Department of
Computer science

Complexity Theory

Course Contents:

1. Turing Machines (TMs)

1.1 Standard TM

1.2 Computability of TM

2. Computability

2.1 Introduction to recursive function theory; primitive and partial recursive functions

2.2 Recursive and recursively enumerable languages

2.3 Turing computable

3. Undecidability

3.1 Turing decidable and Turing acceptable

3.2 Undecidable problems

4. Computational Complexity

4.1 Basics of algorithm analysis: Big O-notation

4.2 Polynomial time and space; nondeterministic polynomial time; P vs NP

Learning Outcomes:

- Equipping the students with techniques of computational complexity like Computability and Undecidability.

Course Description:

- Introduction to various aspects of theory of computation: the Turing Machine (TM) model; computable languages and functions; recursive functions; undecidability and computational complexity.

Complexity Theory

Complexity theory is concerned with how much computational resources are required to solve a given task.

Computational complexity theory focuses on classifying **computational** problems according to their resource usage.

A **computational** problem is a task solved by a **computer**.

1. Turing Machines

Alan Turing

Alan Turing was one of the founding fathers of CS.

- His computer model **, the Turing Machine,** was inspiration of the electronic computer that came two decades later.
- Invented the “Turing Test” used in AI

A Thinking Machine

First Goal of Turing's Machine: A model that can compute anything that a human can compute.

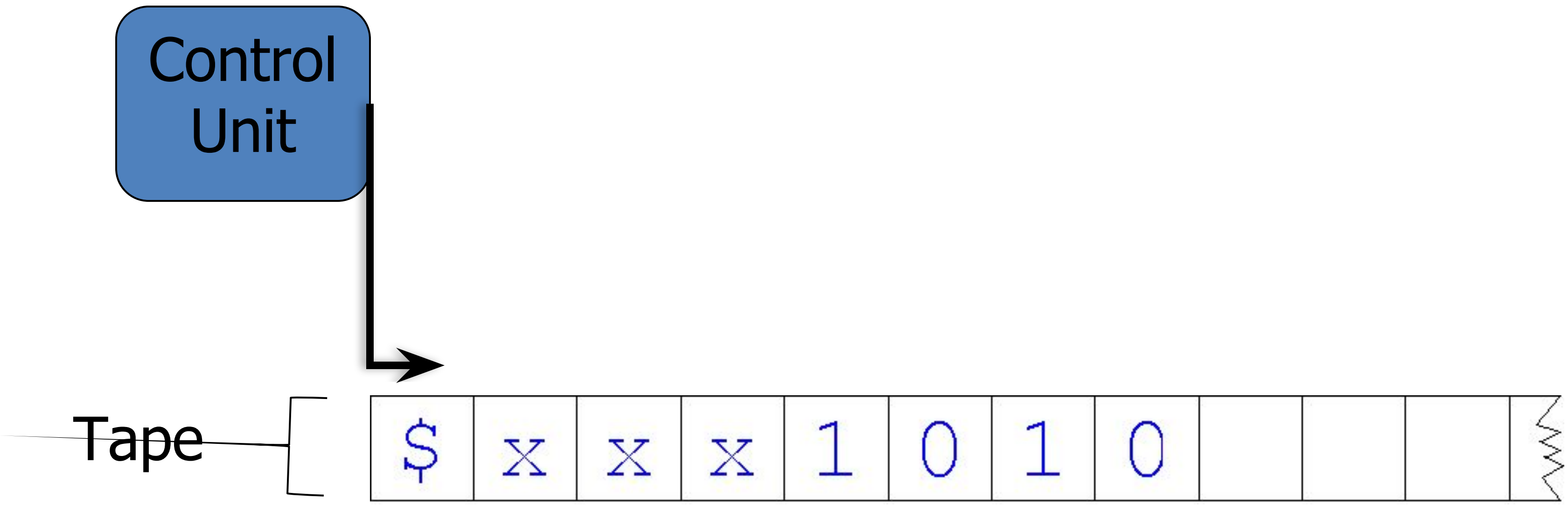
Before invention of electronic computers the term "computer" actually referred to a *person* whose line of work is to calculate numerical quantities!

Turing's Thesis: Any "algorithm" can be carried out by one of his machines

Turing Machine

- Turing machine is a model of general purpose computer.
- A Turing machine can do everything that a real computer can do.
- The Turing machine model uses an infinite tape as its unlimited memory.
- It has a tape head that can read and write symbols and move around on the tape.
- Initially the tape contains only the input string and is blank everywhere else.

- If the machine needs to store information, it may write this information on the tape.
- To read the information that it has written, the machine can move its head back over it.
- The machine continues computing until it decides to produce an output.
- The outputs *accept and reject are obtained by entering designated* accepting and rejecting states.
- If it doesn't enter an accepting or a rejecting state, it will go on forever, never halting.


Control Unit
Tape
$
x
x
x
1
0
1
0

Standard TM

- A TM is given by a tuple: (S, s_0, A, T),

where:

- **S** is a finite list of possible *states* that the machine can be in. The state is the information that the machine can store in the head to make decisions.
- $\mathbf{s_0} \in \mathbf{S}$ is the *initial* state - the state that the machine will be in when it starts a computation.
- **A** is the machine's alphabet, which is the set of symbols which can appear on the machine's tape.
- **T** is the machines *transition function*. This is the real heart of the machine, where the computation is defined. It's a function from the machine state and the alphabet character on the current tape cell to the action that the machine should take.
- The action is a triple consisting of a new value for the machine's state, a character to write onto the current tape cell, and a direction to move the tape head - either left or right.

Example
`Successor` Program

Sample Rules:

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read , write 1, HALT!

Let’s see how they are carried out on a piece of paper that contains the *reverse* binary representation of 47:

Example
`Successor` Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read , write 1, HALT!

1	1	1	1	0	1				

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read , write 1, HALT!

0	1	1	1	0	1				

A Thinking Machine
EG **Successor** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read , write 1, HALT!

0	0	1	1	0	1				

A Thinking Machine
EG **Successor** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read , write 1, HALT!

0	0	0	1	0	1				

A Thinking Machine
EG **Successor** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read , write 1, HALT!

0	0	0	0	0	1				

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read , write 1, HALT!

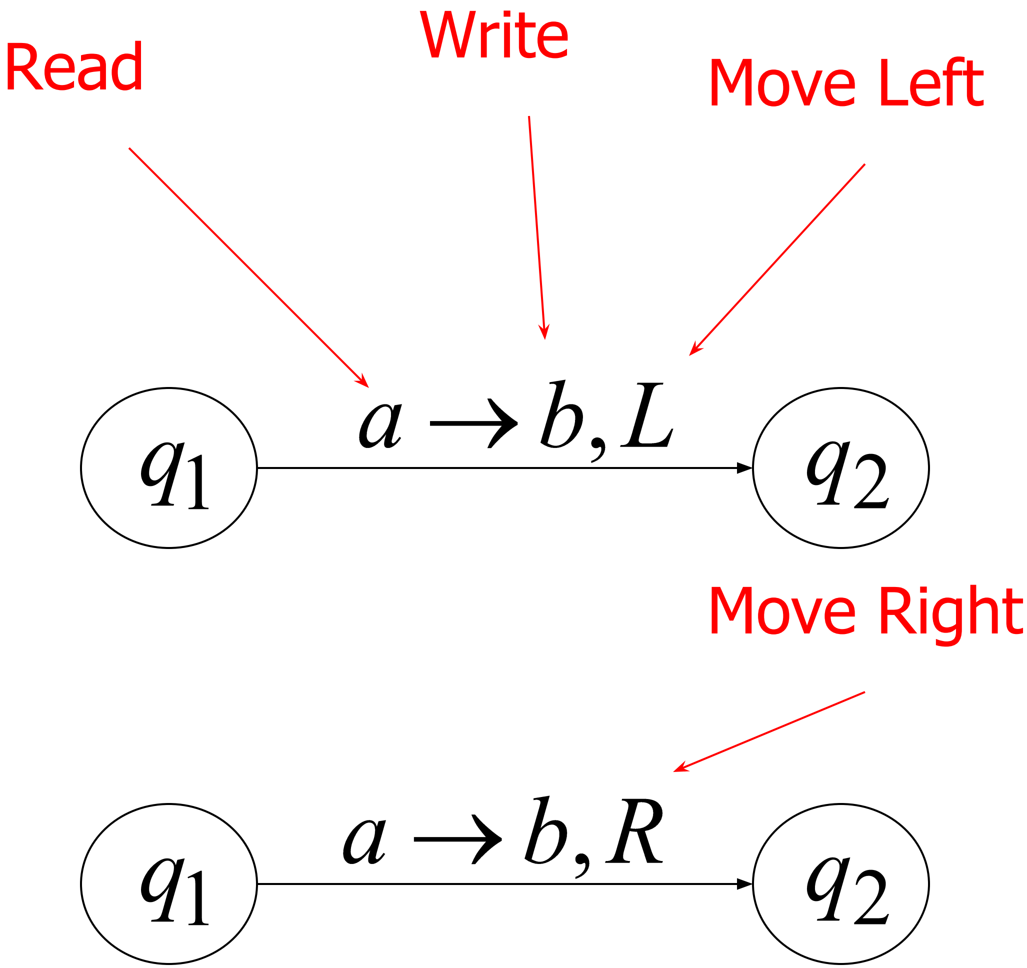
0	0	0	0	1	1				

A Thinking Machine
`Successor` Program

So the successor's output on 111101 was 000011 which is the reverse binary representation of 48.

Similarly, the successor of 127 should be 128:

States & Transitions

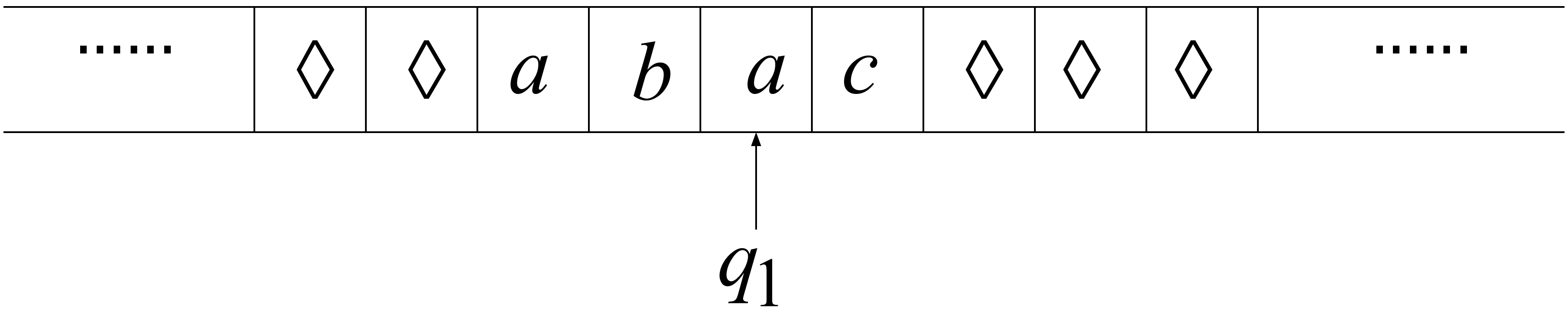


$T(q1,a)=(q2,b,L)$

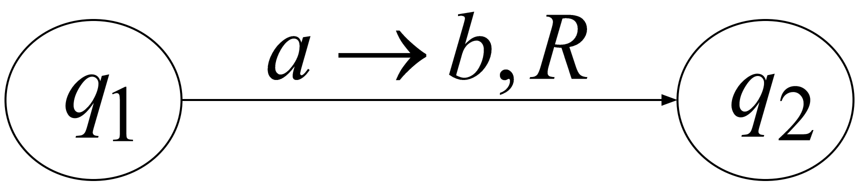
$T(q1,a)=(q2,b,R)$

Example:

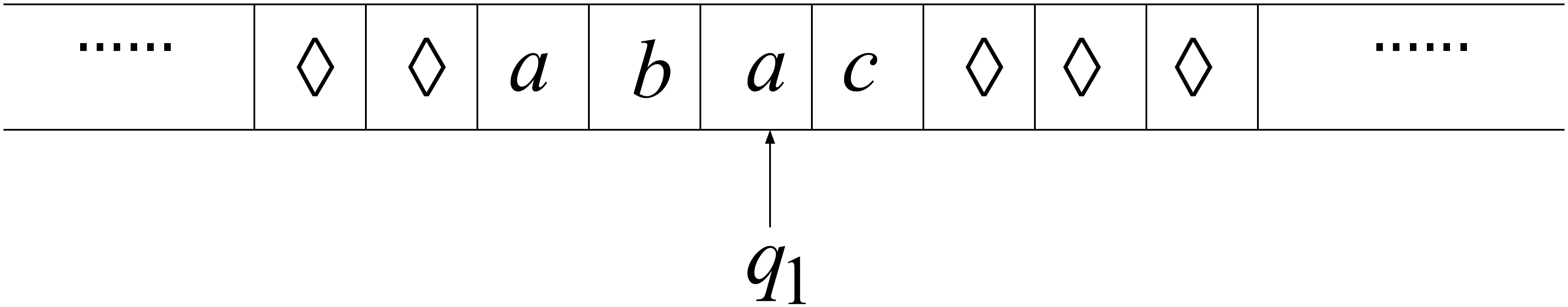
Time 1



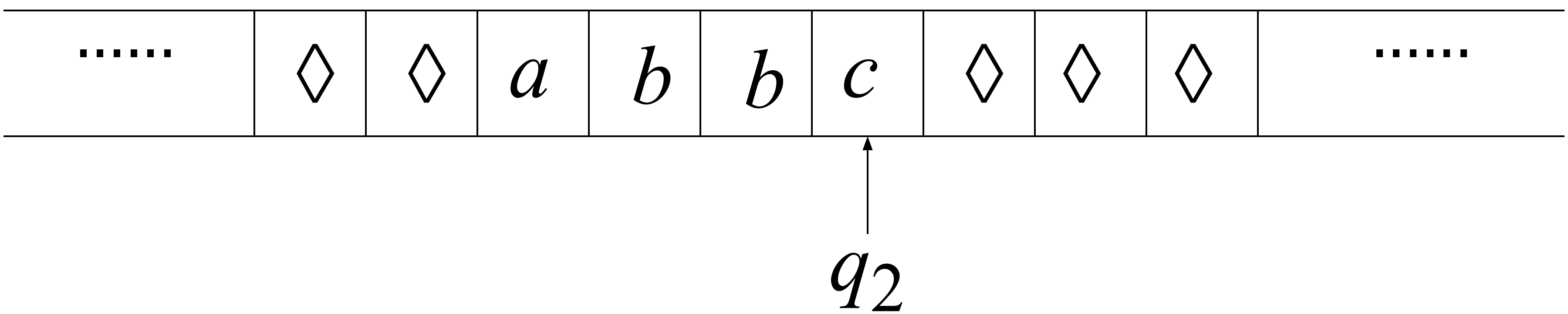
current state

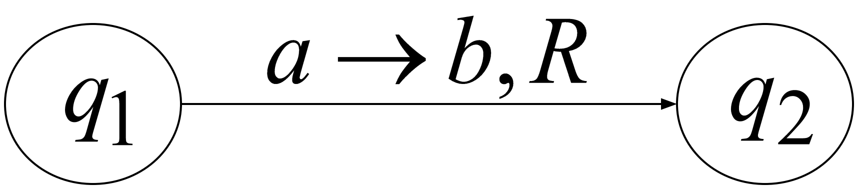


Time 1



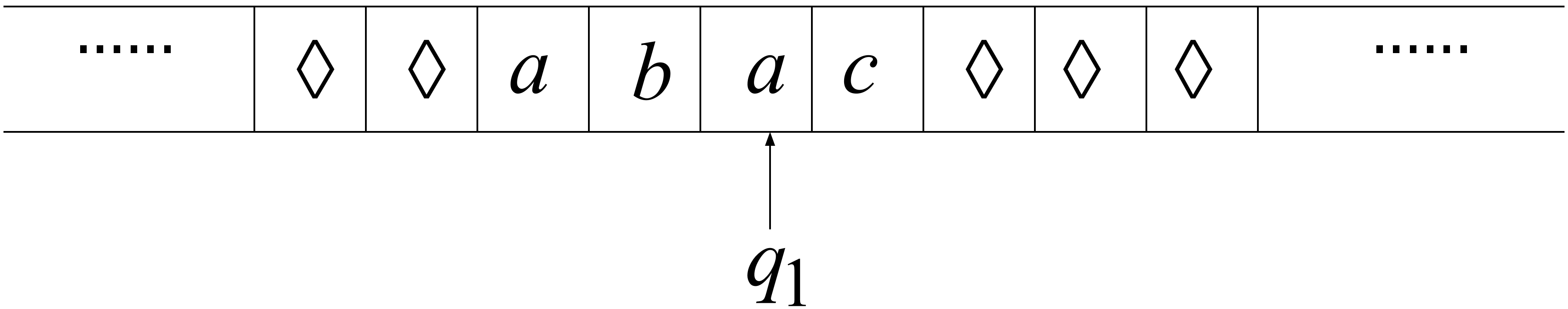
Time 2



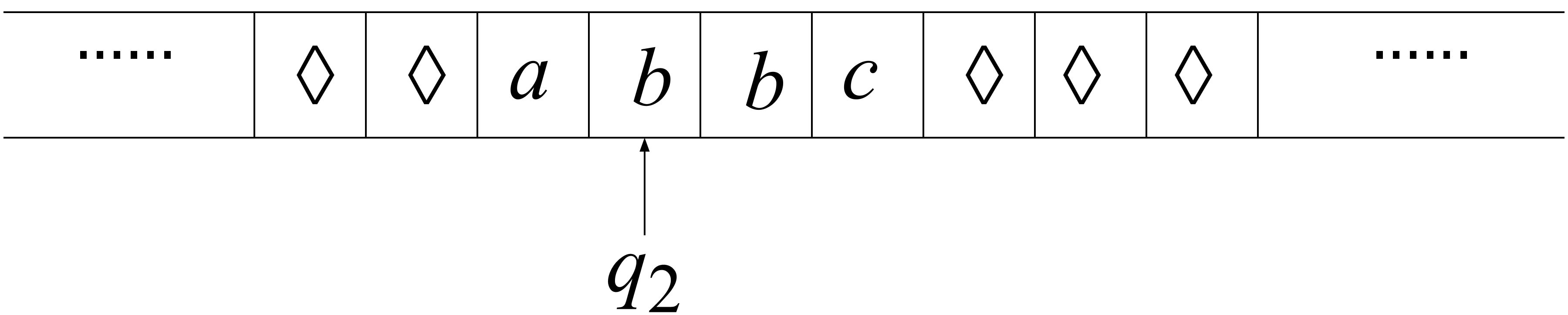


Example:

Time 1



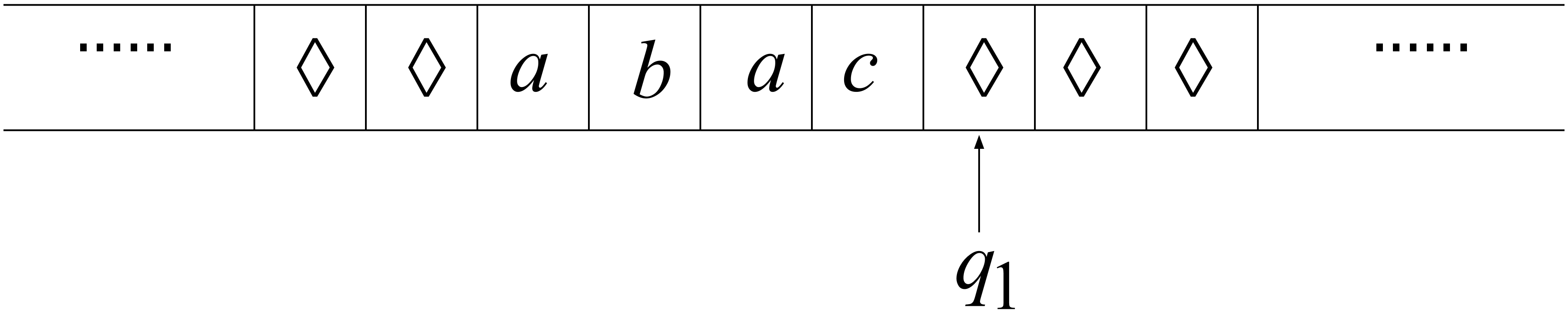
Time 2



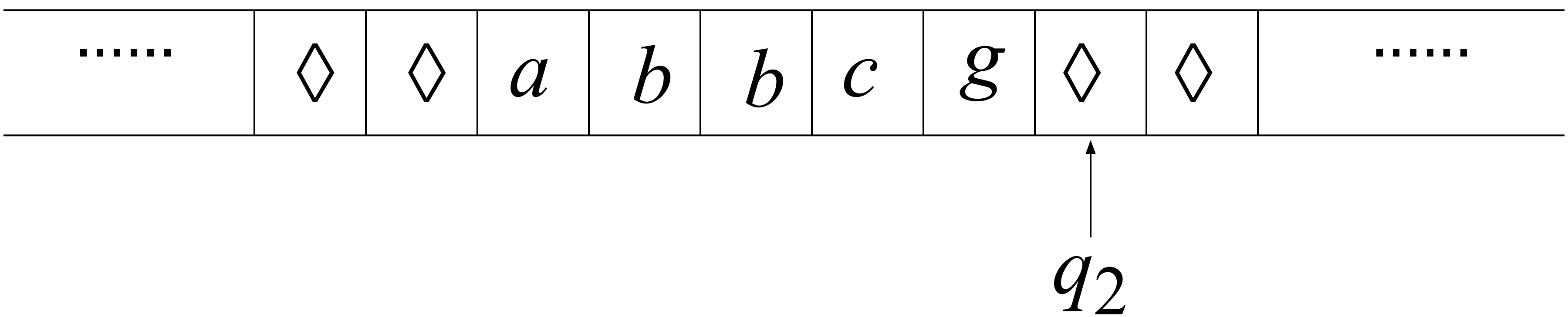
$$q_1 \xrightarrow{a \rightarrow b, L} q_2$$

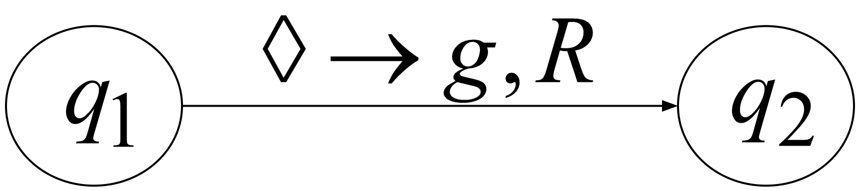
Example:

Time 1



Time 2

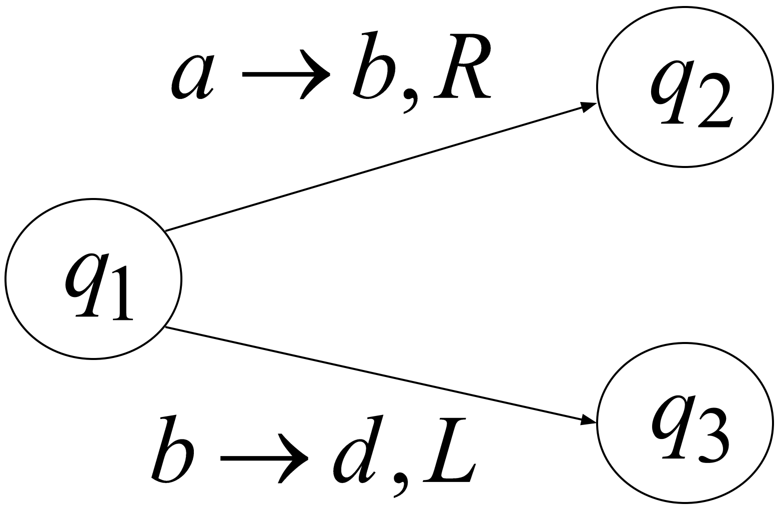




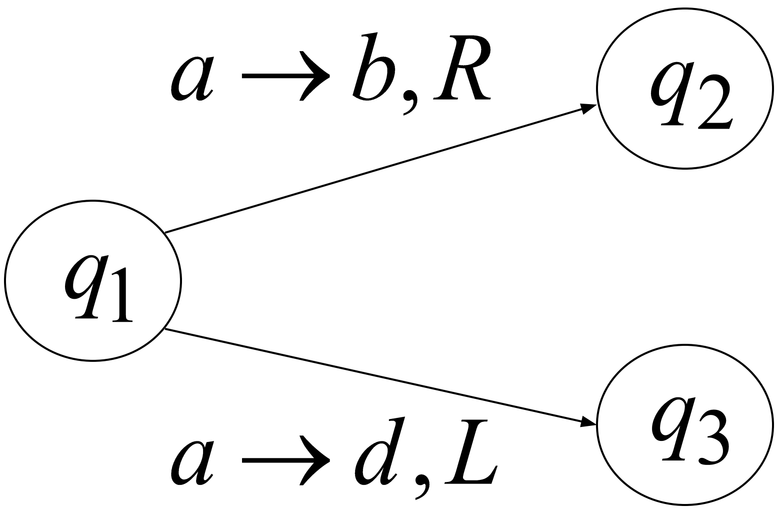
Determinism

Turing Machines are deterministic: for each state there is only one unique Transitions on each symbol.

