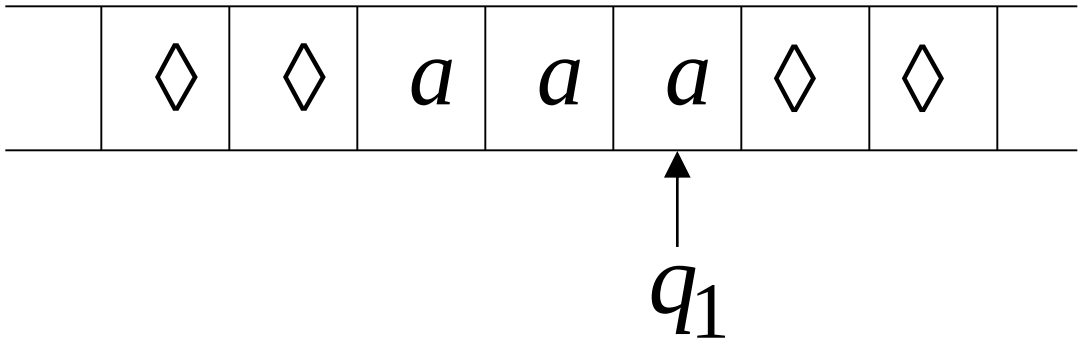



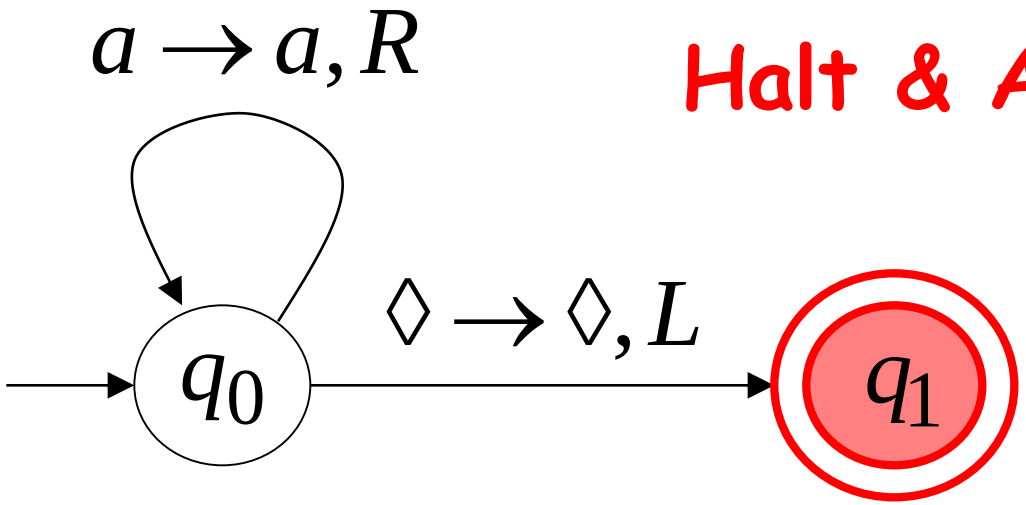
Time 3



Time 4

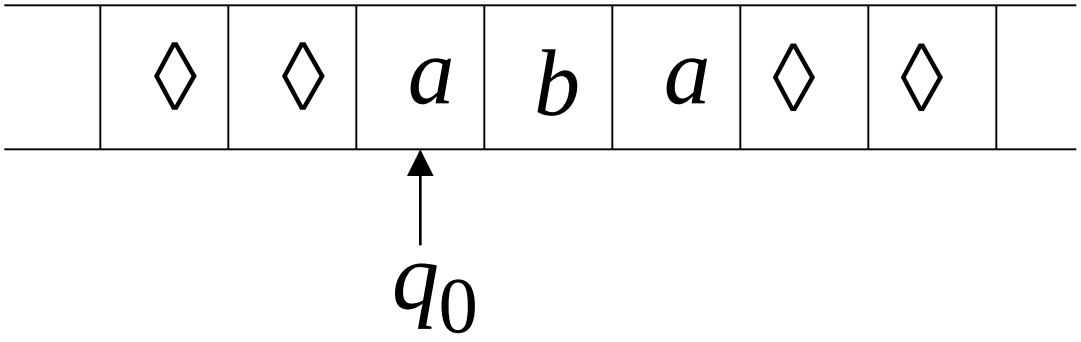


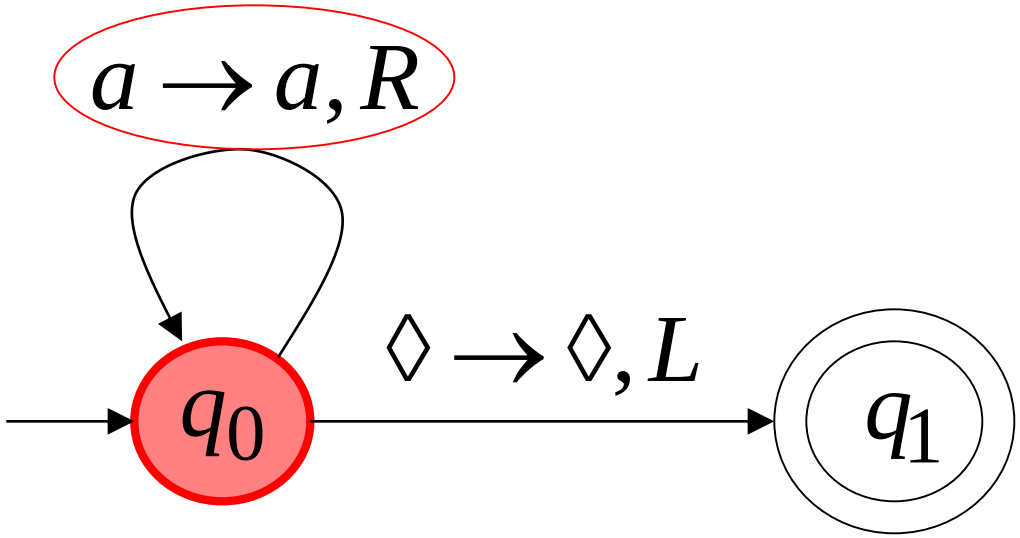




Halt & Accept

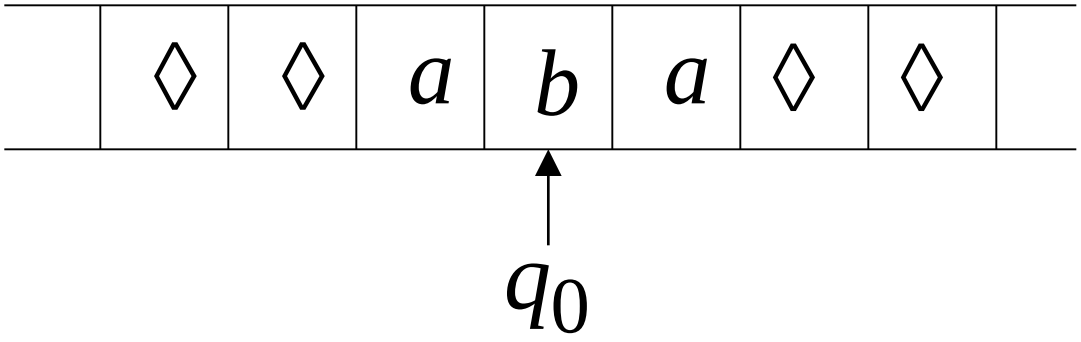
# Rejection Example

Time 0



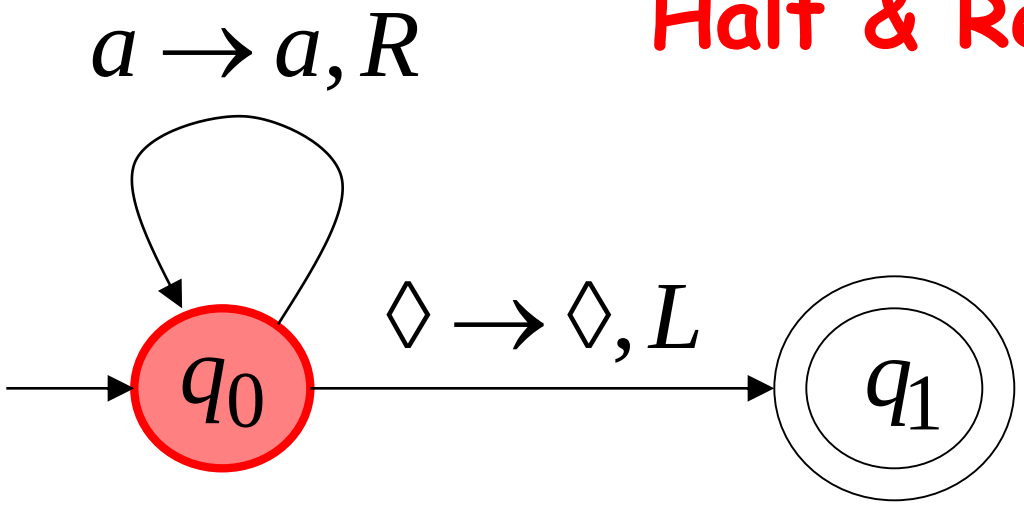



Time 1

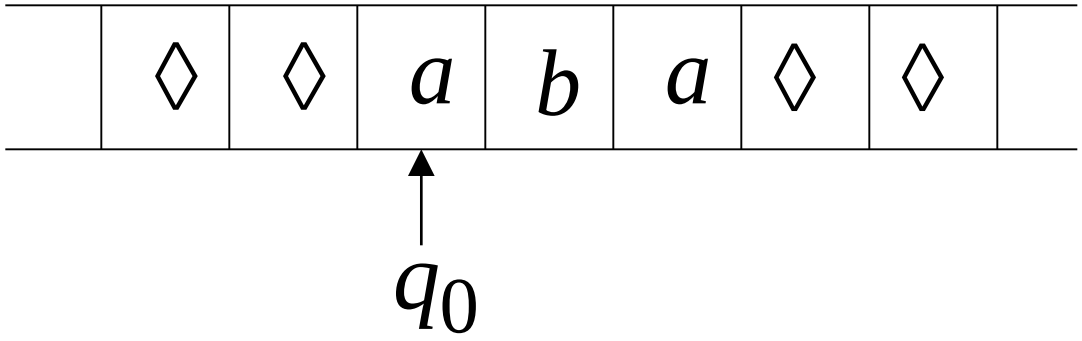



No possible Transition

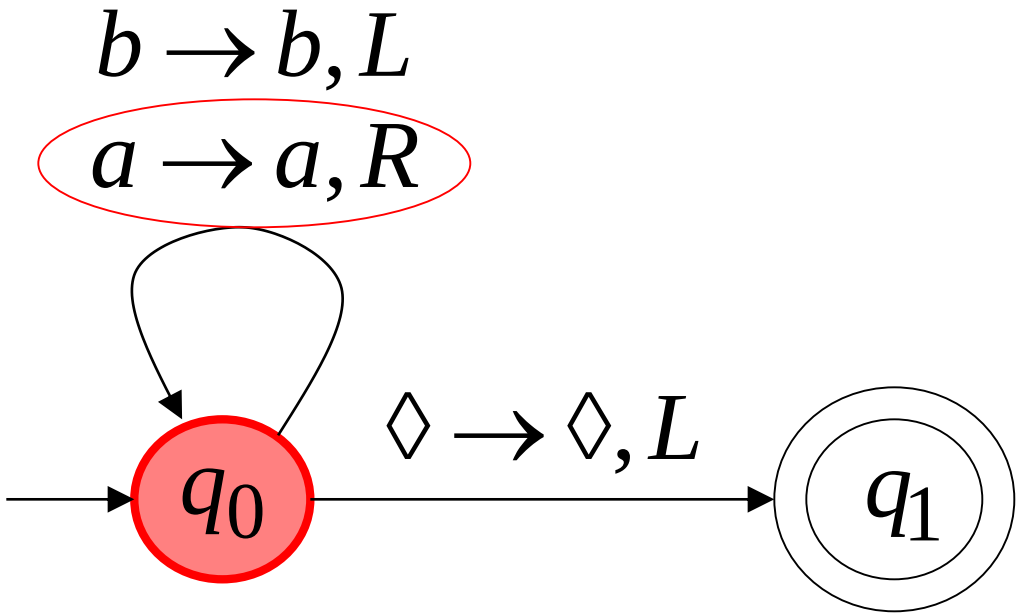
**Halt & Reject**



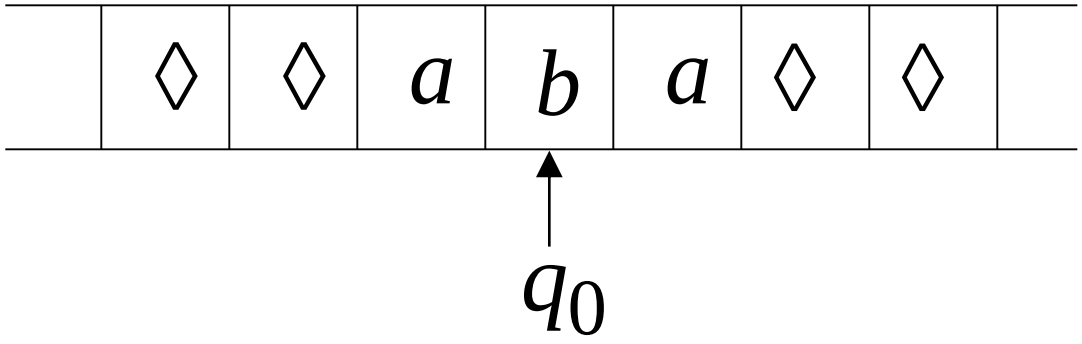
Time 0

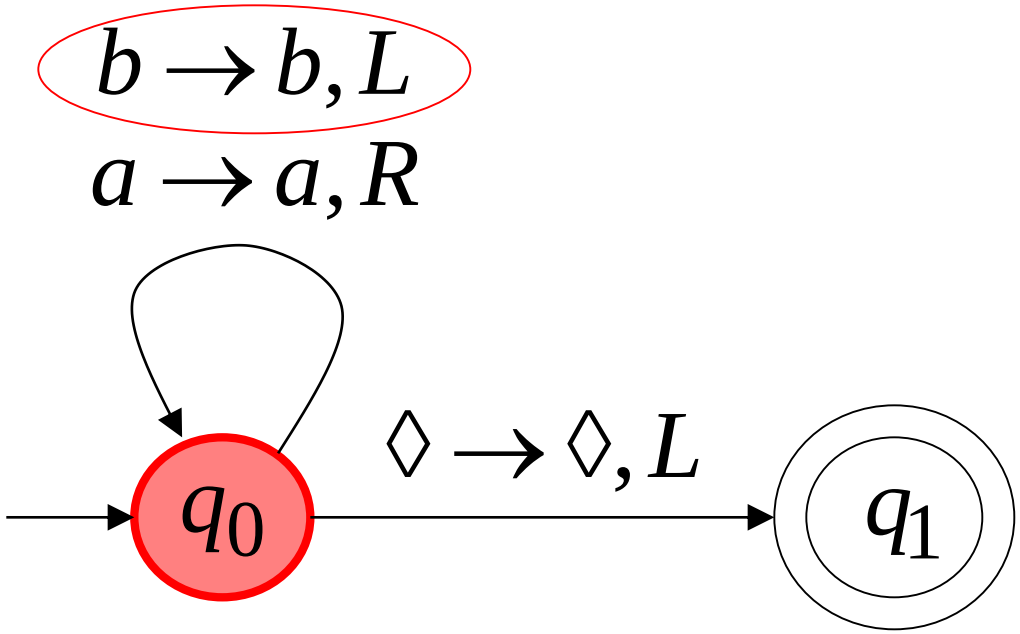



Another Example

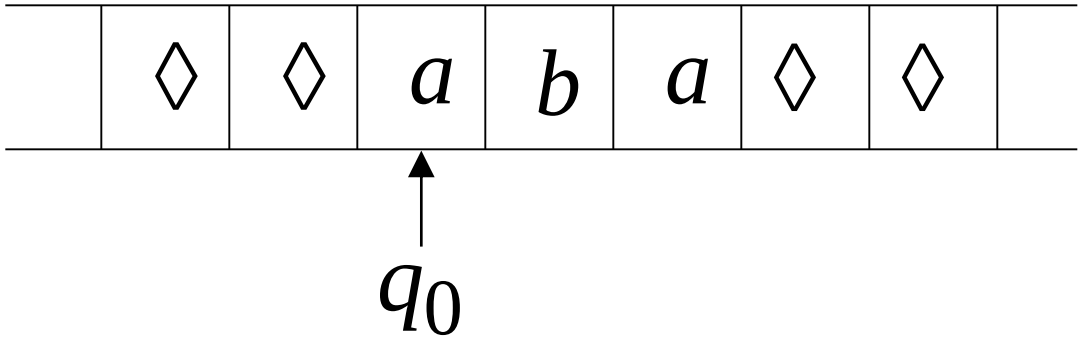


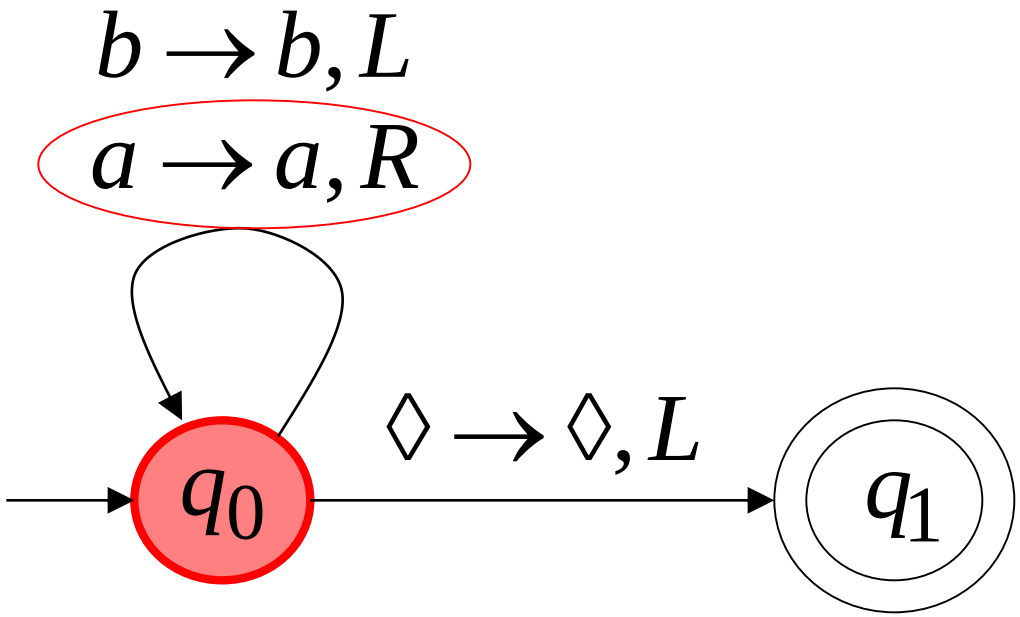
Time 1



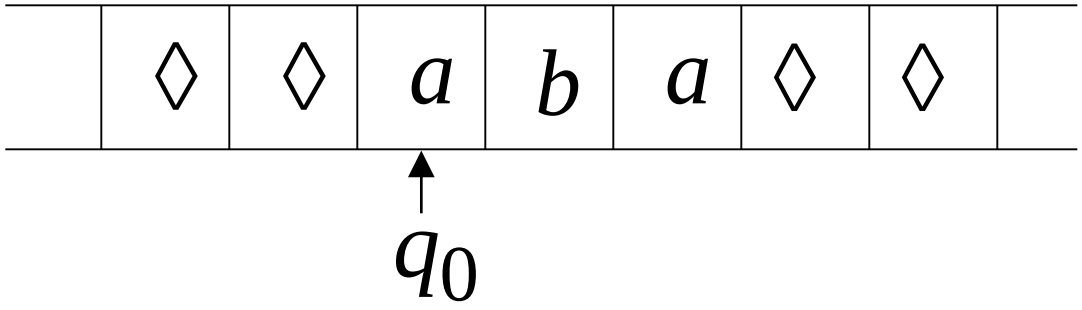



Time 2

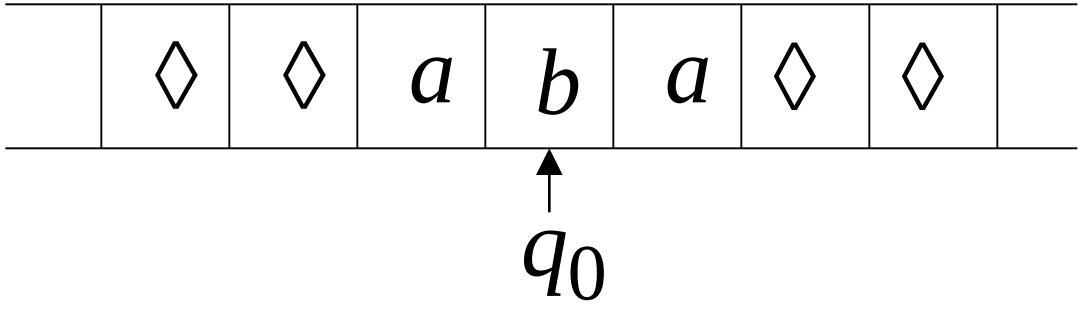


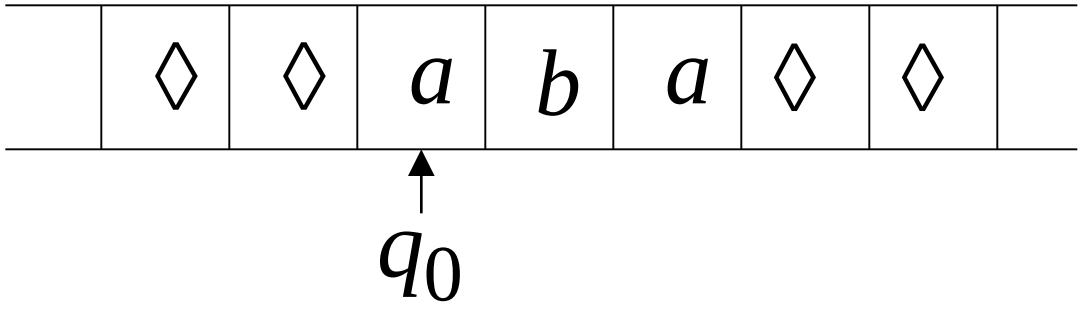
Time 2



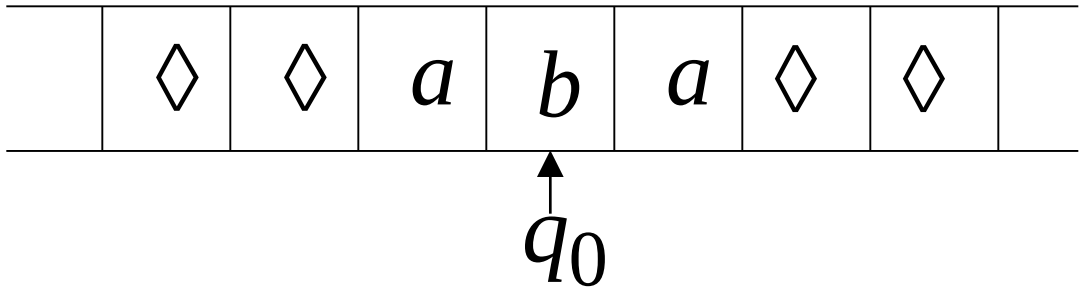

Time 3




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

For $\Sigma = \{a,b\}$, design a Turing machine that accepts

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

- Starting at the leftmost *a*, we check it off by replacing it with some symbol, say *x*.
- We then let the read-write head travel right to find the leftmost *b*, which in turn is checked off by replacing it with another symbol, say *y*.
- After that, we go left again to the leftmost *a*, replace it with an *x*, then move to the leftmost *b*and replace it with *y*, and so on.
- Traveling back and forth this way, we match each *a* with a corresponding *b*. If after some time no a's or b's remain, then the string must be in *L*.

