Turing Machines

Formal Languages and Abstract Machines

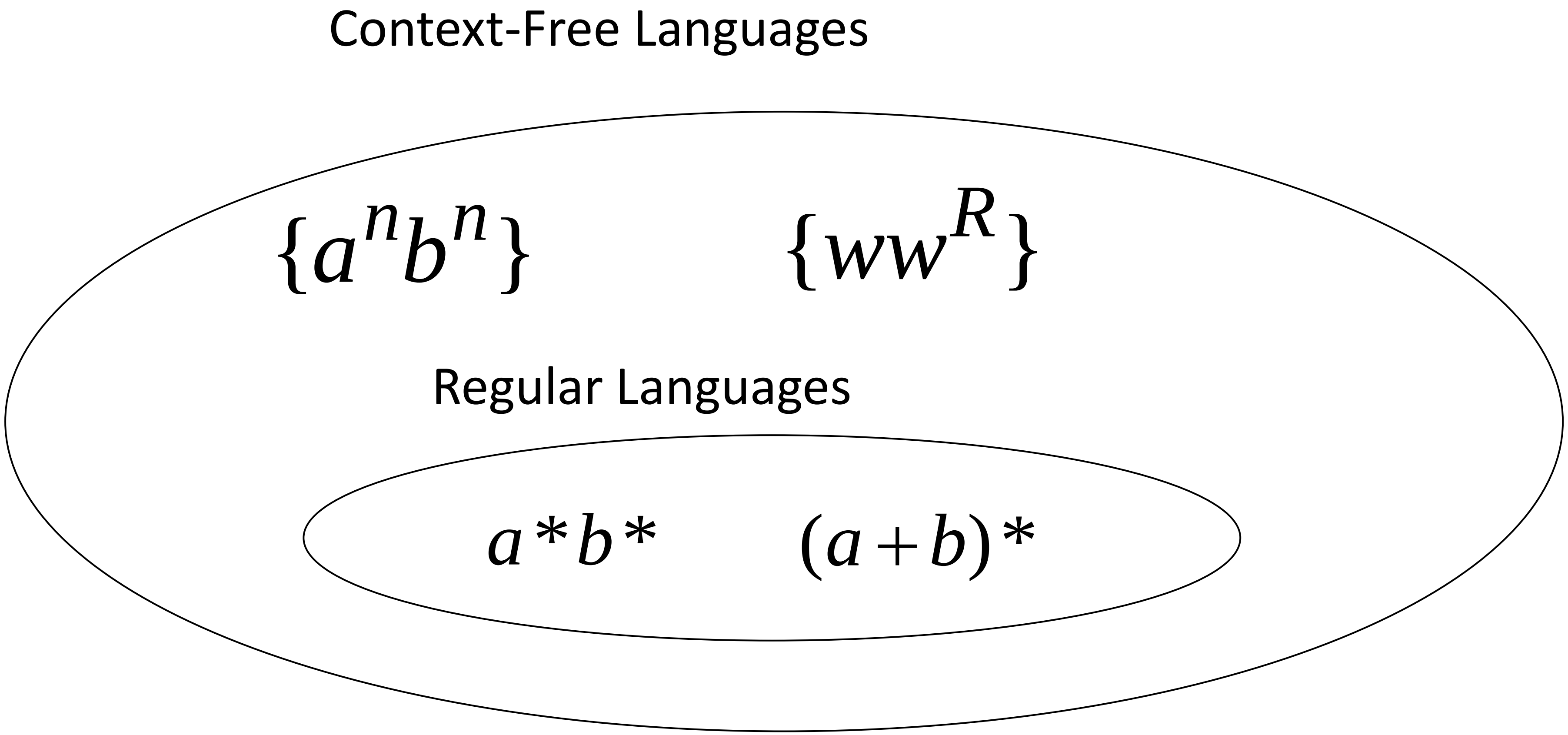
Week 10

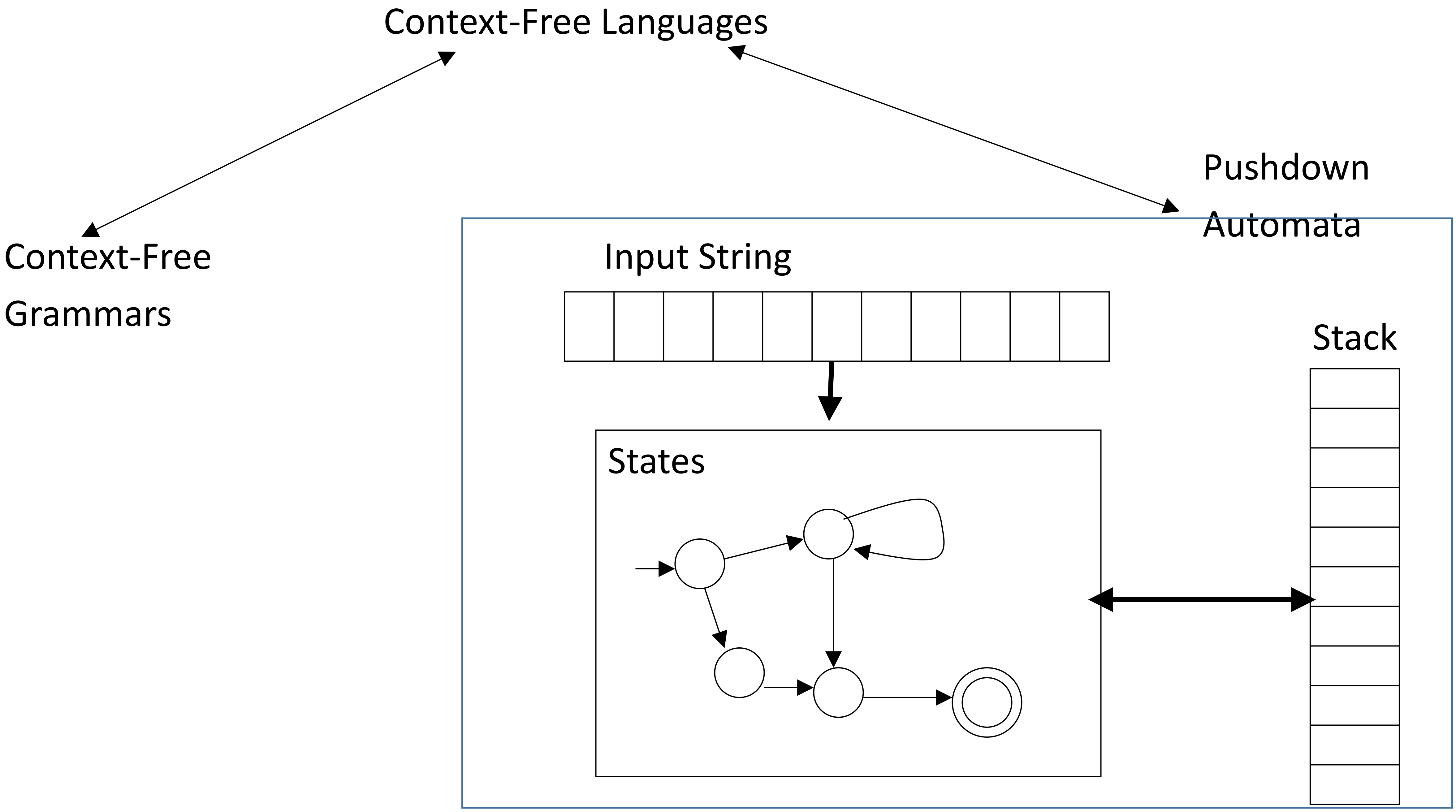
Baris E. Suzek, PhD

Outline

- Last week
- Formal Definition for Turing Machines
- Computing Functions with Turing Machines
- Turing's Thesis
- Variations of the Turing Machine
- Universal Turing Machine
- Countable/uncountable Sets

Context-Free and Regular Languages




Context-Free Languages
Context-Free
Grammars
Pushdown
Automata
Input String
Stack
States

A string is accepted if there is
one computation such that:

All the input is consumed

AND

The last state is a final state

At the end of the computation,
we do not care about the stack contents

A string is rejected

if in every computation with this string:

The input cannot be consumed

OR

The input is consumed and the last state is not a final state

OR

The stack head moves below the bottom of the stack

Formal Definition