Allowed



Not Allowed



TM Ex.   Parenthesis Checker

……	◊	(	(	)	(	(	)	(	)	◊	……

q_0

$T(q_0, () = (q_0, (, R)$

$T(q_0,)) = (q_1, X, L)$

$T(q_0, X) = (q_0, X, R)$

$T(q_0, \lozenge) = (q_2, \lozenge, L)$

$T(q_1, () = (q_0, X, R)$

$T(q_1, X) = (q_1, X, L)$

$T(q_1, \lozenge) = (H, N, -)$

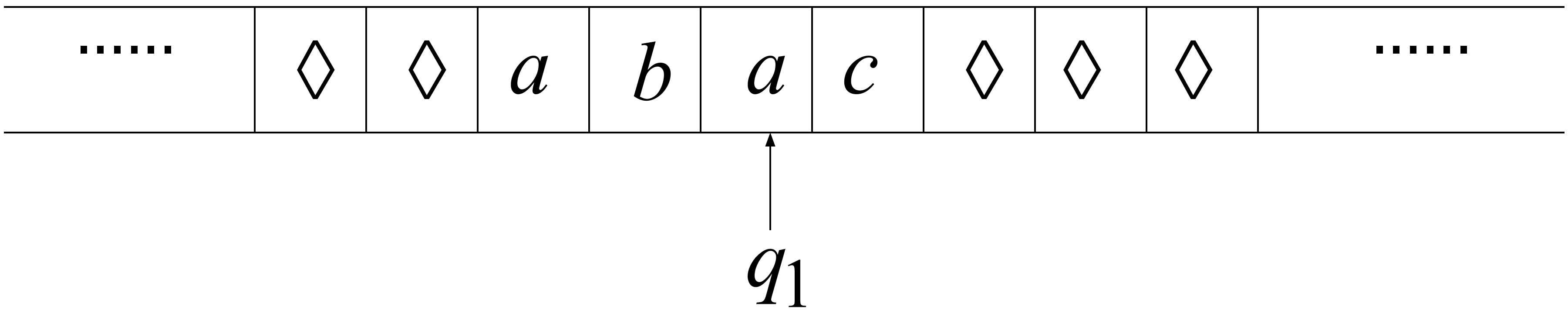
$T(q_2, X) = (q_2, X, L)$

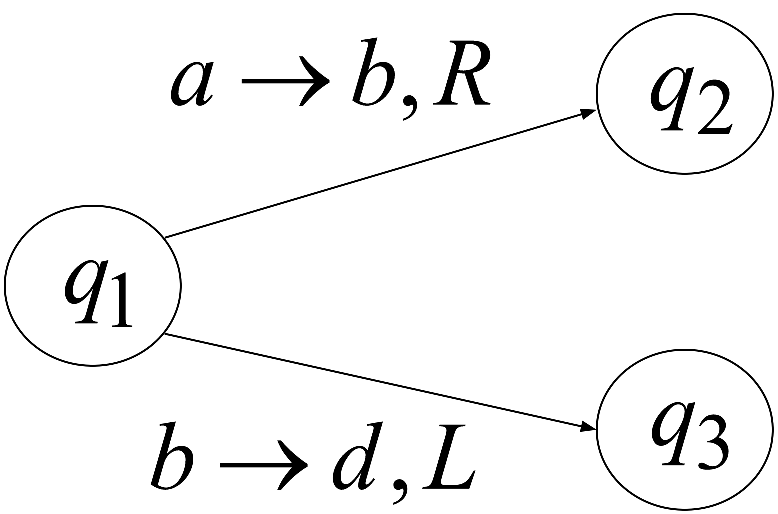
$T(q_2, \lozenge) = (H, Y, -)$

$T(q_2, () = (H, N, -)$

Example:

Partial Transition Function





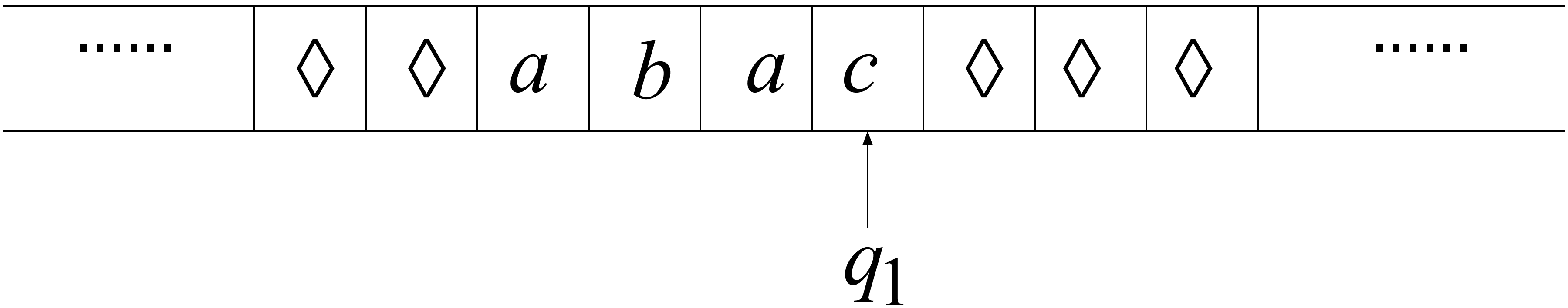
Allowed:

No transition
for input symbol c

Halting

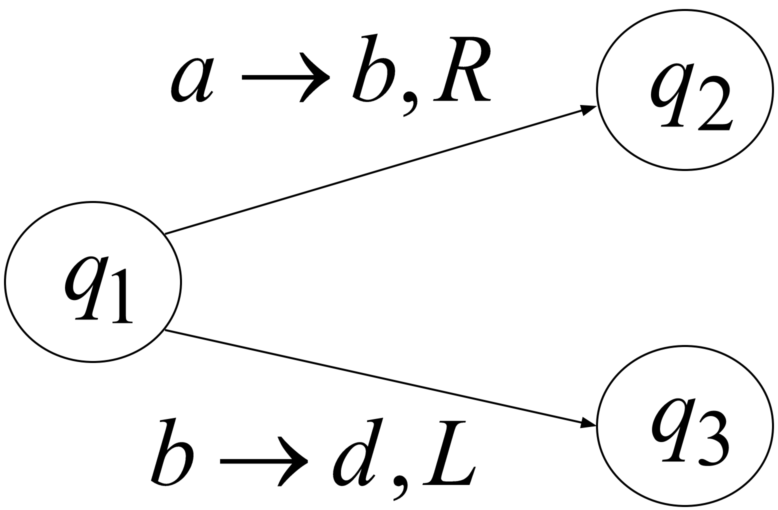
The machine ***halts*** if there are no possible transitions to follow

Example:

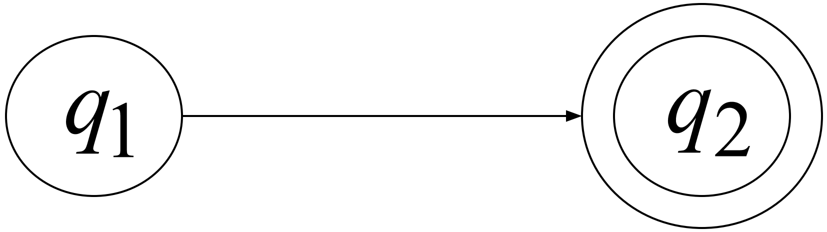


No possible transition

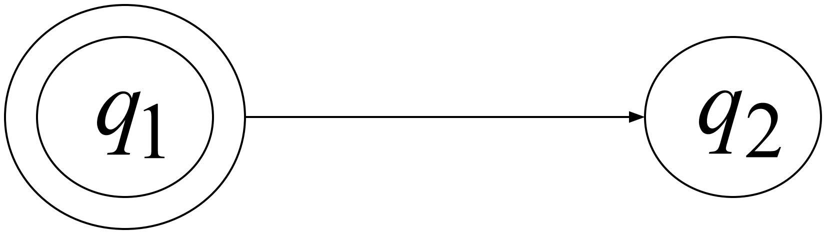
HALT!!!



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

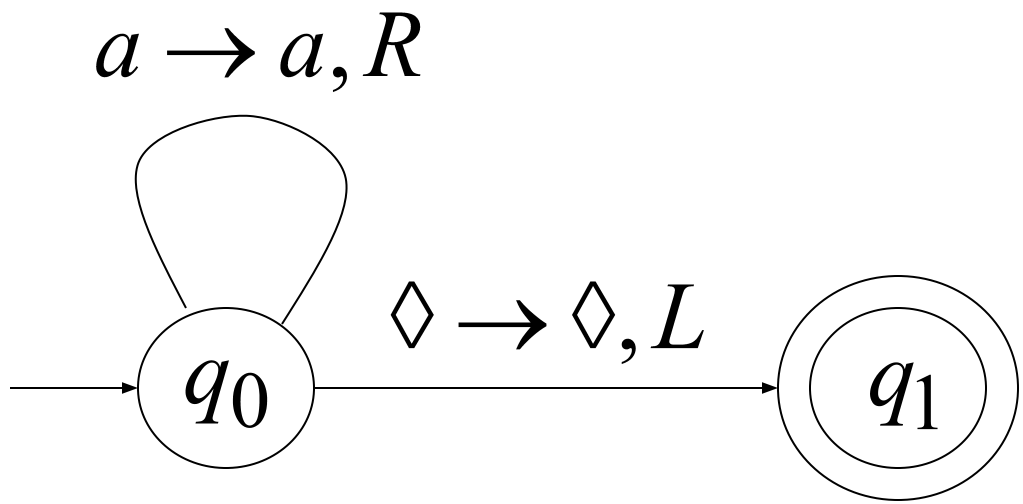
or

If machine enters an *infinite loop*

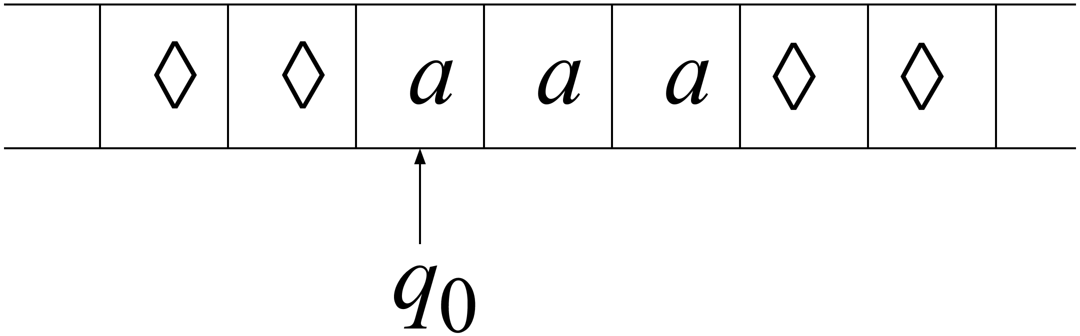
Example

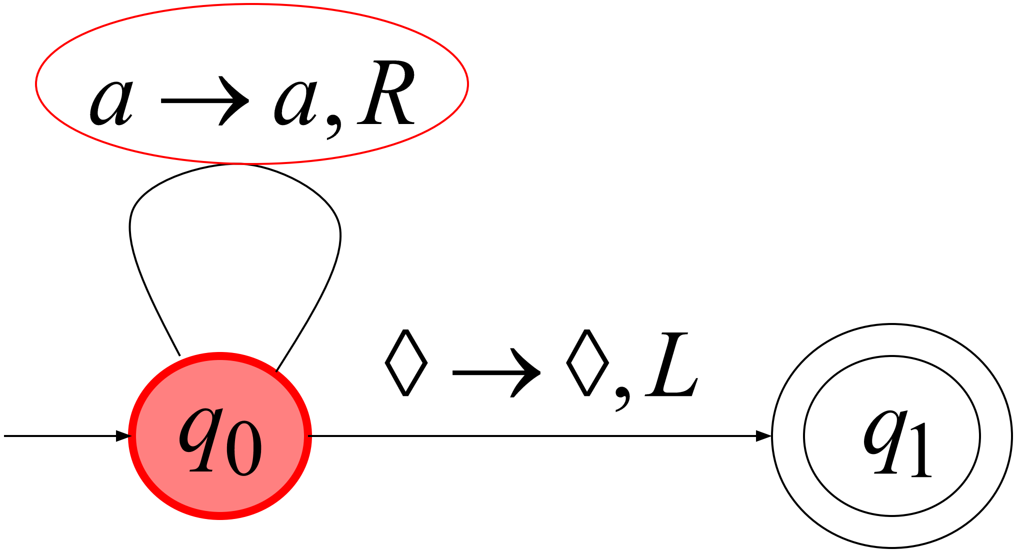
A Turing machine that accepts the language:

$$aa^*$$

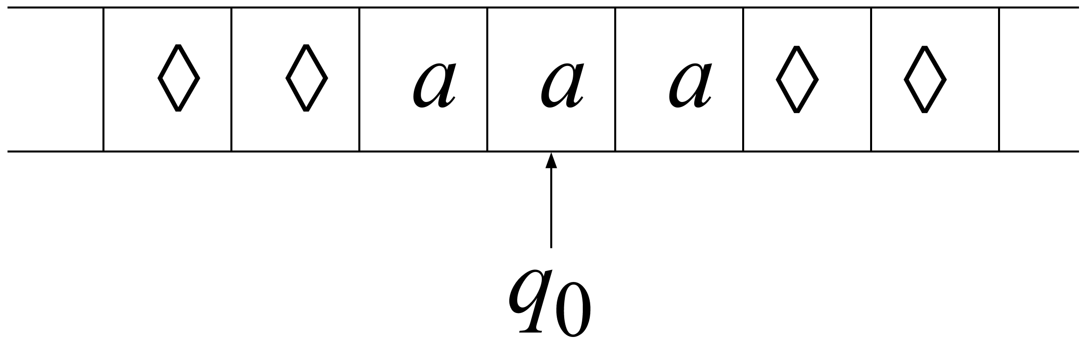


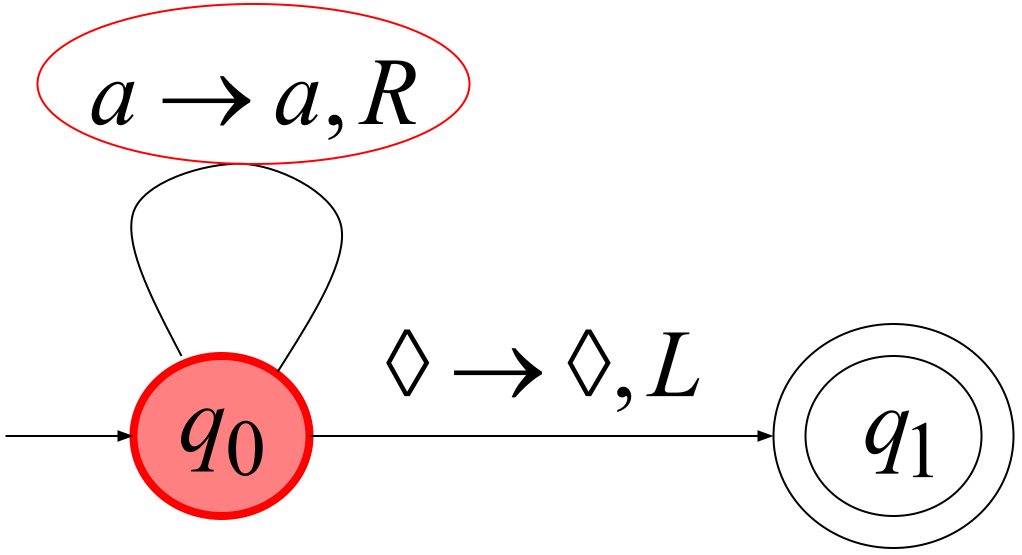
Time 0



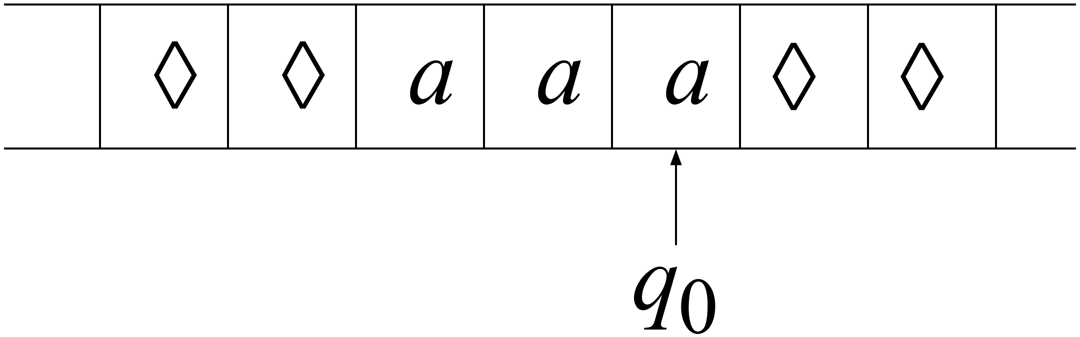


Time 1

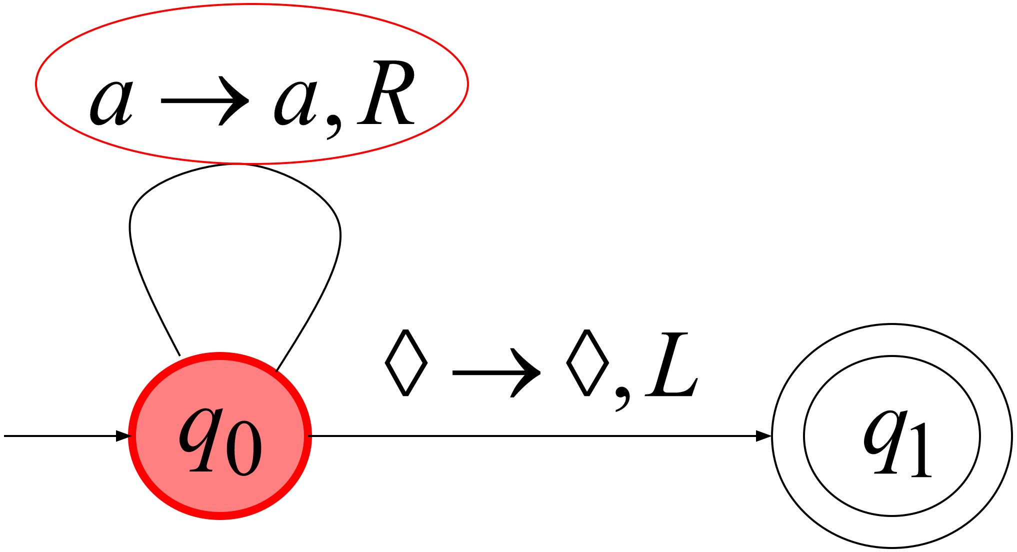




Time 2

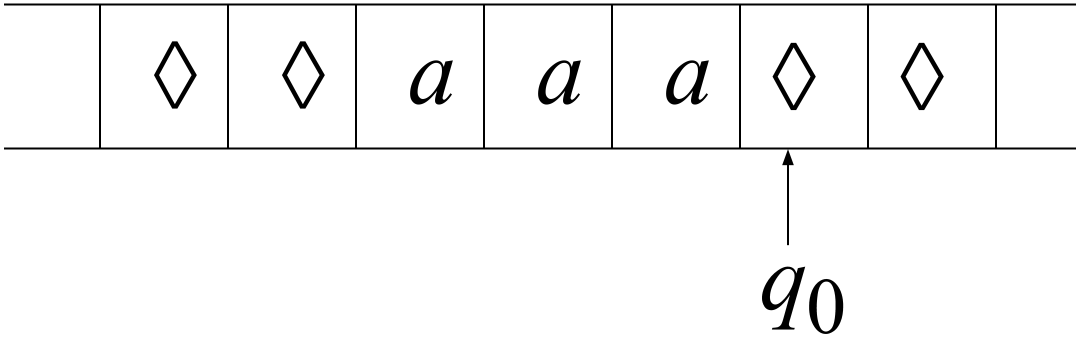


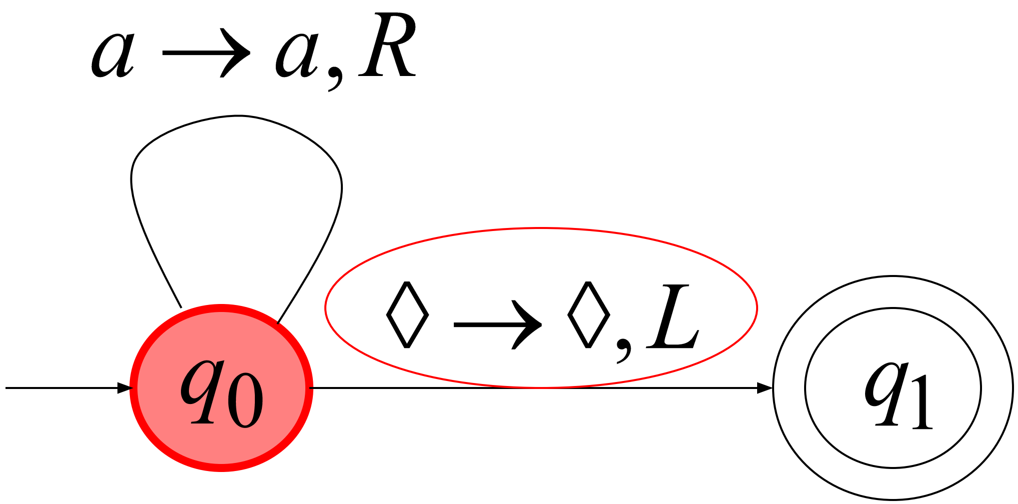
$\Diamond$
$\Diamond$
a
a
a
$\Diamond$
$\Diamond$
q_0



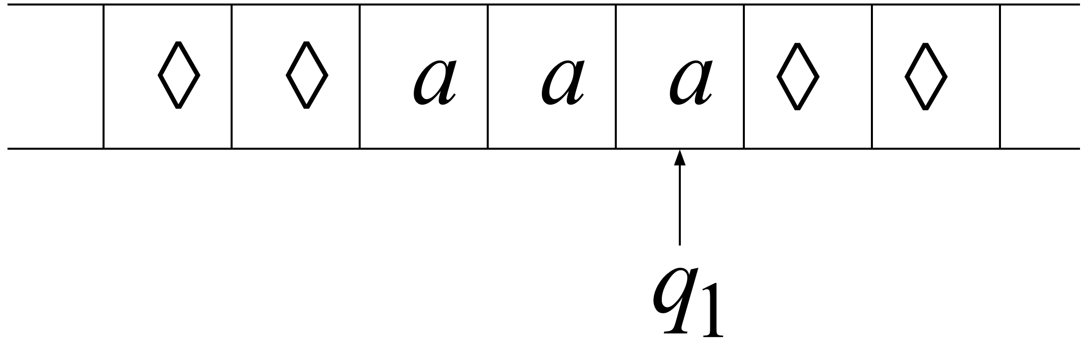
$a \rightarrow a, R$
$\Diamond \rightarrow \Diamond, L$
q_0
q_1

Time 3

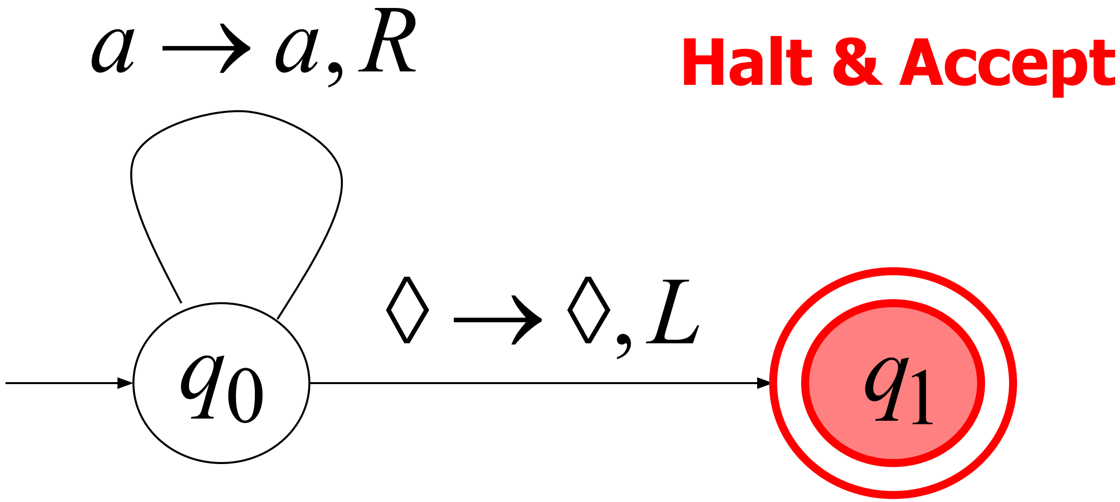




Time 4



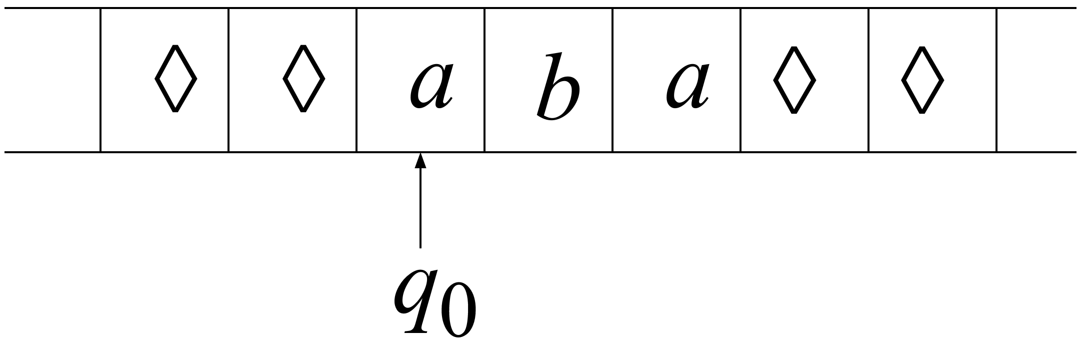
$\Diamond$ $\Diamond$ a a a $\Diamond$ $\Diamond$
q_1