$Q = \{q_0, q_1, q_2, q_3, q_4\}, F = \{q_4\}, \Sigma = \{a, b\}, \Gamma = \{a, b, x, y, \}$.

The transitions can be broken into several parts.

Phase 1:

$$\delta(q_0, a) = (q_1, x, R),$$

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

$$\delta(q_1, y) = (q_1, y, R),$$

$$\delta(q_1, b) = (q_2, y, L),$$

- The set replaces the leftmost *a* with an *x*, then causes the read-write head to travel right to the first *b*, replacing it with a *y*.
- When the *y* is written, the machine enters a state $q_2$, indicating that an *a* has been successfully paired with a *b*.

The next set of transitions reverses the direction until an $x$ is encountered, repositions the read-write head over the leftmost $a$, and returns control to the initial state.

$$\delta(q_2,y)=(q_2,y,L),$$

$$\delta(q_2,a)=(q_2,a,L),$$

$$\delta(q_2,x)=(q_0,x,R),$$

We are now back in the initial state $q_0$, ready to deal with the next $a$ and $b$.

Final check.... is made to see if all *a*'s and *b*'s have been replaced (to detect input where an *a* follows a *b*)

$$\delta(q_0,y)=(q_3,y,R),$$

$$\delta(q_3,y)=(q_3,y,R),$$

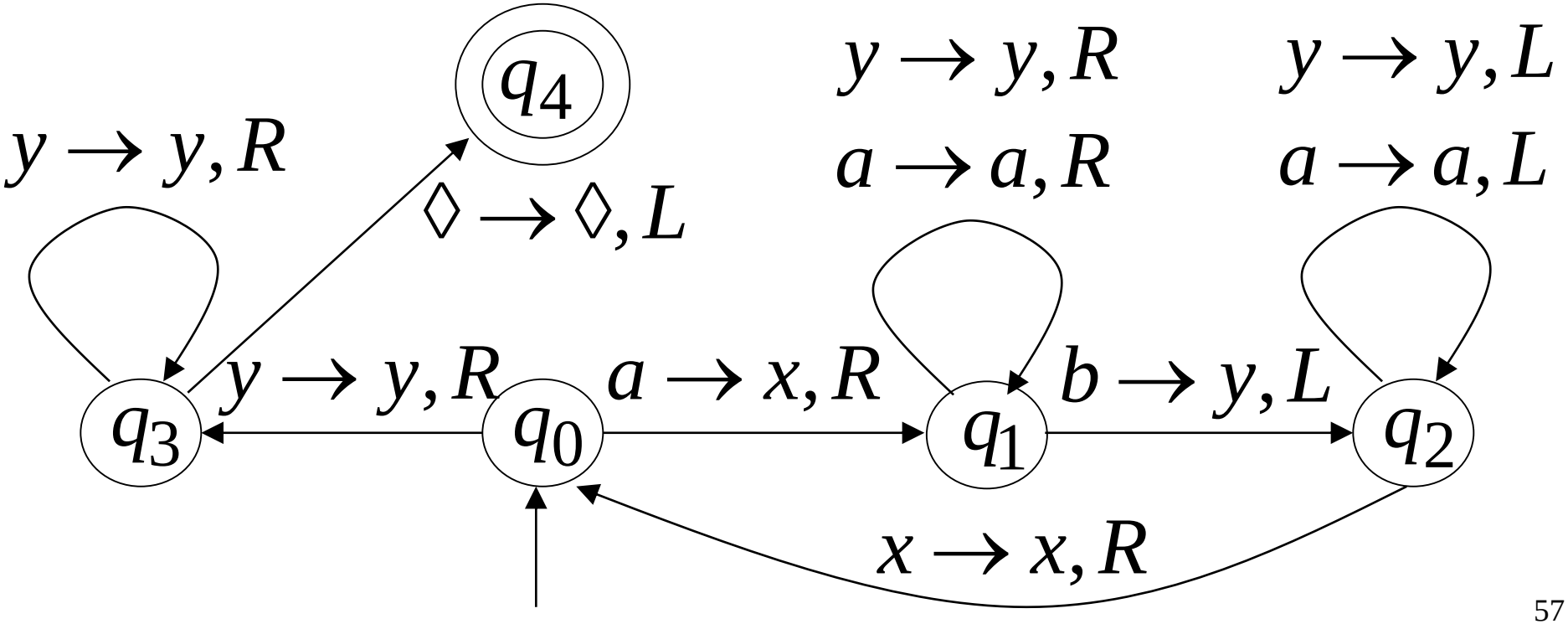
$$\delta(q_3,\square)=(q_4,\square,R),$$

The particular input *aabb* gives the following successive instantaneous descriptions:

$$
\begin{aligned}
q_0aabb &\vdash xq_1abb \vdash xaq_1bb \vdash xq_2ayb \\
&\vdash q_2xayb \vdash xq_0ayb \vdash xxq_1yb \\
&\vdash xxyq_1b \vdash xxq_2yy \vdash xq_2xyy \\
&\vdash xxq_0yy \vdash xxyq_3y \vdash xxyyq_3\square \\
&\vdash xxyy\square q_4\square.
\end{aligned}
$$

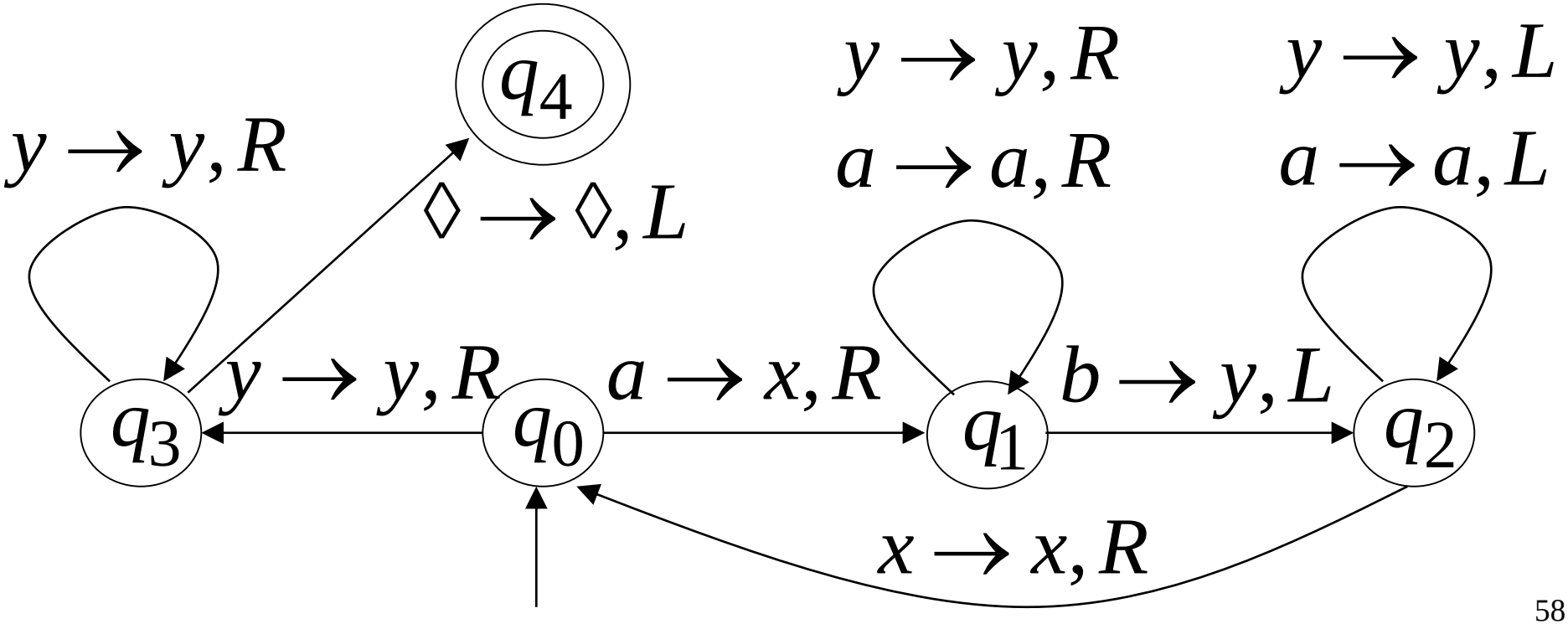
# Another Turing Machine Example

Language?

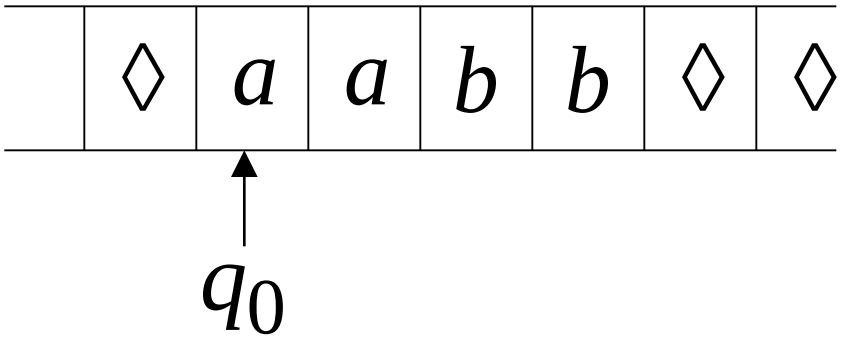


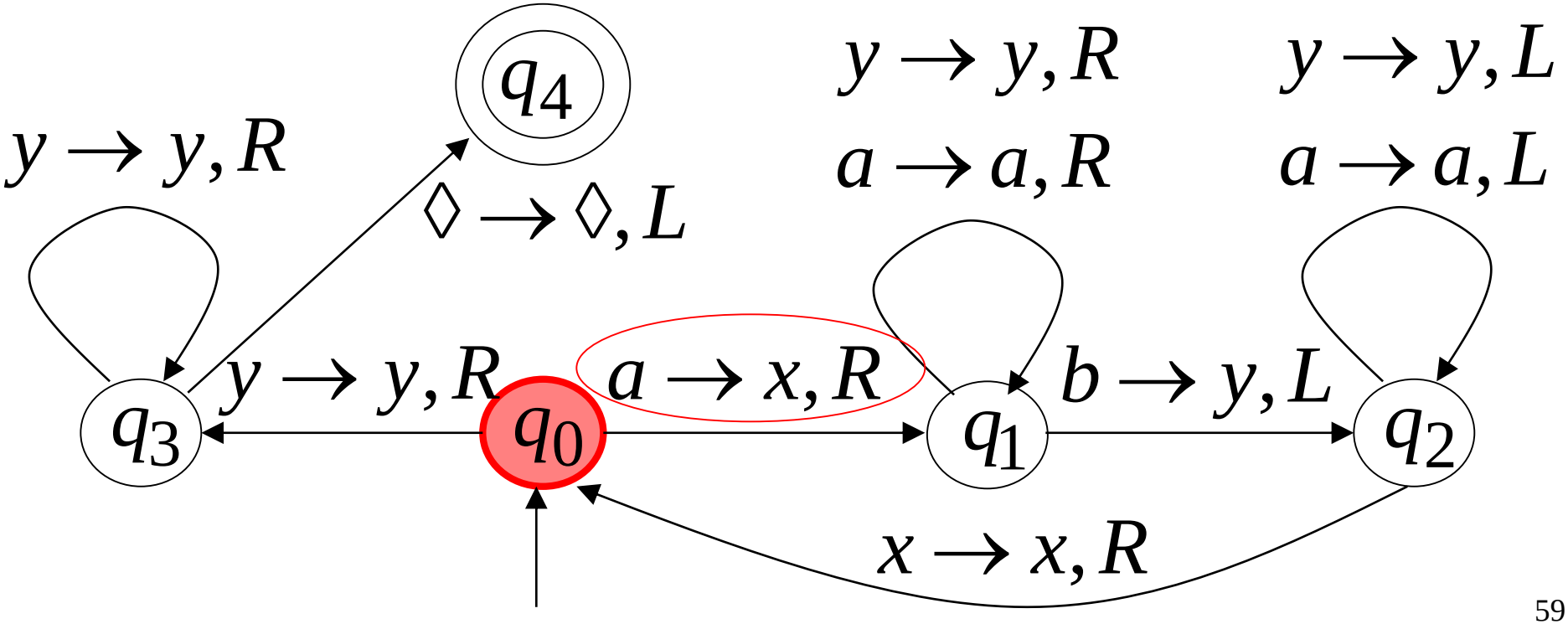
# Another Turing Machine Example

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

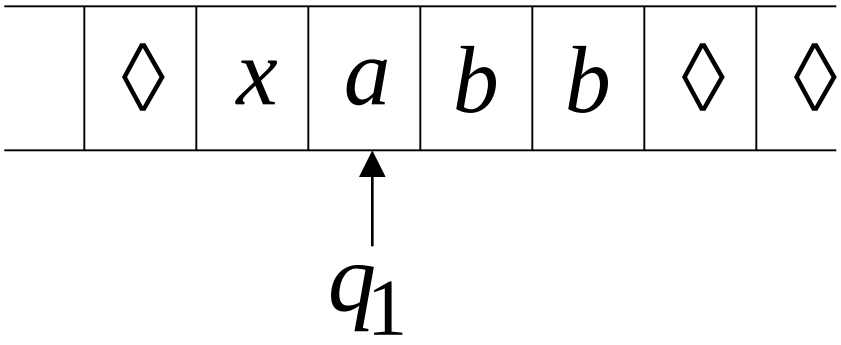


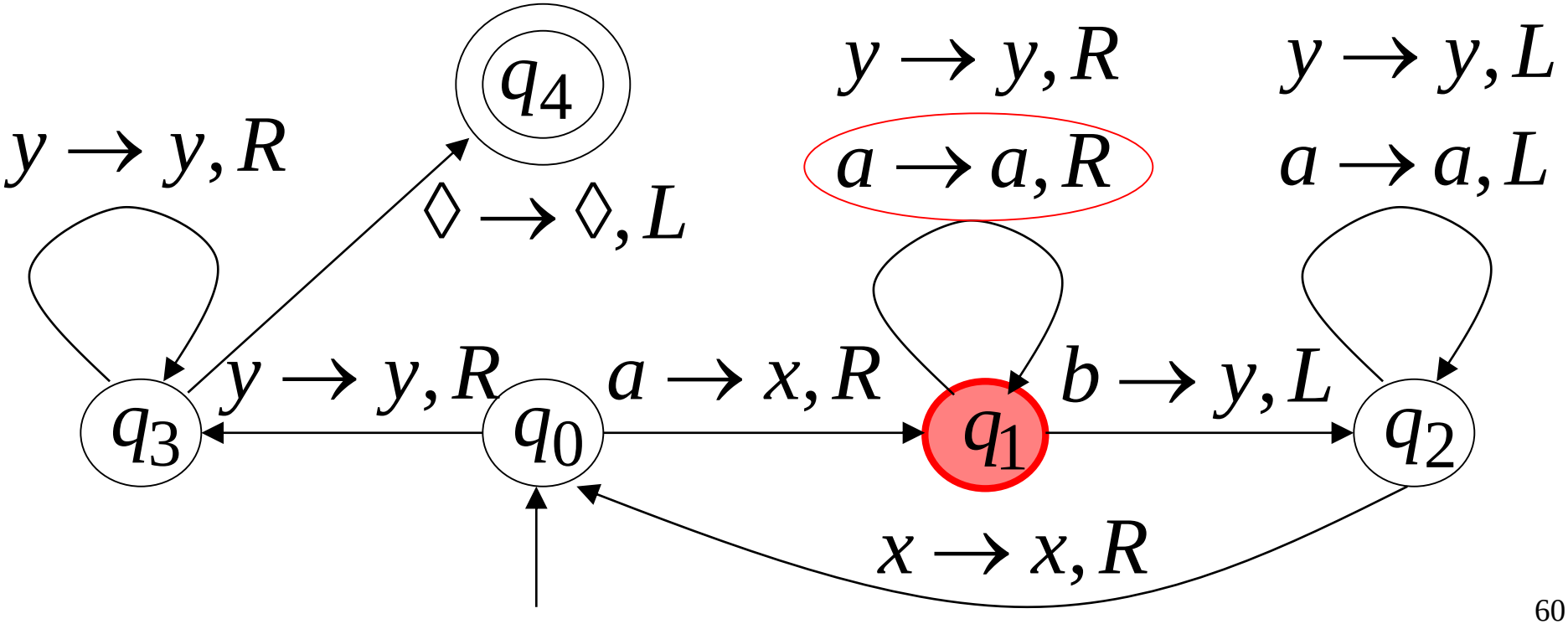
Time 0


◊ a a b b ◊ ◊
$q_0$

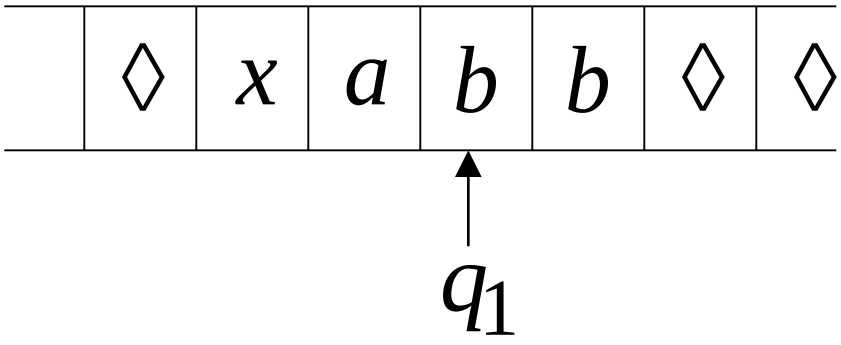

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

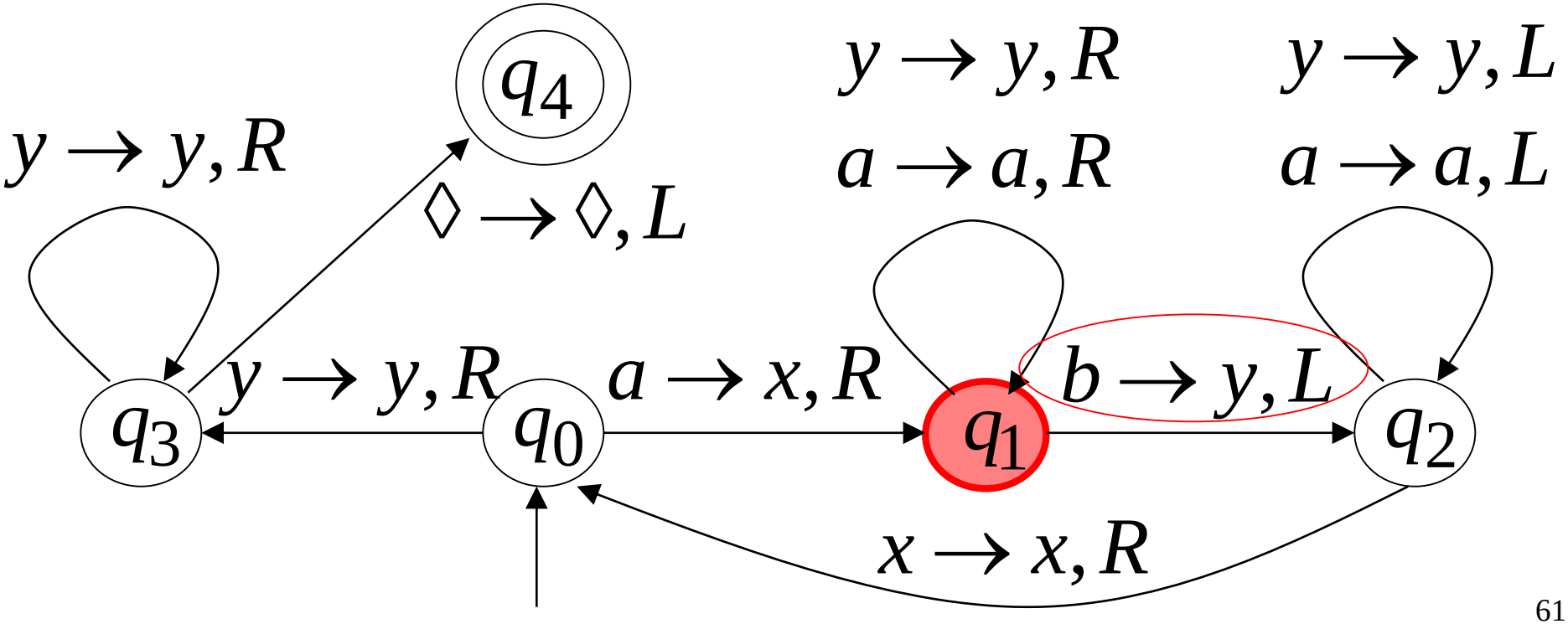
Time 1


◊ x a b b ◊ ◊
$q_1$

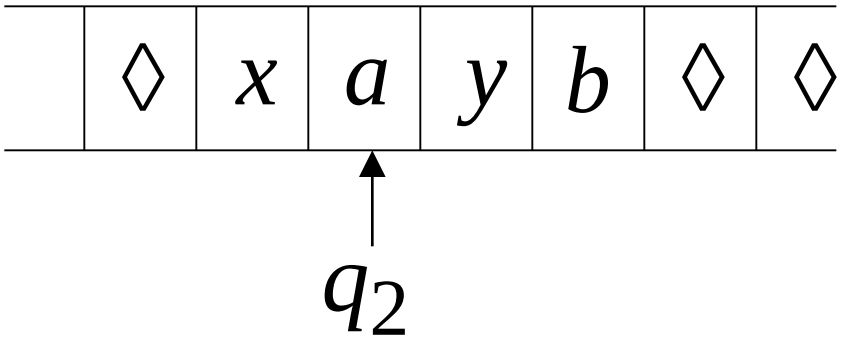

$q_4$
$y \to y, R$
$y \to y, R$
$y \to y, L$
$a \to a, R$
$a \to a, L$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