Non-Deterministic Pushdown Automaton

NPDA

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

States

Input alphabet

Stack alphabet

Transition function

Initial state

Stack start symbol

Final states

Instantaneous Description

$$(q,\ u,\ s)$$

Current state

Remaining input

Current stack contents

Union

Context-free languages
are closed under: **Union**

L_1 is context free

L_2 is context free

$\Rightarrow$ $L_1 \cup L_2$ is context-free

Concatenation

Context-free languages
are closed under: **Concatenation**

L_1 is context free

L_2 is context free

$\Rightarrow$ $L_1 L_2$ is context-free

Star Operation

Context-free languages
are closed under: **Star-operation**

L is context free $\Rightarrow$ L^* is context-free

Intersection

Context-free languages are **not** closed under: **intersection**

L_1 is context free

L_2 is context free

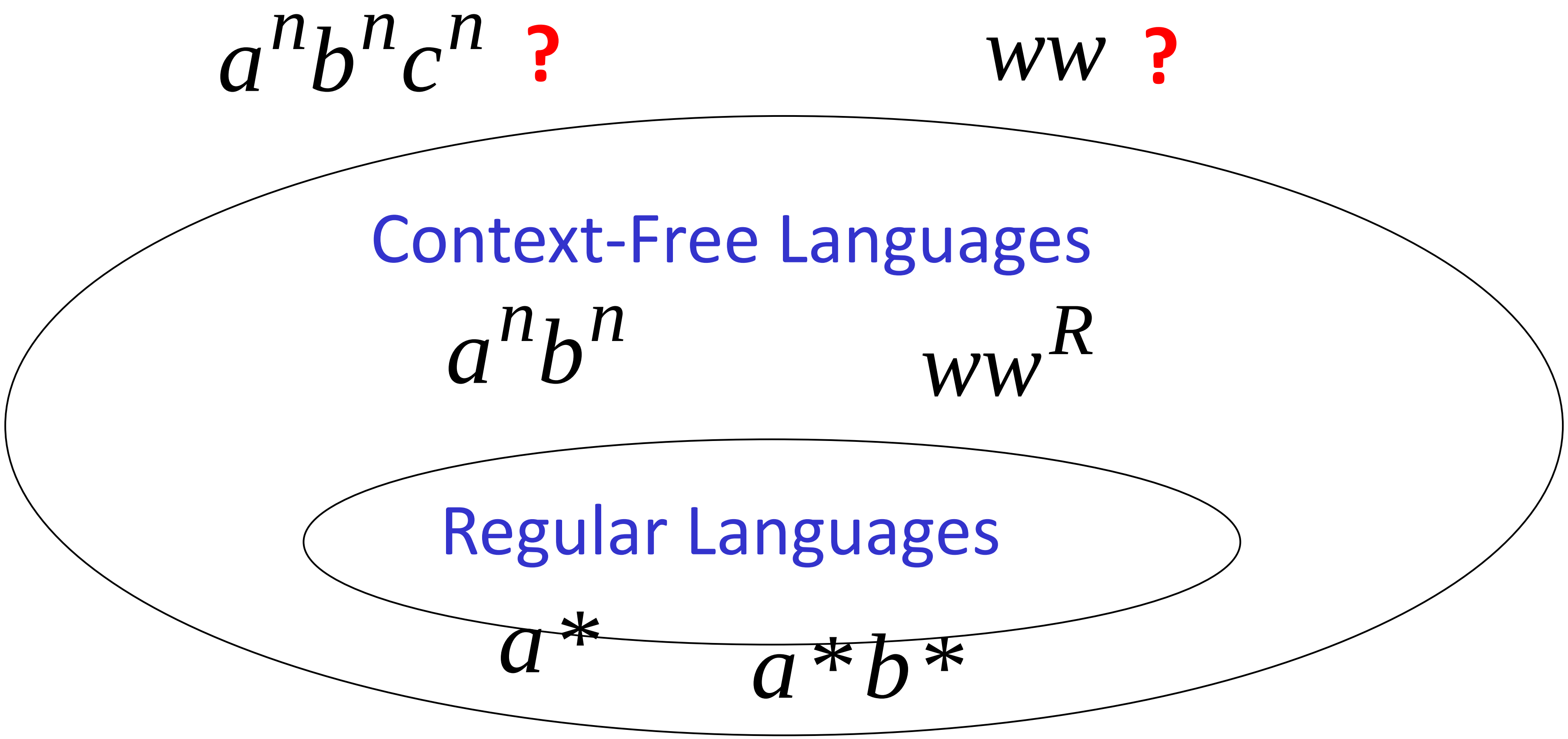
$\Rightarrow$ $L_1 \cap L_2$ **not** necessarily context-free