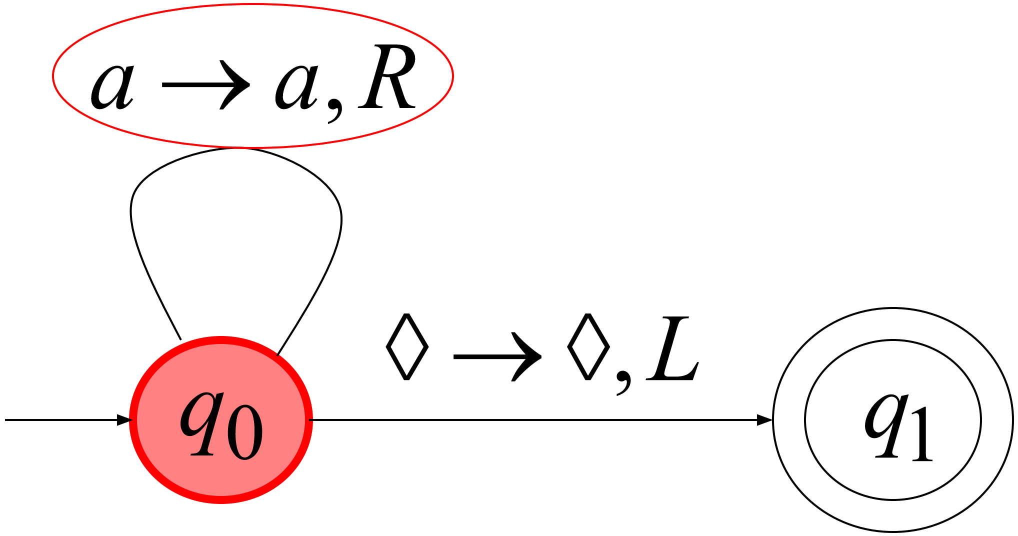


$a \rightarrow a, R$
Halt & Accept
$\Diamond \rightarrow \Diamond, L$
q_0
q_1

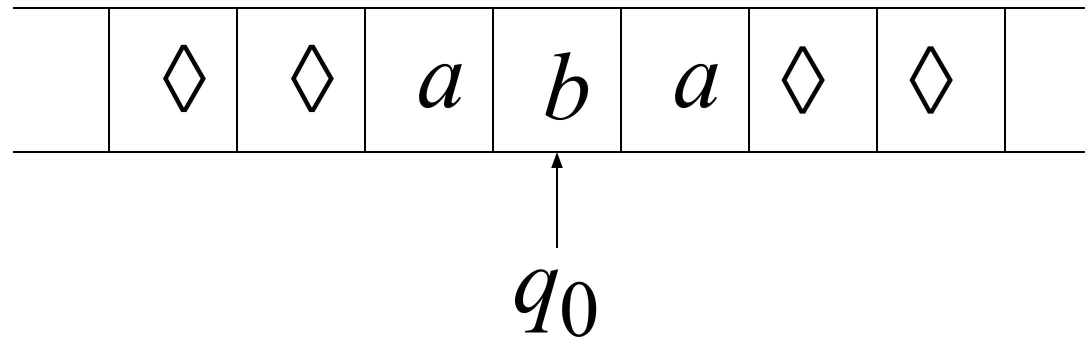
Rejection Example

Time 0



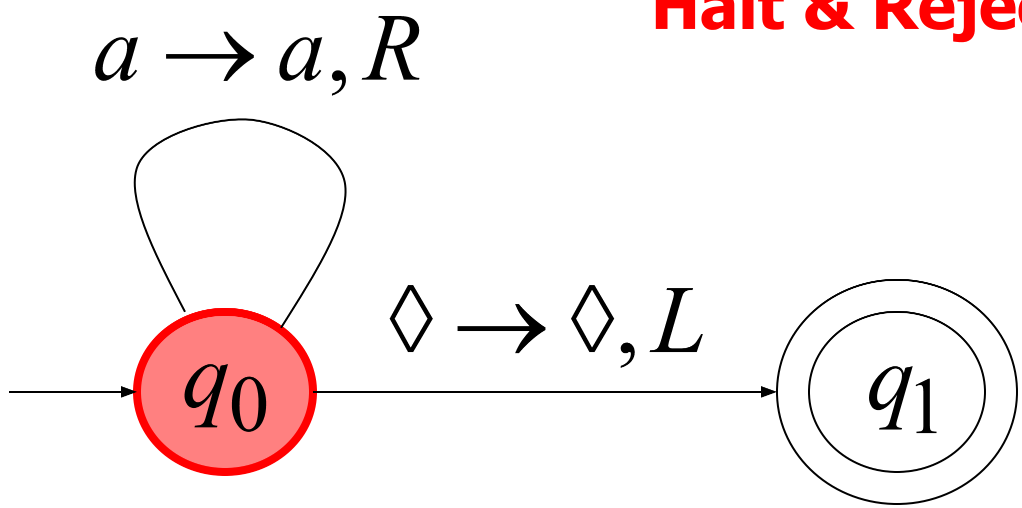


Time 1



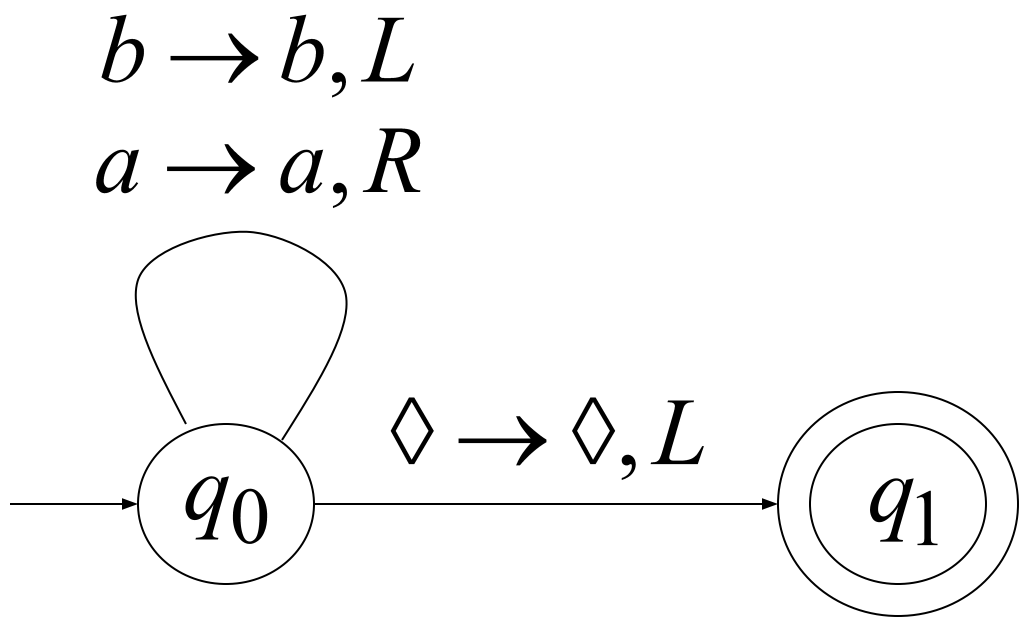
No possible Transition

Halt & Reject

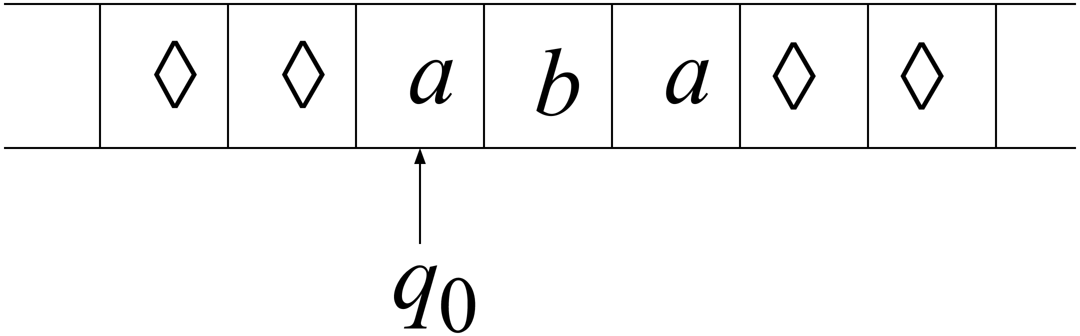


Infinite Loop Example

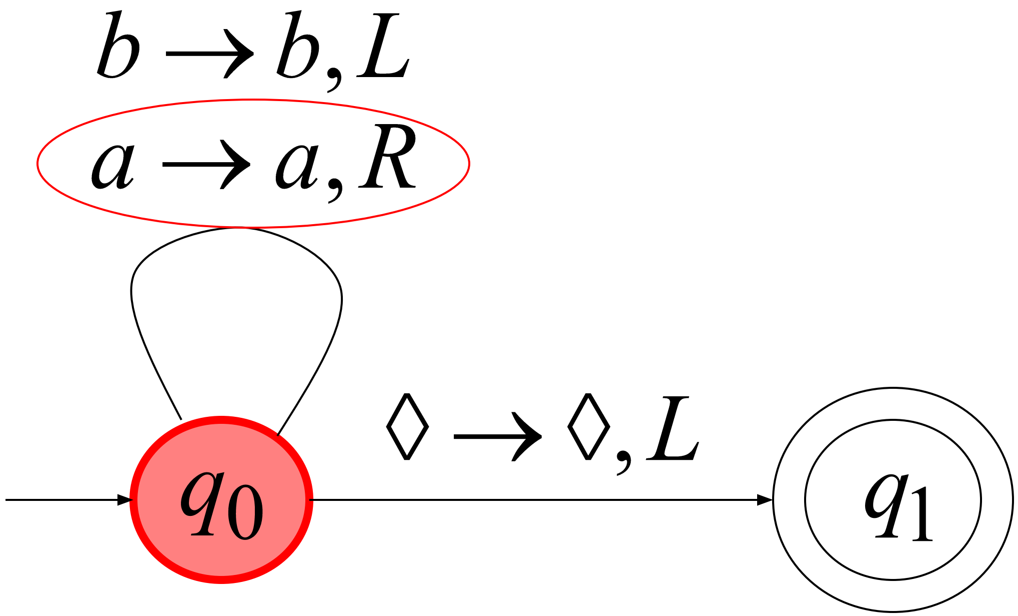
A Turing machine for language $aa^* + b(a+b)^*$



Time 0

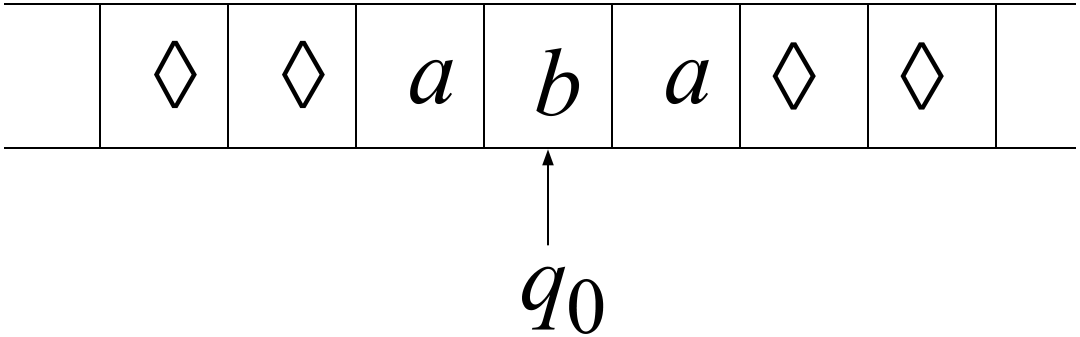


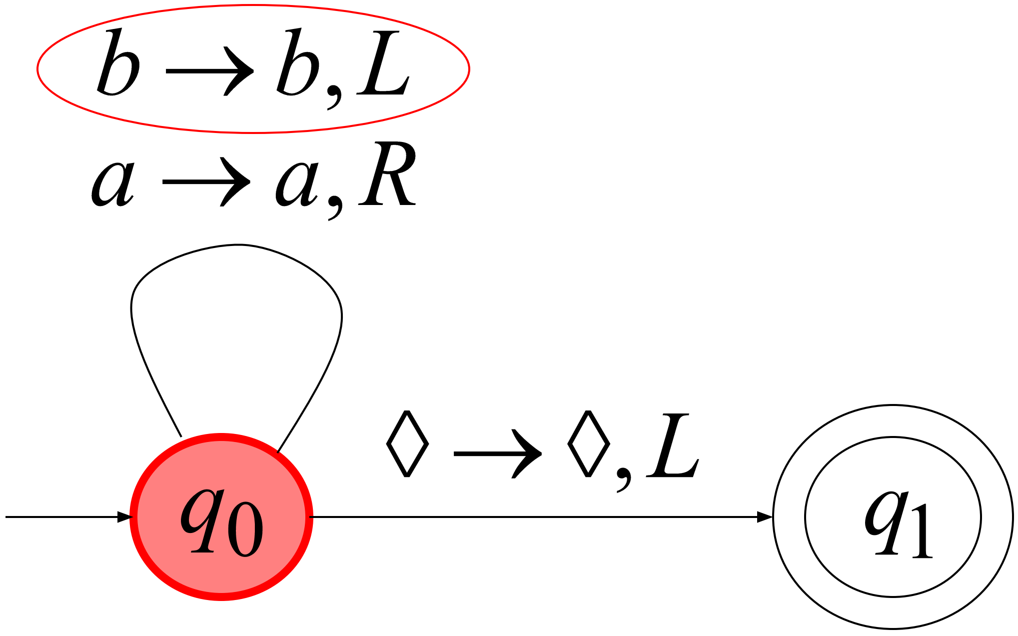
◊
◊
a
b
a
◊
◊
q_0



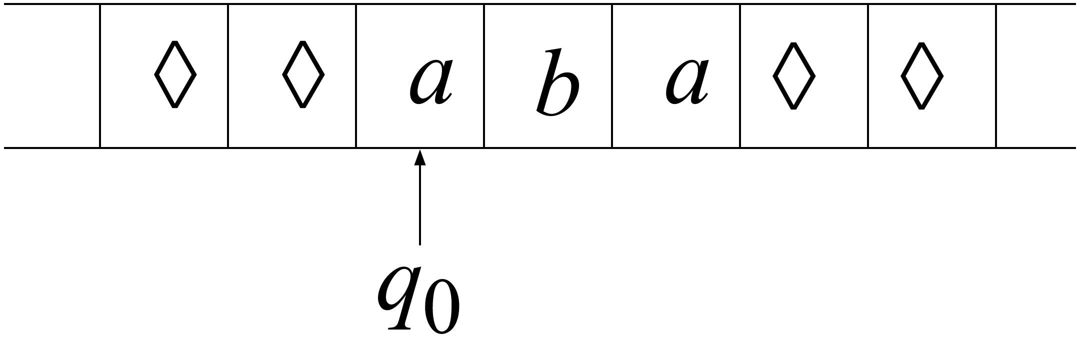
$b \rightarrow b, L$
$a \rightarrow a, R$
$\Diamond \rightarrow \Diamond, L$
q_0
q_1

Time 1

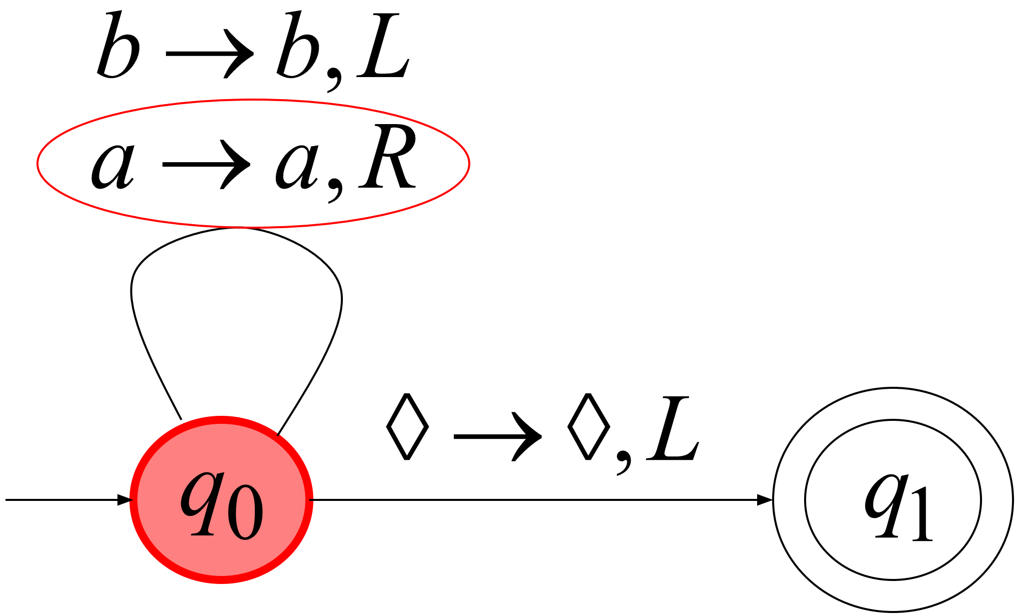




Time 2

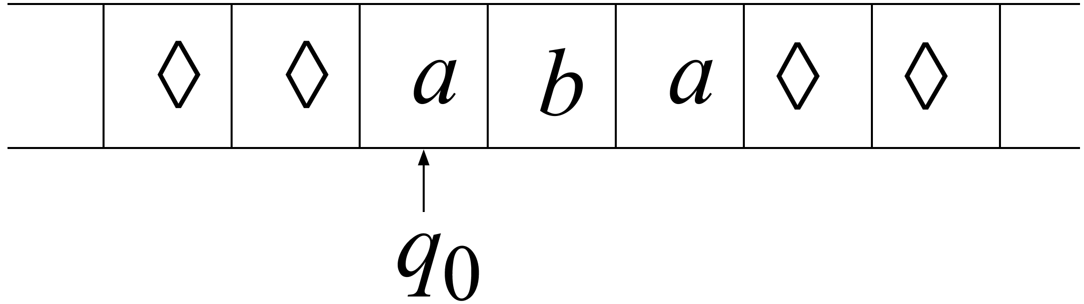


◊ ◊ a b a ◊ ◊
q_0

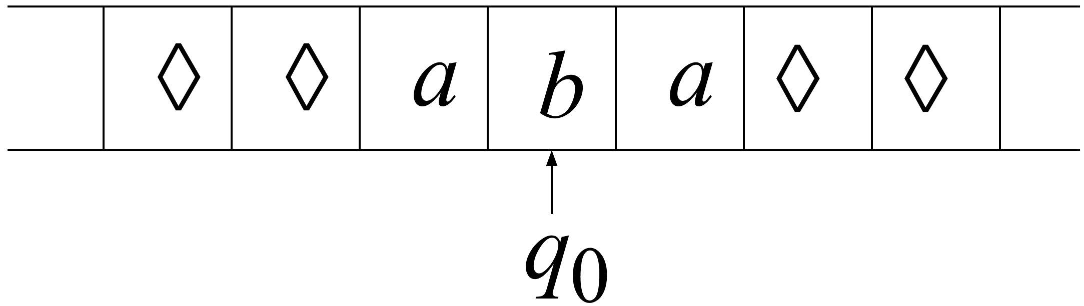


$b \rightarrow b, L$
$a \rightarrow a, R$
$\Diamond \rightarrow \Diamond, L$
q_0
q_1

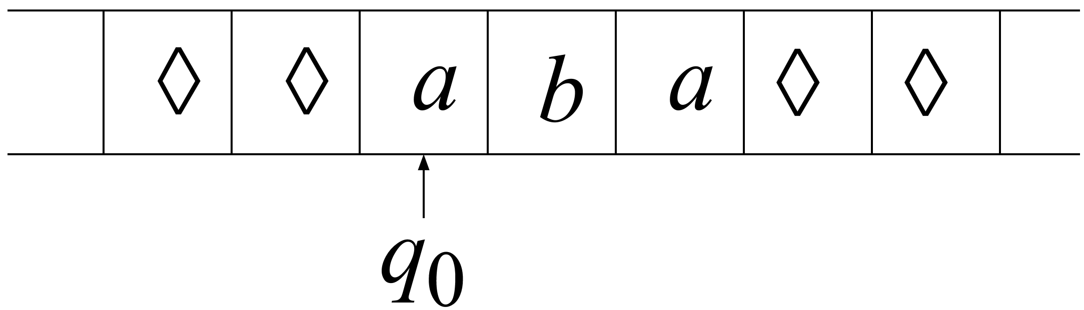
Time 2



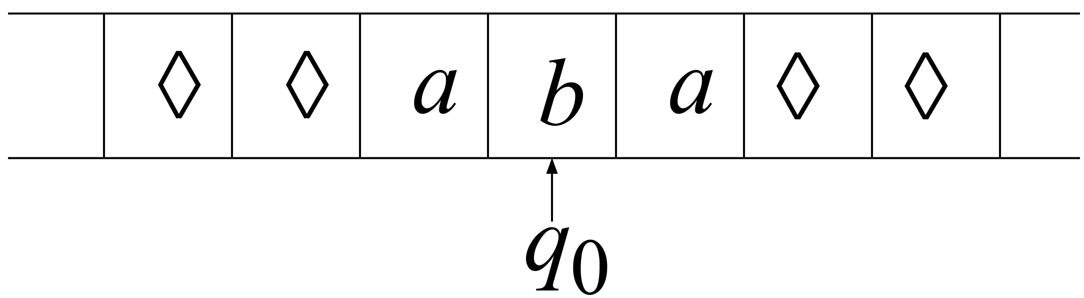
Time 3



Time 4



Time 5



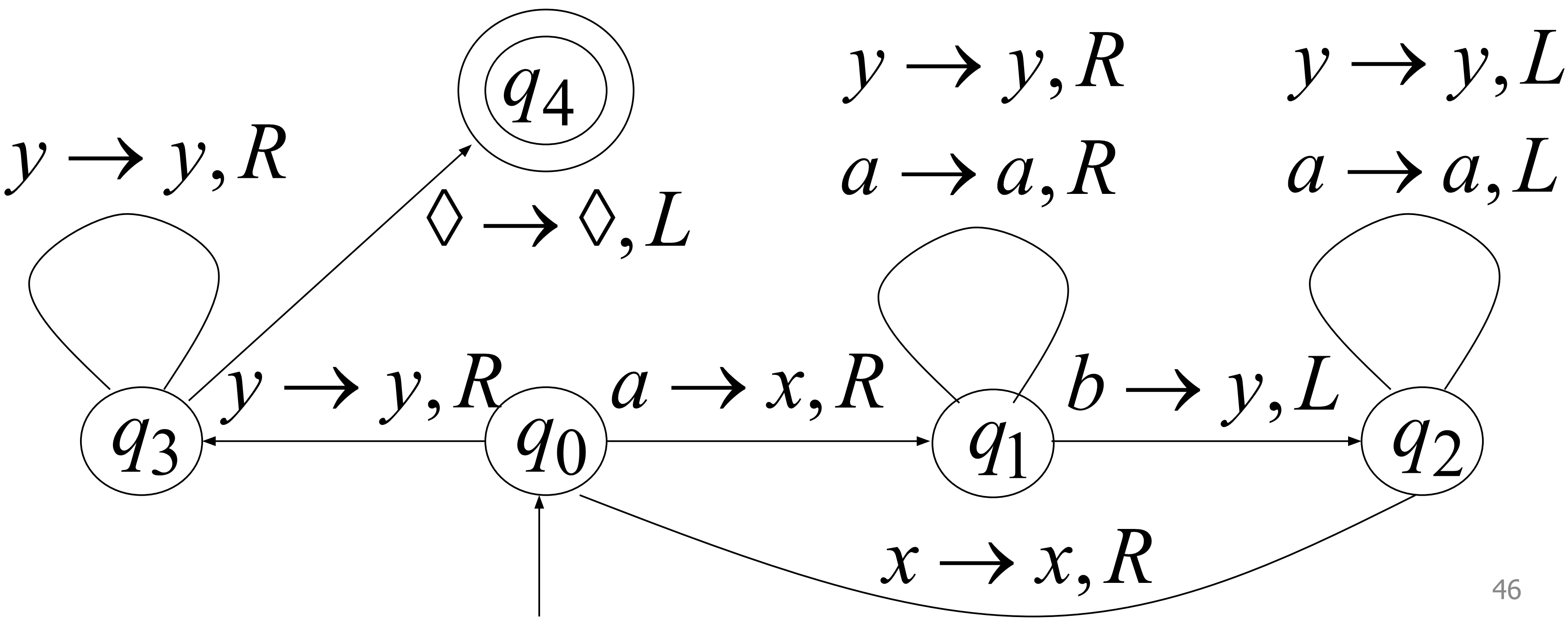
Infinite loop

Because of the infinite loop:

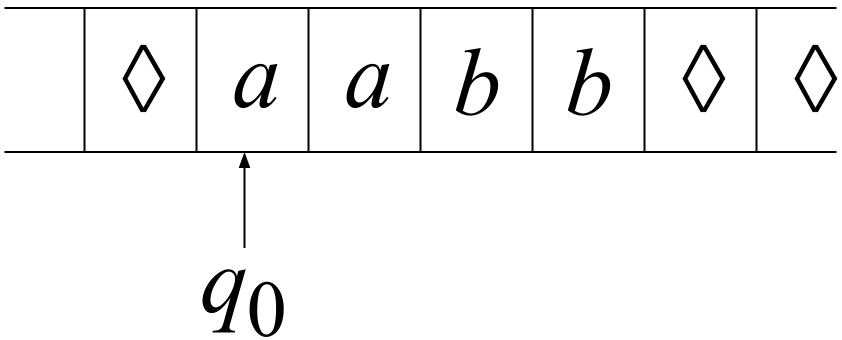
- The final state cannot be reached
- The machine never halts
- The input is not accepted

Another Turing Machine Example

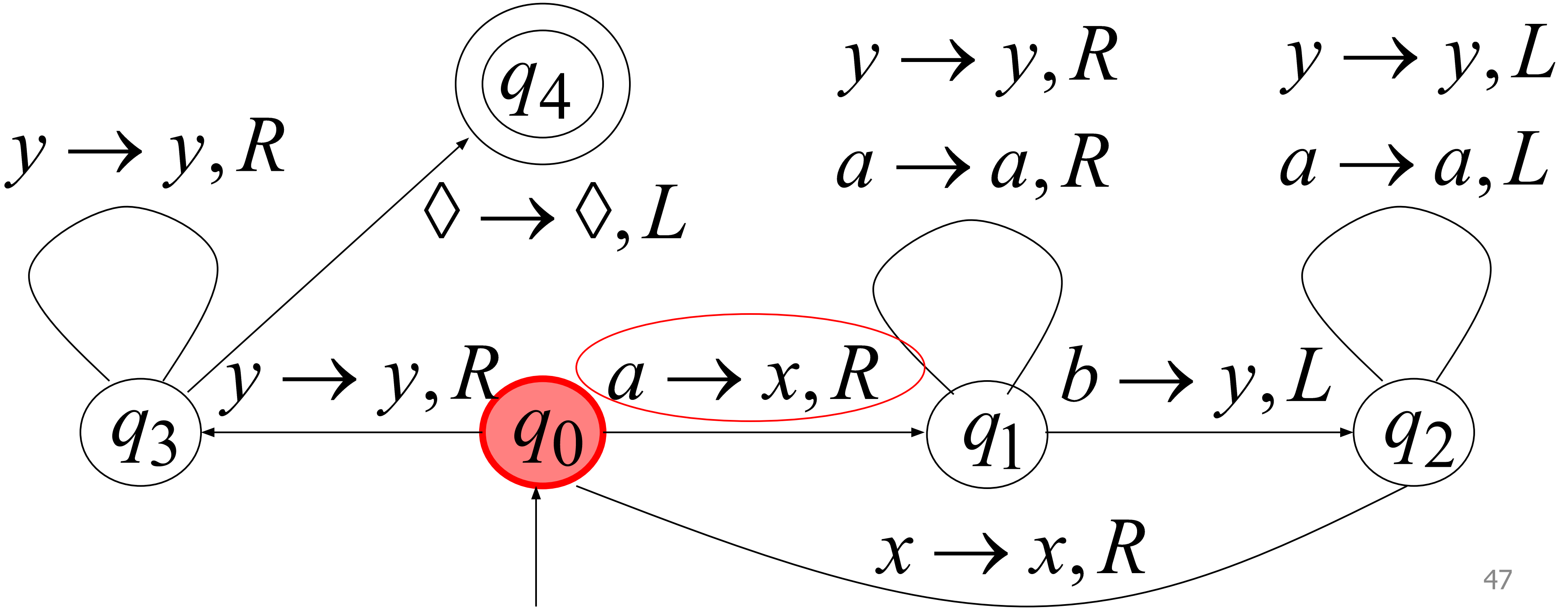
Turing machine for the language $\{a^n b^n\}$