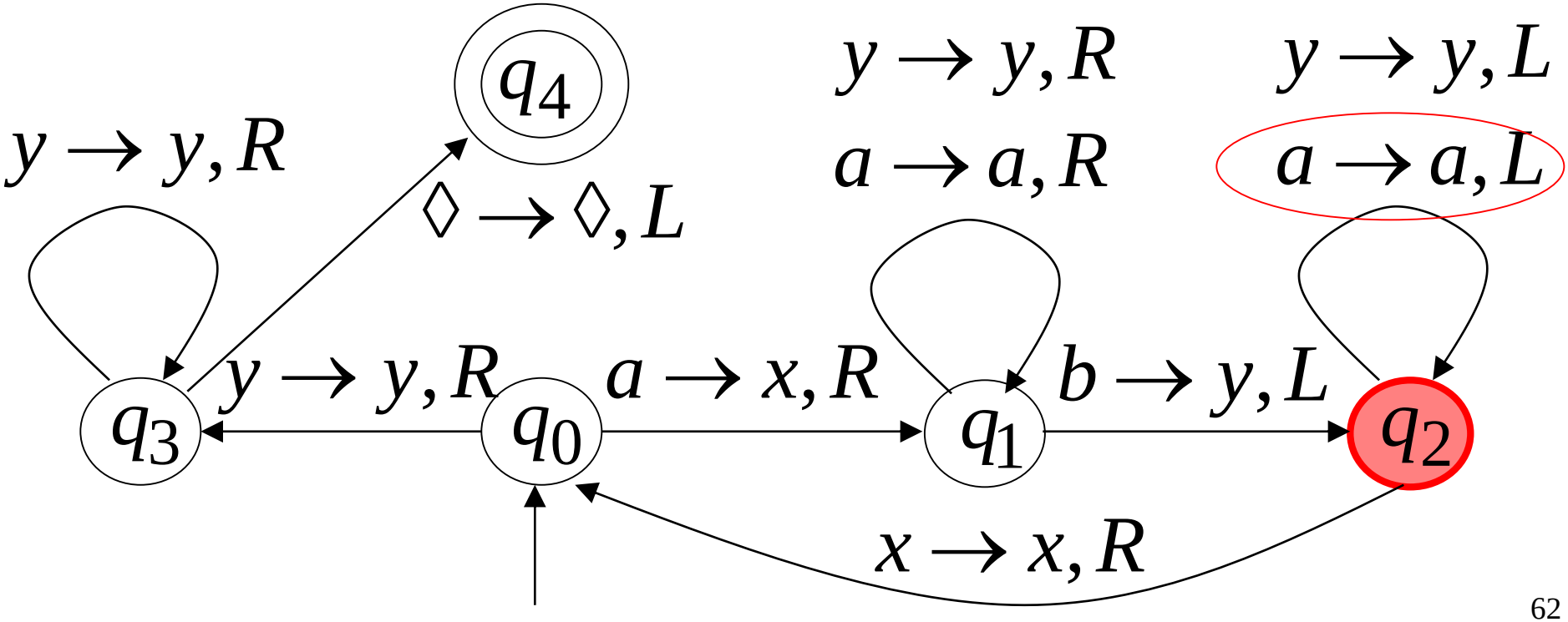
Time 2


◊ x a b b ◊ ◊
$q_1$

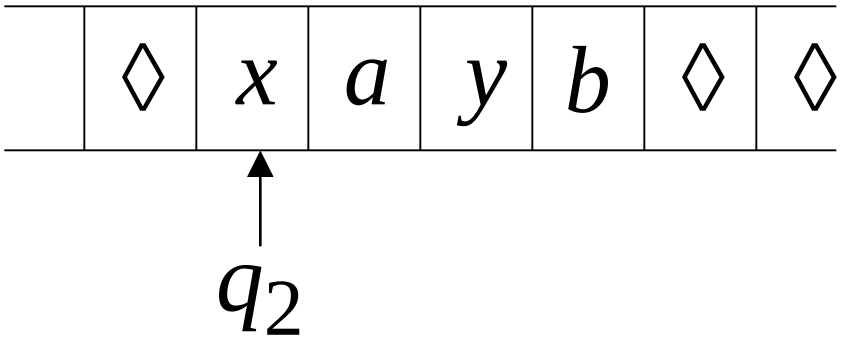

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

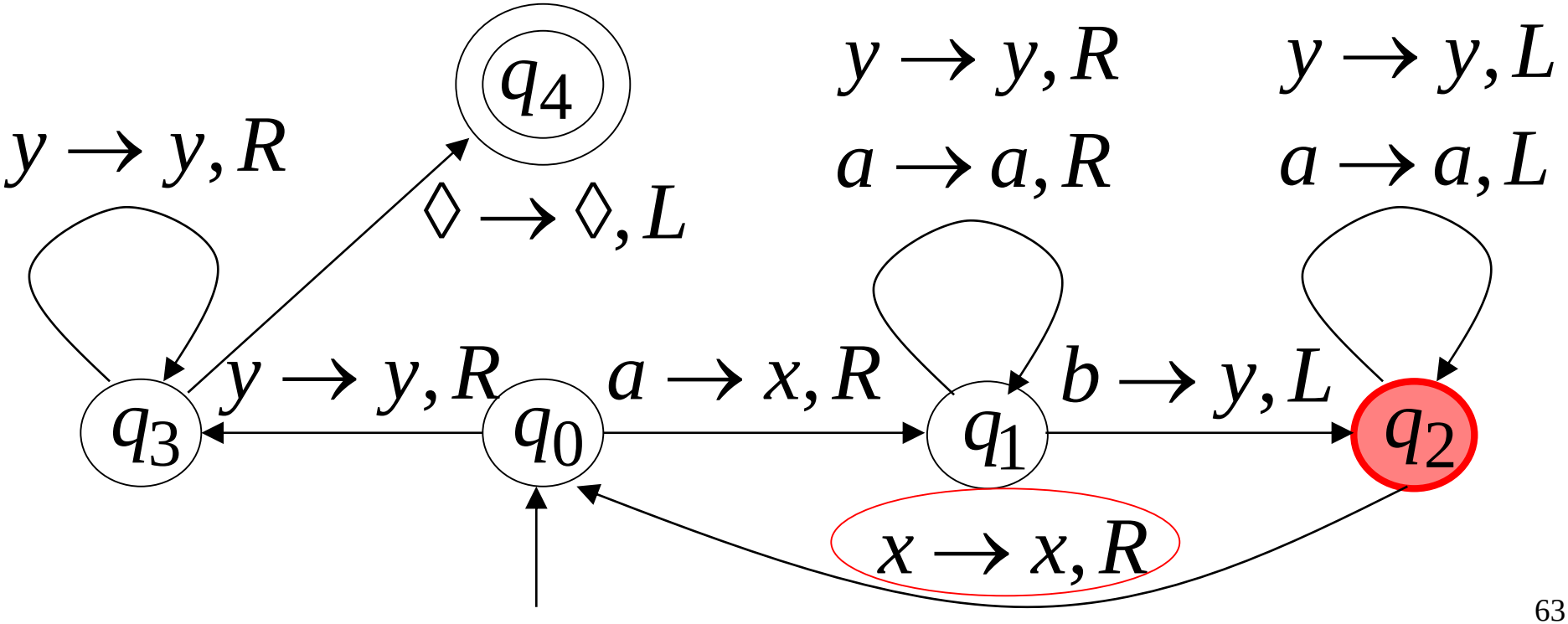
Time 3


◊ x a y b ◊ ◊
$q_2$

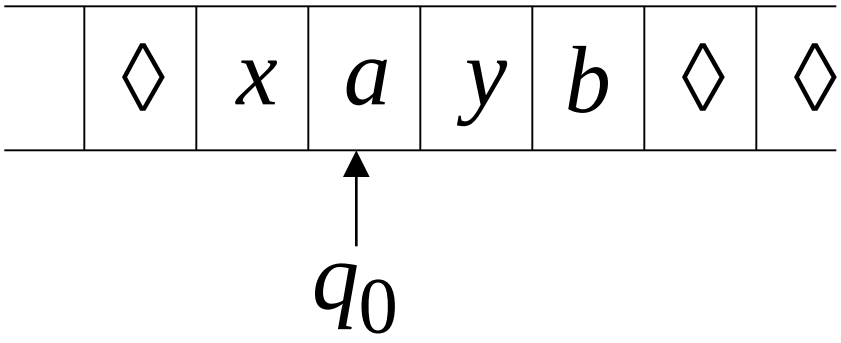

$q_4$
$y \to y, R$
$\lozenge \to \lozenge, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$q_3$
$y \to y, R$
$q_0$
$a \to x, R$
$q_1$
$b \to y, L$
$q_2$
$x \to x, R$

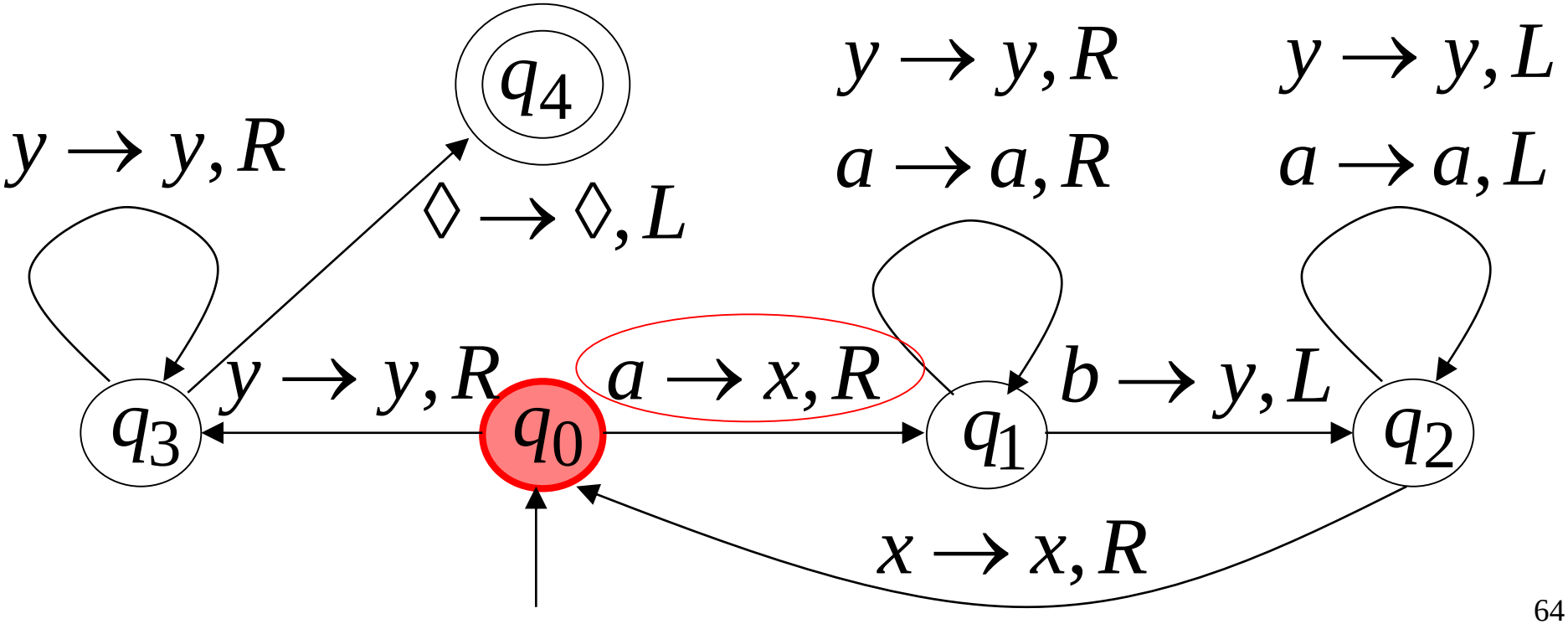
Time 4


◊ x a y b ◊ ◊
$q_2$

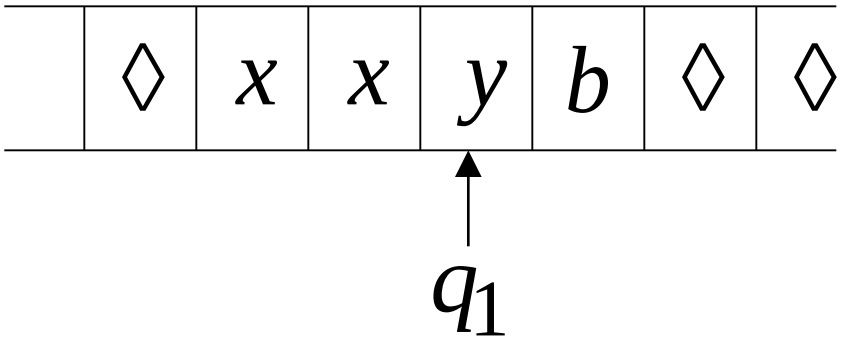

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

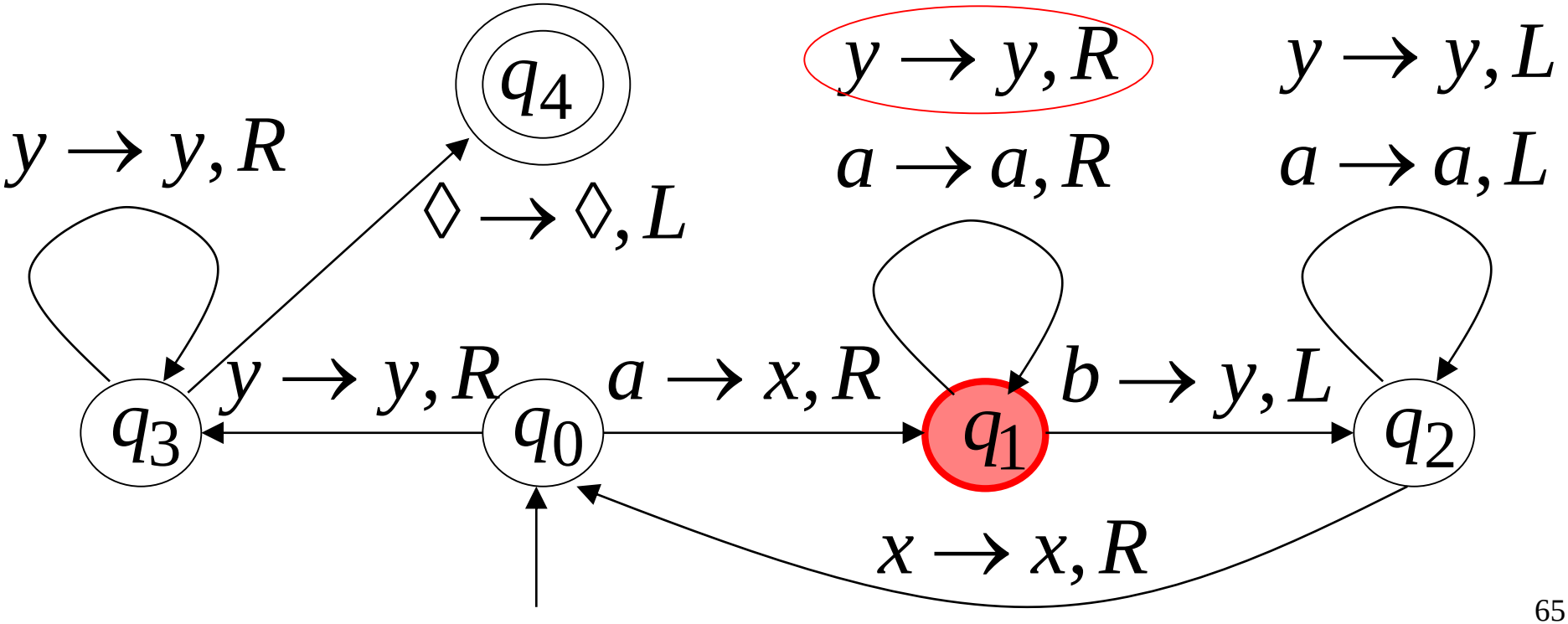
Time 5


◊ x a y b ◊ ◊
$q_0$

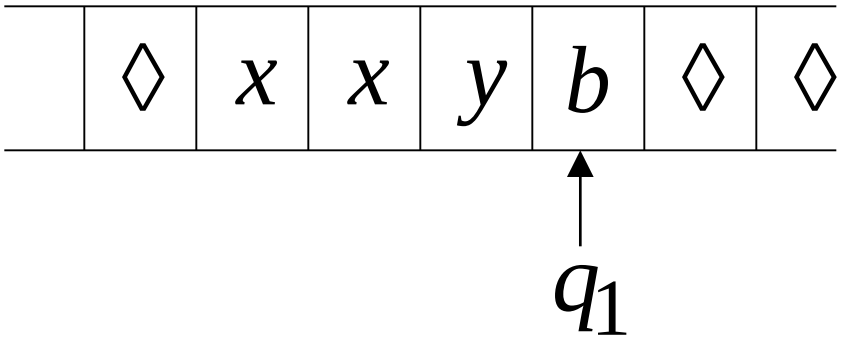

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

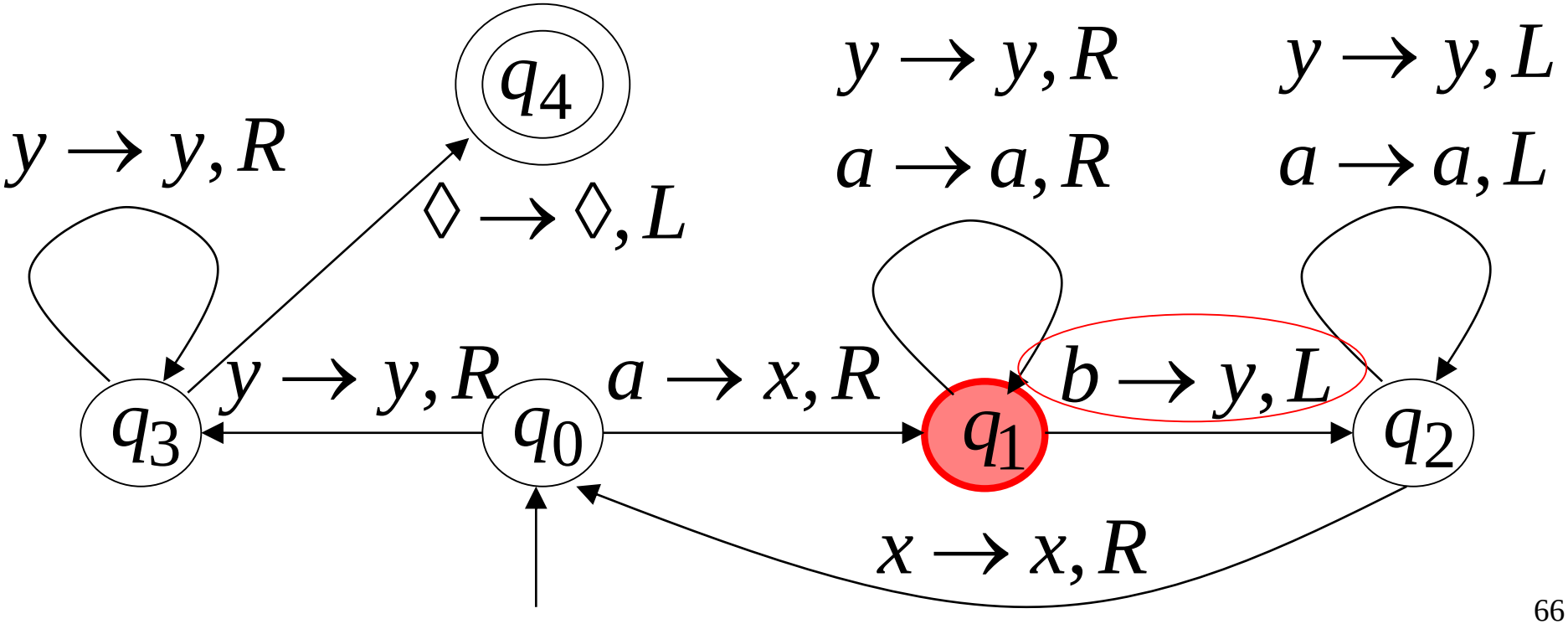
Time 6


◊ x x y b ◊ ◊
$q_1$

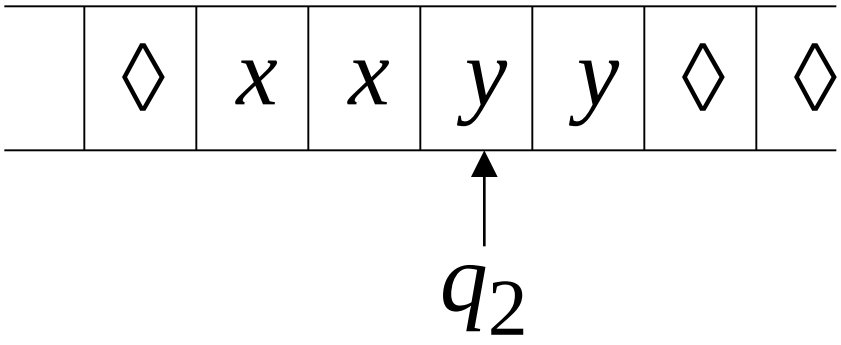

$q_4$
$y \to y, R$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$\Diamond \to \Diamond, L$
$q_3$
$y \to y, R$
$q_0$
$a \to x, R$
$q_1$
$b \to y, L$
$q_2$
$x \to x, R$