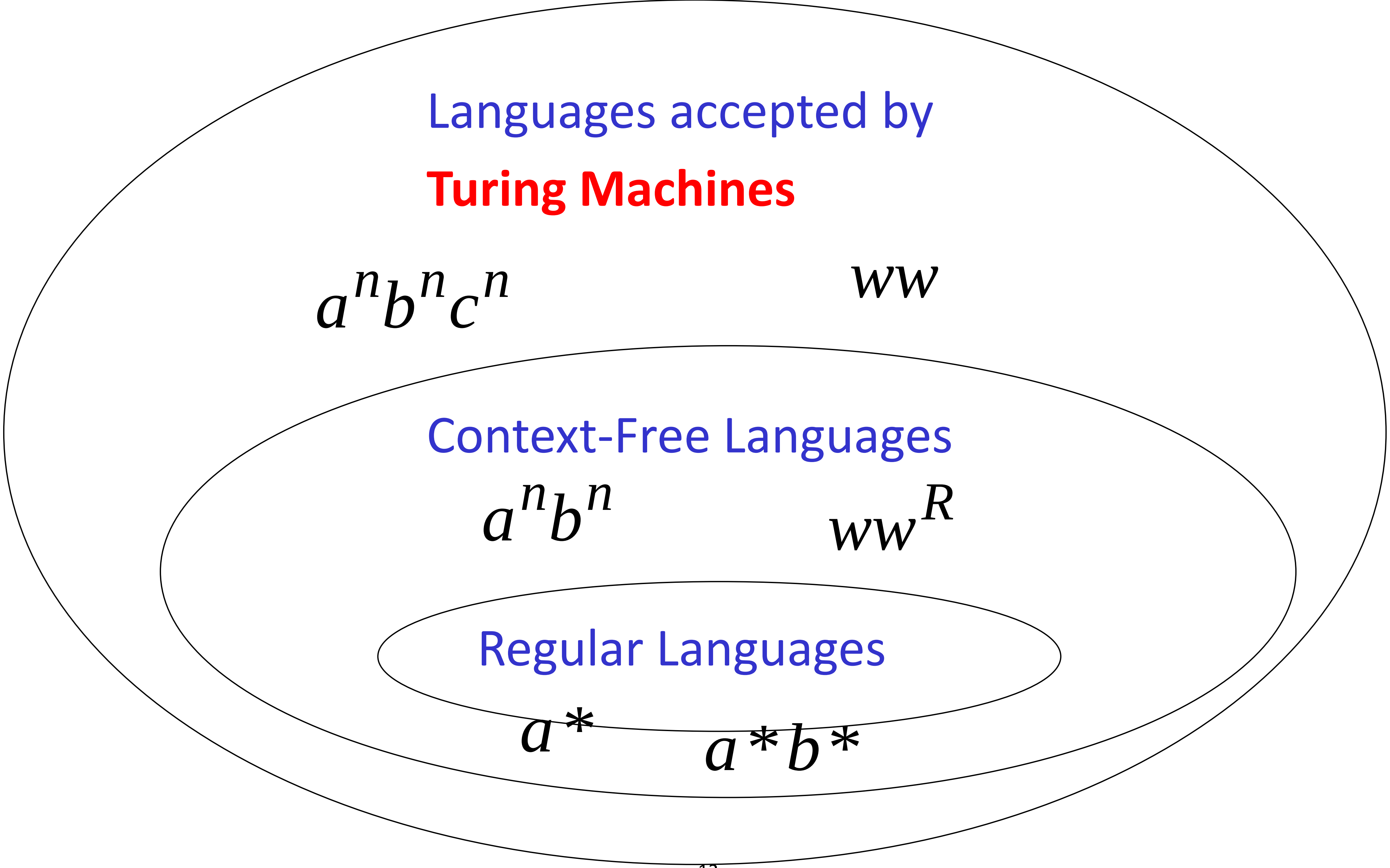
Complement

Context-free languages are **not** closed under: **complement**

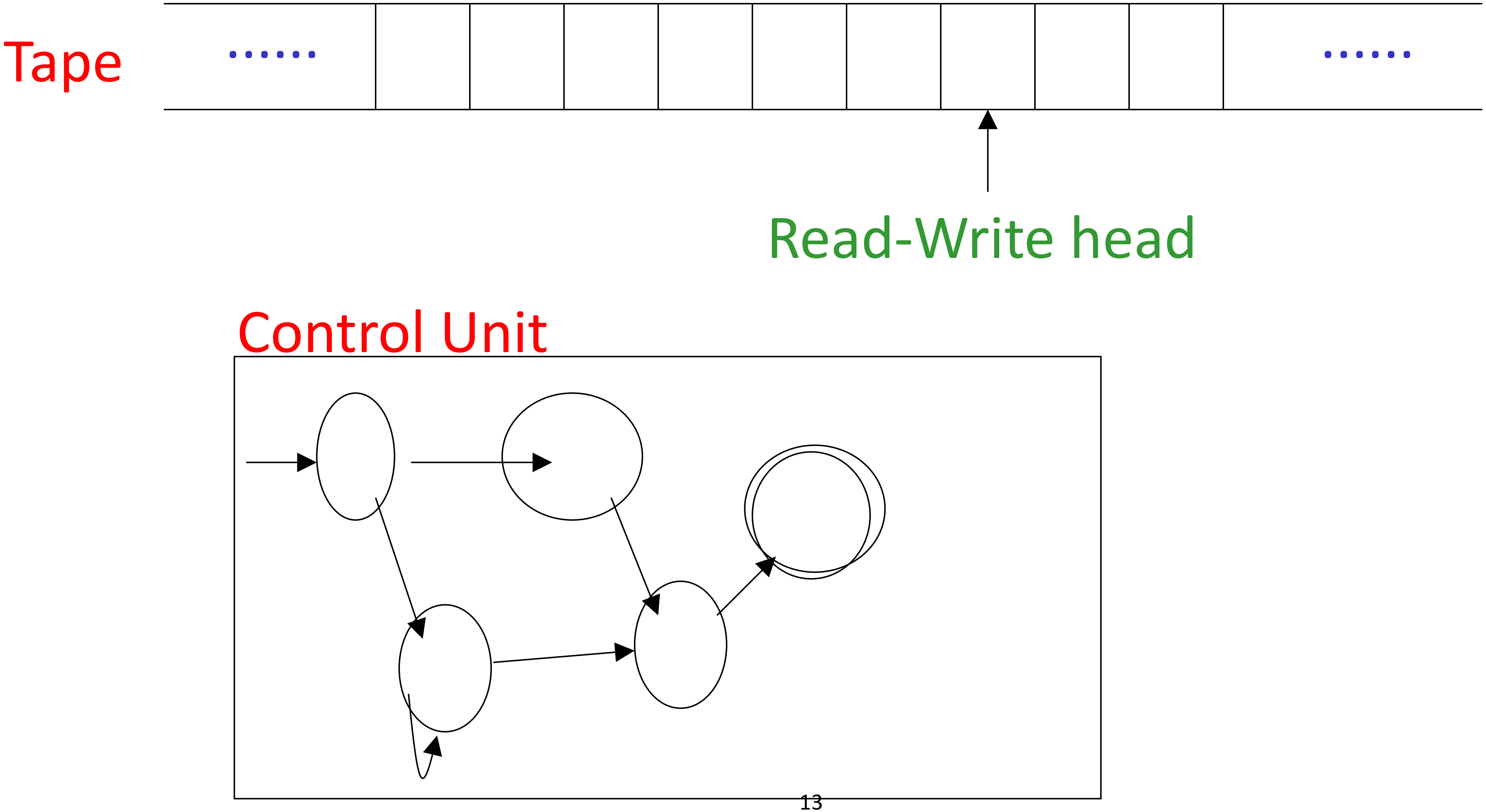
L is context free $\Rightarrow$ $\overline{L}$ **not** necessarily context-free