Time 0

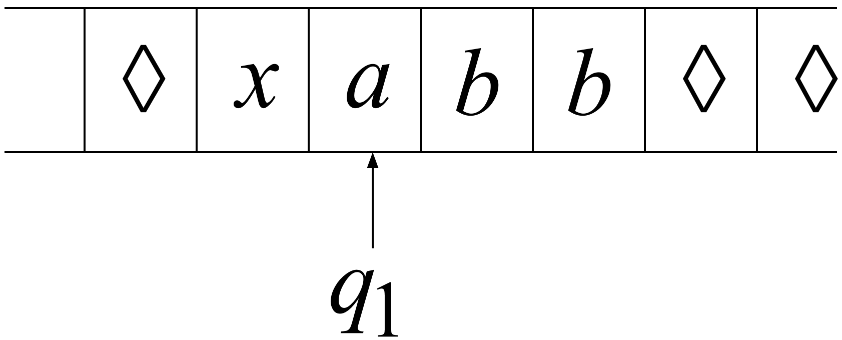


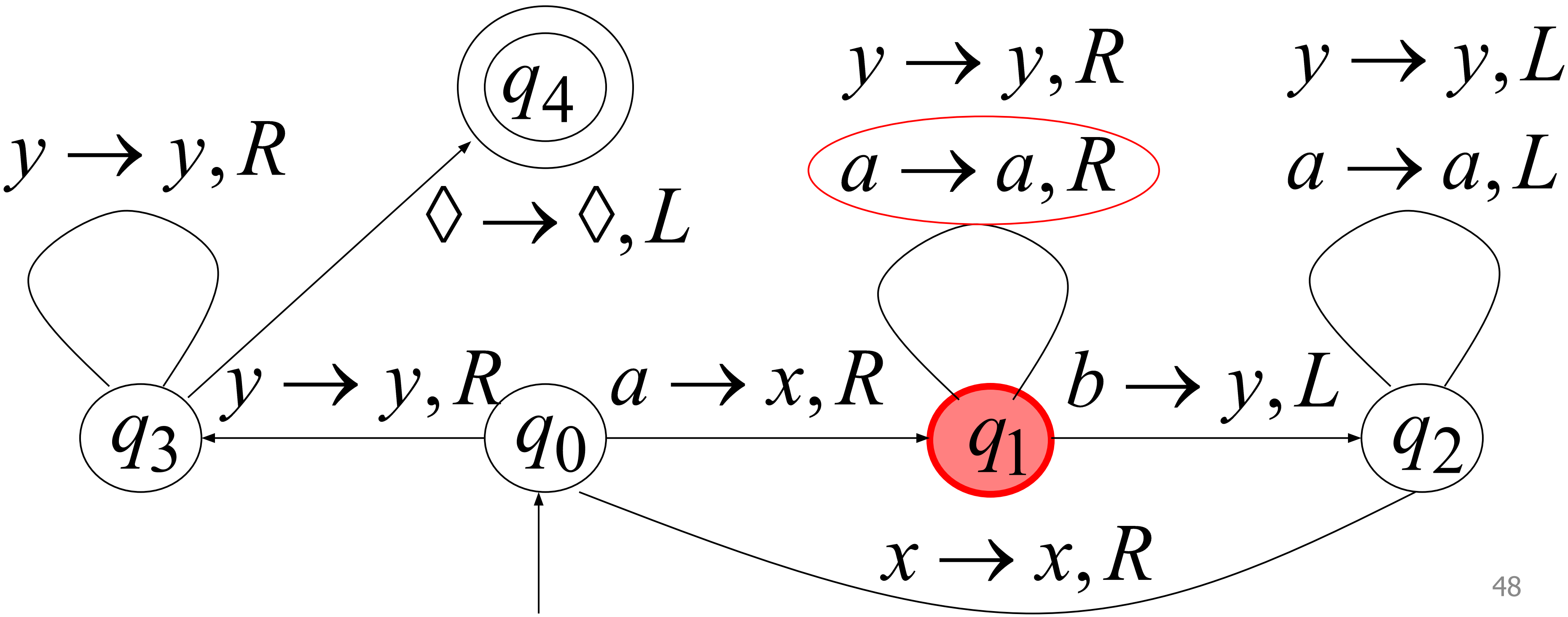
◊
a
a
b
b
◊
◊
q_0



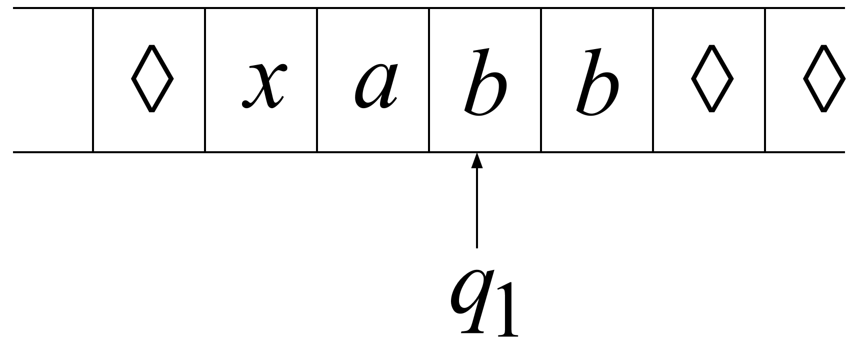
q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

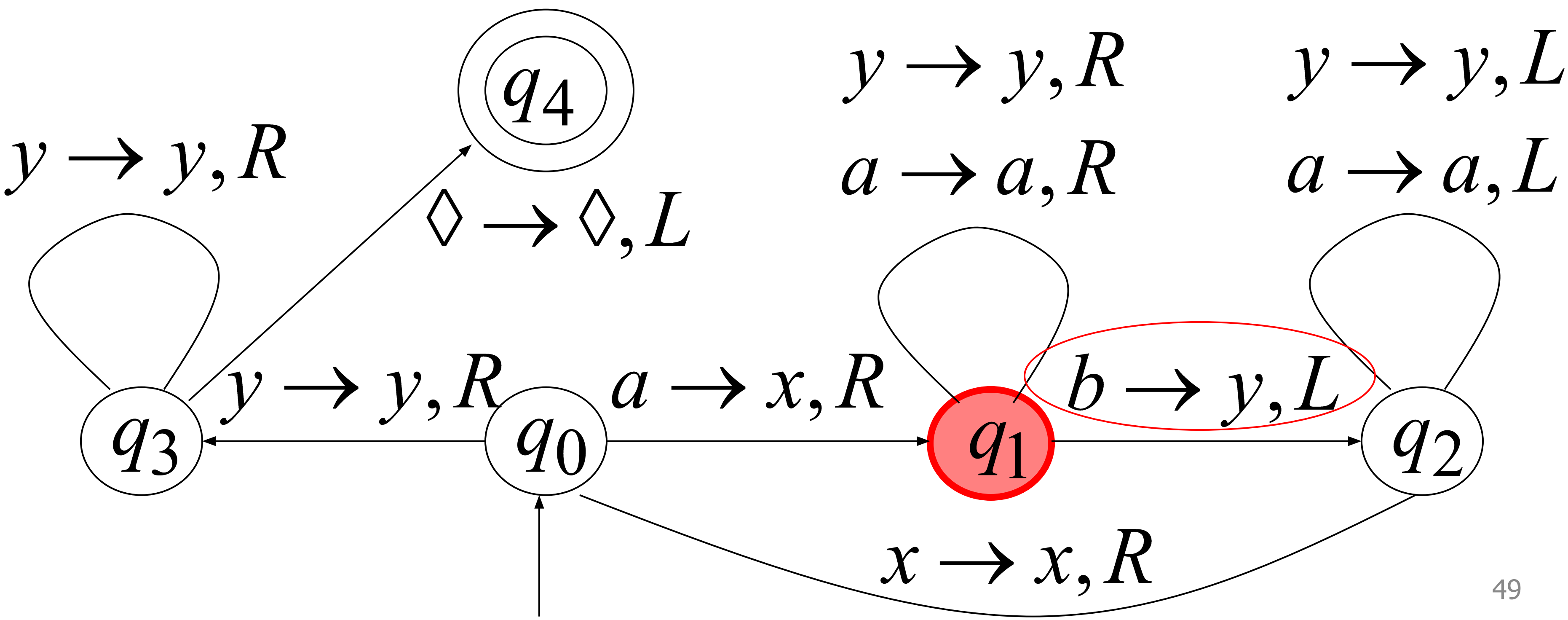
Time 1


◊
x
a
b
b
◊
◊
q_1

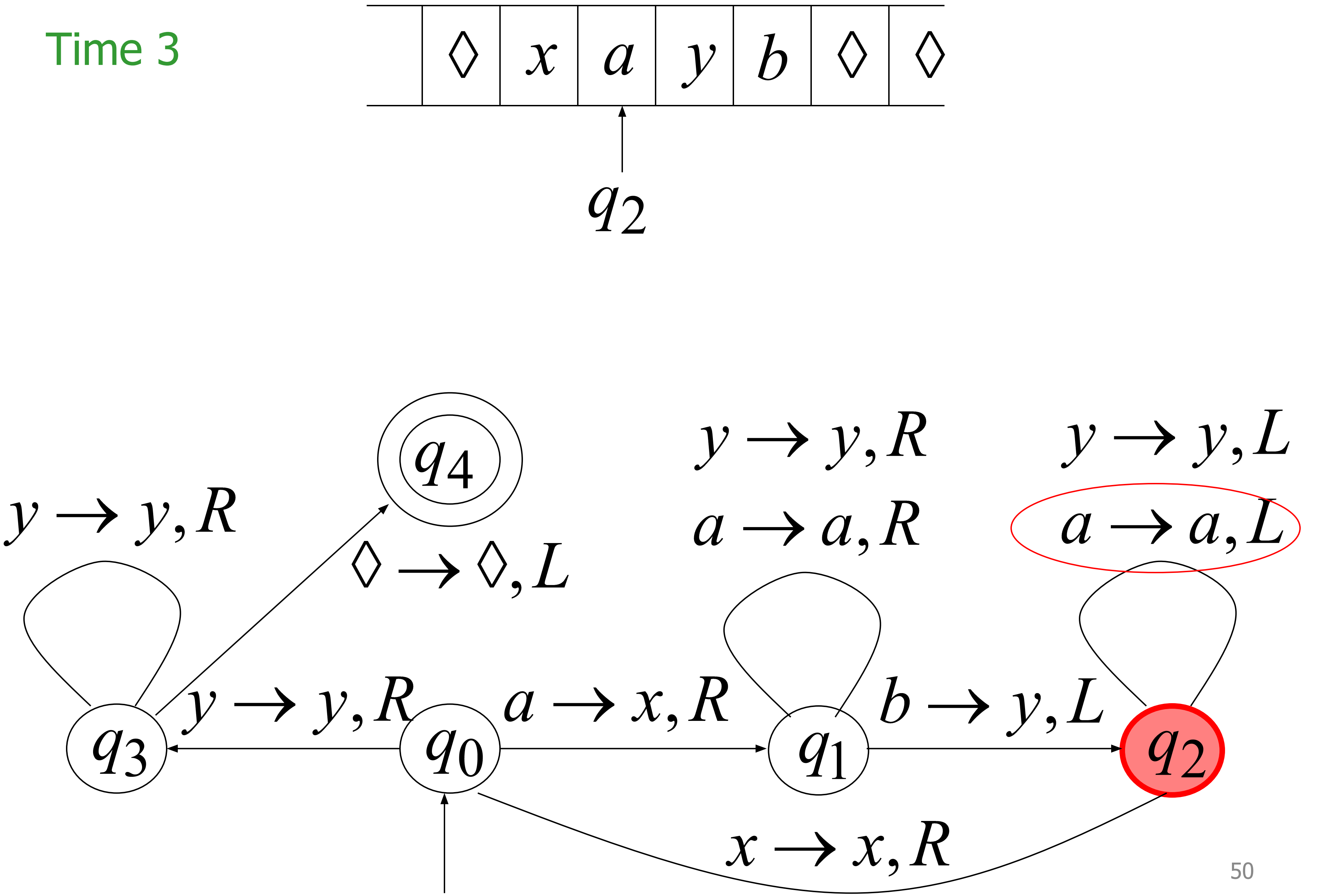

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
q_3
q_0
q_1
q_2
$x \to x, R$

Time 2


◊
x
a
b
b
◊
◊
q_1

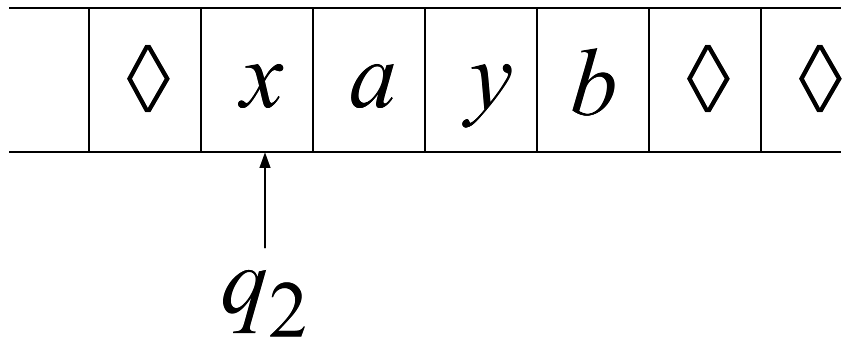

q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

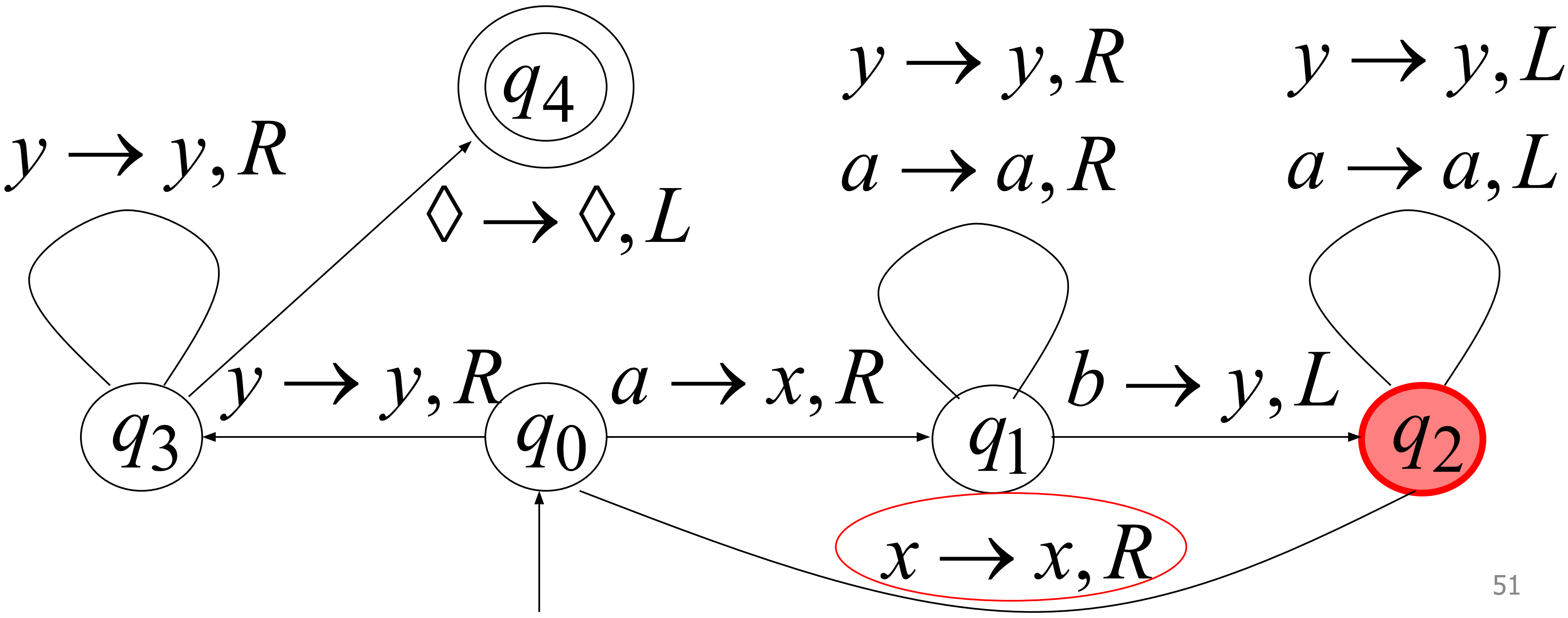
Time 3



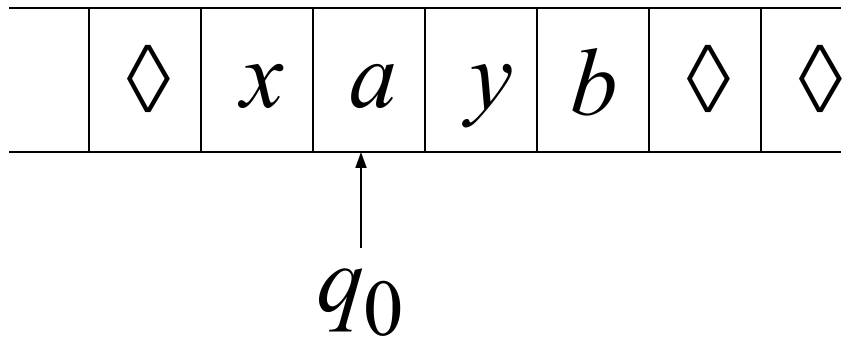
◊
x
a
y
b
◊
◊
q_2
$y \to y, R$
q_4
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
$y \to y, R$
$a \to x, R$
$b \to y, L$
q_3
q_0
q_1
q_2
$x \to x, R$

Time 4

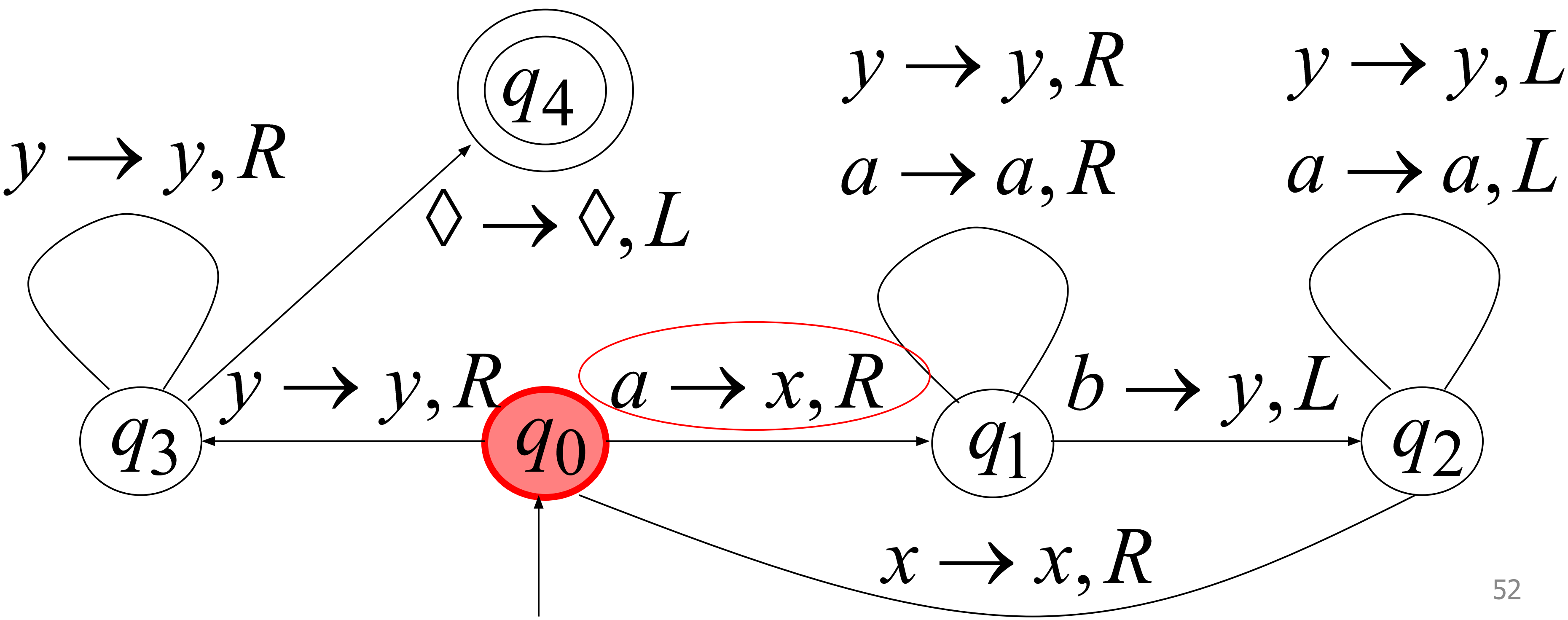

◊
x
a
y
b
◊
◊
q_2


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

Time 5

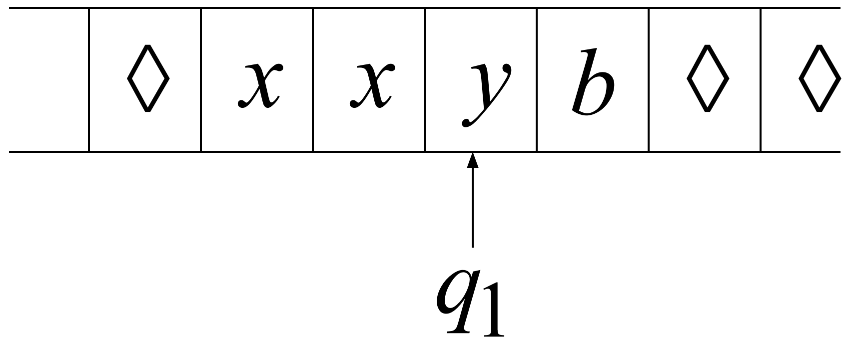


$\Diamond$ x a y b $\Diamond$ $\Diamond$
q_0

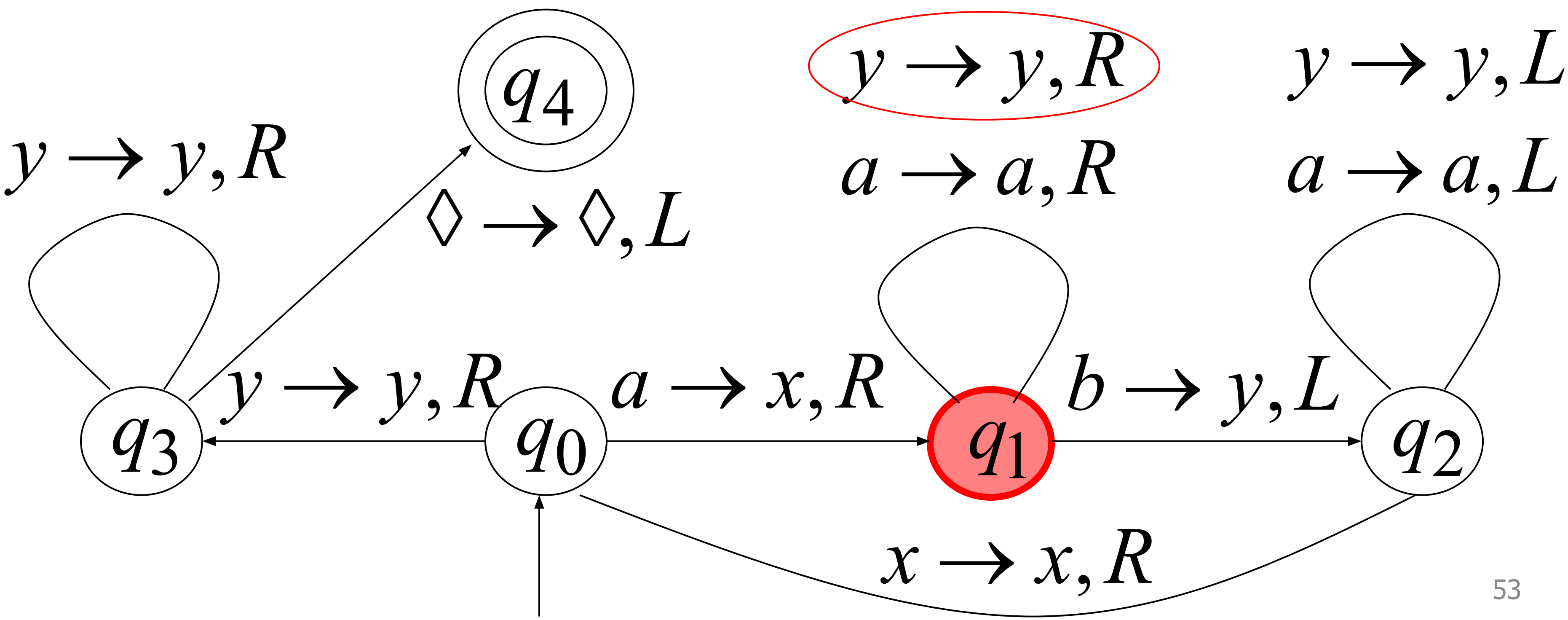


q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 6

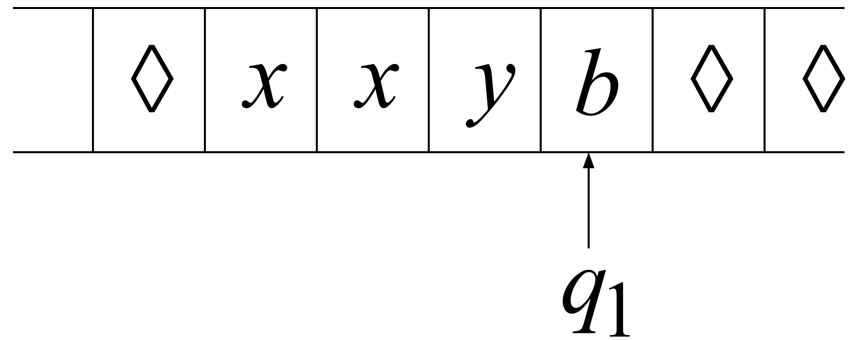


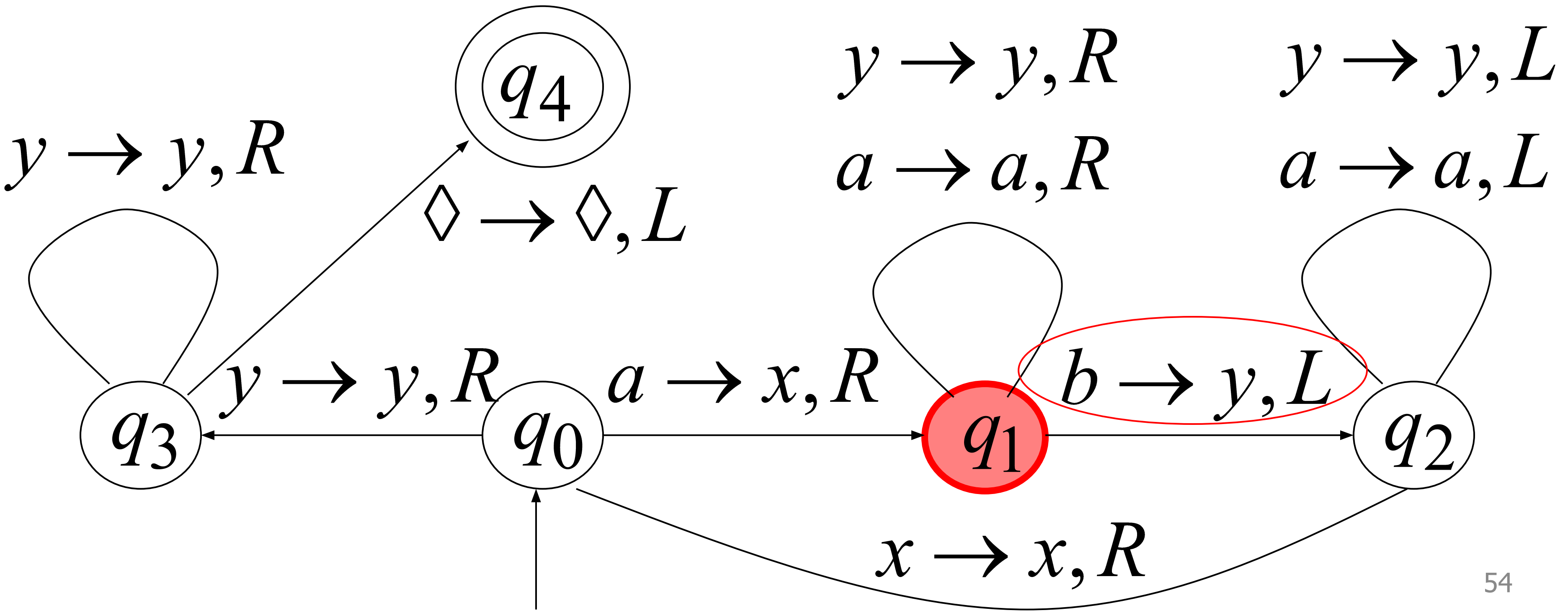
◊
x
x
y
b
◊
◊
q_1



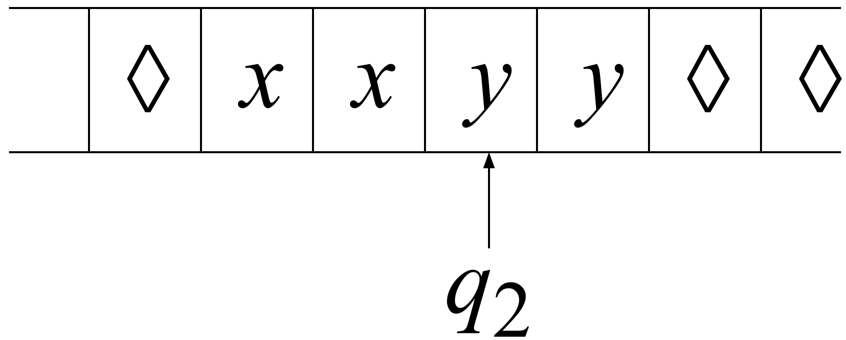
q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