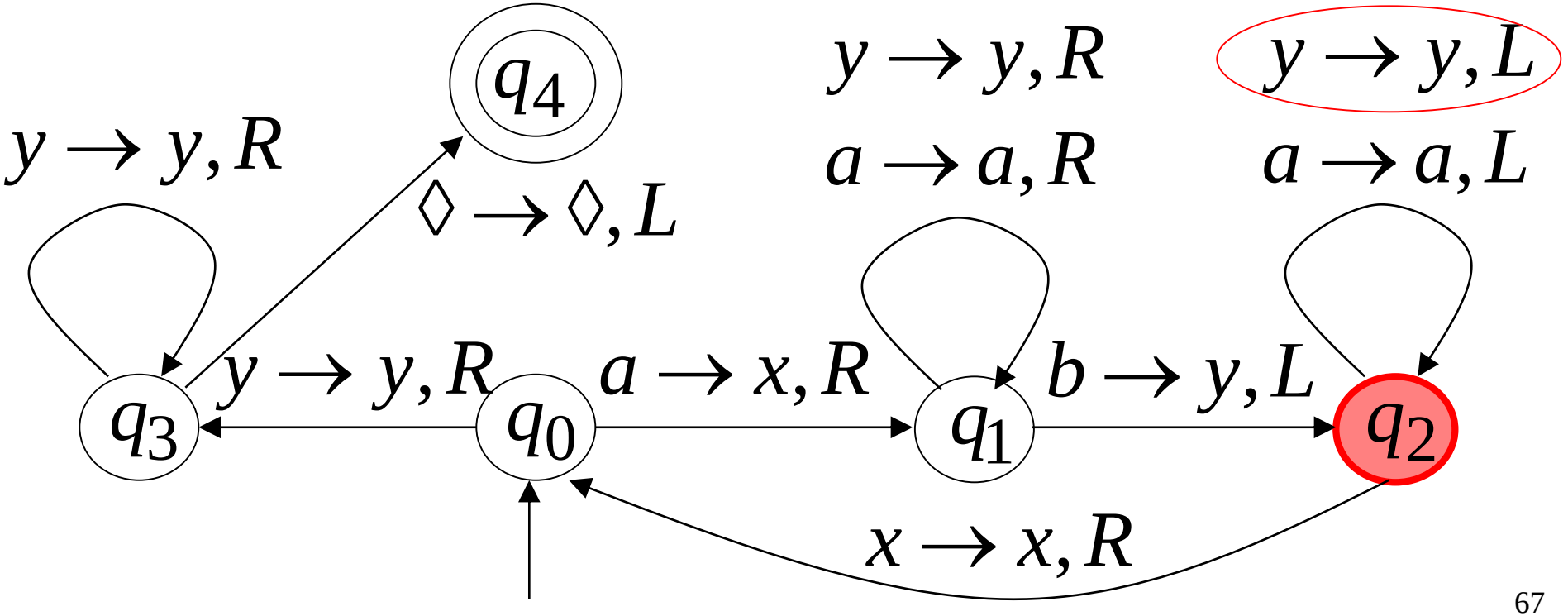
Time 7


◊ x x y b ◊ ◊
$q_1$

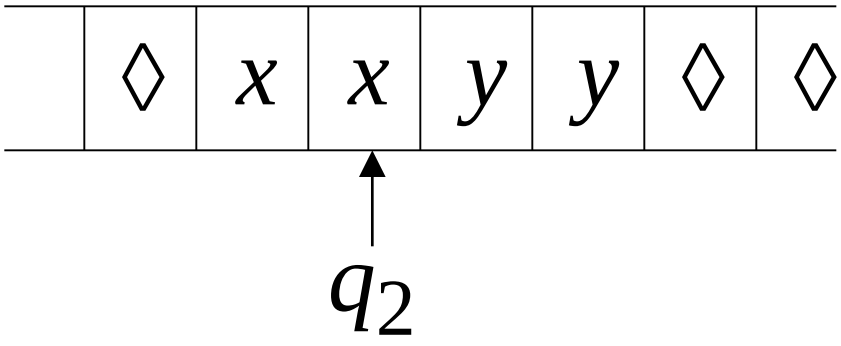

$q_4$
$y \to y, R$
$\lozenge \to \lozenge, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$q_3$
$y \to y, R$
$q_0$
$a \to x, R$
$q_1$
$b \to y, L$
$q_2$
$x \to x, R$

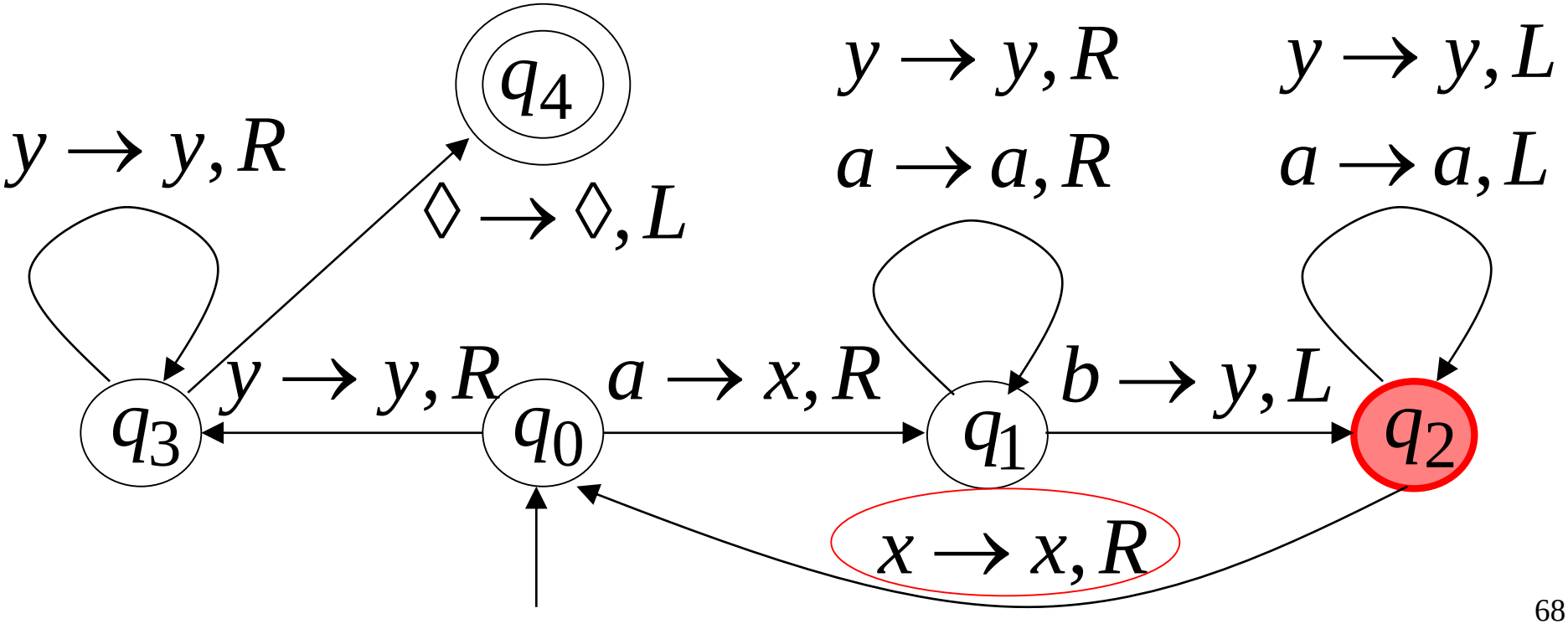
Time 8


◊ x x y y ◊ ◊
$q_2$

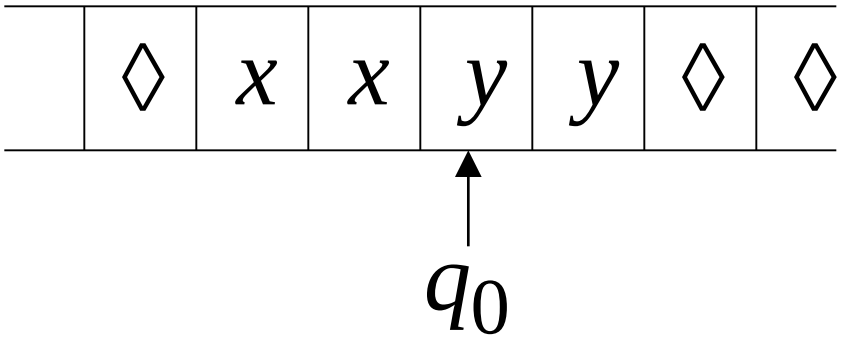

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$q_3$
$y \to y, R$
$q_0$
$a \to x, R$
$q_1$
$b \to y, L$
$q_2$
$x \to x, R$

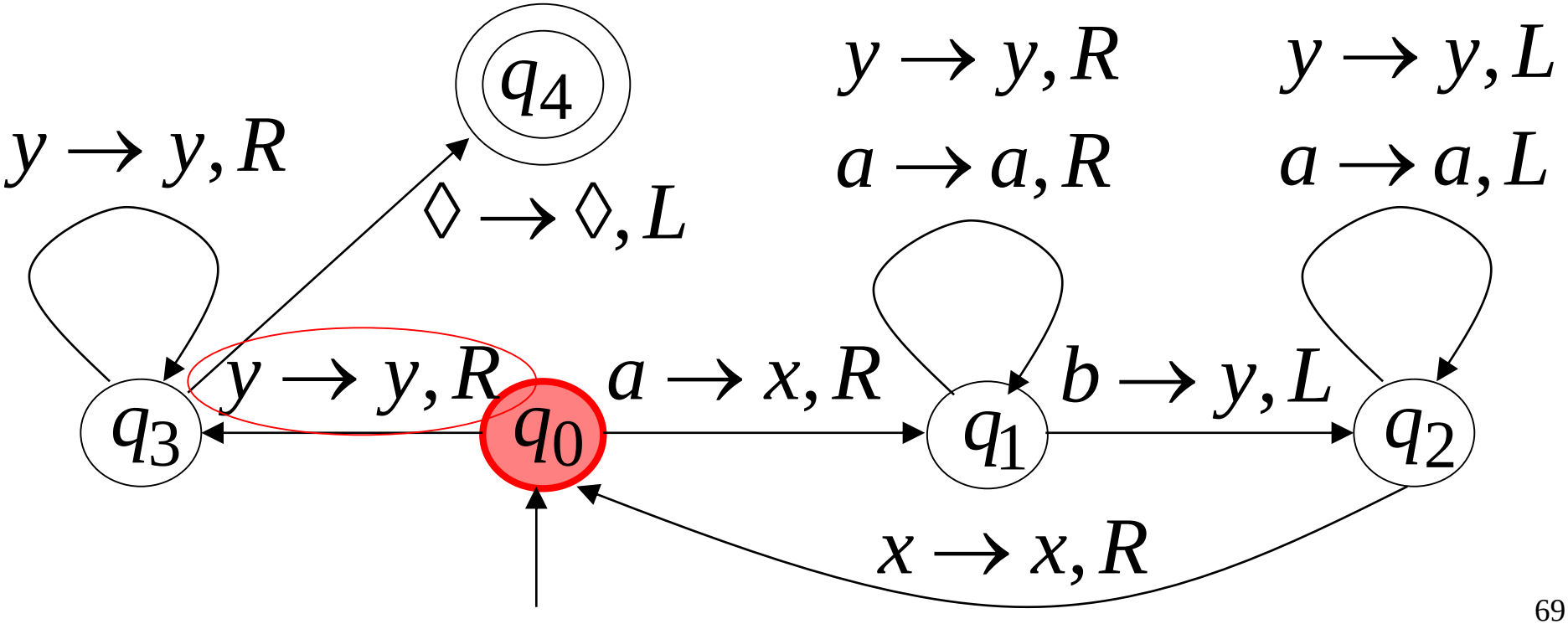
Time 9


◊ x x y y ◊ ◊
$q_2$

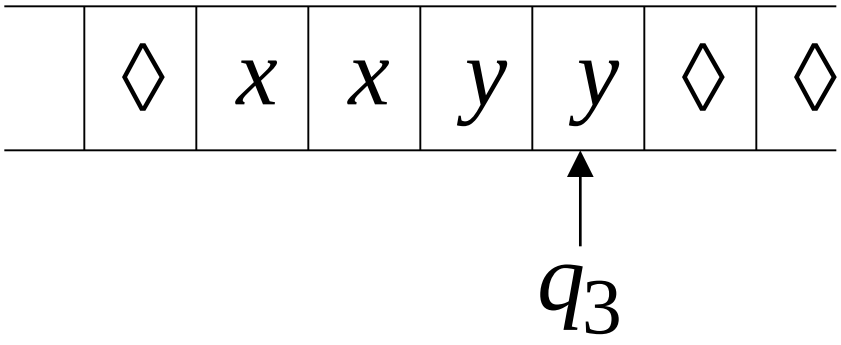

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

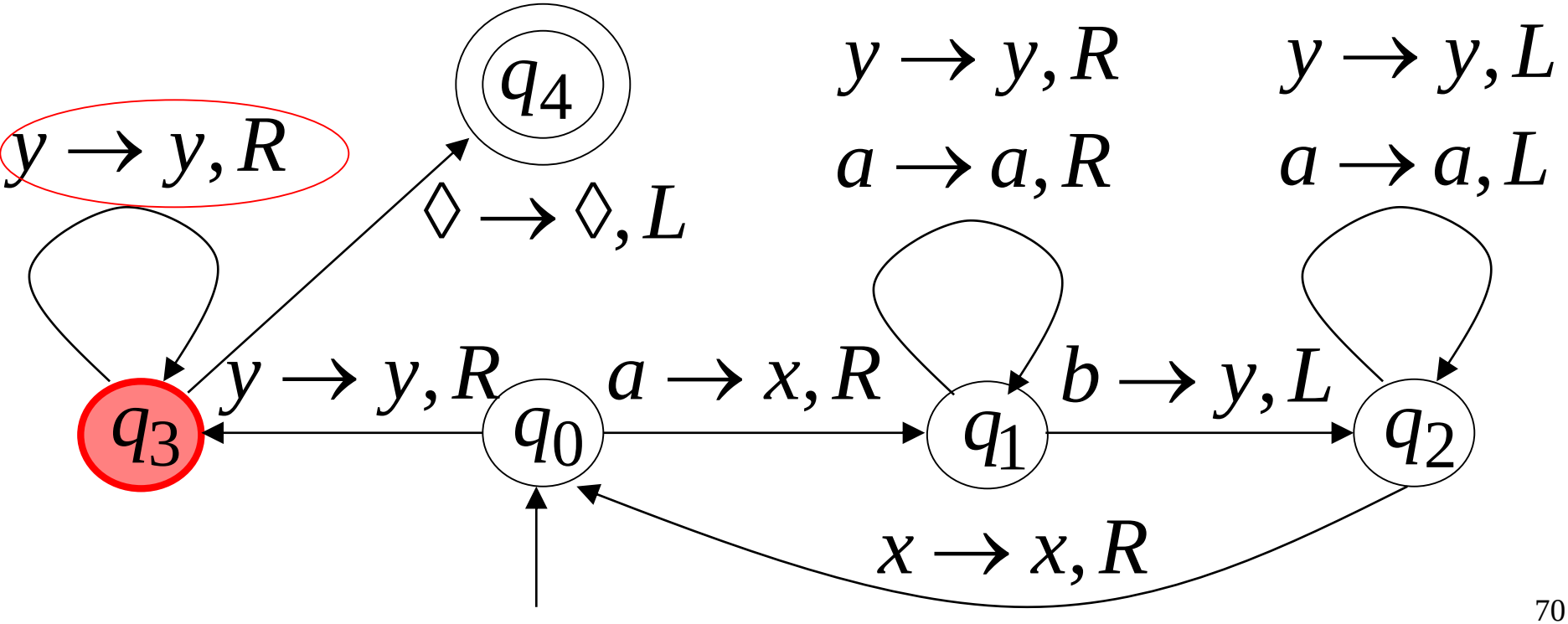
Time 10


◊ x x y y ◊ ◊
$q_0$

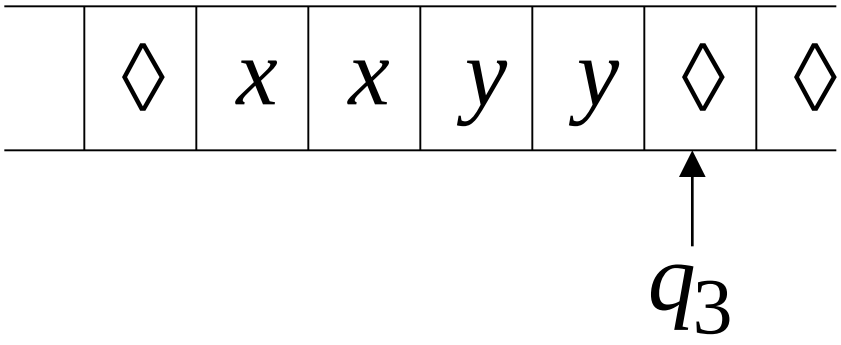

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

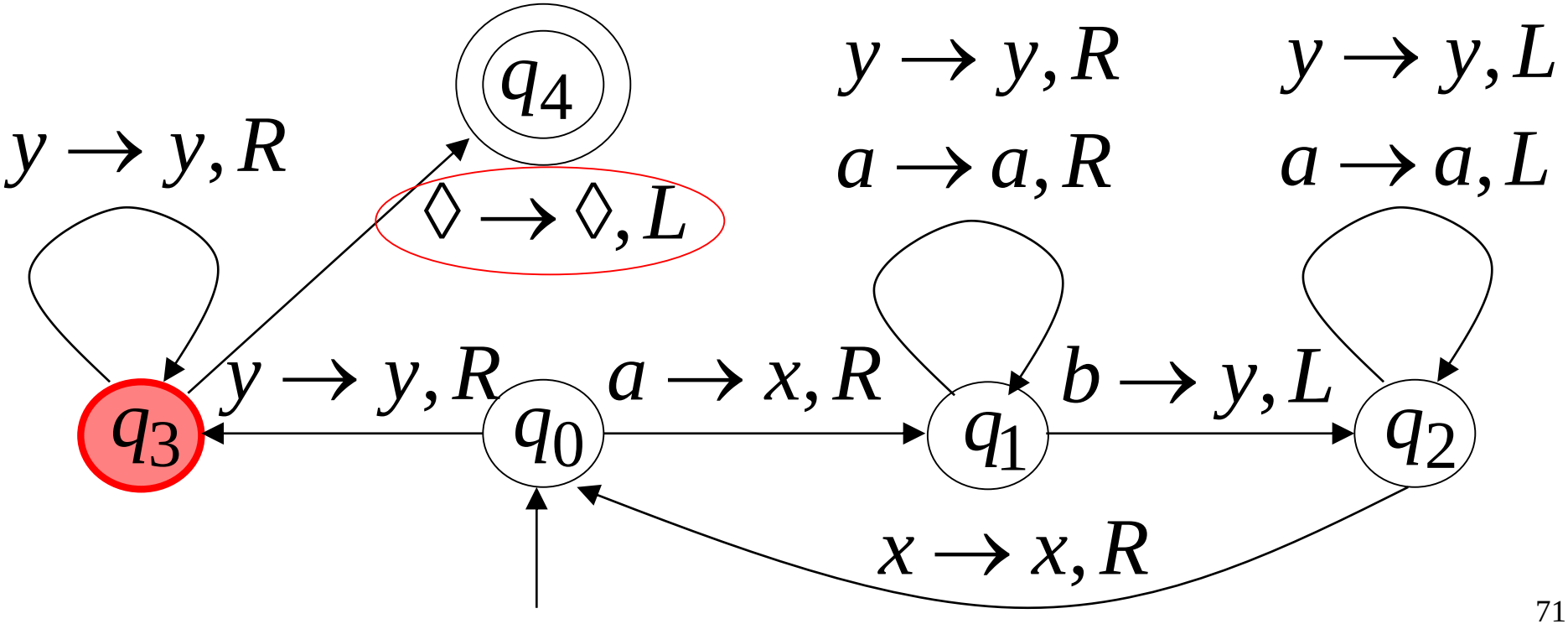
Time 11


◊ x x y y ◊ ◊
$q_3$

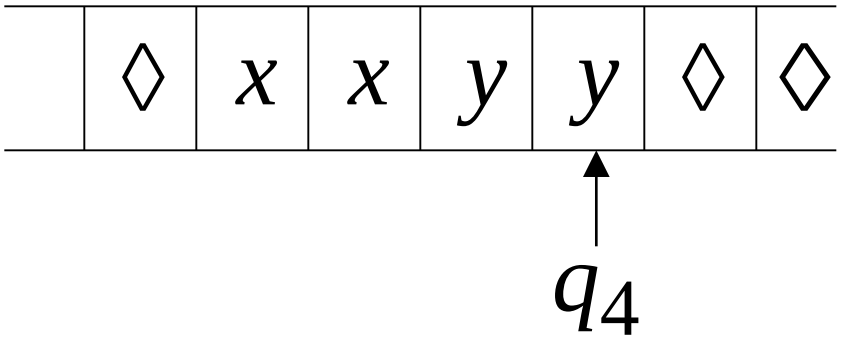

$q_4$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
$q_3$
$q_0$
$q_1$
$q_2$
$x \to x, R$