The Language Hierarchy



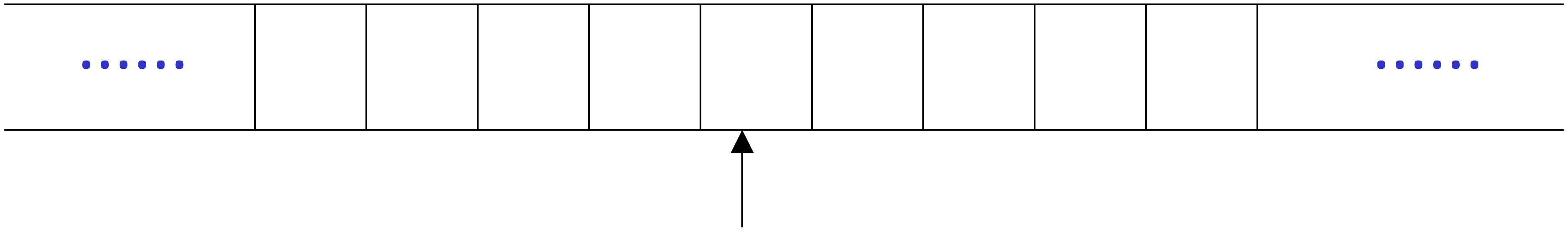

Languages accepted by
Turing Machines
$a^n b^n c^n$
ww
Context-Free Languages
$a^n b^n$
ww^R
Regular Languages
a^*
$a^* b^*$

A Turing Machine



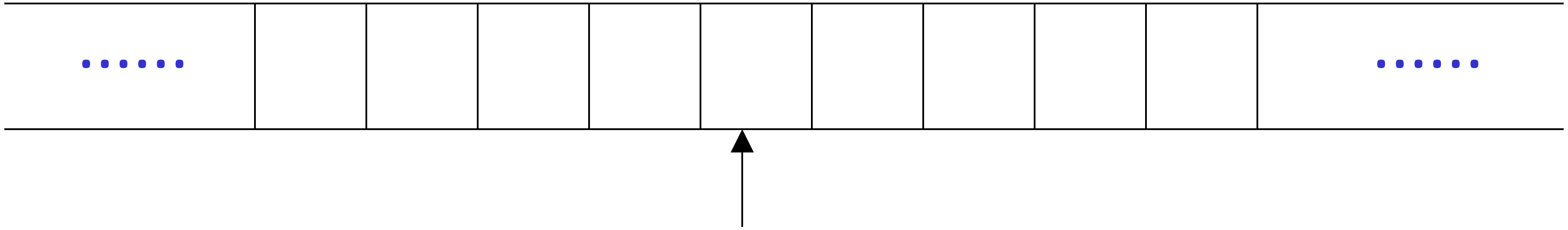
The Tape

No boundaries -- infinite length



Read-Write head

The head moves Left or Right

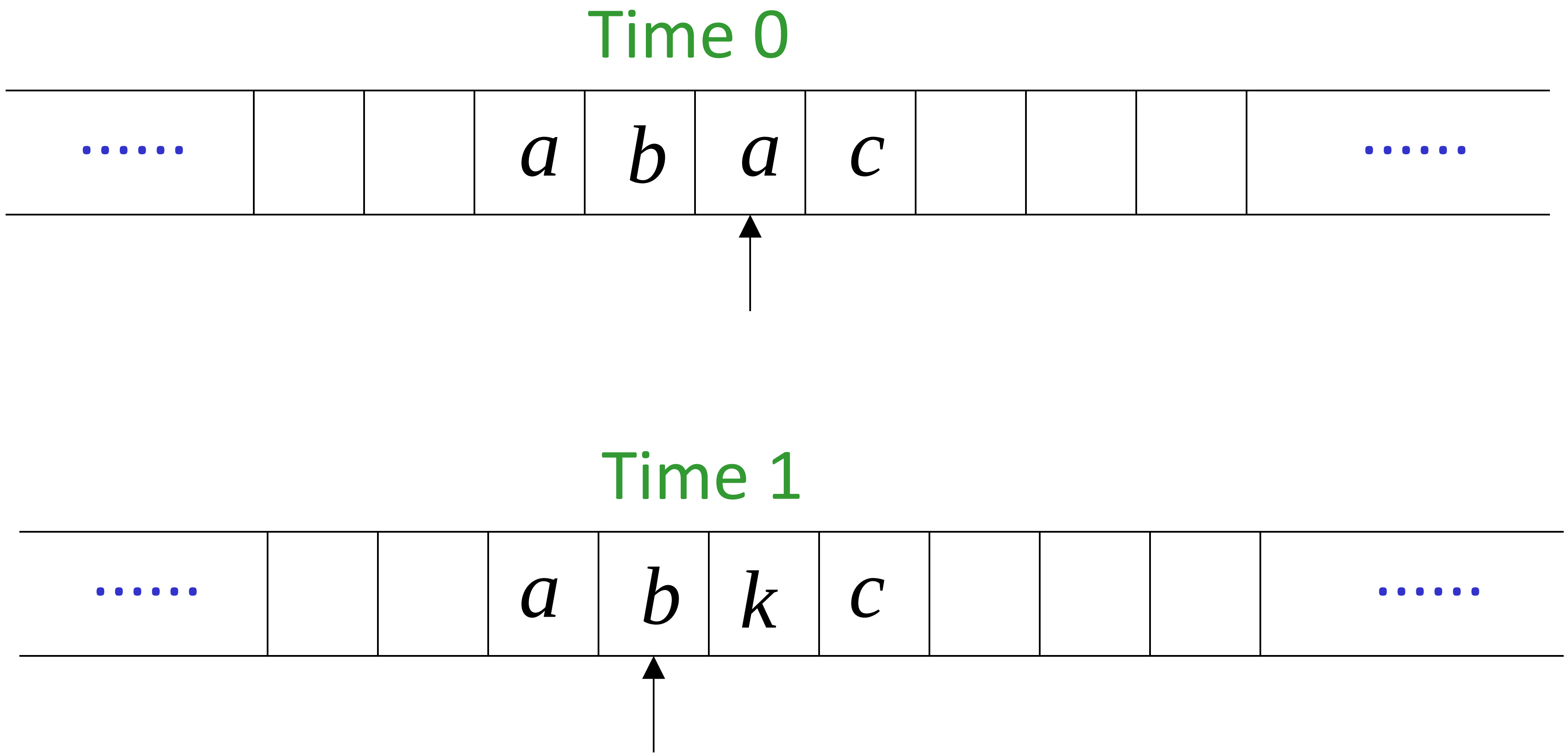


Read-Write head

The head at each time step:

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

Example:

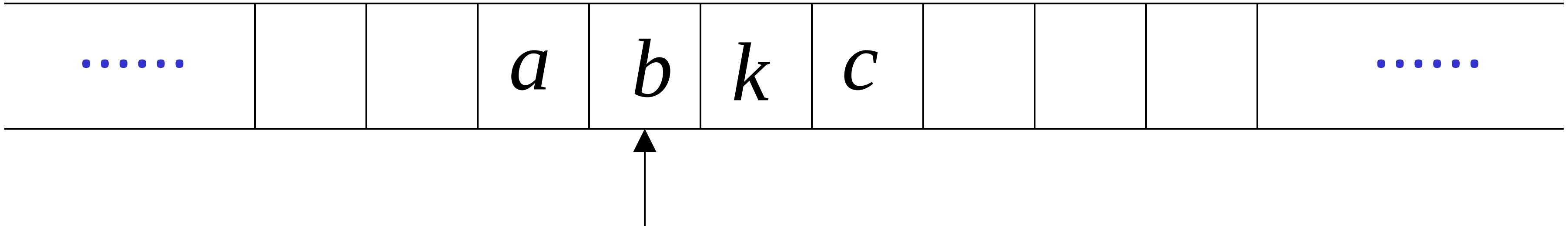


1. Reads a

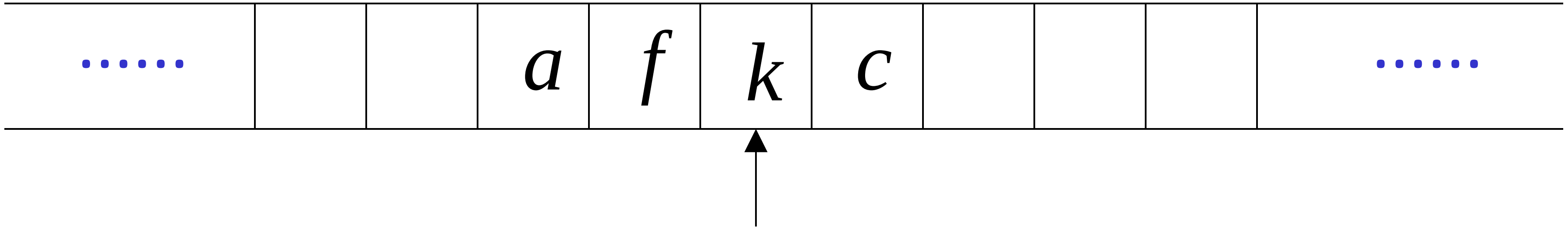
2. Writes k

3. Moves Left

Time 1

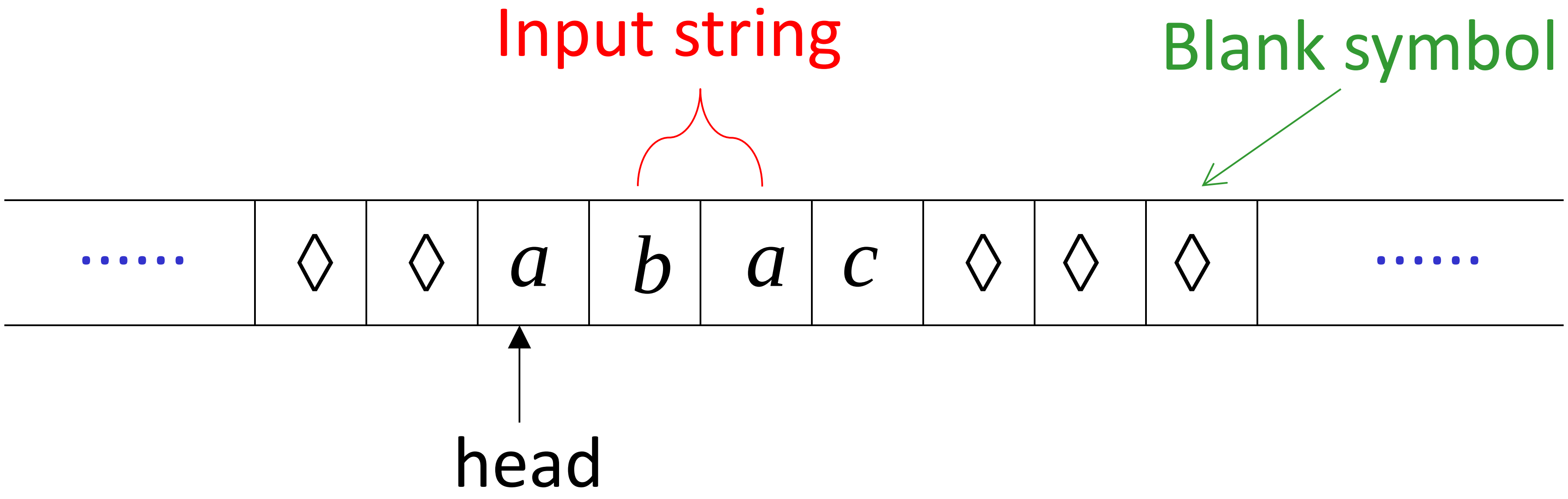


Time 2