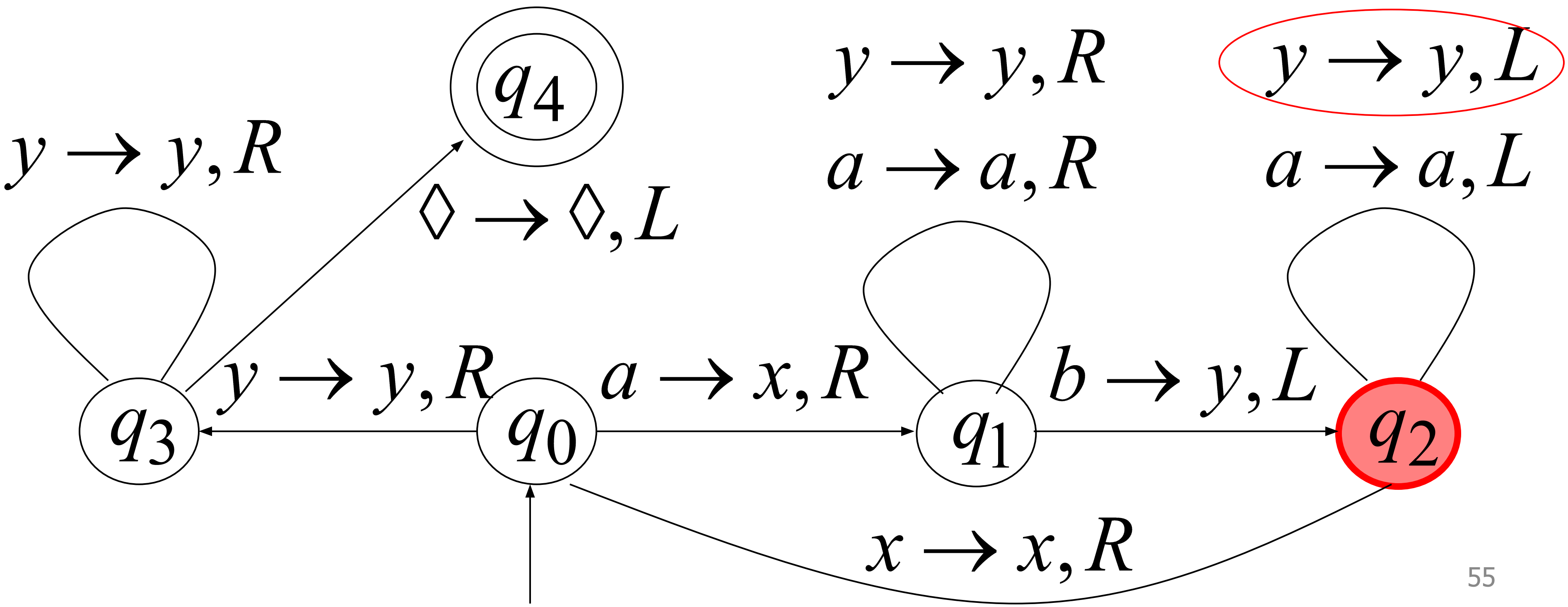
Time 7


◊
x
x
y
b
◊
◊
q_1

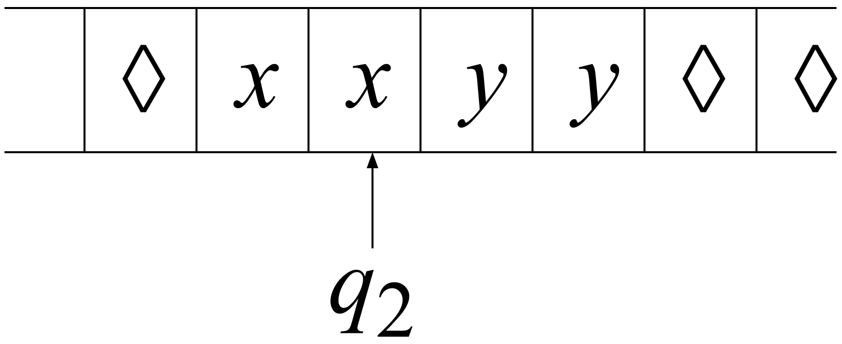

q_4
$y \rightarrow y, R$
$\lozenge \rightarrow \lozenge, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

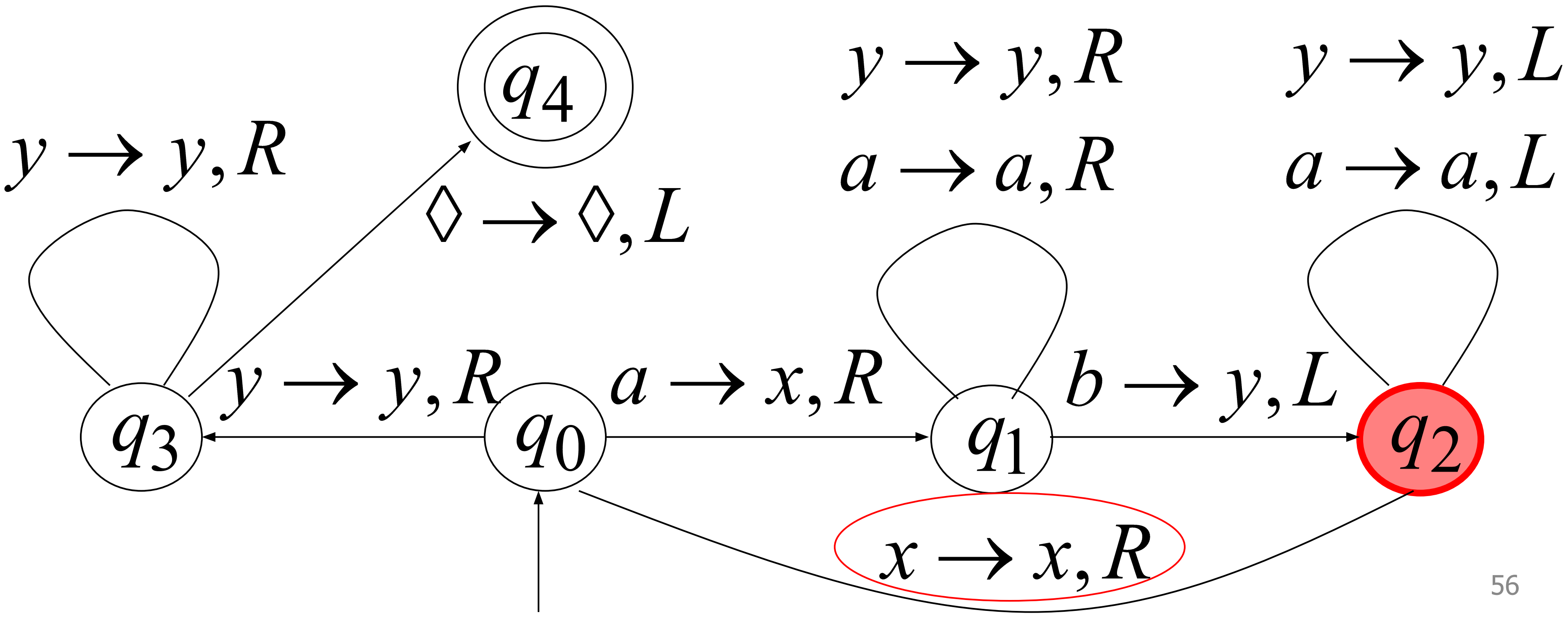
Time 8


◊
x
x
y
y
◊
◊
q_2

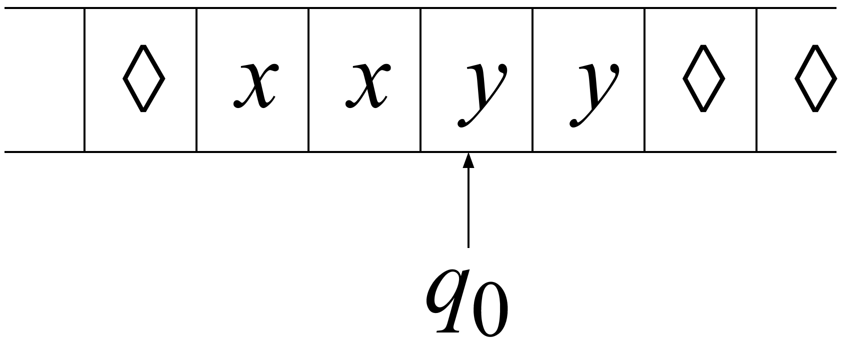

q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

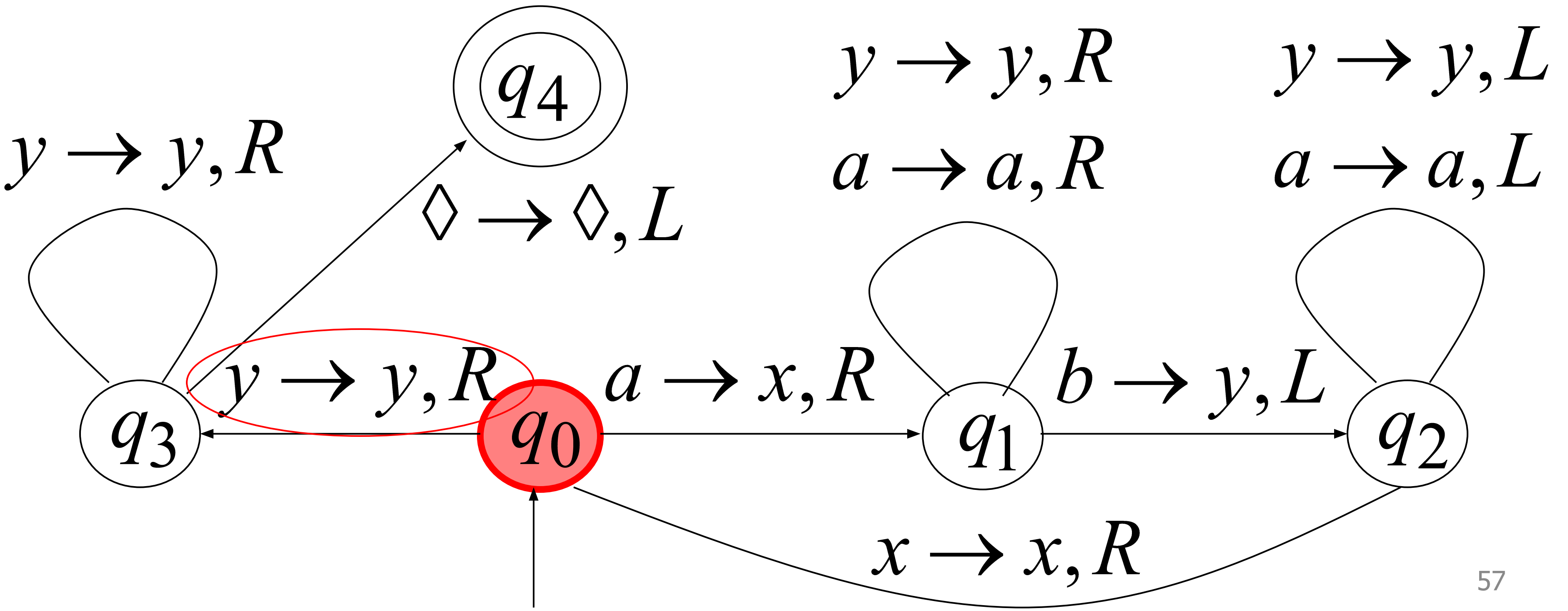
Time 9


◊
x
x
y
y
◊
◊
q_2

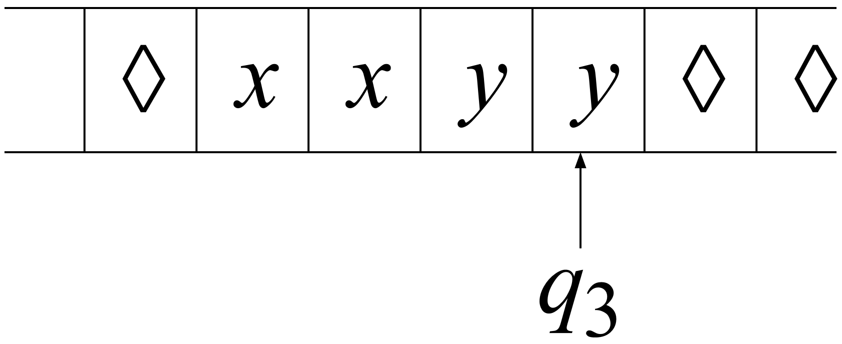

q_4
$y \rightarrow y, R$
$\lozenge \rightarrow \lozenge, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 10

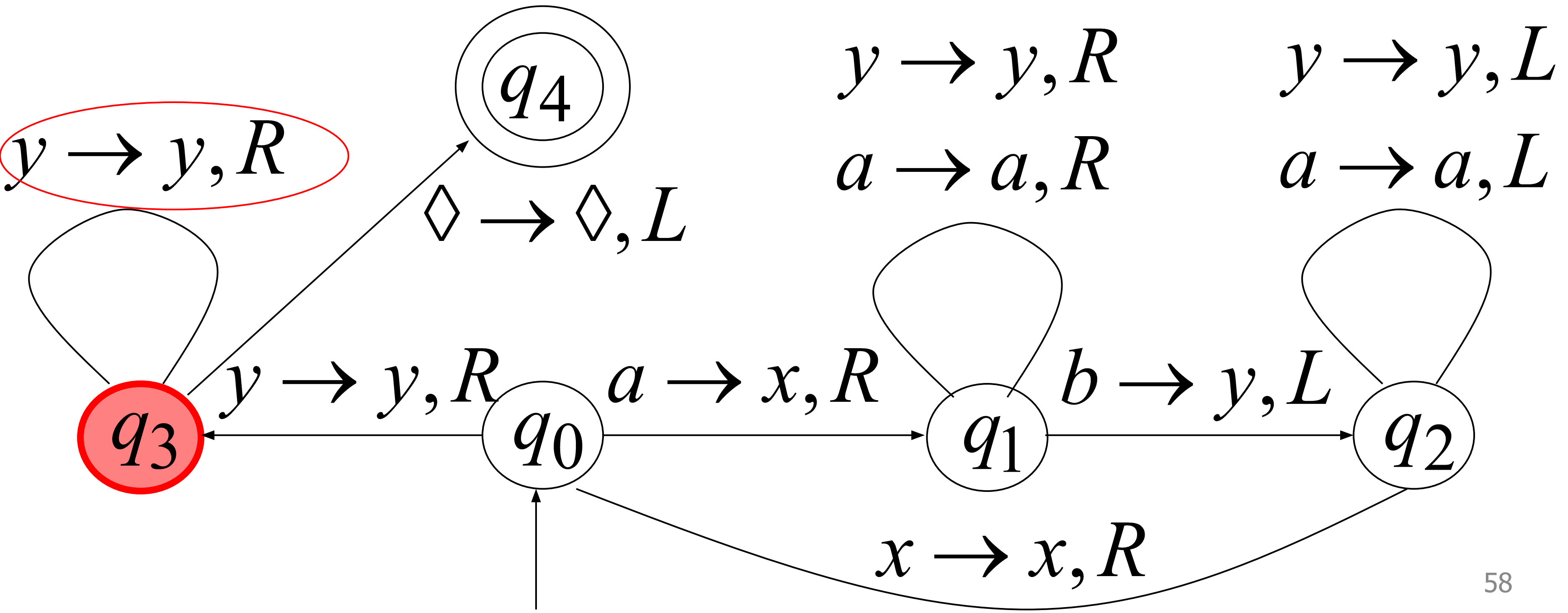

◊
x
x
y
y
◊
◊
q_0


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

Time 11

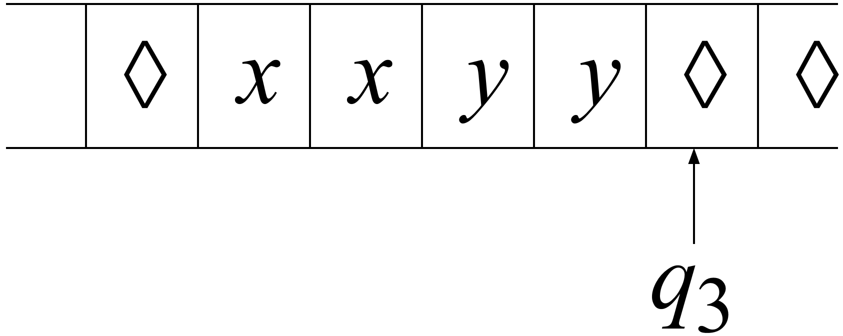


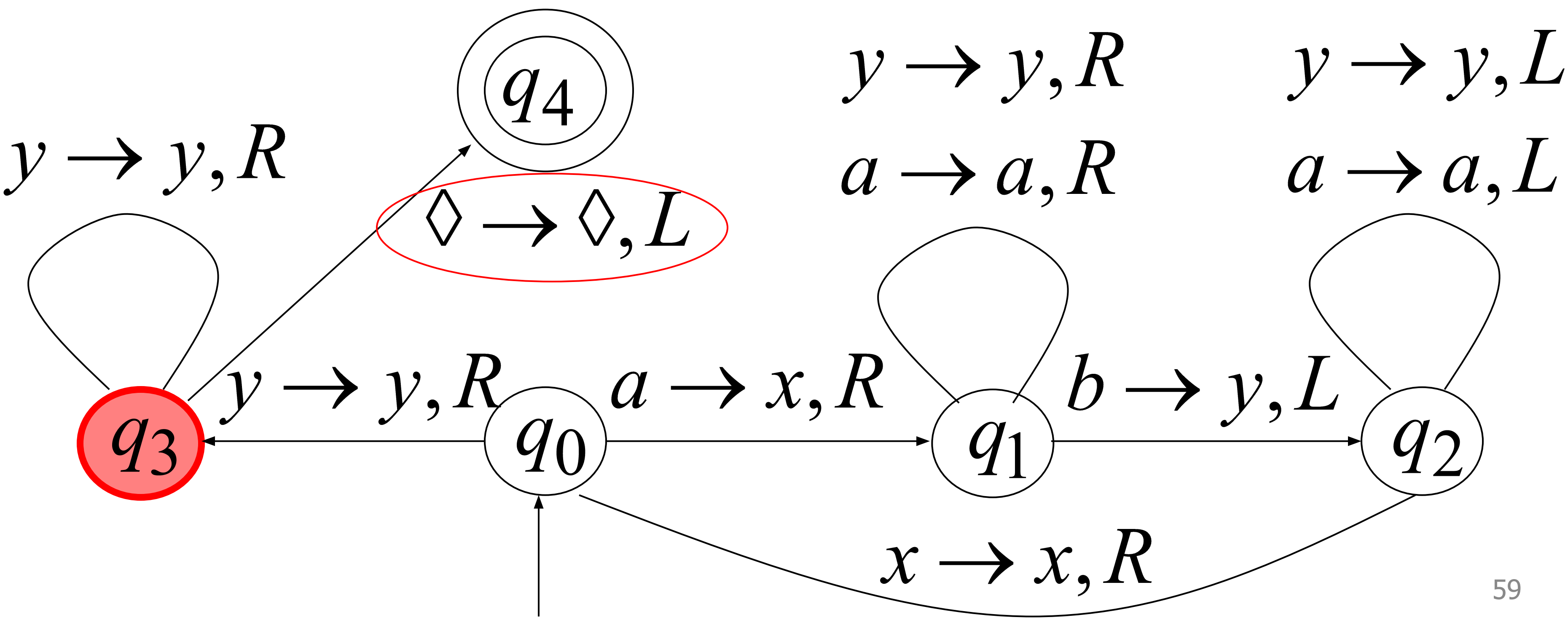
◊
x
x
y
y
◊
◊
q_3



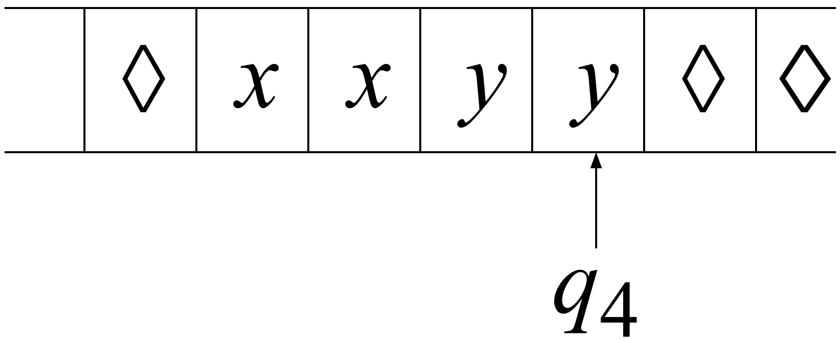
q_4
$y \rightarrow y, R$
$\lozenge \rightarrow \lozenge, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 12

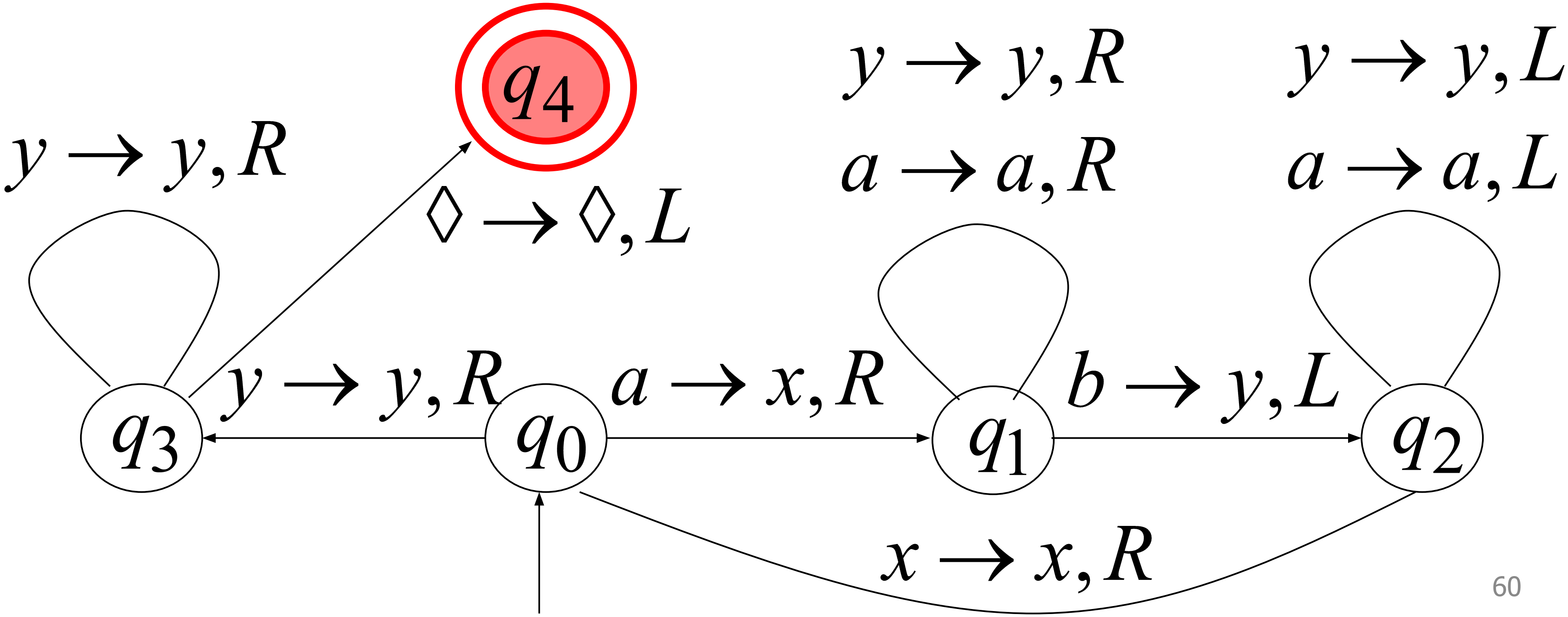




Time 13



Halt & Accept



Exercise:
Convert the parenthesis checker TM rules into state diagram.