Time 12

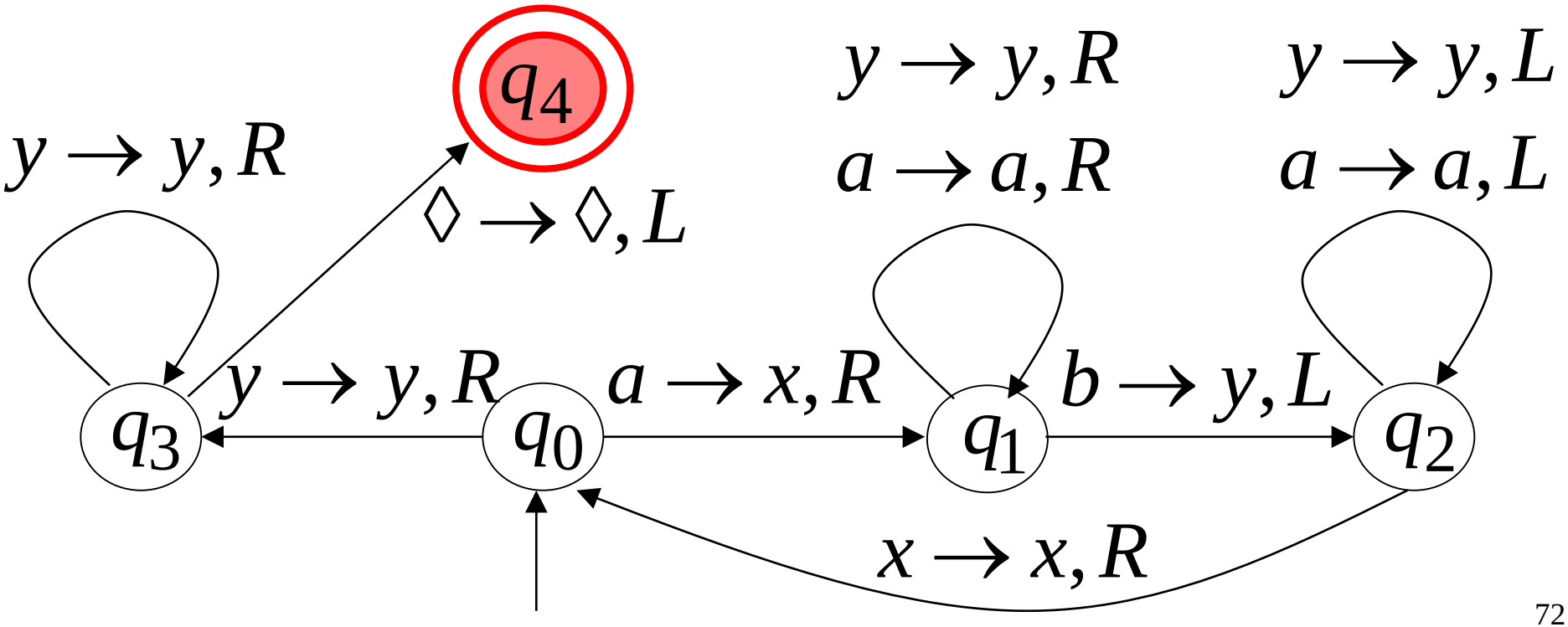

◊ x x y y ◊ ◊
$q_3$


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

Time 13




**Halt & Accept**



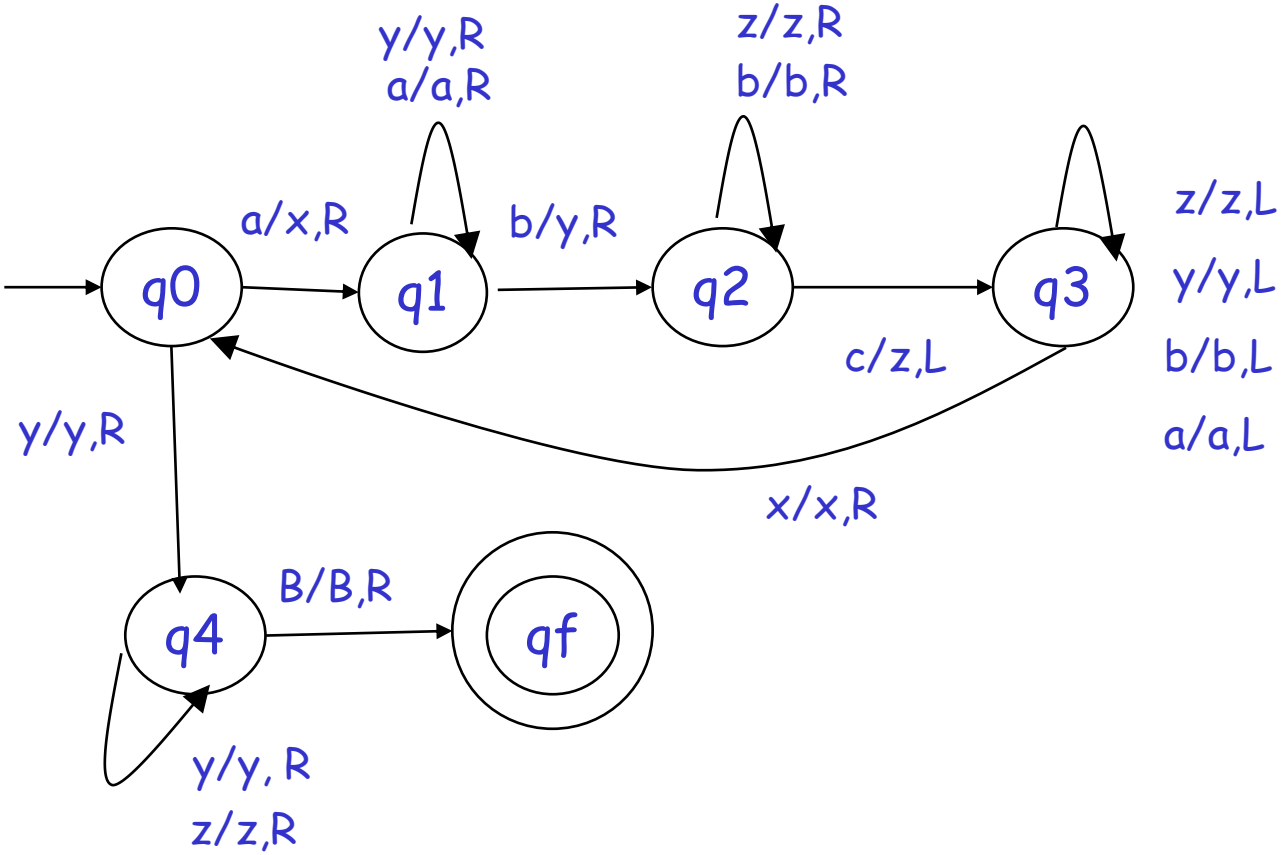
## Observation:

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

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

we can easily construct a machine for the language

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



# Turing Machines as Transducers

Turing machine transducer $M$ as an implementation of a function $f$ defined by

$$\widehat{w} = f(w),$$

provided that

$$q_0 w \overset{*}{\vdash}_M q_f \widehat{w},$$

for some final state $q_f$.

Example 1:

Given two positive integers *x* and *y*, design a Turing machine that computes *x* + *y*.

- We will assume that *w*(*x*) and *w*(*y*) are on the tape in unary notation, separated by a single 0, with the read-write head on the leftmost symbol of *w*(*x*).
- After the computation, *w* (*x*+ *y*) will be on the tape followed by a single 0, and the read-write head will be positioned at the left end of the result.

$$q_0 w(x) 0 w(y) \overset{*}{\vdash} q_f w(x+y) 0,$$

# Integer Domain

Decimal: 5

Binary: 101

Unary: 11111

We prefer **unary** representation:

easier to manipulate with Turing machines

$$q_0 w(x)\,0\,w(y) \overset{*}{\vdash} q_f w(x+y)\,0,$$

To achieve this, we construct $M=(Q,\Sigma,\Gamma,\delta,q_0,\ ,F)$, with $Q=\{q_0,q_1,q_2,q_3,q_4\}$, $F=\{q_4\}$,

$$\delta(q_0,1)=(q_0,1,R),$$

$$\delta(q_0,0)=(q_1,1,R),$$

$$\delta(q_1,1)=(q_1,1,R),$$

$$\delta(q_1,\Box)=(q_2,\Box,L),$$

$$\delta(q_2,1)=(q_3,0,L),$$

$$\delta(q_3,1)=(q_3,1,L),$$

$$\delta(q_3,\Box)=(q_4,\Box,R),$$

Instantaneous Description

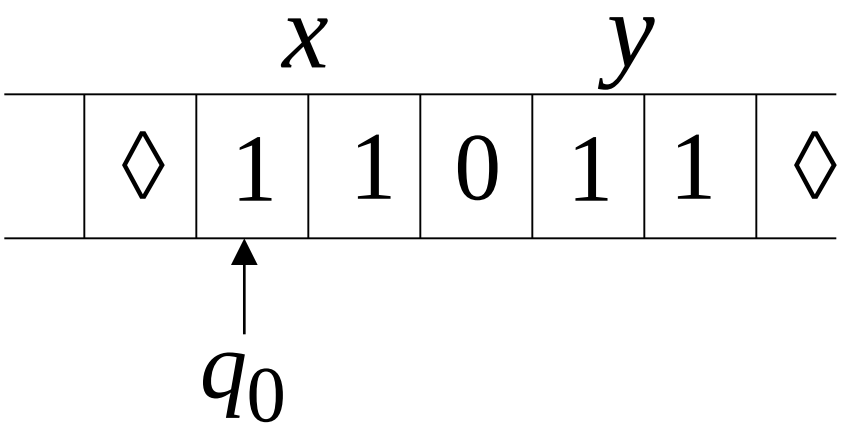
$$\begin{aligned} q_0111011 &\vdash 1q_011011 \vdash 11q_01011 \vdash 111q_0011 \\ &\vdash 1111q_111 \vdash 11111q_11 \vdash 111111q_1\Box \\ &\vdash 11111q_21 \vdash 1111q_310 \\ &\overset{*}{\vdash} q_3\Box111110 \vdash q_4111110. \end{aligned}$$

# Execution Example:

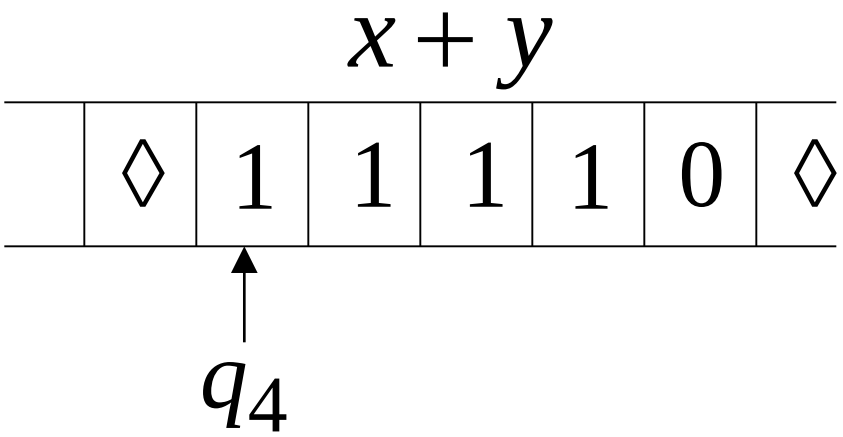
$x = 11$ (2)

$y = 11$ (2)