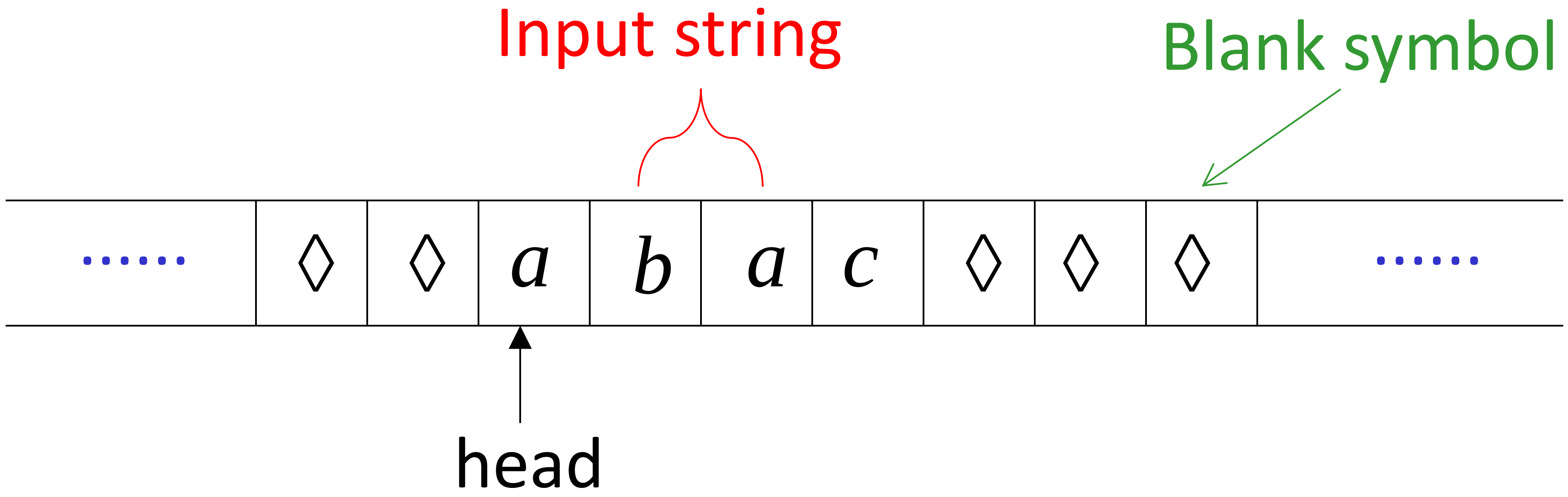


1. Reads b
2. Writes f
3. Moves Right

The Input String

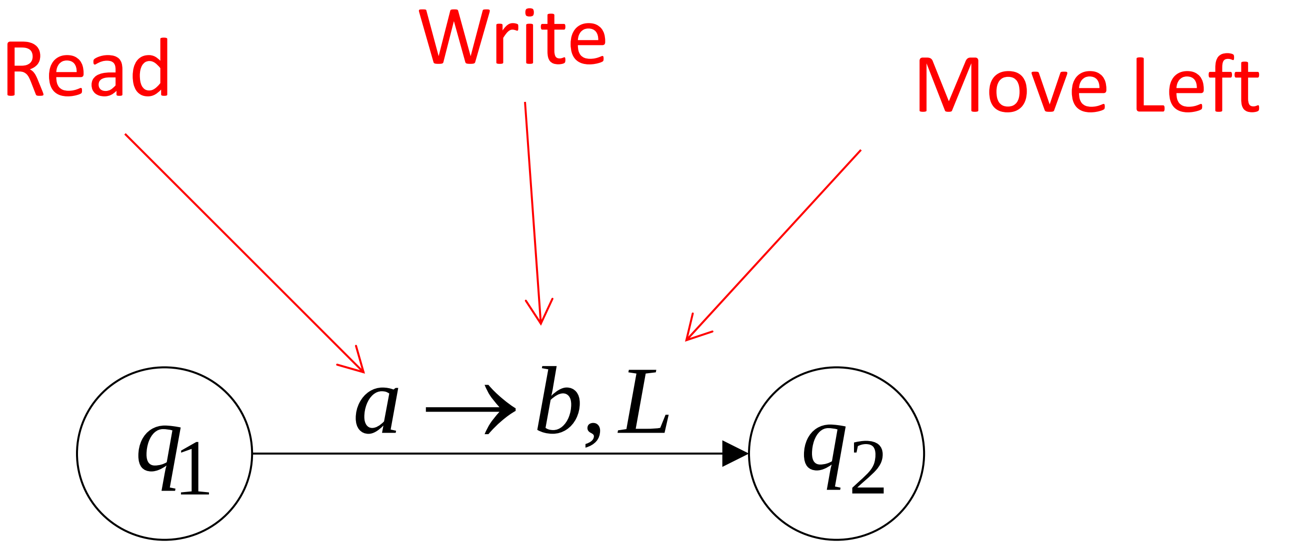


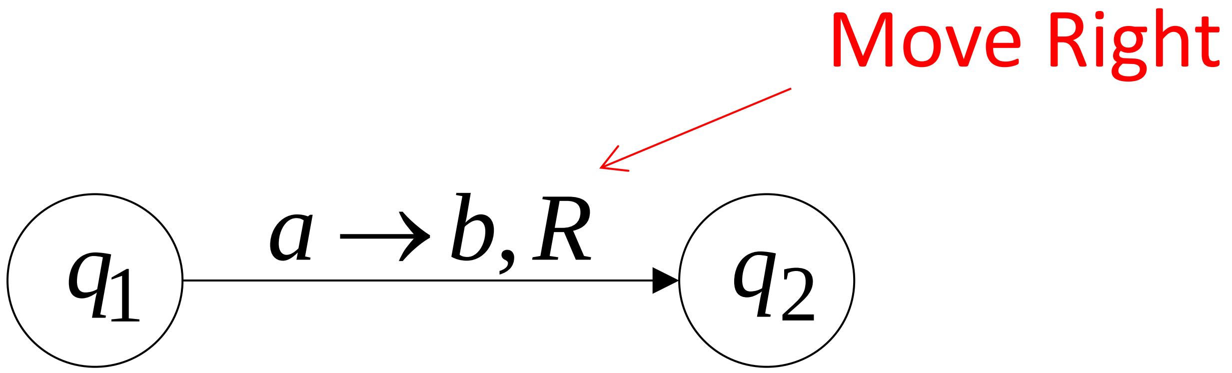
Head starts at the leftmost position of the input string