Standard Turing Machine

The machine we described is the standard:

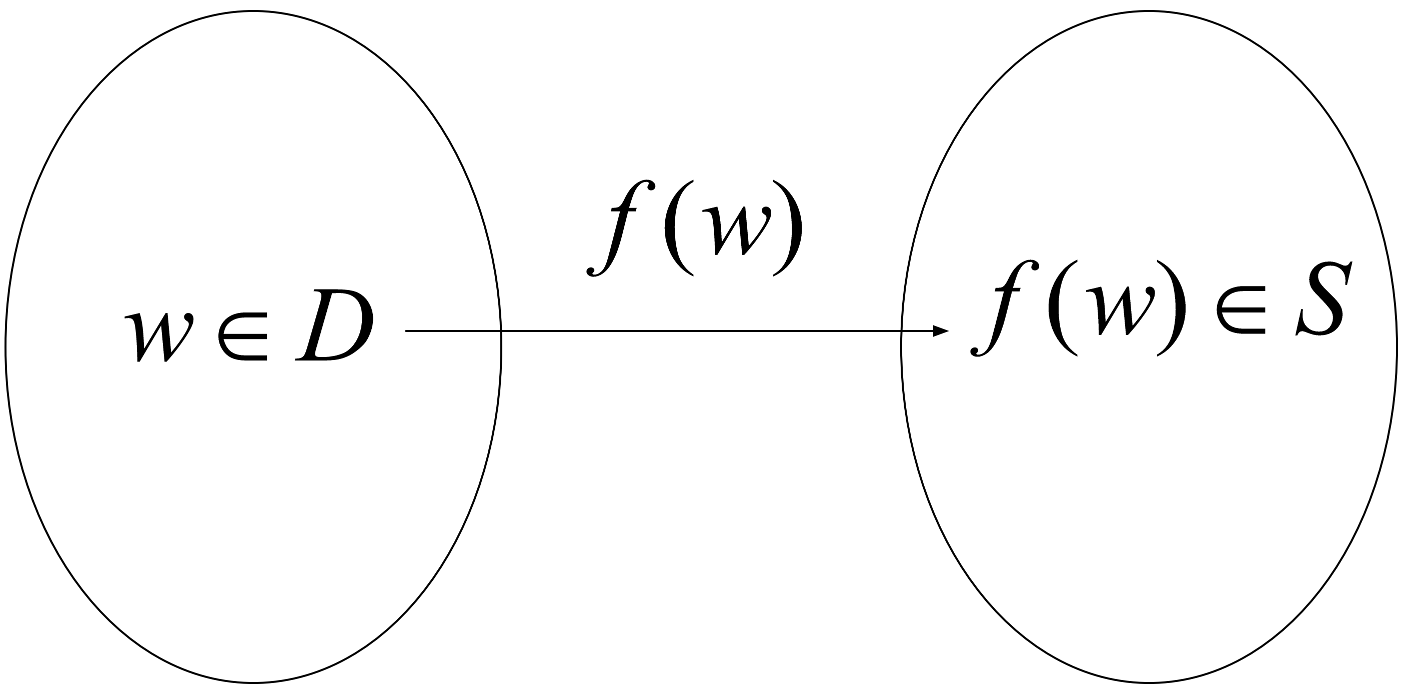
- Deterministic
- Infinite tape in both directions
- Tape is the input/output file

Computing Functions
with
Turing Machines

A function $f(w)$ has:

Domain: D

Result Region: S



A function may have many parameters:

Example: Addition function

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

Integer Domain

Decimal: 5

Binary: 101

Unary: 11111

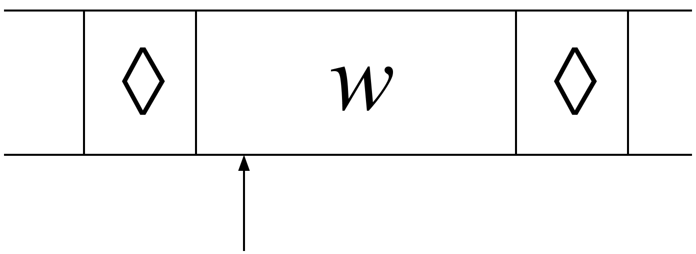
We prefer **unary** representation:

easier to manipulate with Turing machines

Definition:

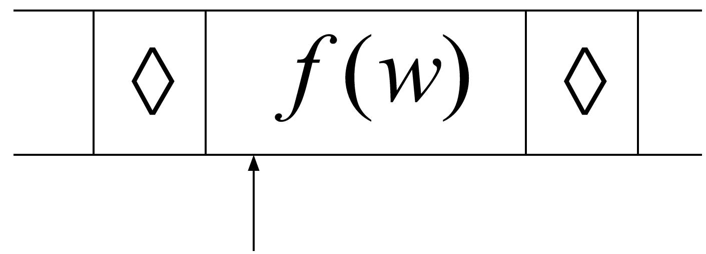
A function f is computable if there is a Turing Machine M with a transition from initial state to final state.

Initial configuration



q_0 initial state

Final configuration



q_f final state

For all $w \in D$ Domain

In other words:

The functions that can be implemented by a TM are said to be **computable functions.**

Example

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

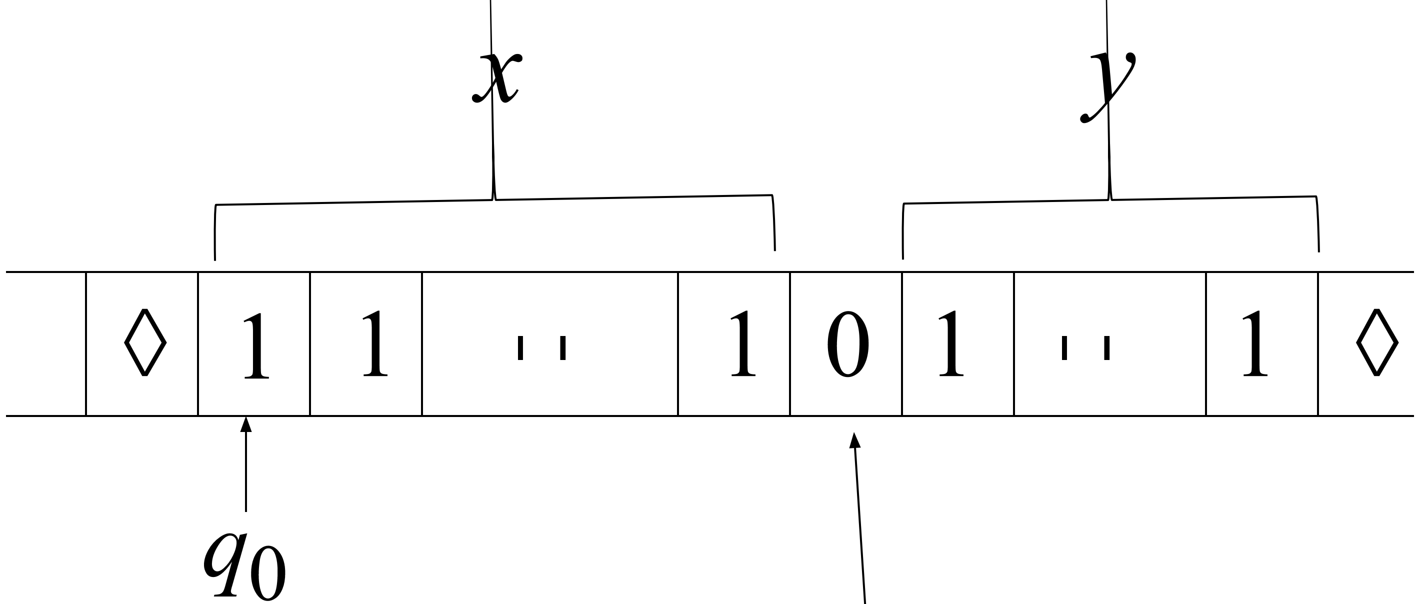
x, y are integers

Turing Machine:

Input string: $x0y$ unary

Output string: $xy0$ unary

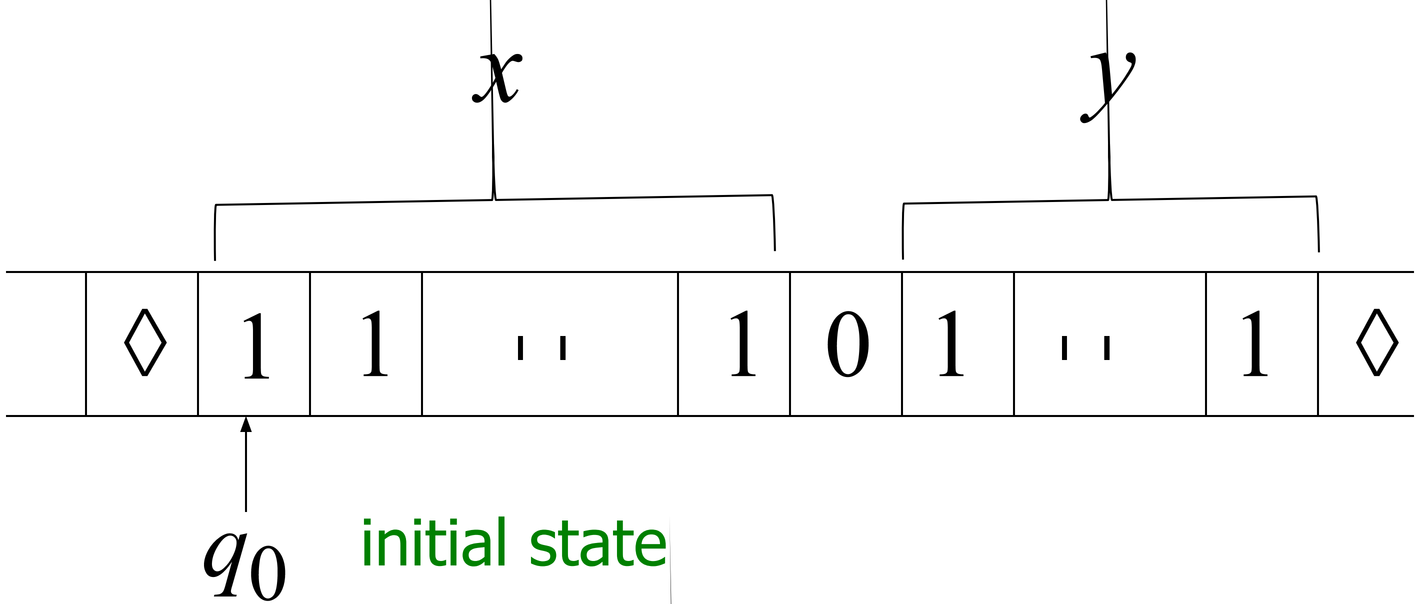
Start



initial state

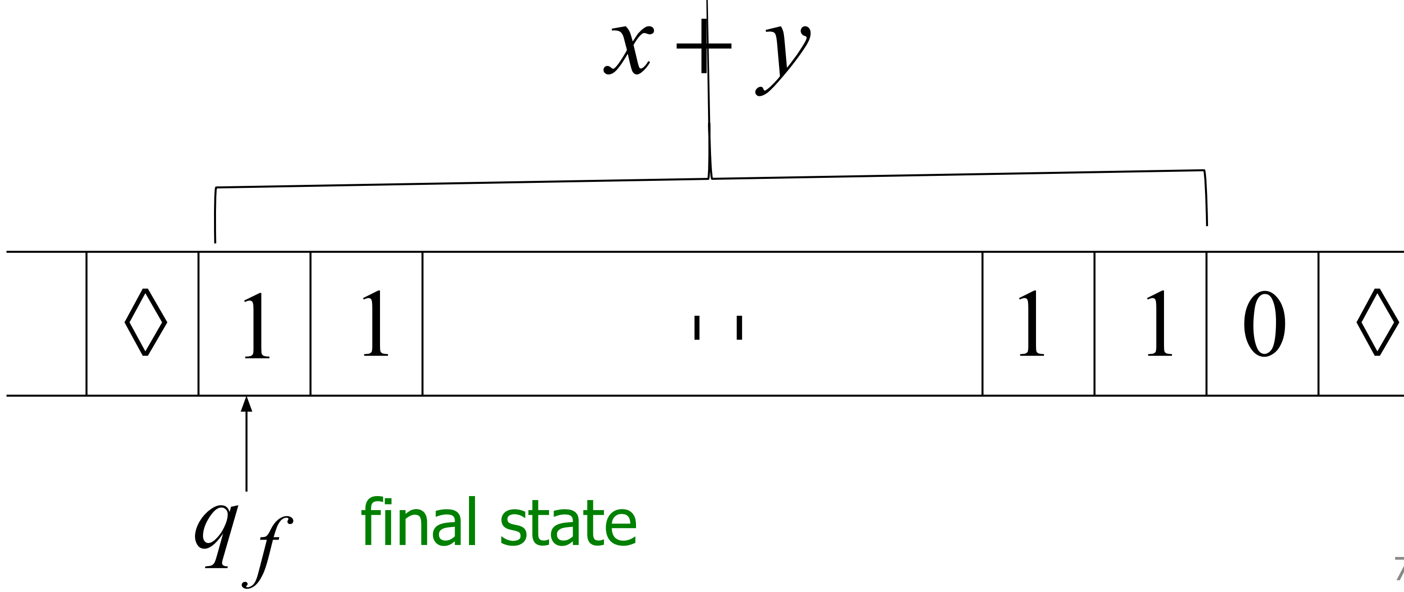
The 0 is the delimiter that
separates the two numbers

Start



initial state

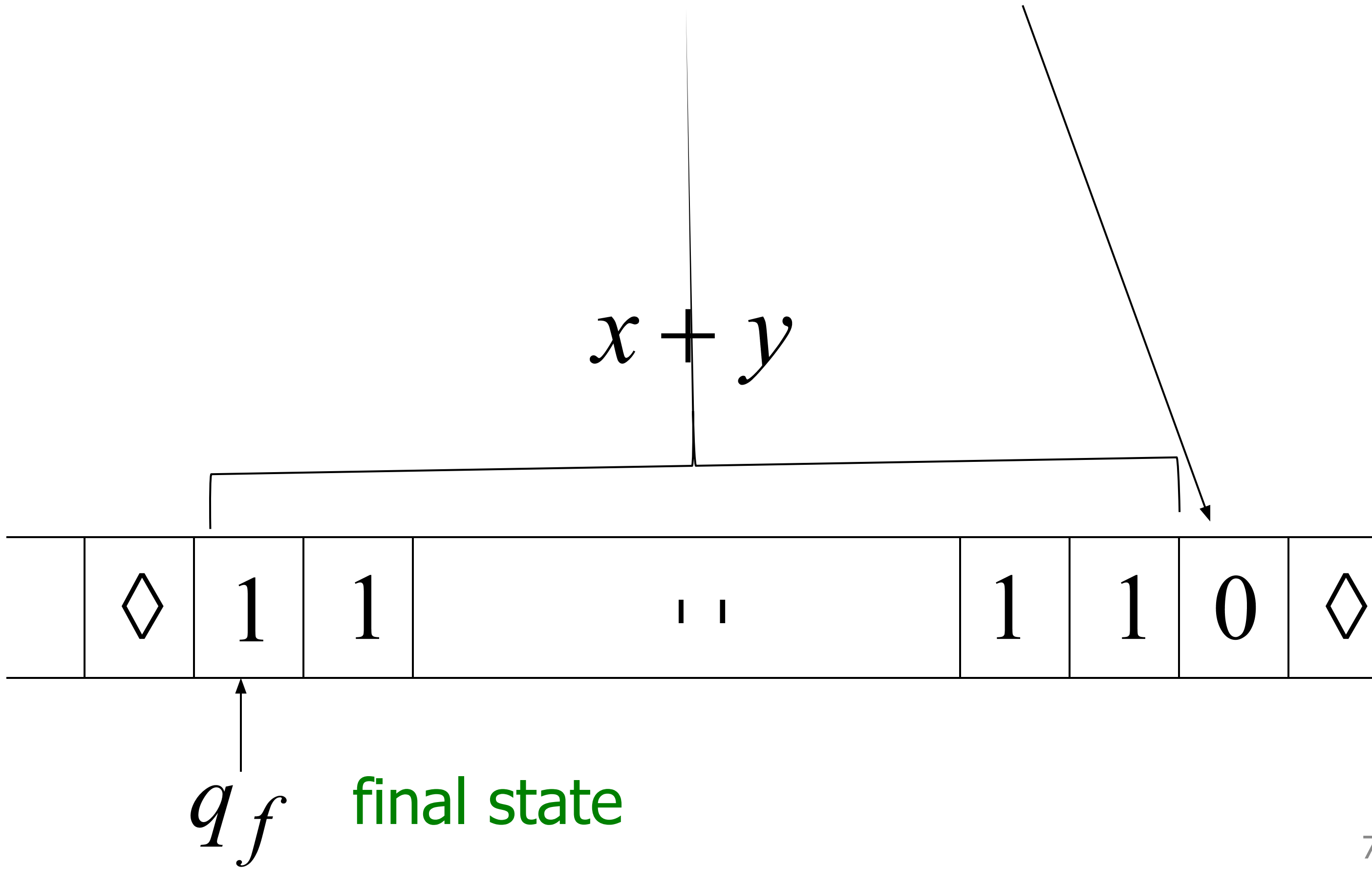
Finish



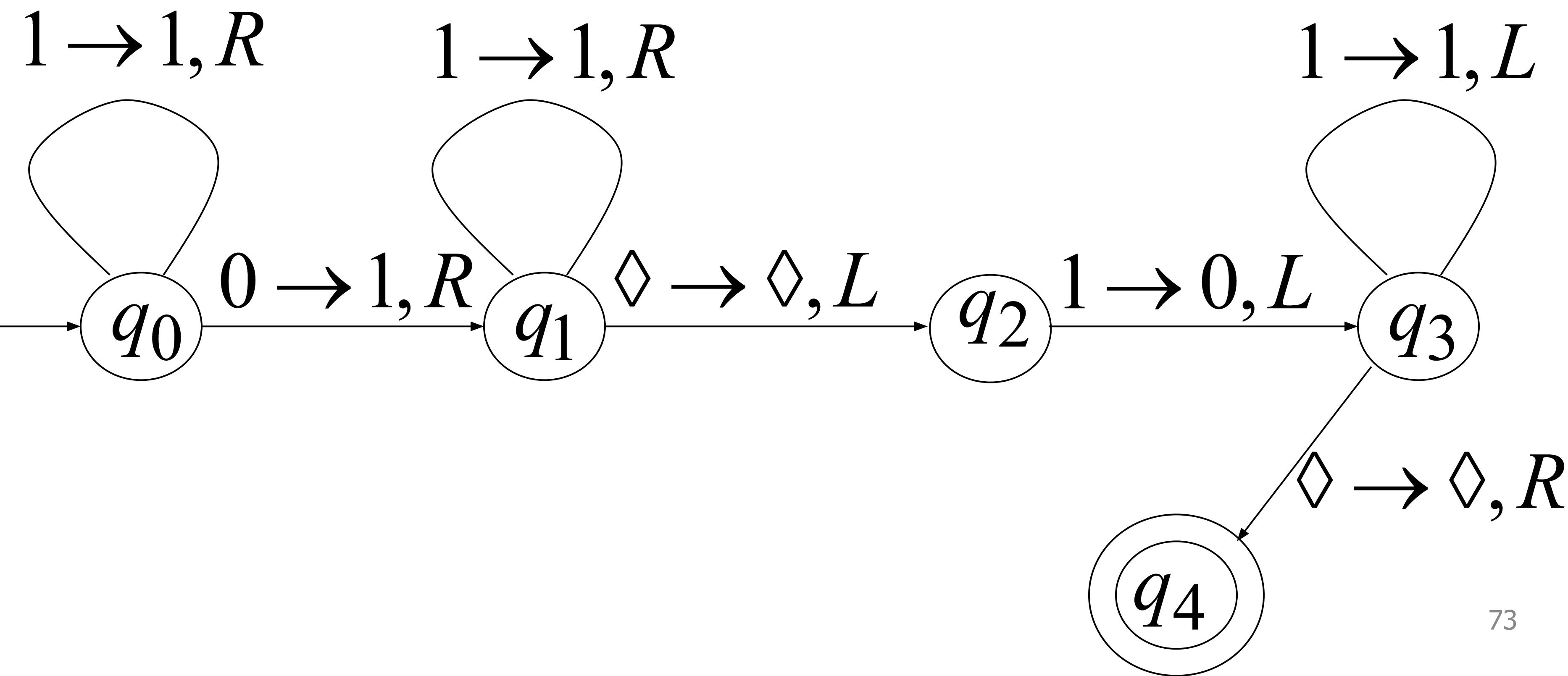
final state

The 0 helps when we use
the result for other operations

Finish



Turing machine for function $f(x, y) = x + y$

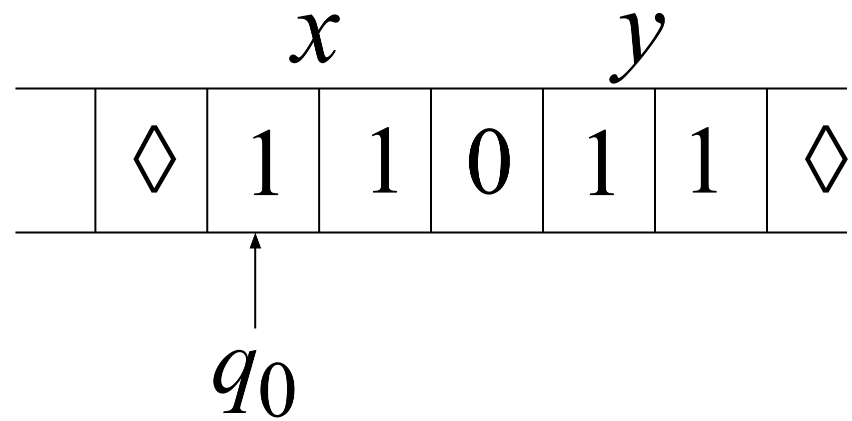


Execution Example:

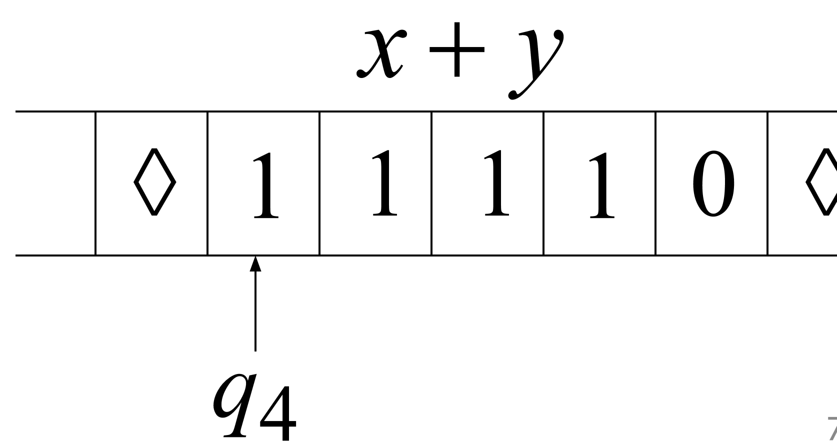
$$x = 11 \quad {}_{(2)}$$

$$y = 11 \quad {}_{(2)}$$

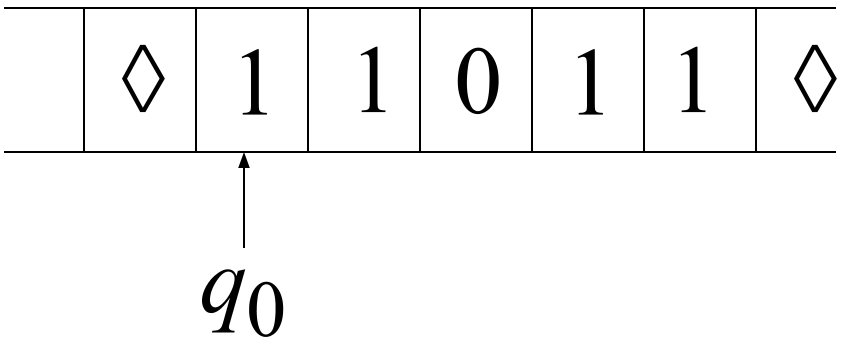
Time 0

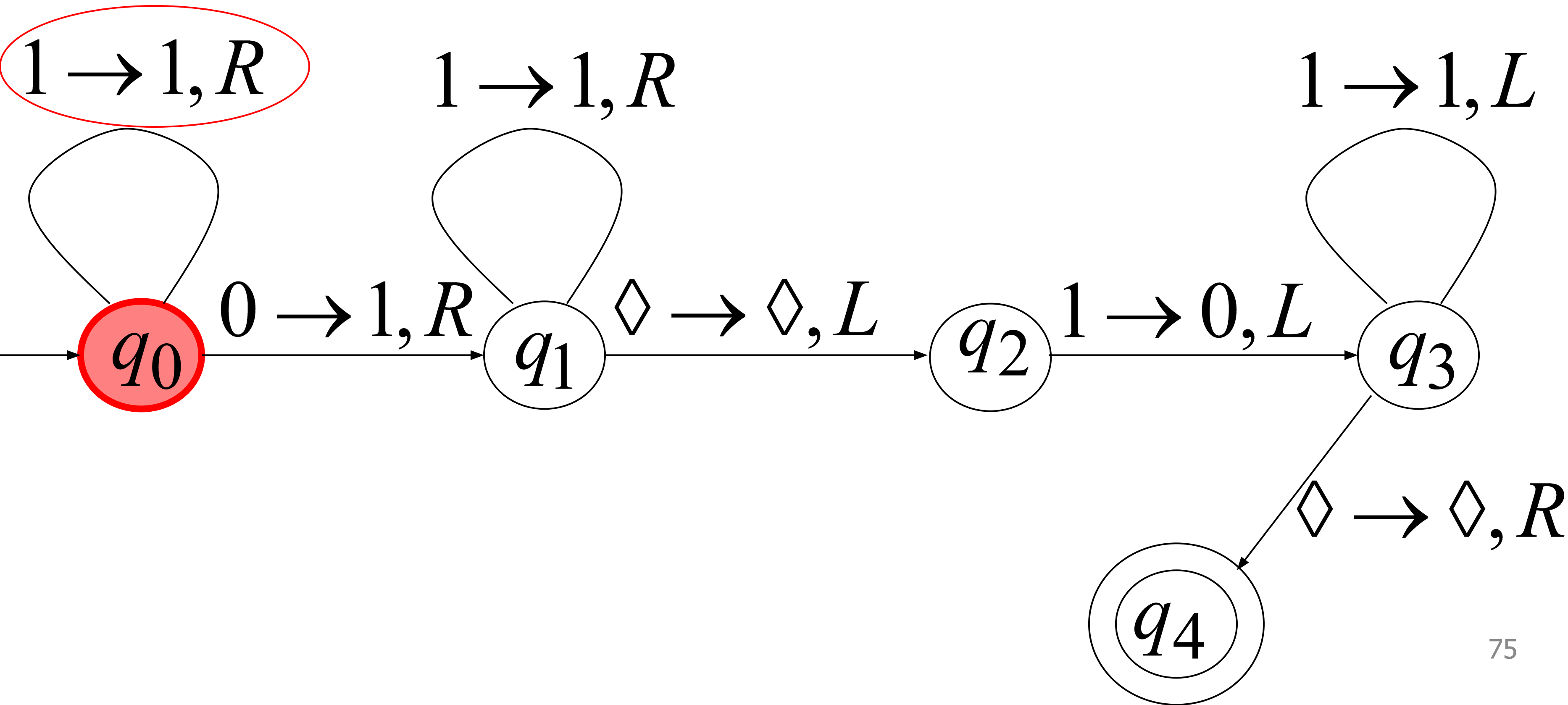


Final Result

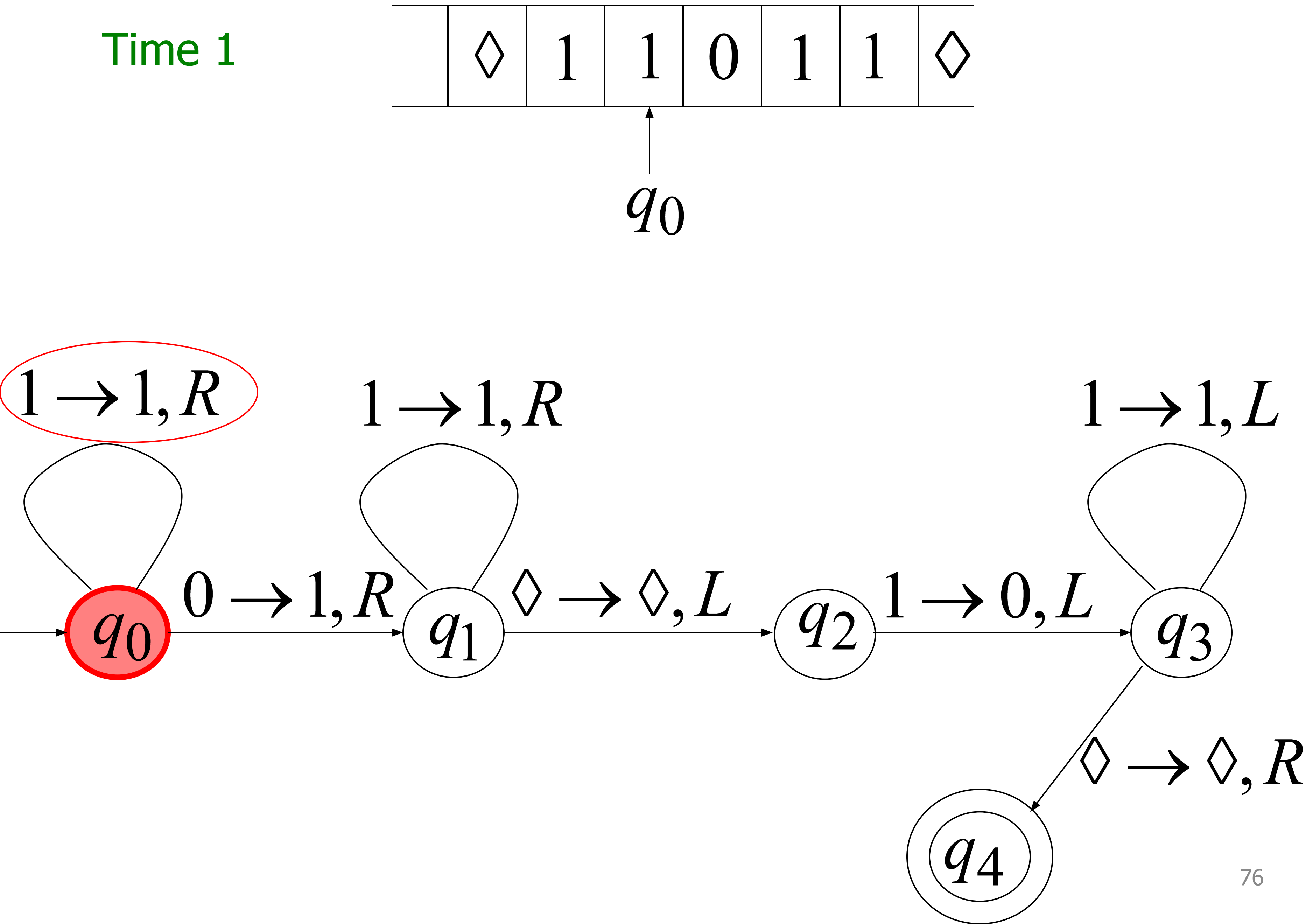


Time 0

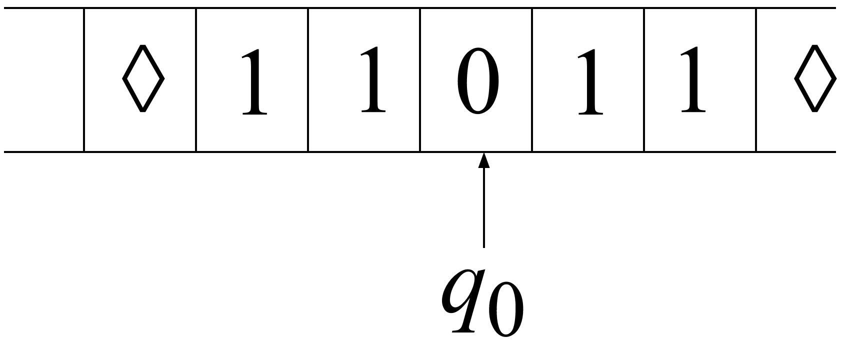

◊ 1 1 0 1 1 ◊
q_0


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

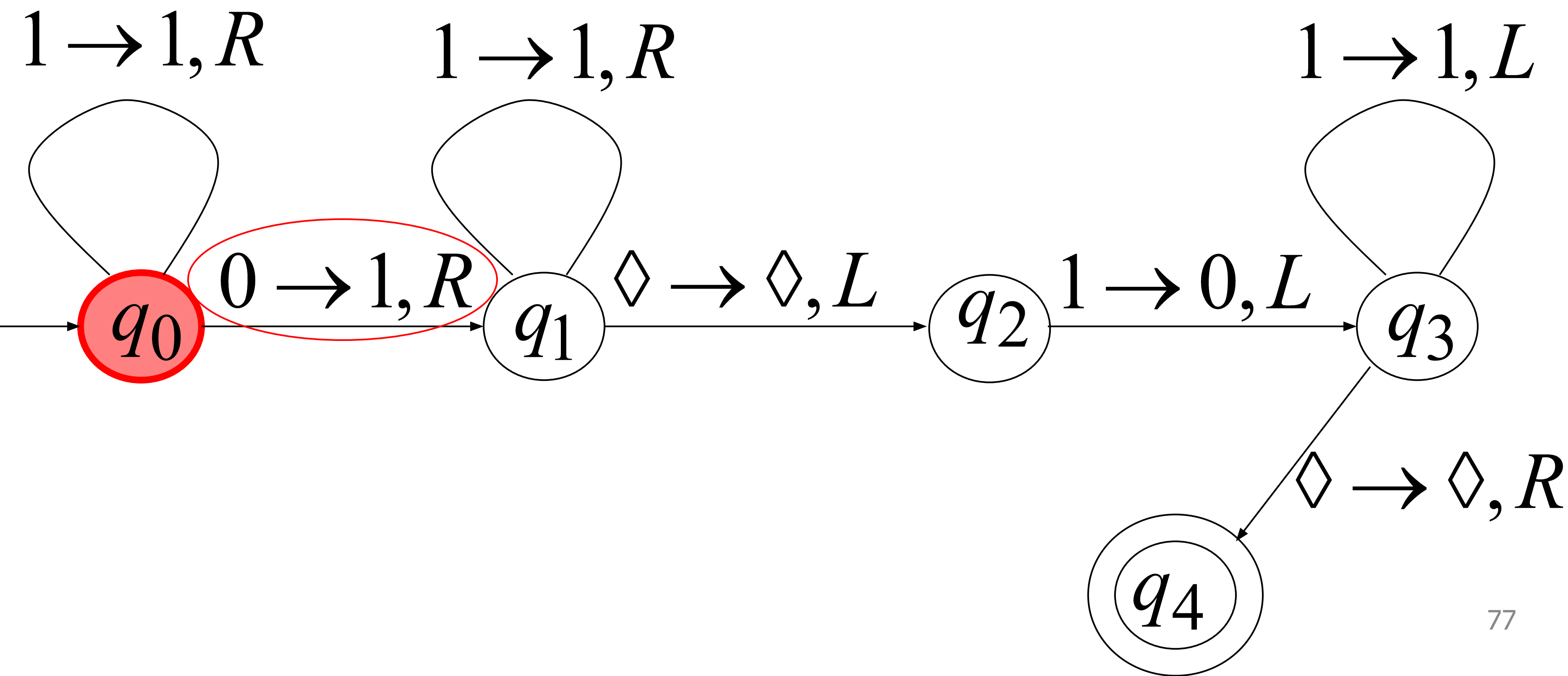
Time 1


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

Time 2

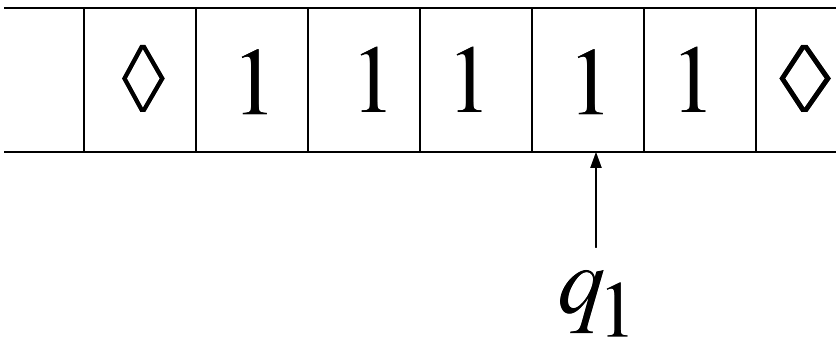


◊
1
1
0
1
1
◊
q_0

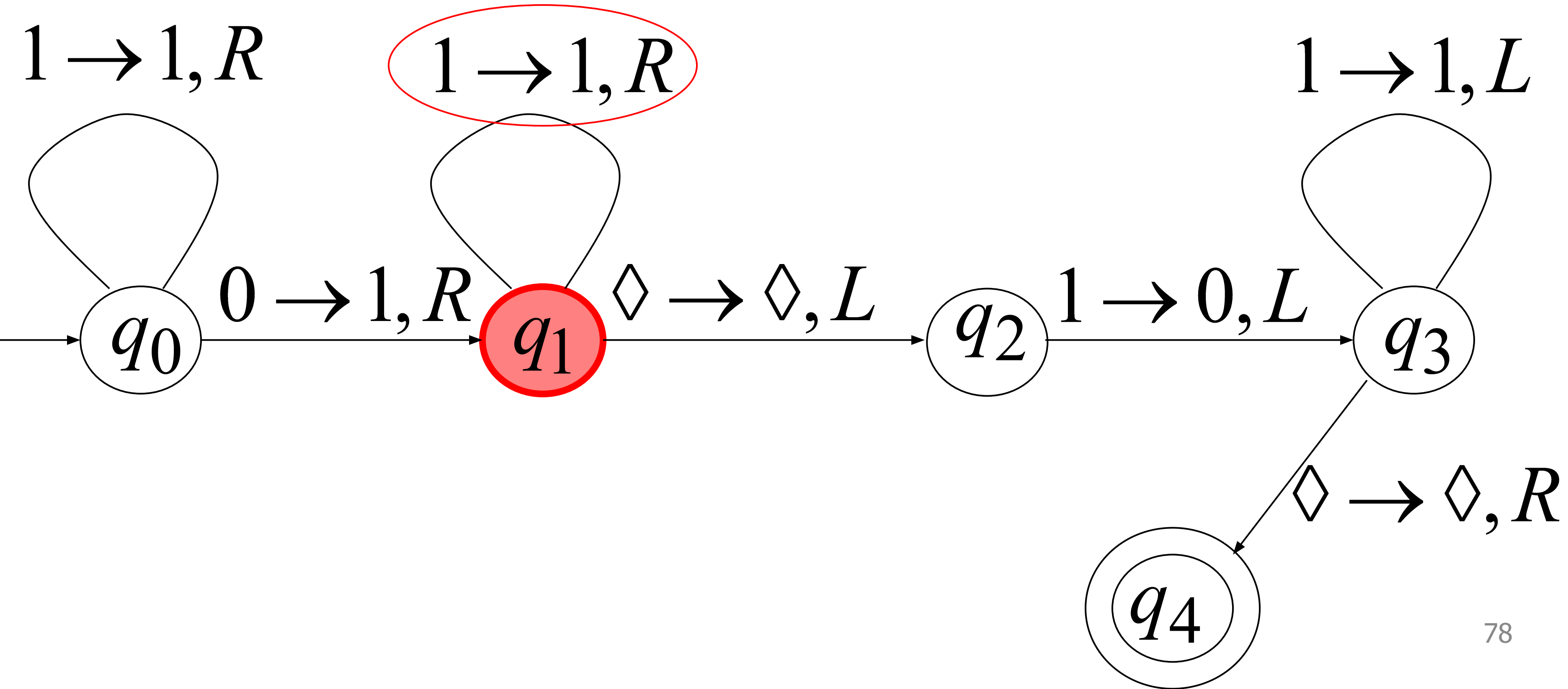


$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
q_0
$0 \rightarrow 1, R$
q_1
$\lozenge \rightarrow \lozenge, L$
q_2
$1 \rightarrow 0, L$
q_3
$\lozenge \rightarrow \lozenge, R$
q_4

Time 3

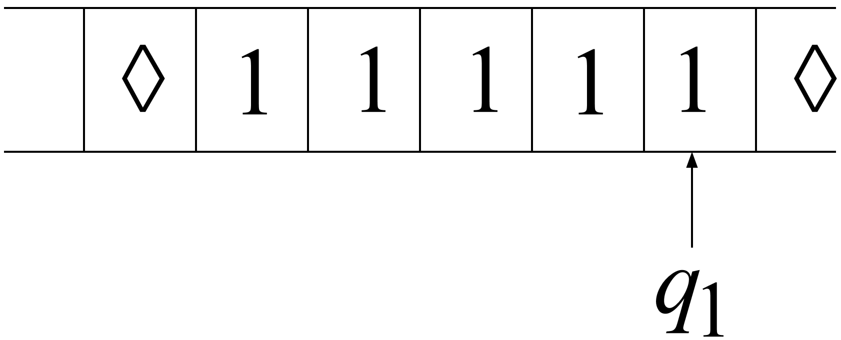


◊
1
1
1
1
1
◊
q_1

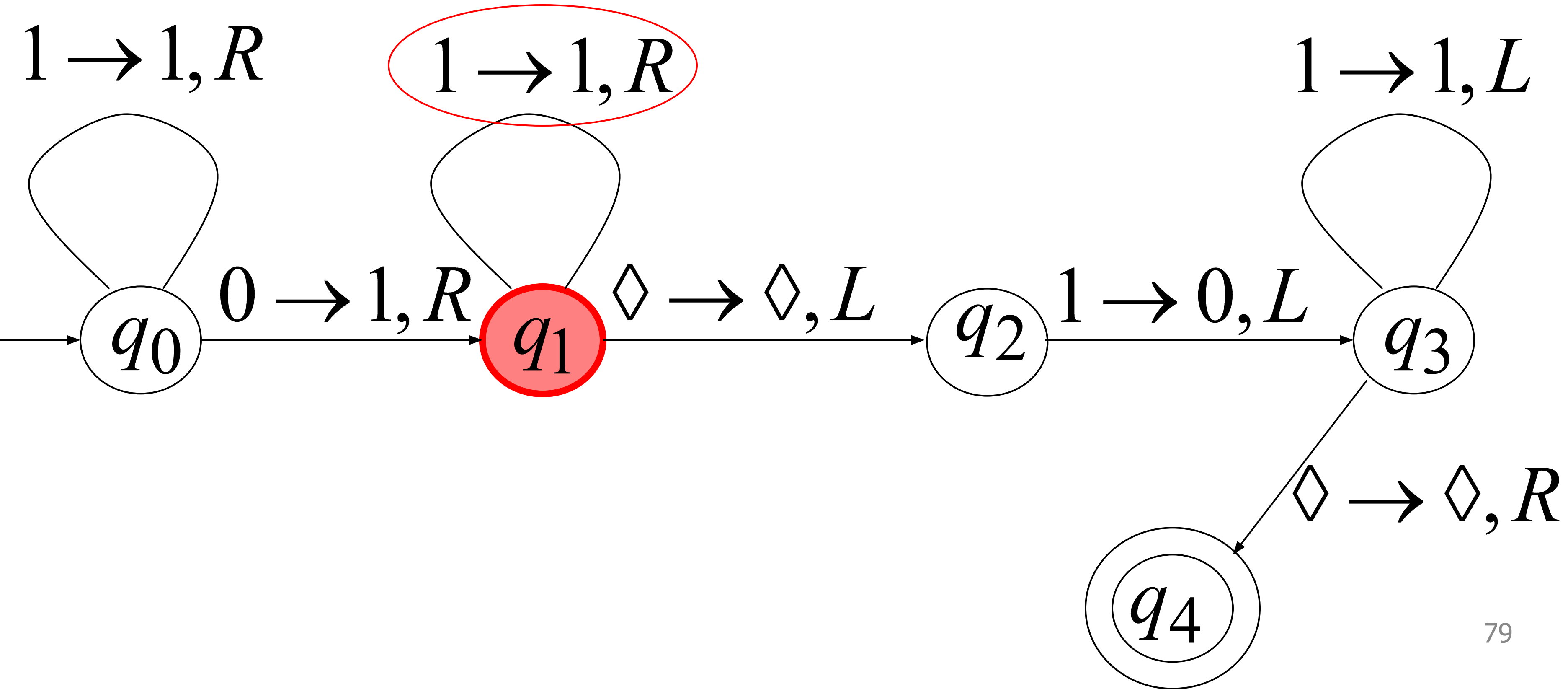


$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
$0 \rightarrow 1, R$
$\Diamond \rightarrow \Diamond, L$
$1 \rightarrow 0, L$
$\Diamond \rightarrow \Diamond, R$
q_0
q_1
q_2
q_3
q_4

Time 4

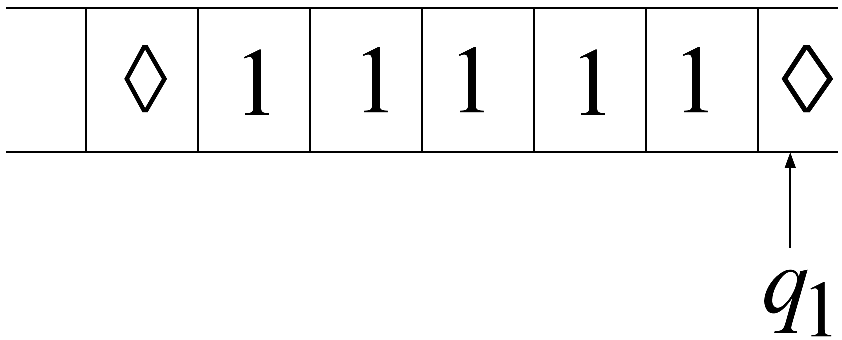


◊
1
1
1
1
1
◊
q_1

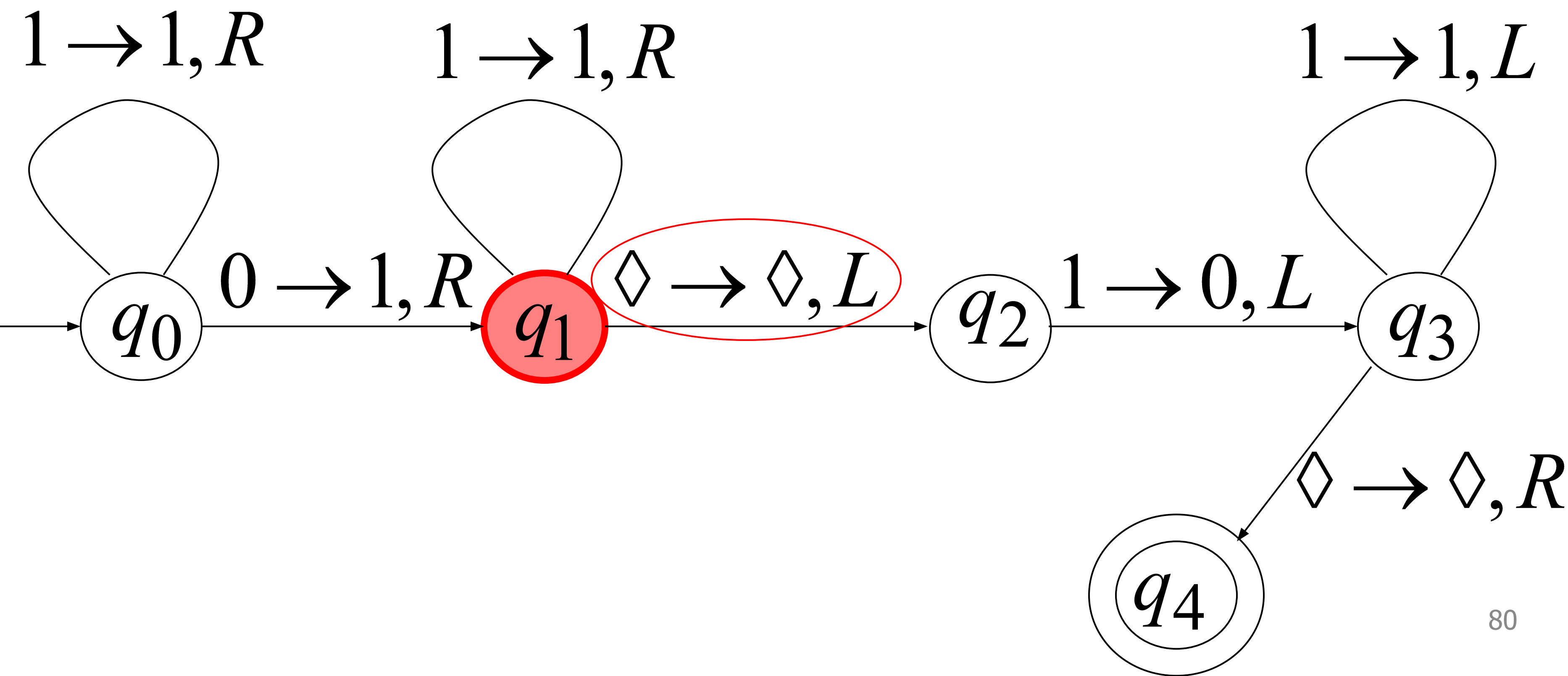


$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
q_0
$0 \rightarrow 1, R$
q_1
$\Diamond \rightarrow \Diamond, L$
q_2
$1 \rightarrow 0, L$
q_3
$\Diamond \rightarrow \Diamond, R$
q_4

Time 5

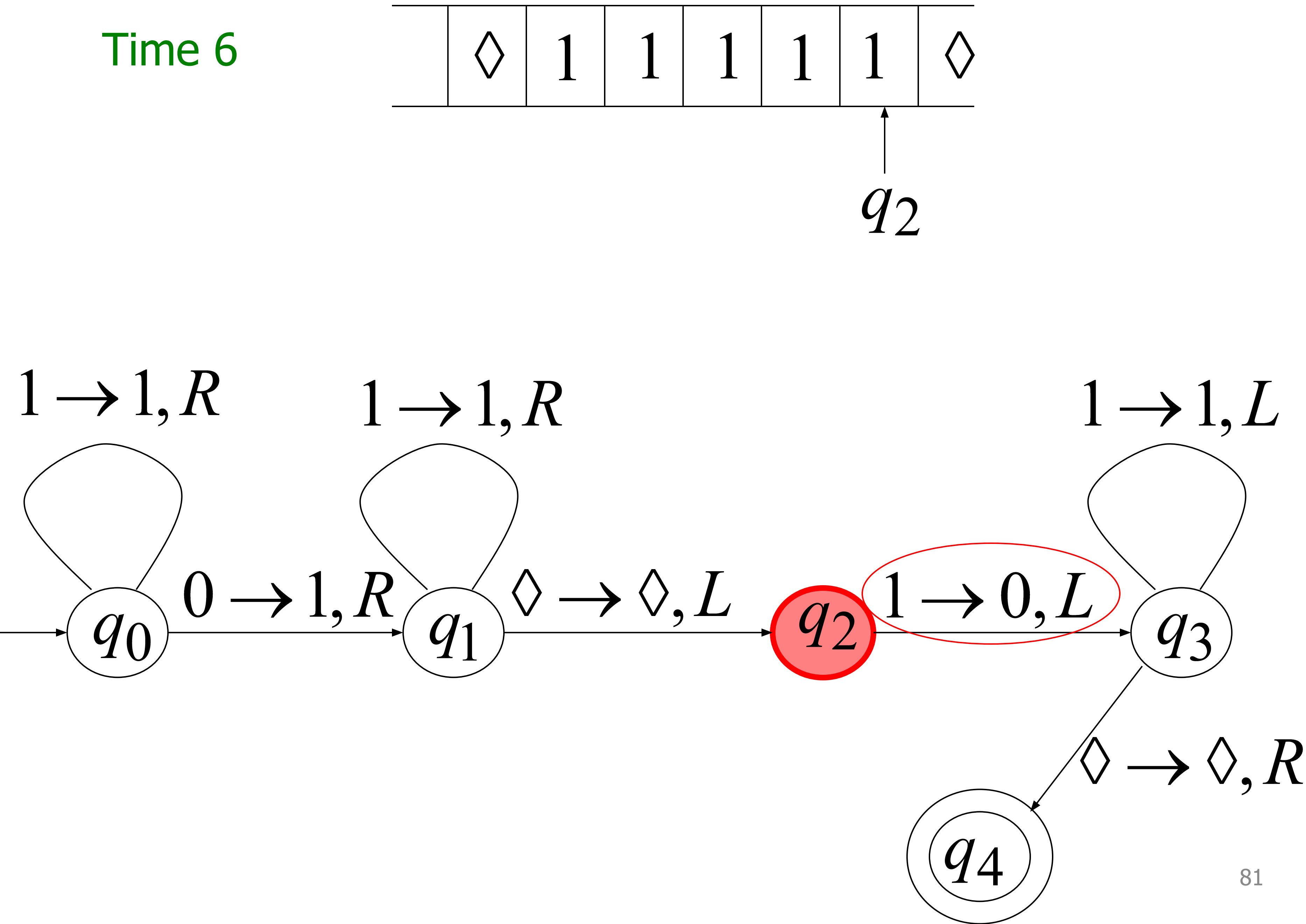


◊ 1 1 1 1 1 ◊
q_1



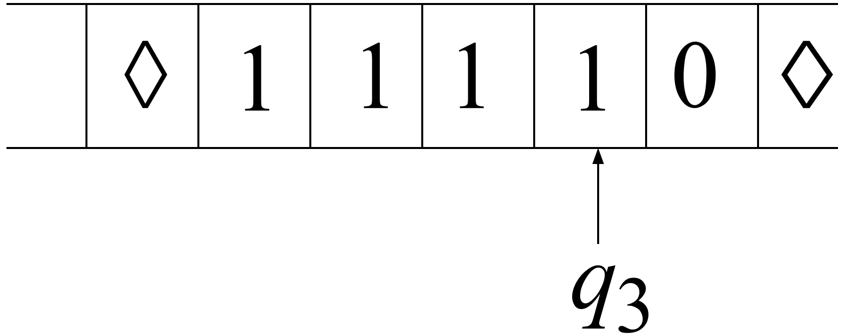
$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
q_0
$0 \rightarrow 1, R$
q_1
$\Diamond \rightarrow \Diamond, L$
q_2
$1 \rightarrow 0, L$
q_3
$\Diamond \rightarrow \Diamond, R$
q_4