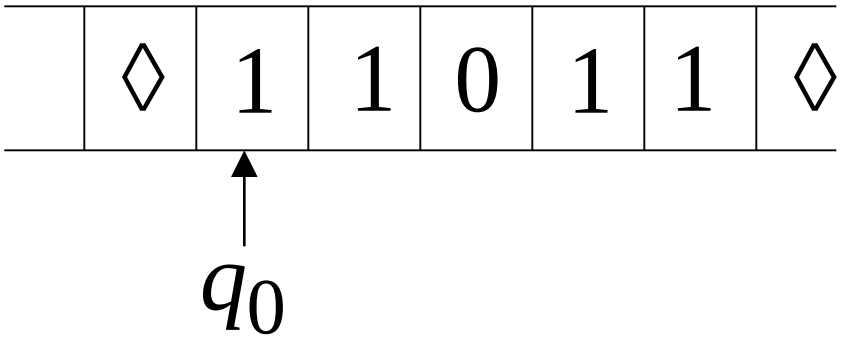
Time 0

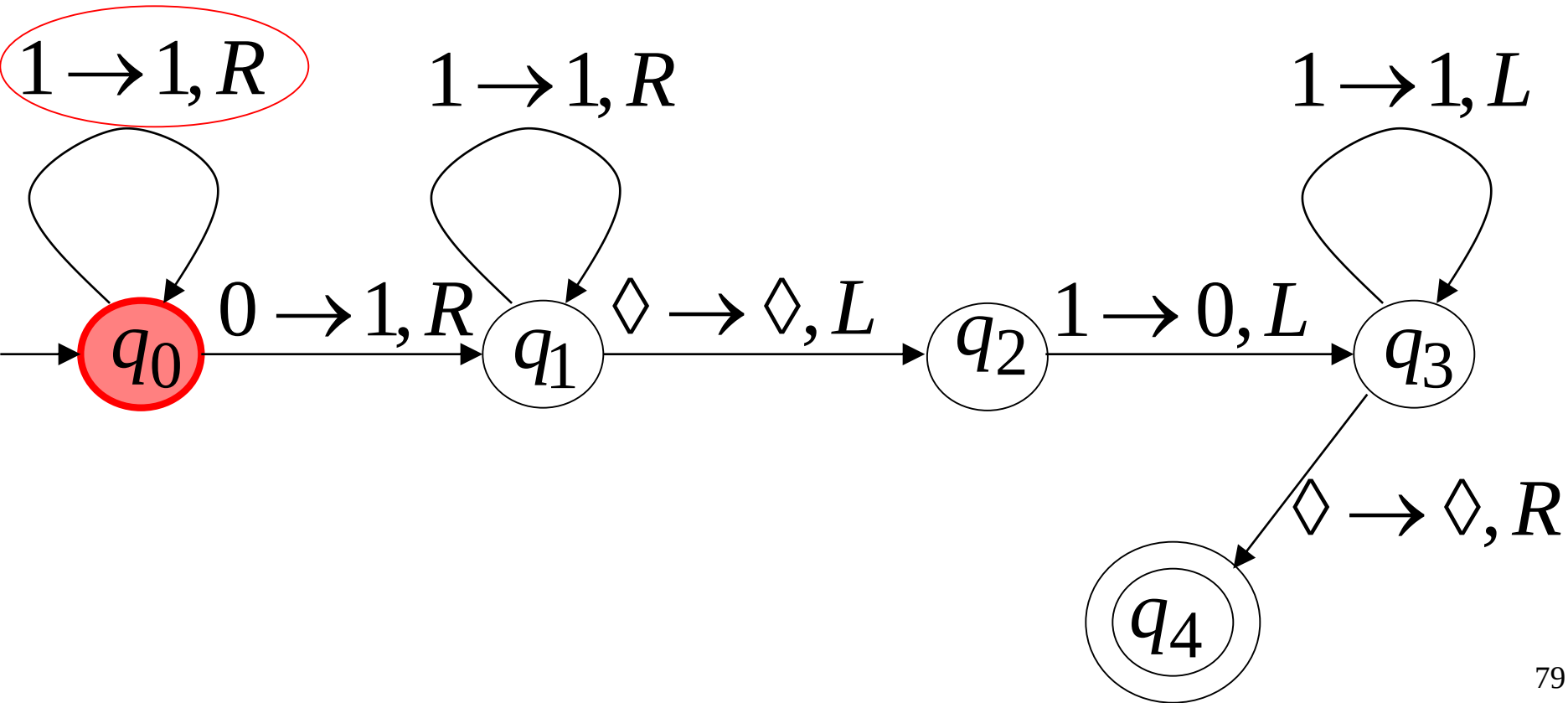



Final Result

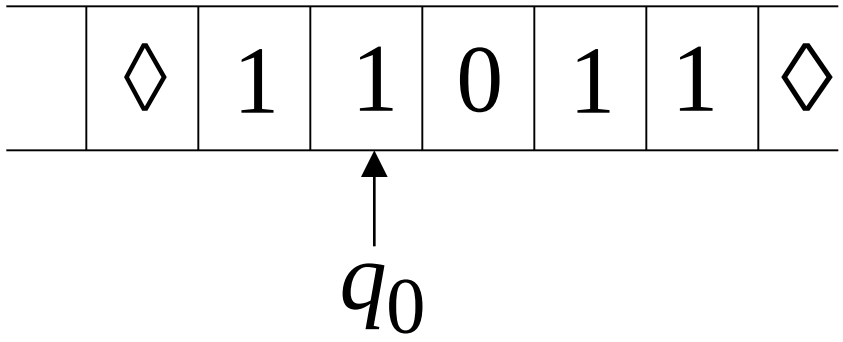


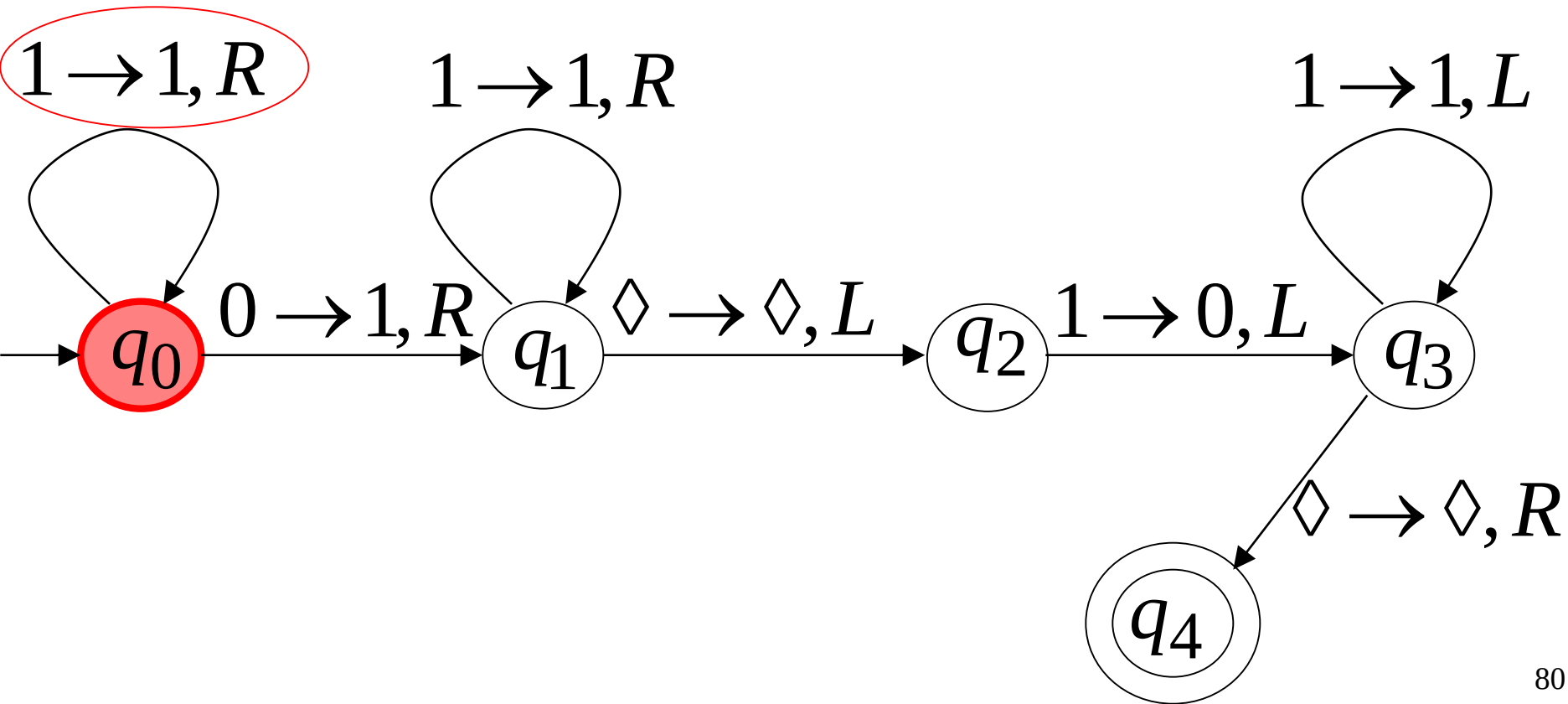
Time 0


◊ 1 1 0 1 1 ◊
$q_0$

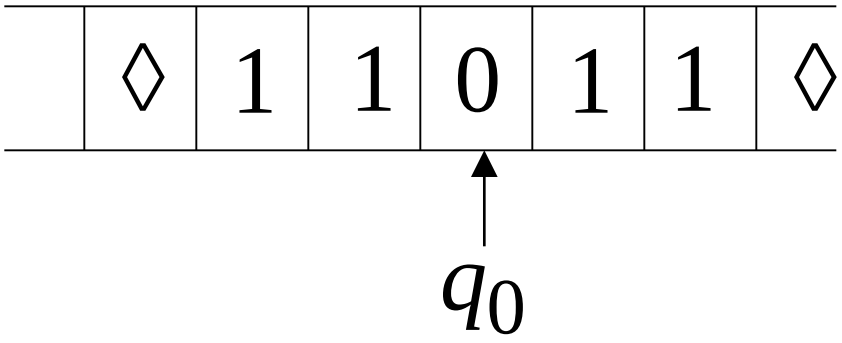

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

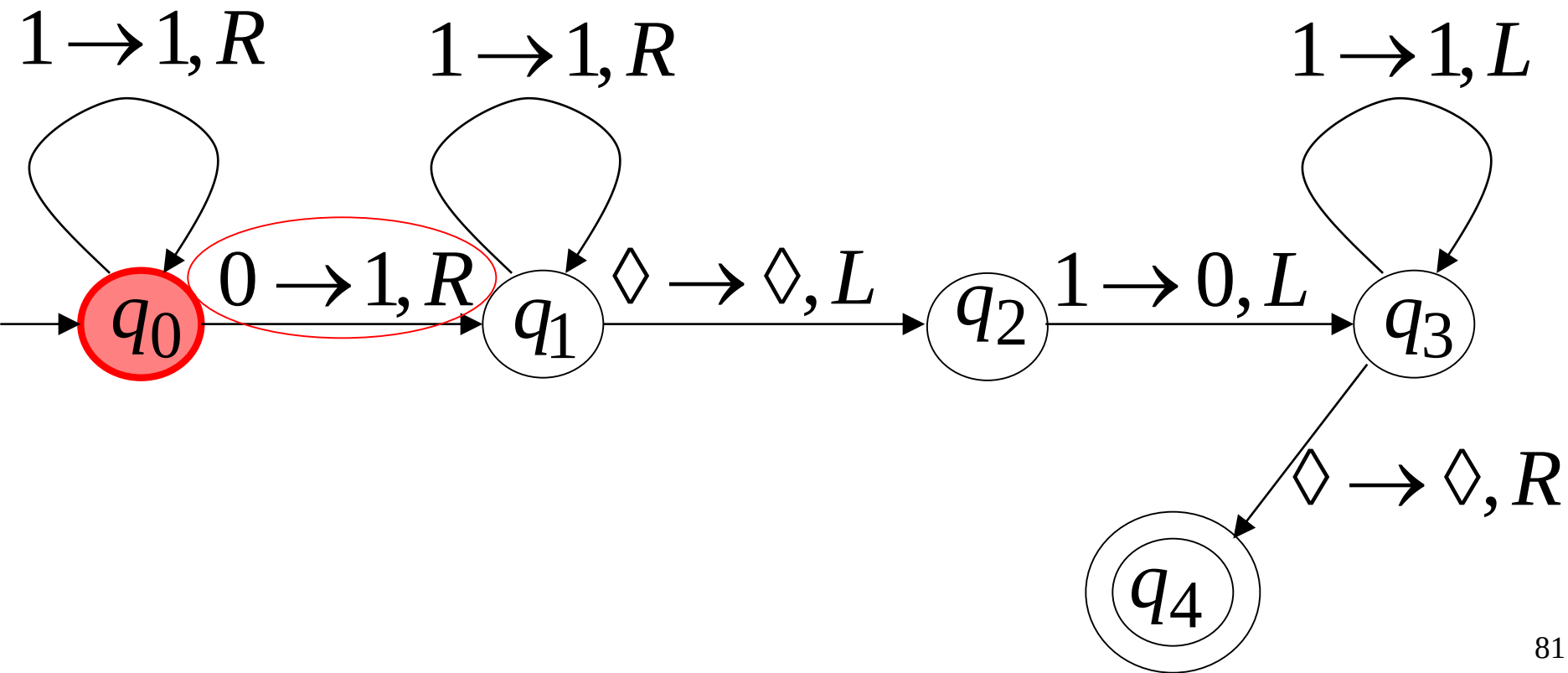
Time 1


◊ 1 1 0 1 1 ◊
$q_0$

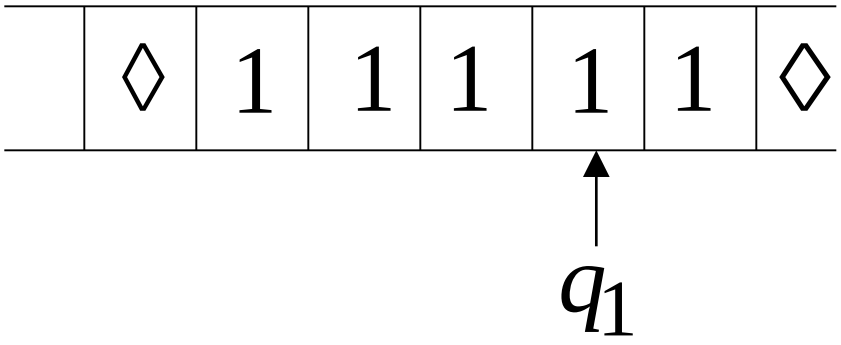

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

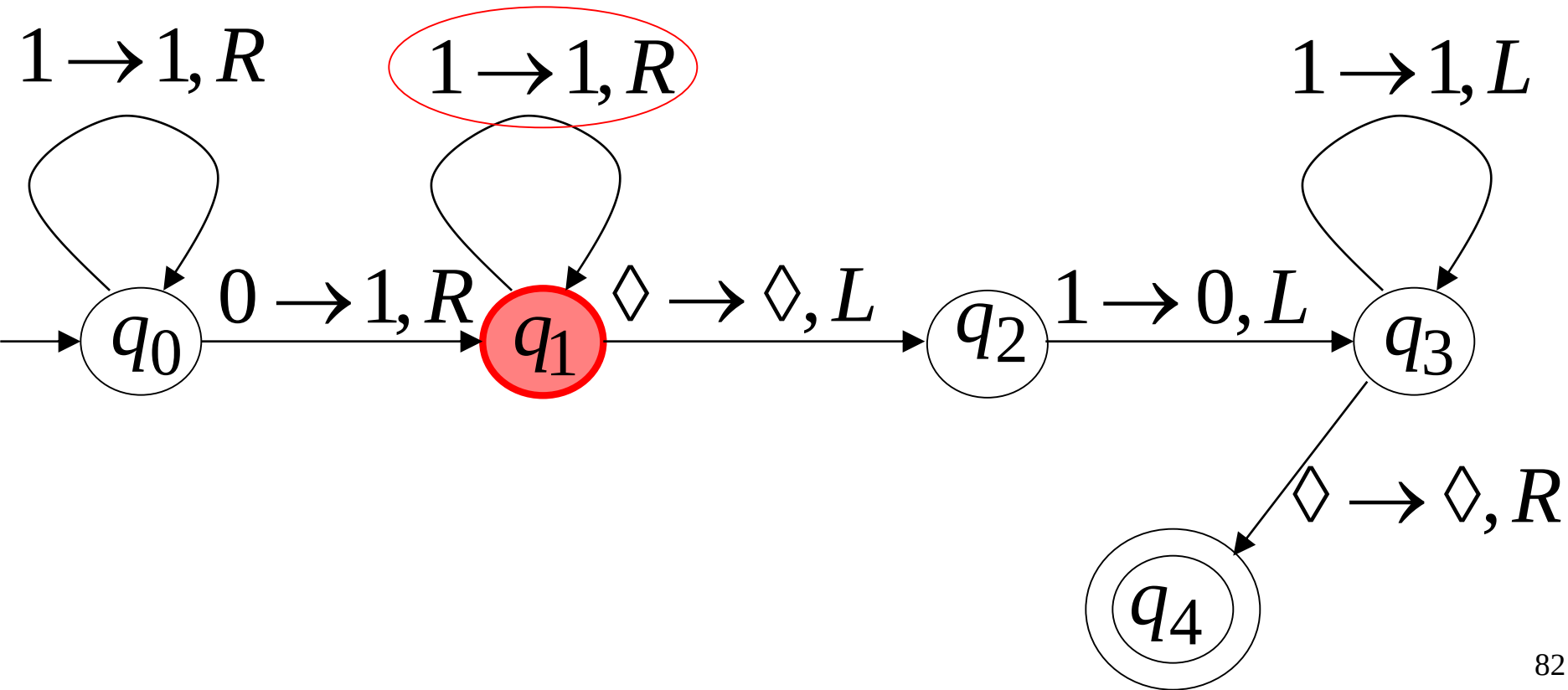
Time 2


◊ 1 1 0 1 1 ◊
$q_0$

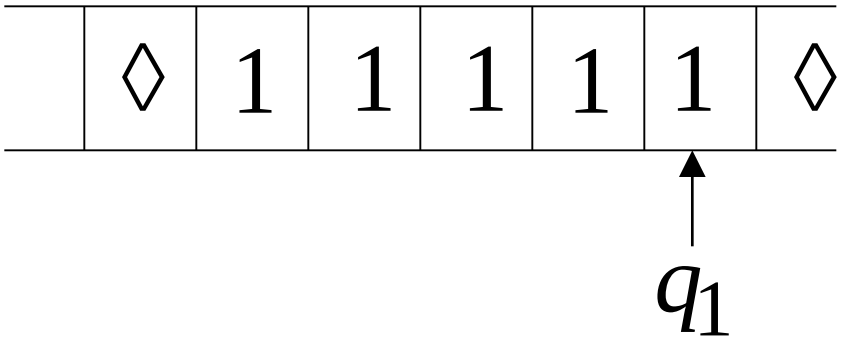

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

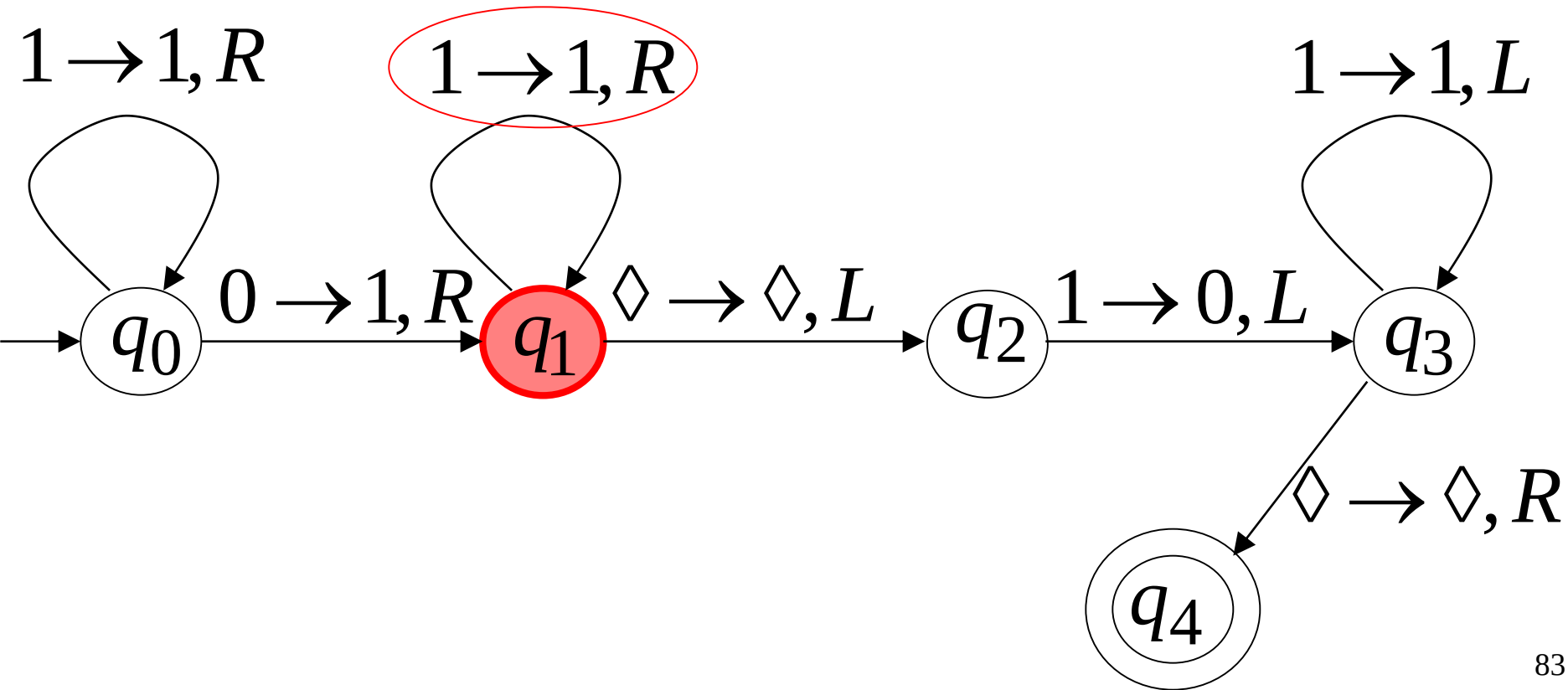
Time 3


◊ 1 1 1 1 1 ◊
$q_1$

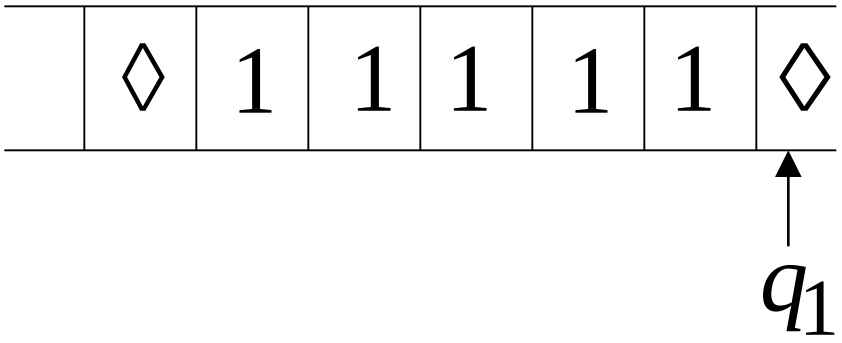

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

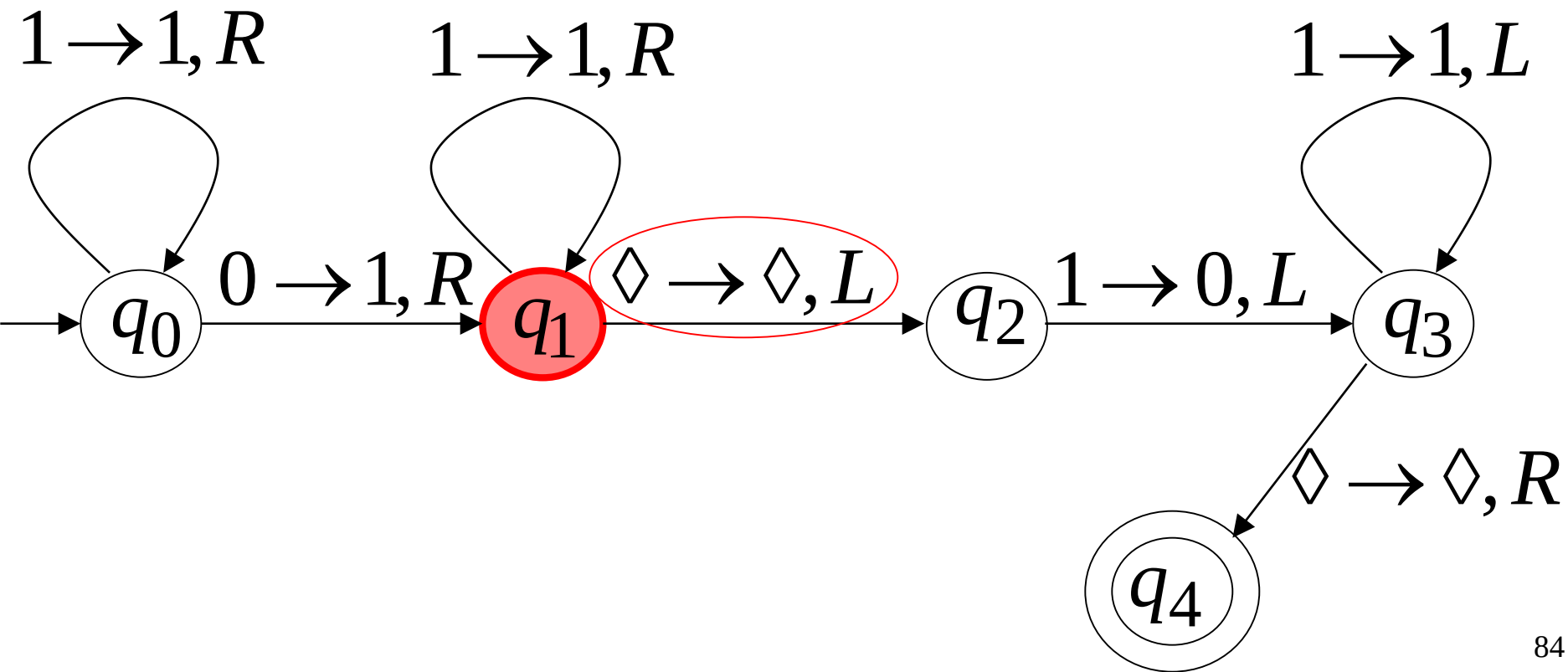
Time 4


◊ 1 1 1 1 1 ◊
$q_1$

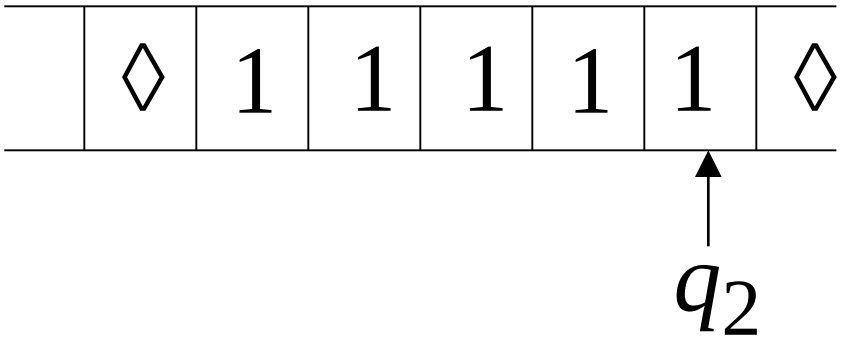

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

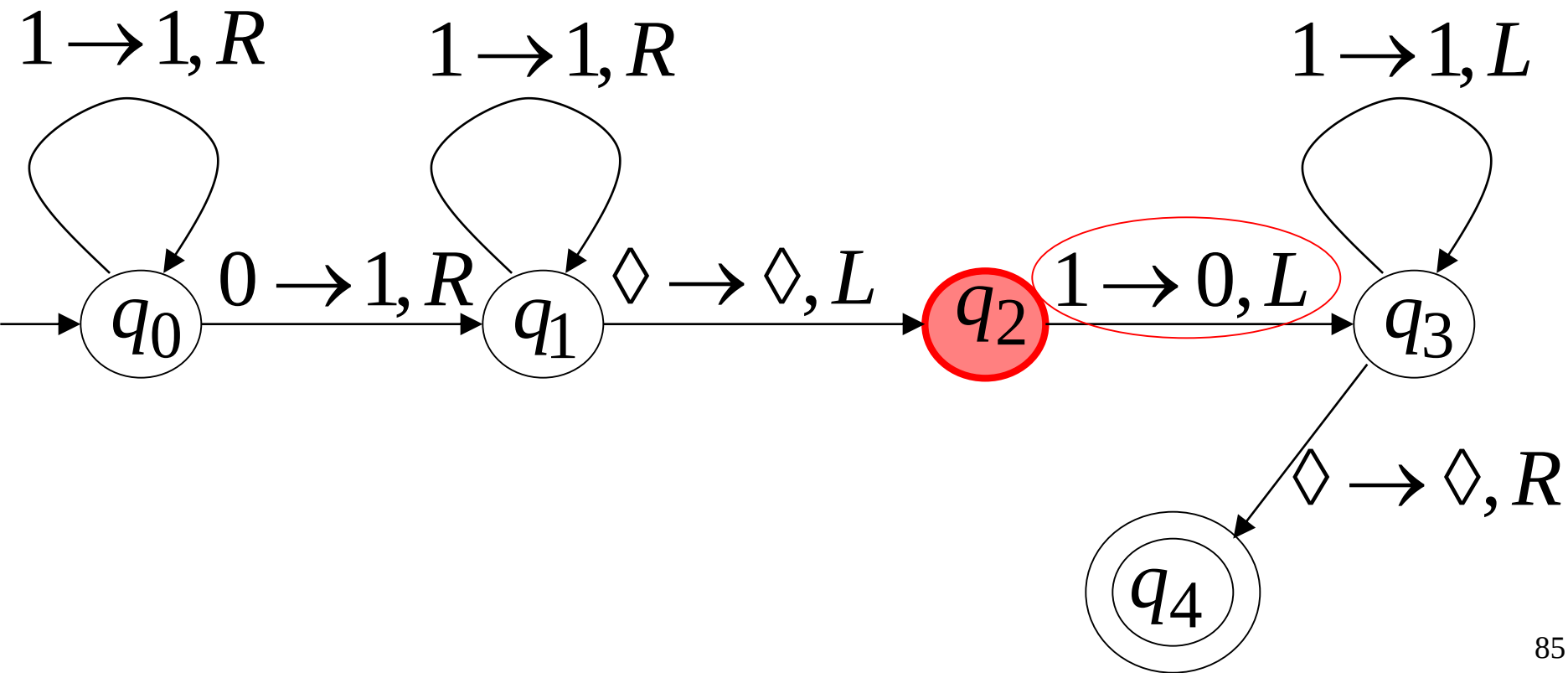
Time 5


◊ 1 1 1 1 1 ◊
$q_1$

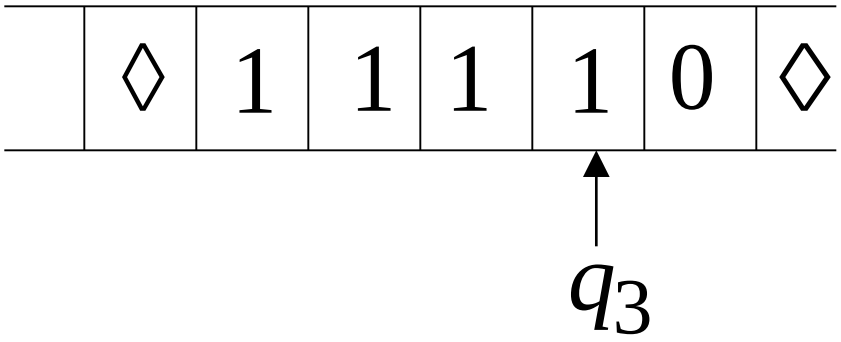

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

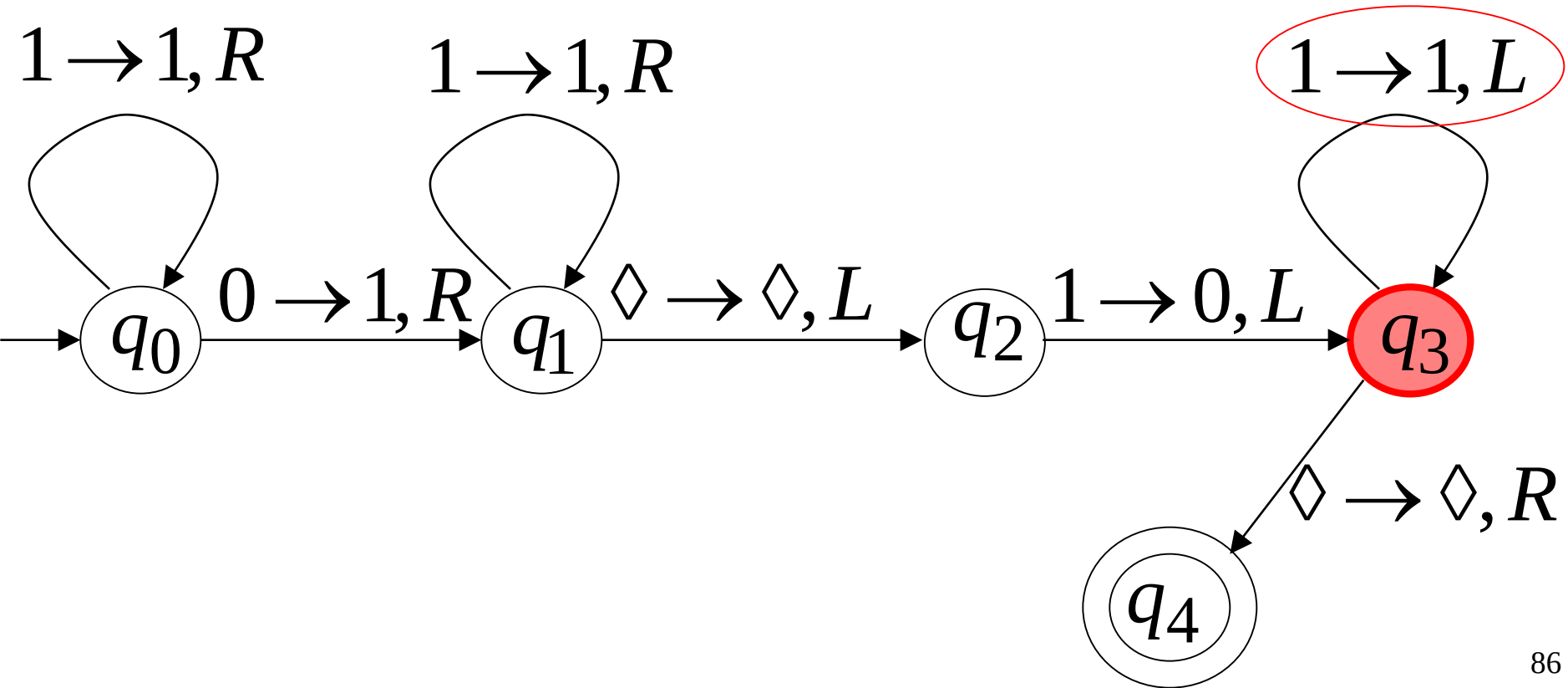
Time 6


◊ 1 1 1 1 1 ◊