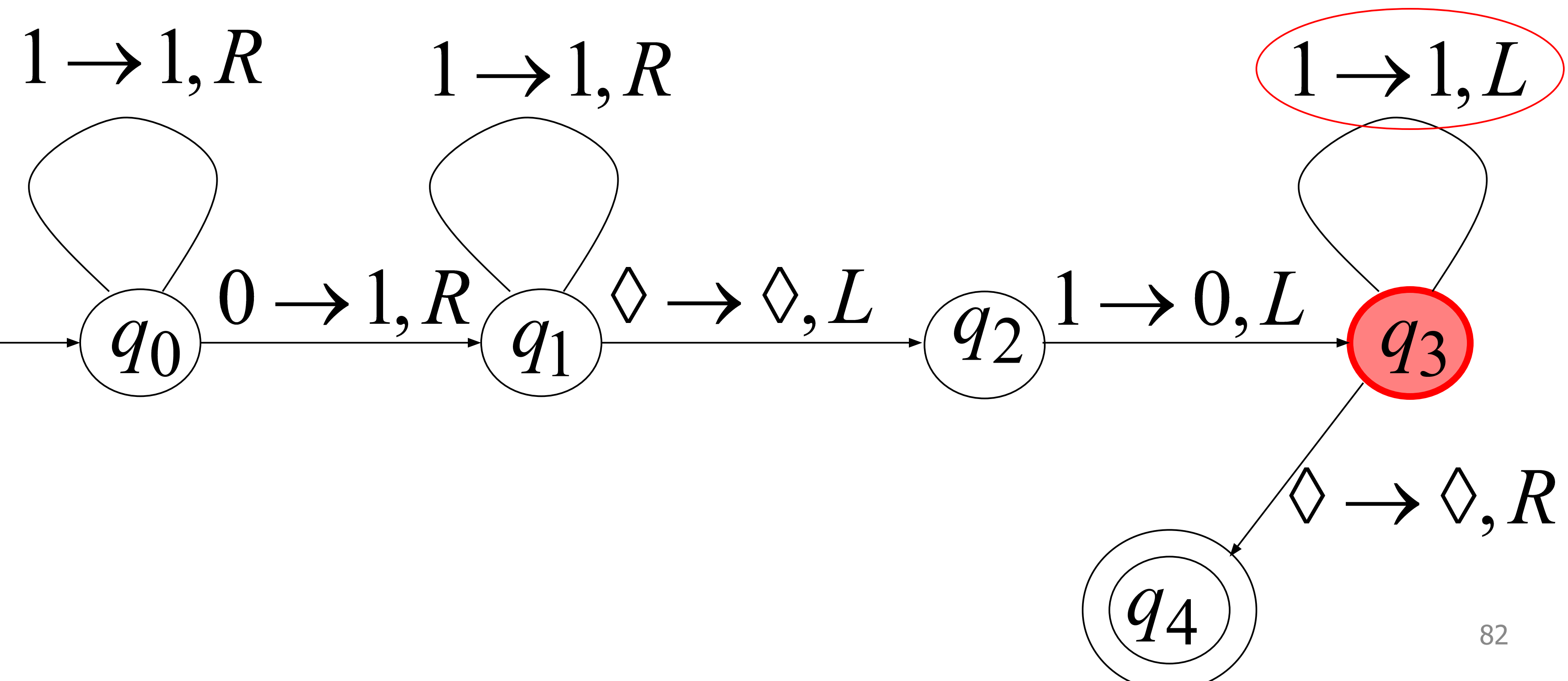
Time 6



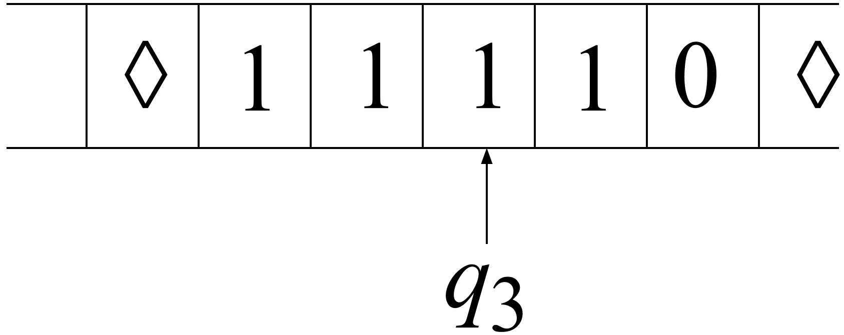
◊ 1 1 1 1 1 ◊
q_2
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
q_0
q_1
q_2
q_3
q_4

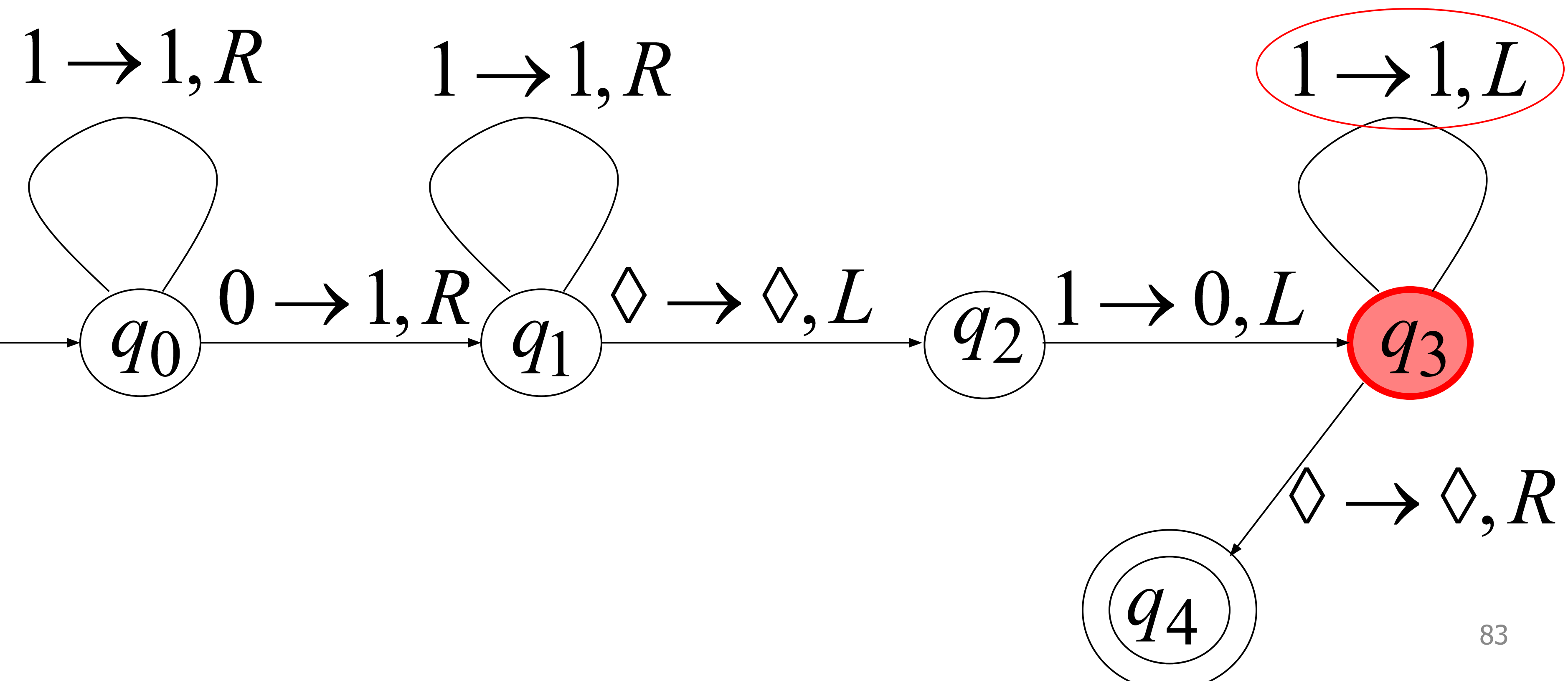
Time 7



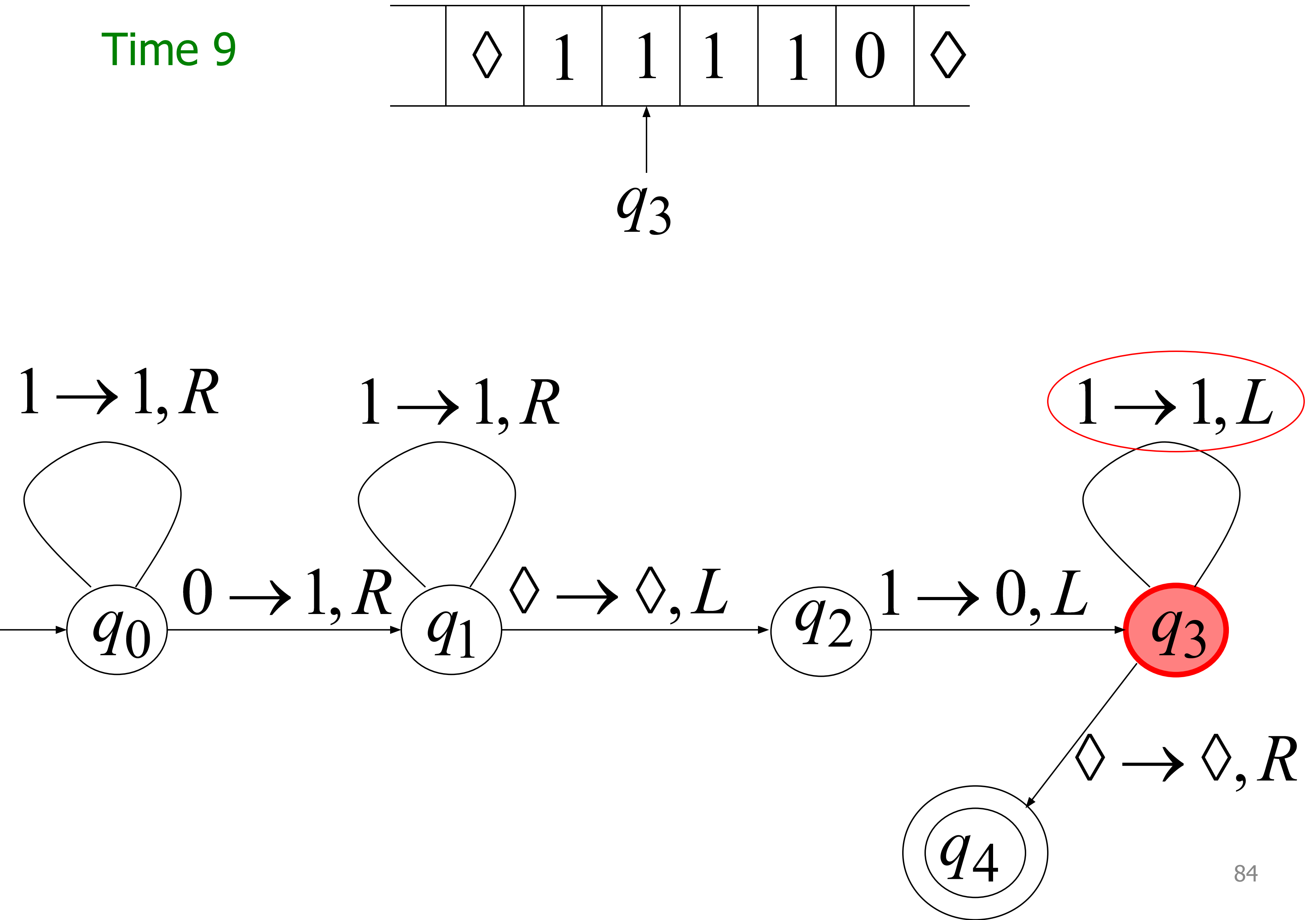


Time 8


◊ 1 1 1 1 0 ◊
q_3

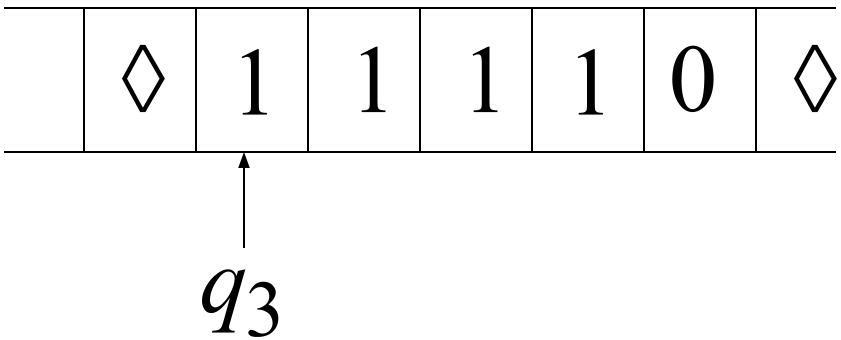

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\Diamond \to \Diamond, L$
$1 \to 0, L$
$\Diamond \to \Diamond, R$
q_0
q_1
q_2
q_3
q_4

Time 9

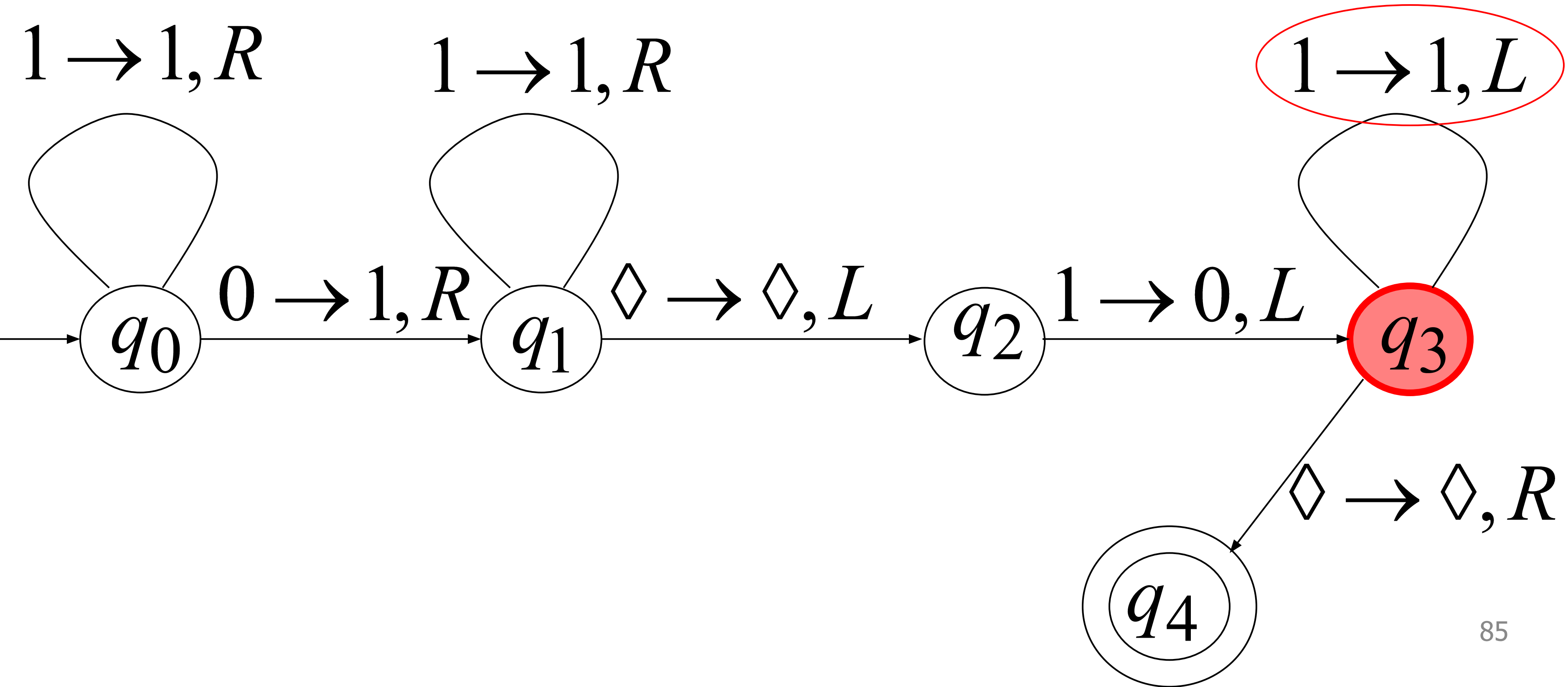


◊
1
1
1
1
0
◊
q_3
$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
$0 \rightarrow 1, R$
$\lozenge \rightarrow \lozenge, L$
$1 \rightarrow 0, L$
$\lozenge \rightarrow \lozenge, R$
q_0
q_1
q_2
q_3
q_4

Time 10

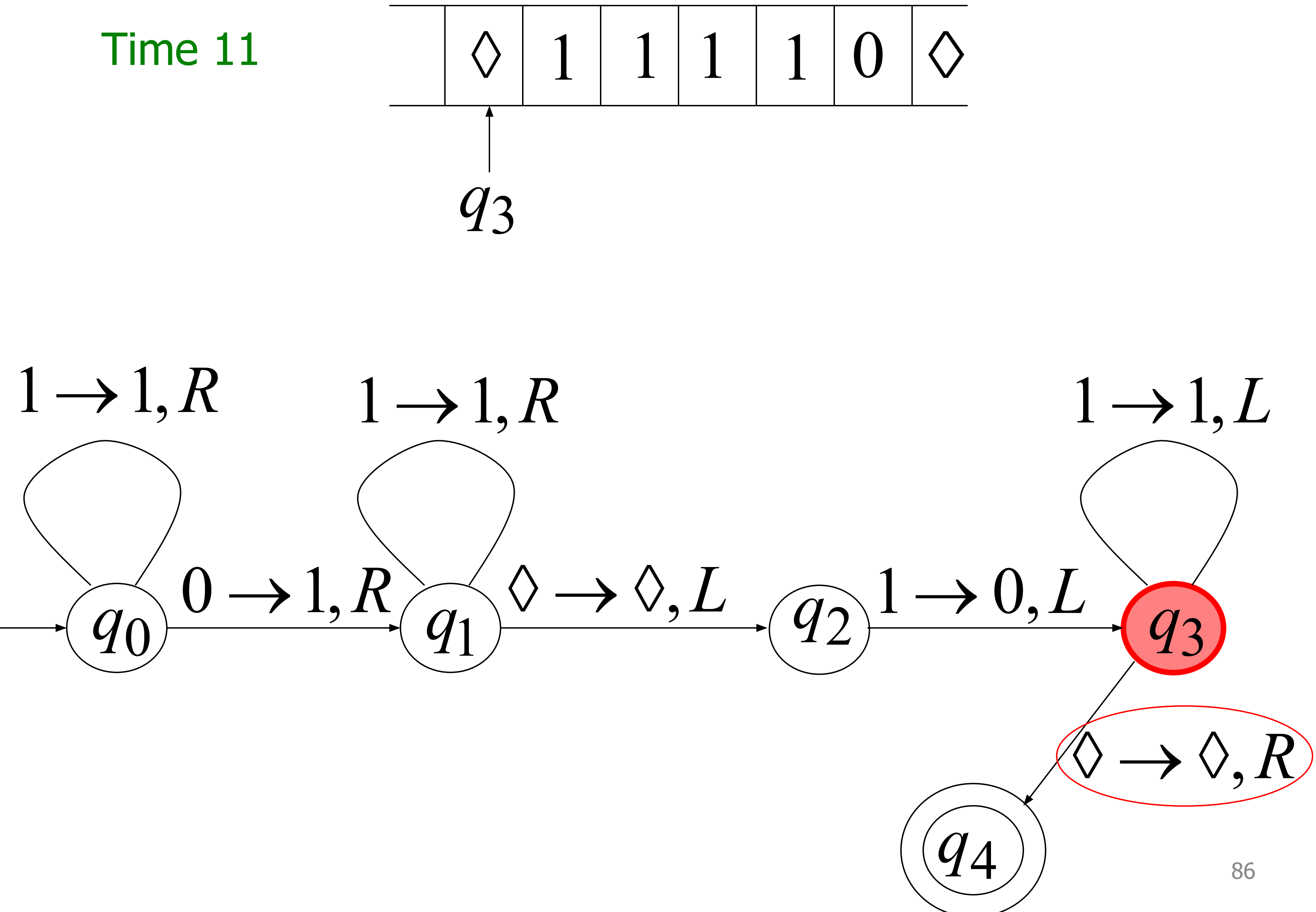


◊
1
1
1
1
0
◊
q_3



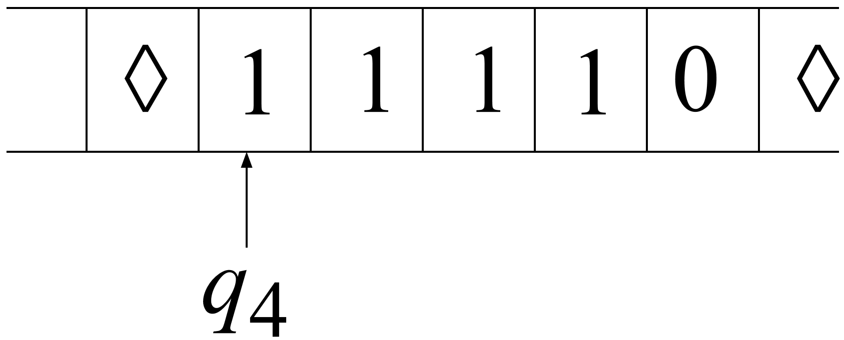
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\lozenge \to \lozenge, L$
$1 \to 0, L$
$\lozenge \to \lozenge, R$
q_0
q_1
q_2
q_3
q_4

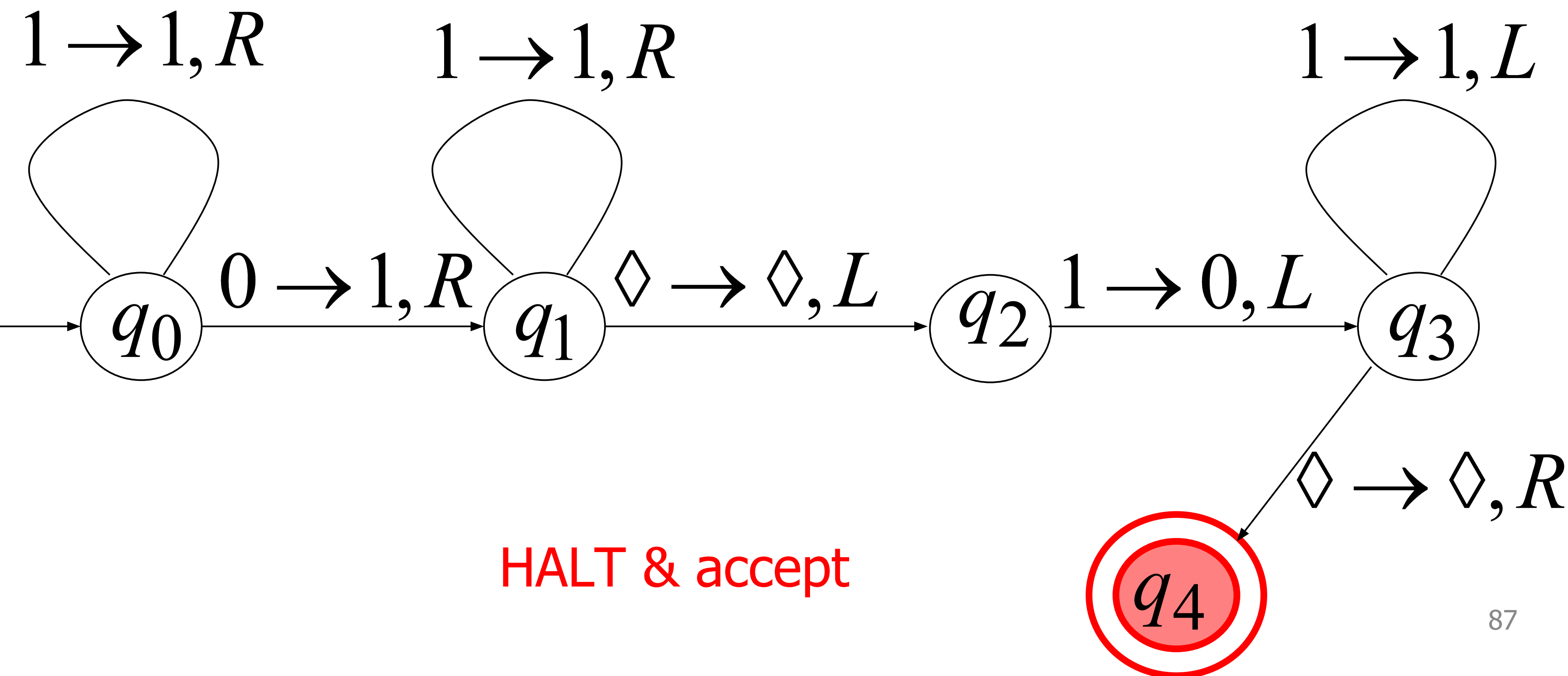
Time 11



◊
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





HALT & accept

Group Assignment #1

1. Mention a TM example for acceptance and rejection cases other than the examples given in the slide.
2. Is the function f(x)=2X is computable?
3. Construct a TM to:

   a)subtract two Unary Numbers.

   b) evaluate the function f(x)=X+Y+Z, where X,Y and Z are all unary numbers.

   c) compare two strings
4. Construct a TM that accepts the language

   a) {ab, aab, aaab, aaaab, ….}

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

NB: Show your TM using state diagram.

The End