$q_2$

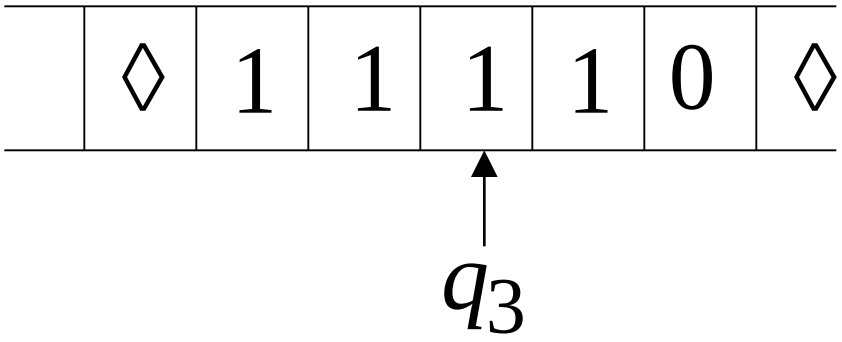

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

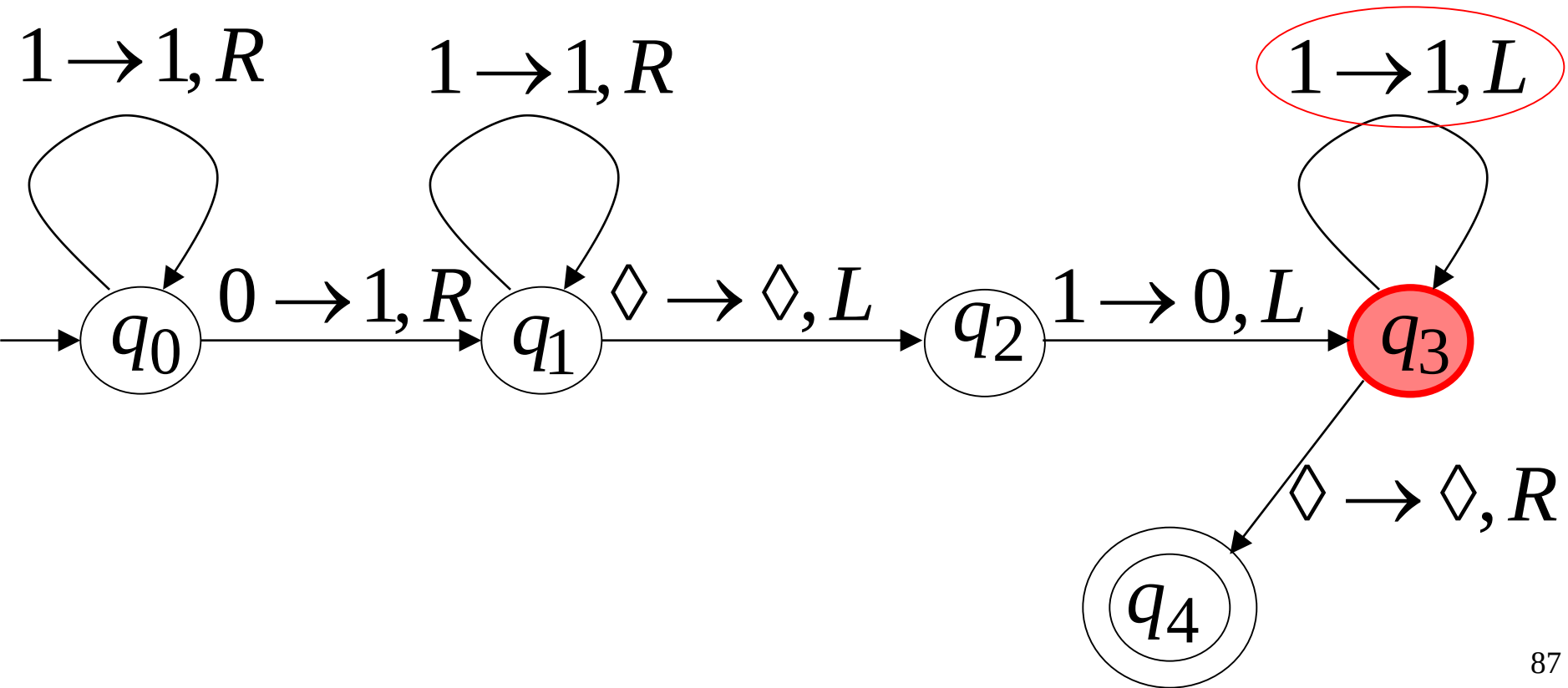
Time 7


◊ 1 1 1 1 0 ◊
$q_3$

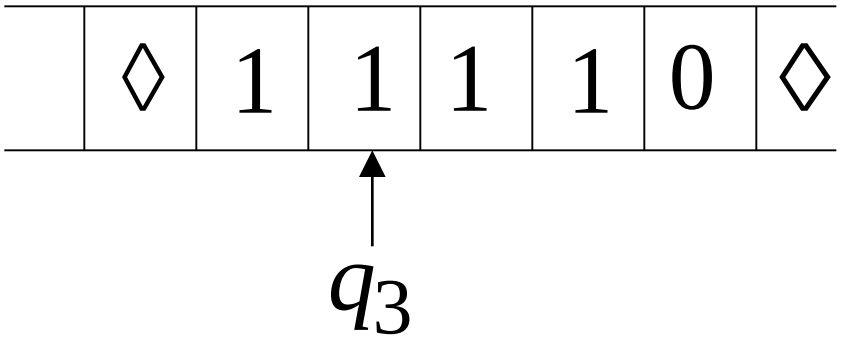

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

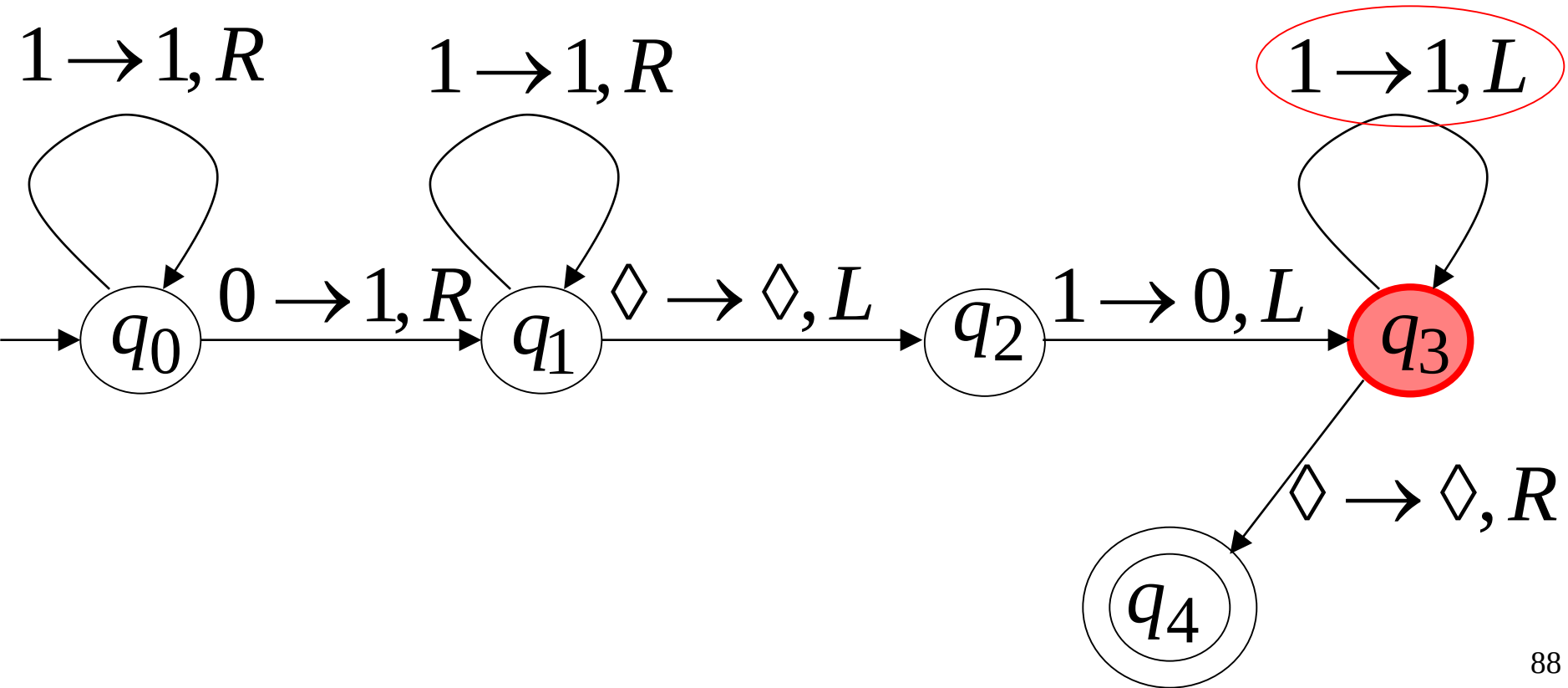
Time 8


◊ 1 1 1 1 0 ◊
$q_3$

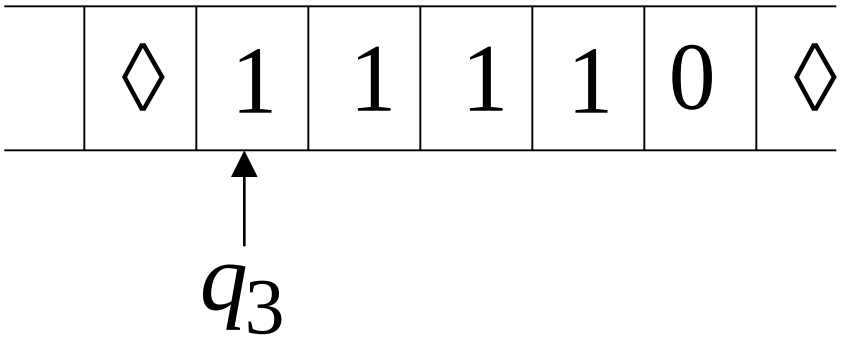

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

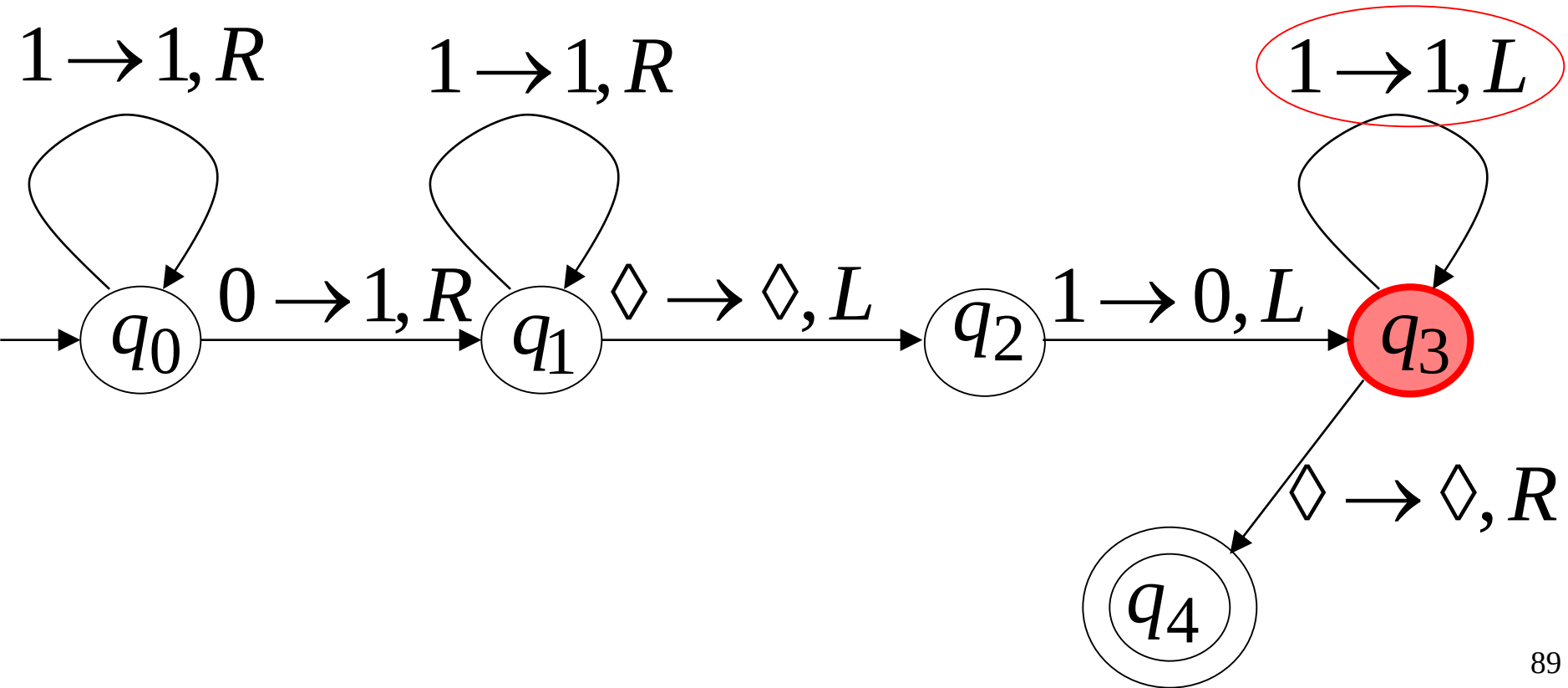
Time 9


◊ 1 1 1 1 0 ◊
$q_3$

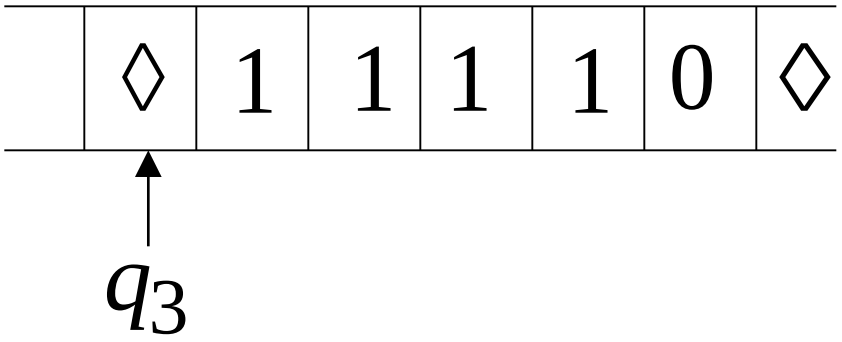

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

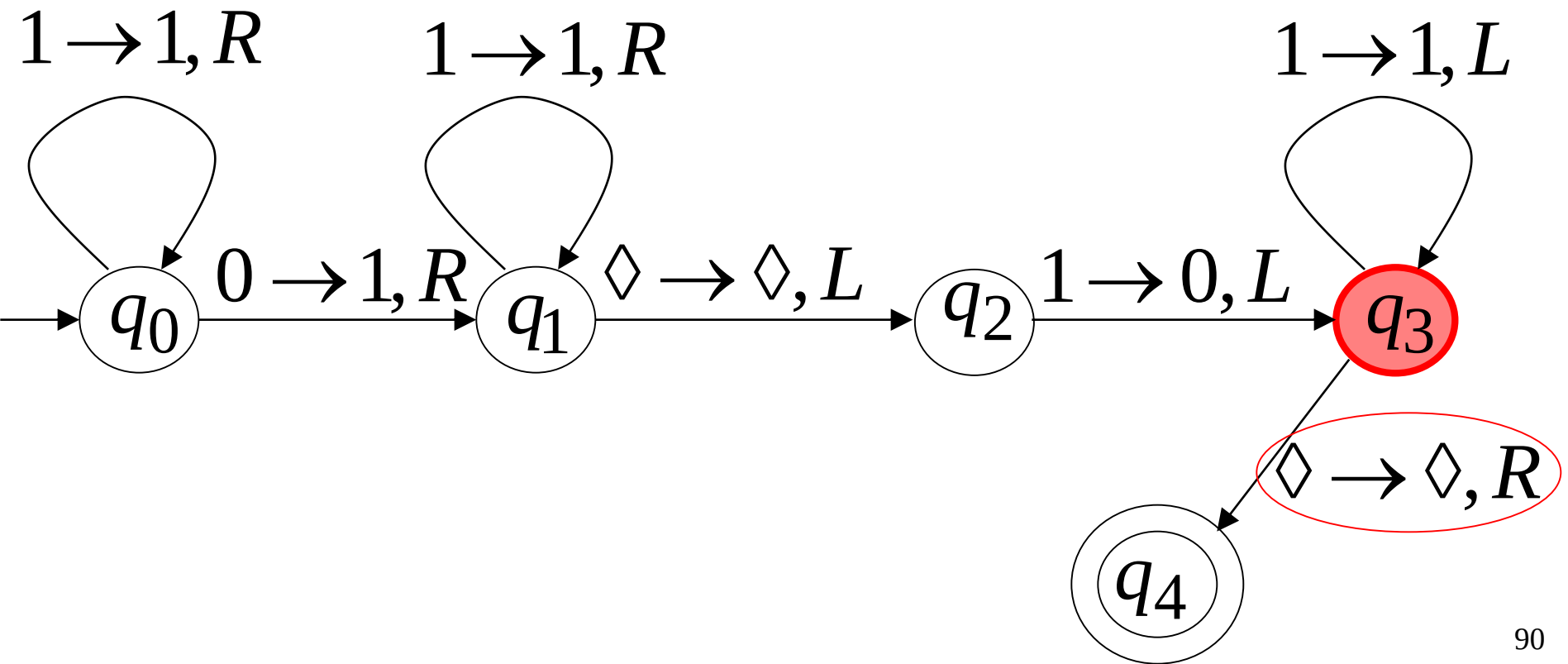
Time 10


◊ 1 1 1 1 0 ◊
$q_3$

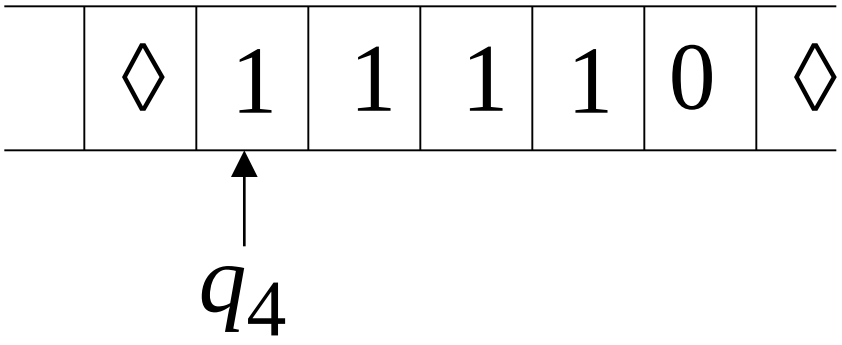

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

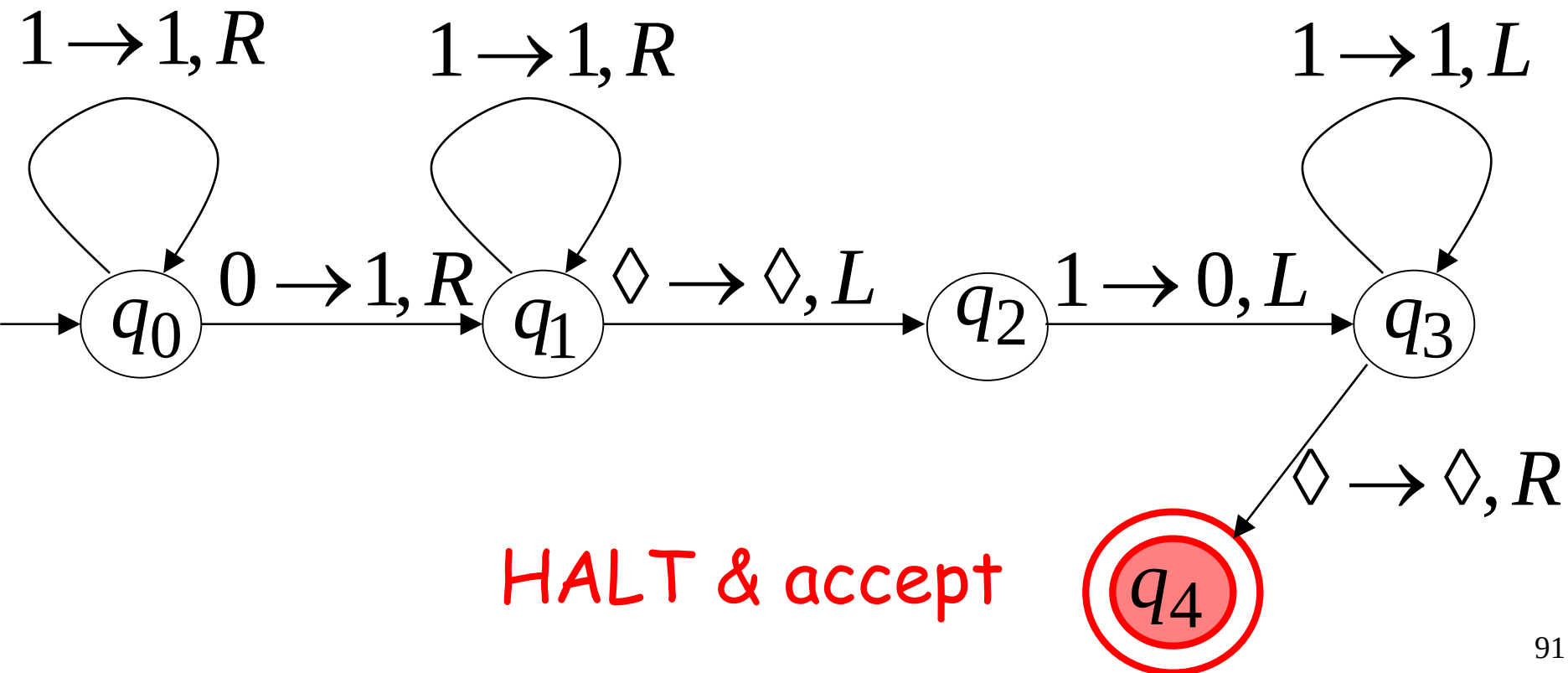
Time 11


◊ 1 1 1 1 0 ◊
$q_3$


$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$

Time 12


◊ 1 1 1 1 0 ◊
$q_4$


$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
$q_0$
$q_1$
$q_2$
$q_3$
$q_4$
HALT & accept

Design a Turing machine that copies strings of 1's. More precisely, find a machine that performs the computation

$q_0 w \overset{*}{\vdash} q_f ww$, for any $w \in \{1\}^+$.

To solve the problem, we implement the following intuitive process:

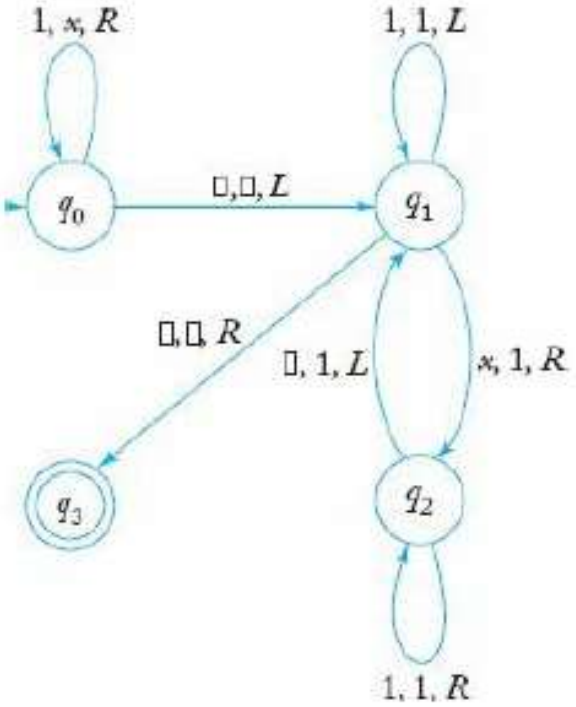
**1.** Replace every 1 by an *x*.

**2.** Find the rightmost *x* and replace it with 1.

**3.** Travel to the right end of the current nonblank region and create a 1 there.

**4.** Repeat Steps 2 and 3 until there are no more x's.

## Transition Graph




## Instantaneous Description

$$
\begin{aligned}
q_0 11 &\vdash x q_0 1 \vdash x x q_0 \square \vdash x q_1 x \\
&\vdash x 1 q_2 \square \vdash x q_1 11 \vdash q_1 x 11 \\
&\vdash 1 q_2 11 \vdash 11 q_2 1 \vdash 111 q_2 \square \\
&\vdash 11 q_1 11 \vdash 1 q_1 111 \\
&\vdash q_1 1111 \vdash q_1 \square 1111 \vdash q_3 1111.
\end{aligned}
$$

A function may have many parameters:

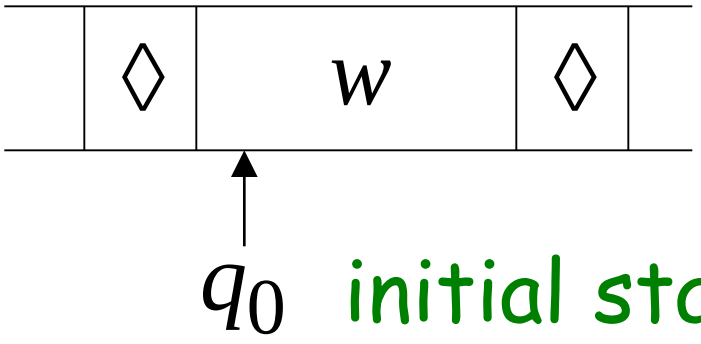
Example: Addition function

$$f(x, y) = x + y$$

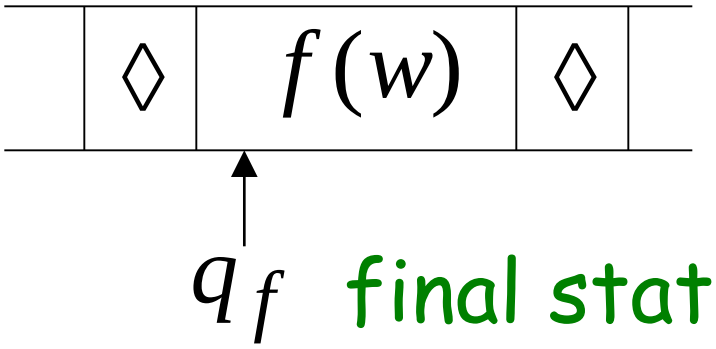
# Definition:

A function $f$ is computable if there is a Turing Machine $M$ such that:

Initial configuration




Final configuration




For all $w \in D$ Domain

In other words:

A function $f$ is computable if there is a Turing Machine $M$ such that:

$$q_0\, w \overset{*}{\succ} q_f\, f(w)$$

Initial Configuration

Final Configuration

For all $w \in D$ Domain

# Example

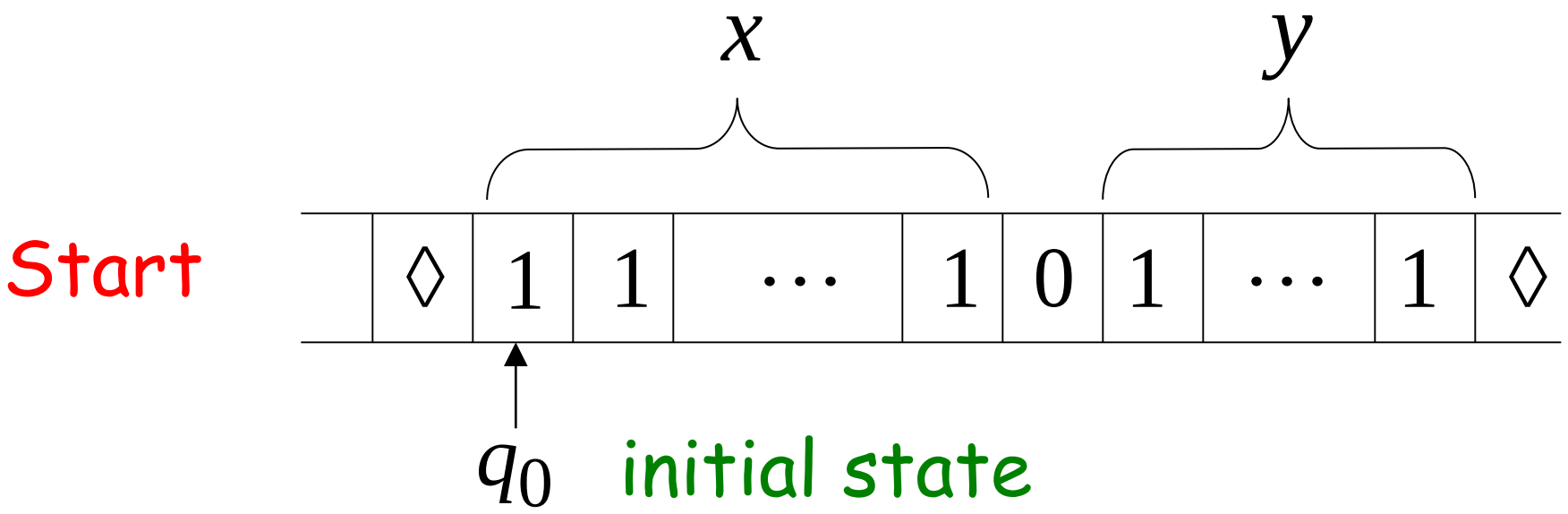
The function $f(x, y) = x + y$ is computable

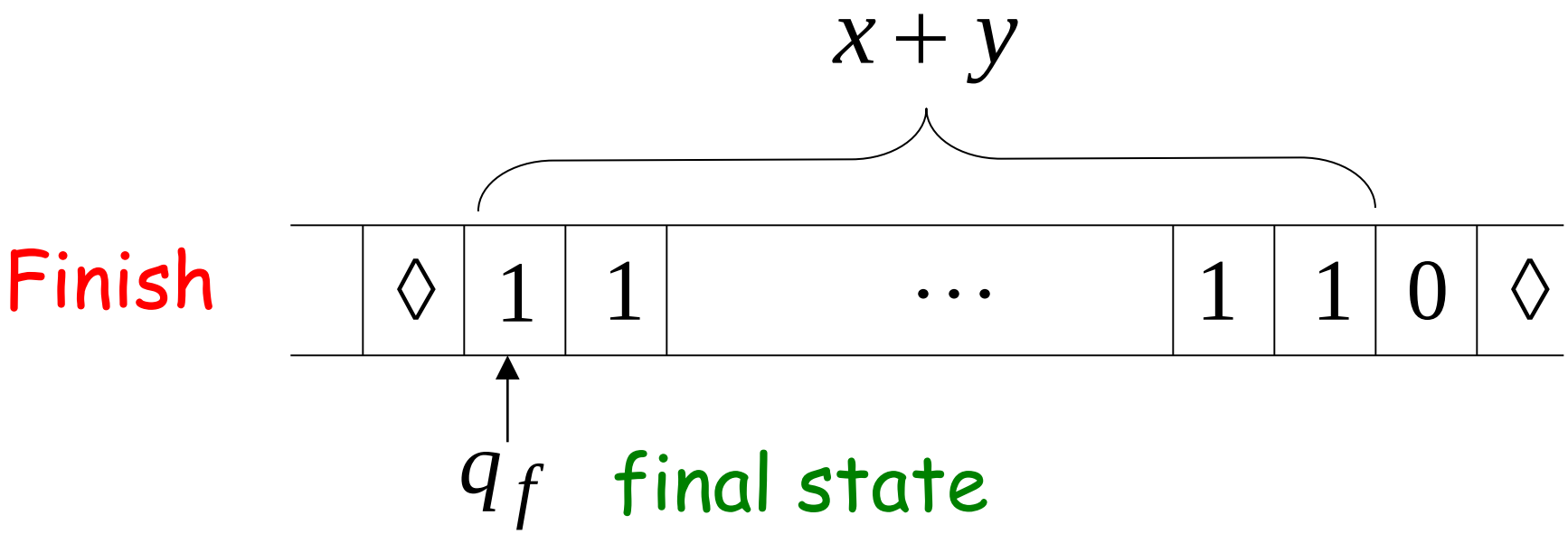
$x, y$ are integers

Turing Machine:

Input string: $x0y$ unary

Output string: $xy0$ unary


x
y
Start
◊ 1 1 ⋯ 1 0 1 ⋯ 1 ◊
$q_0$ initial state


x + y
Finish
◊ 1 1 ⋯ 1 1 0 ◊
$q_f$ final state

# Another Example

The function $f(x) = 2x$ is computable

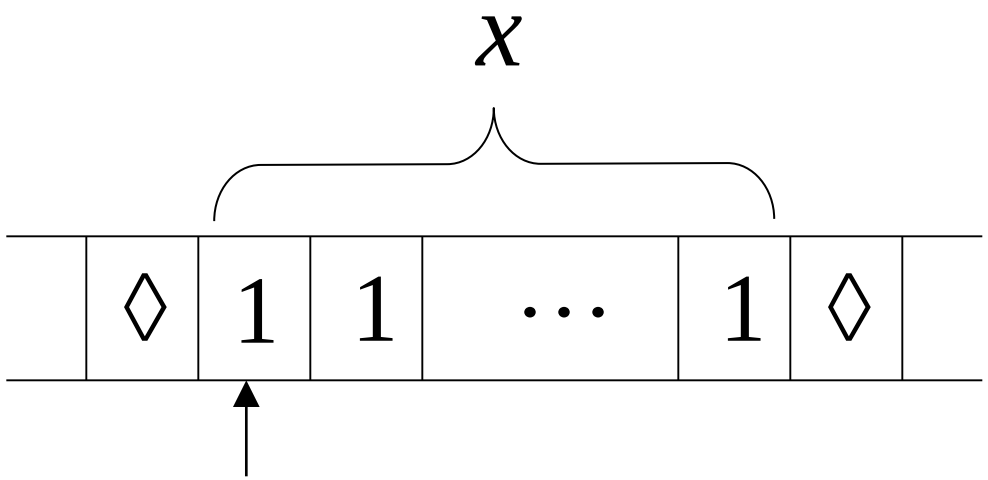
$x$ is integer

Turing Machine:

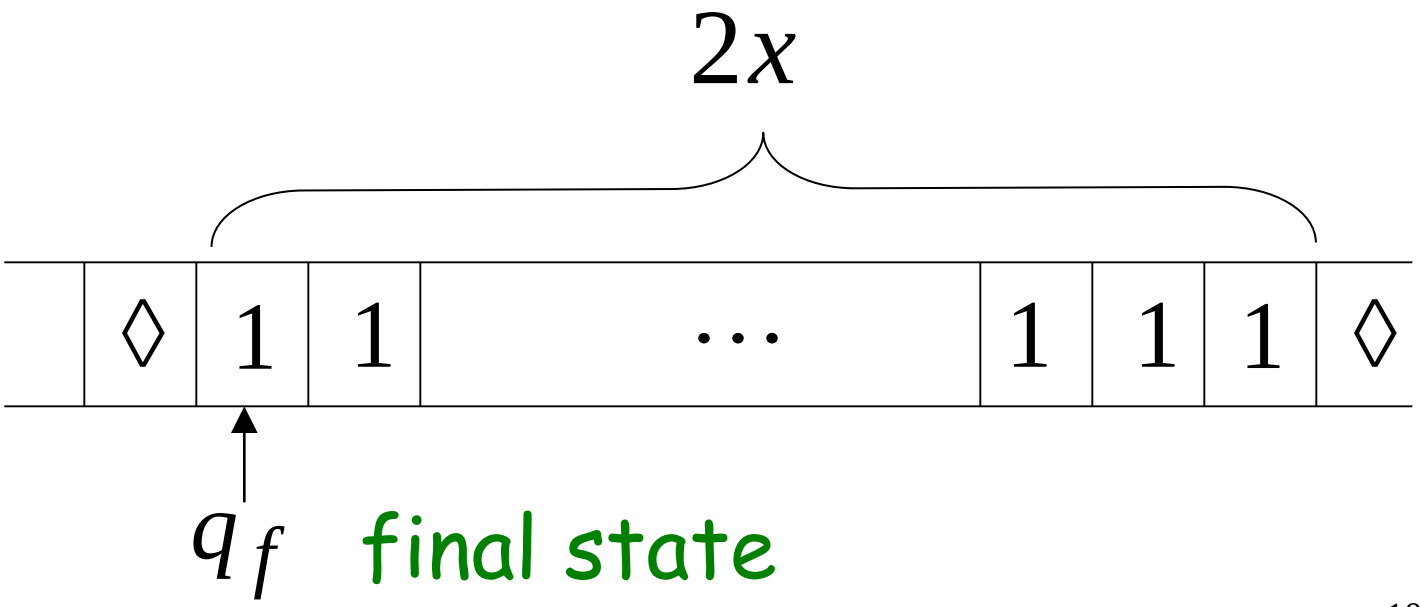
Input string: $x$ unary

Output string: $xx$ unary

Start




Finish

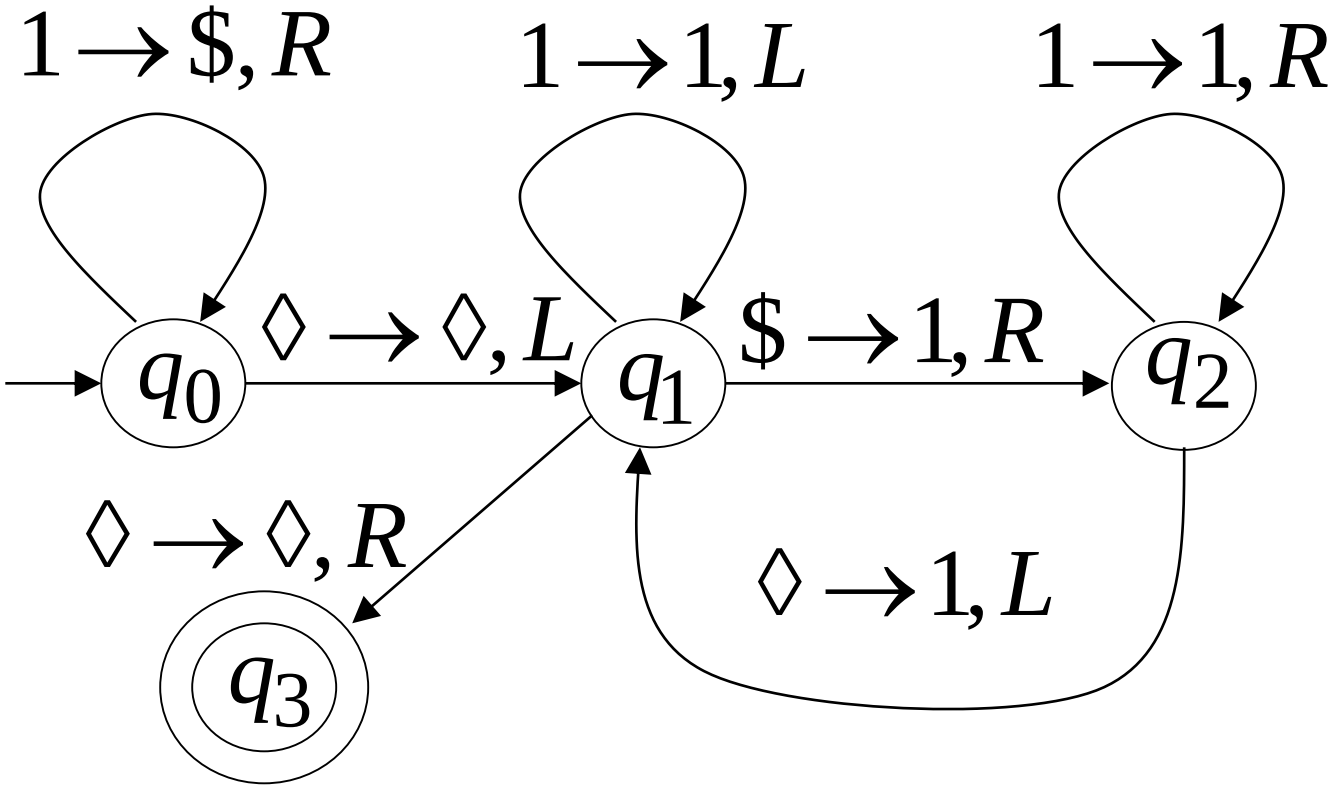


# Turing Machine Pseudocode for $f(x) = 2x$

- Replace every 1 with $
- Repeat:
    - Find rightmost $, replace it with 1
    - Go to right end, insert 1

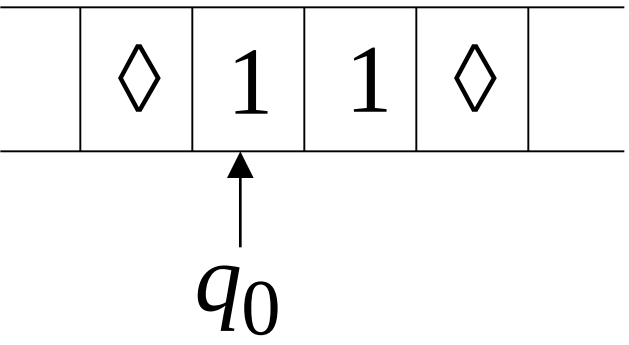
  Until no more $ remain

# Turing Machine for $f(x) = 2x$



# Example

Start




Finish