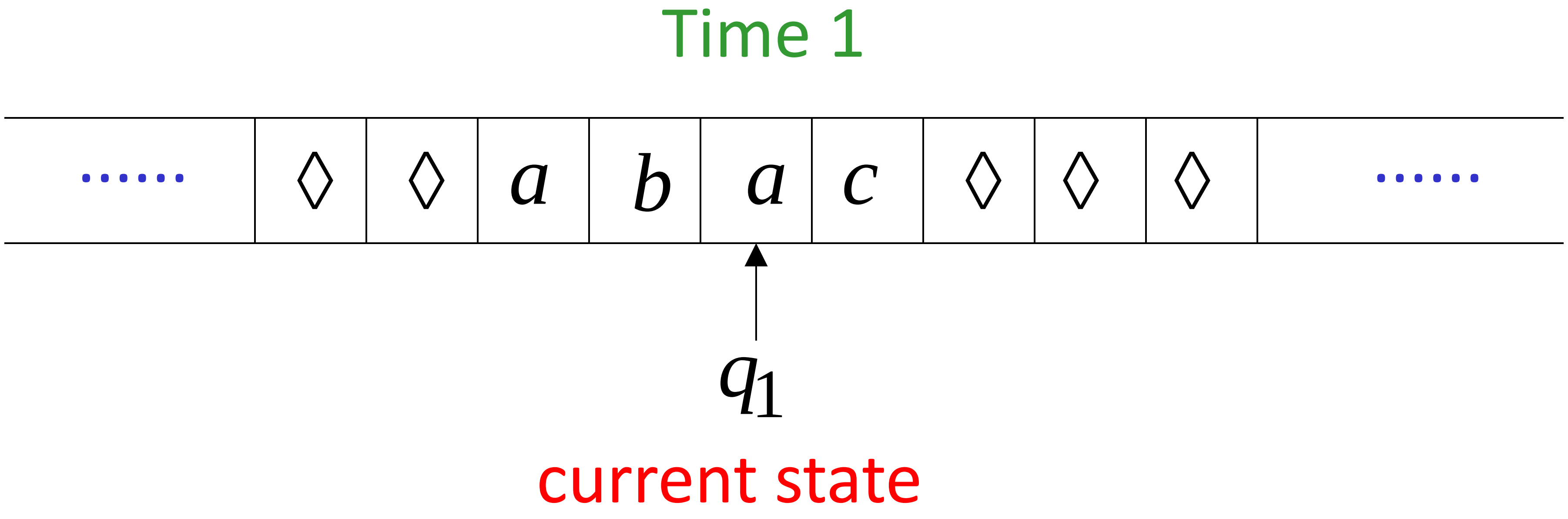
Remark: The input string is never empty

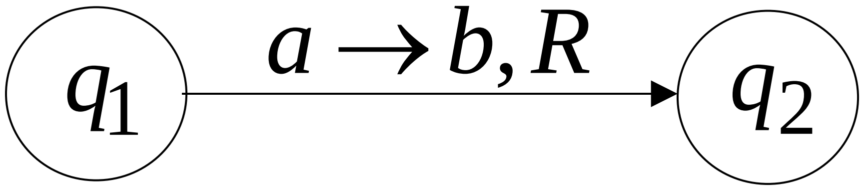
States & Transitions



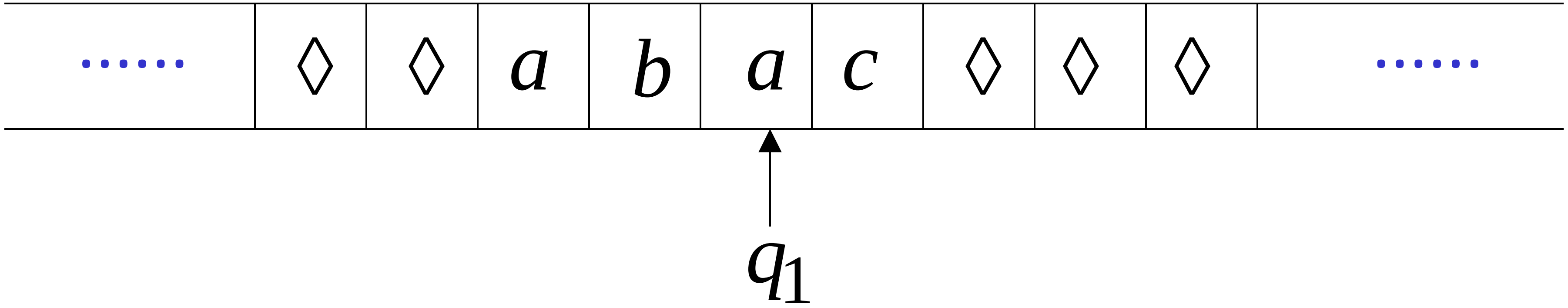


Example:

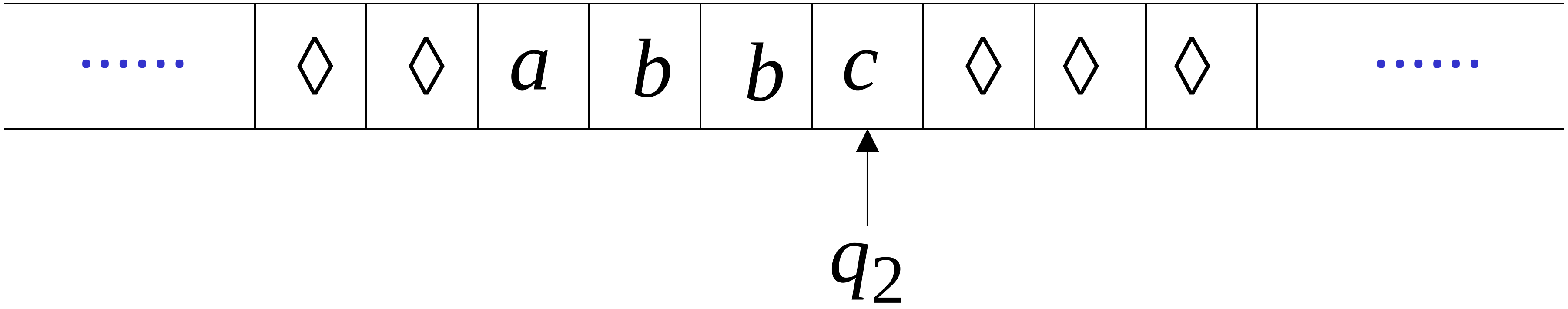


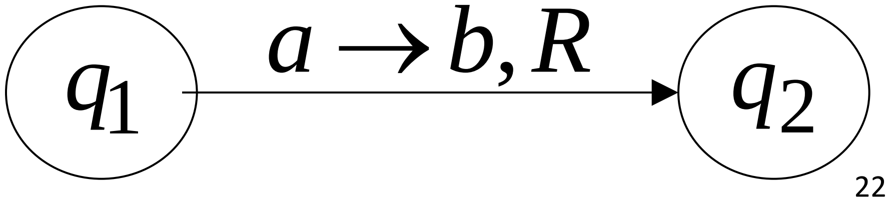


Time 1



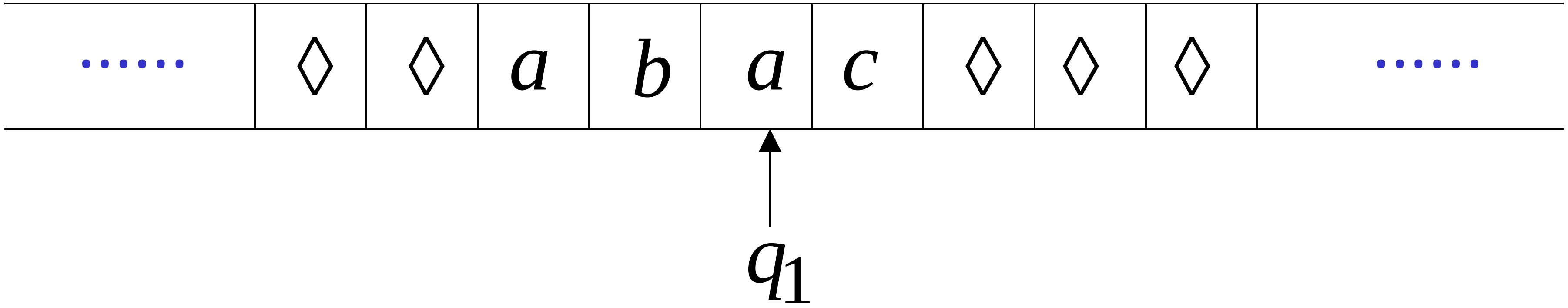
Time 2



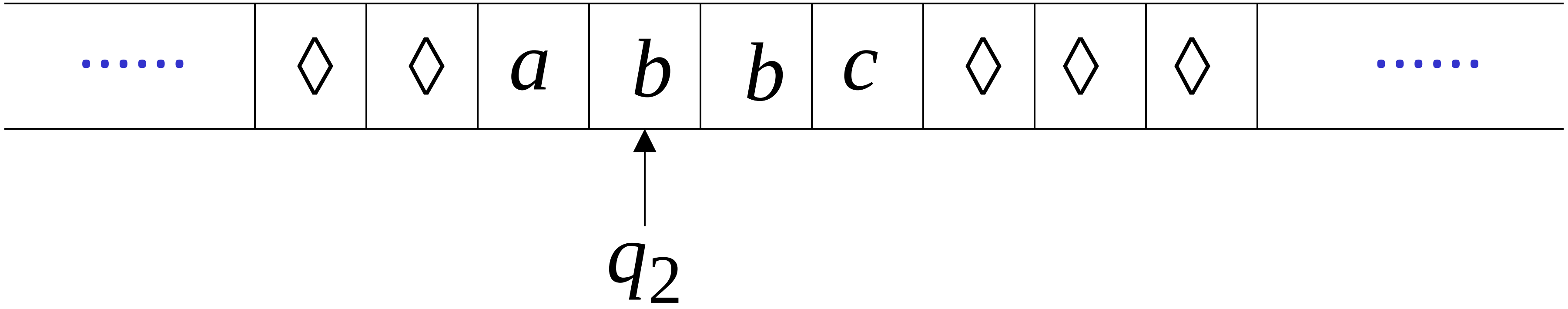


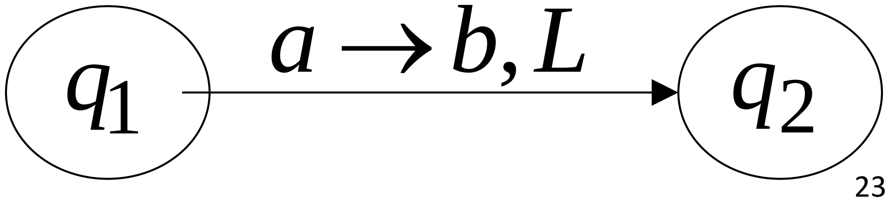
Example:

Time 1

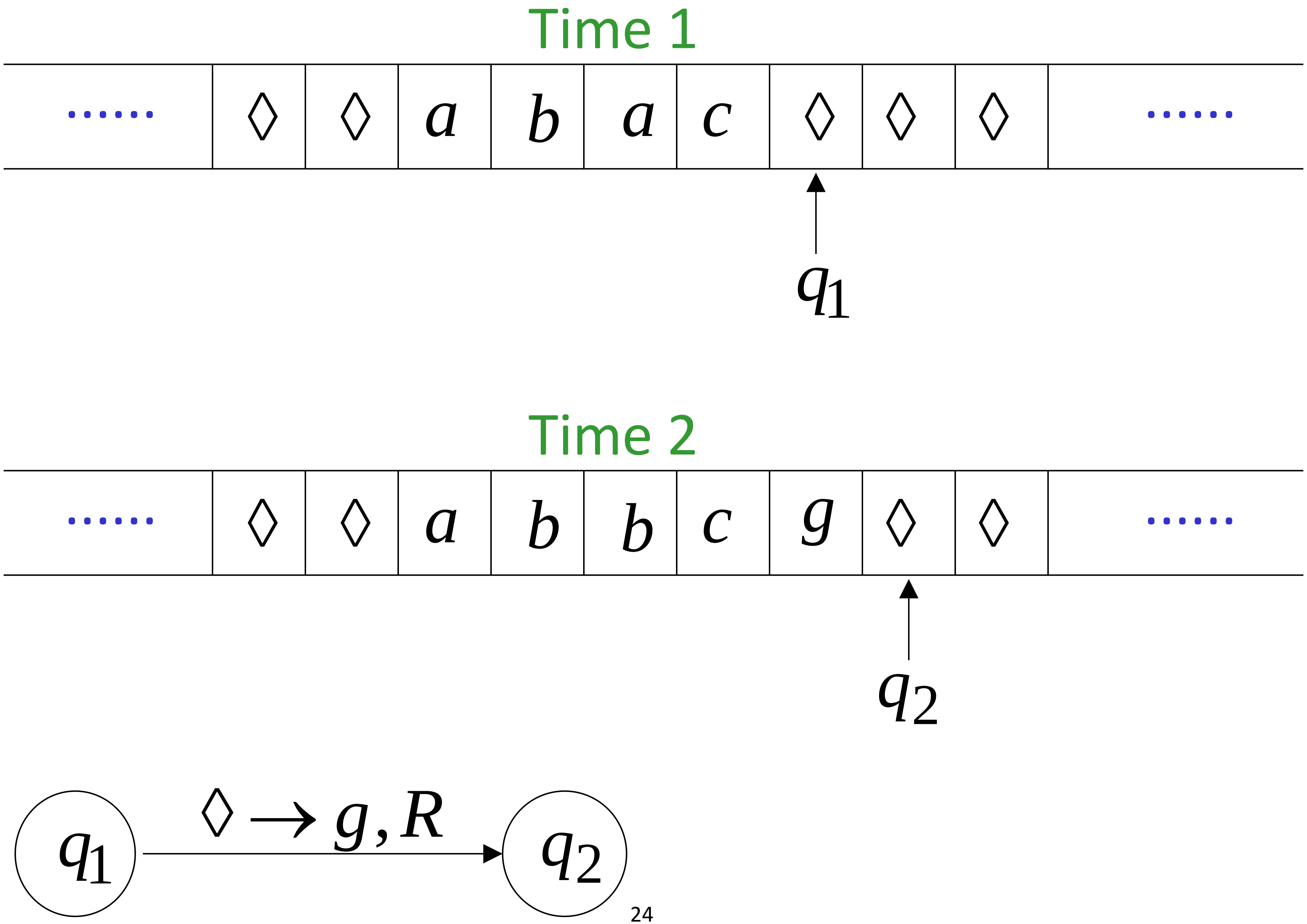


Time 2





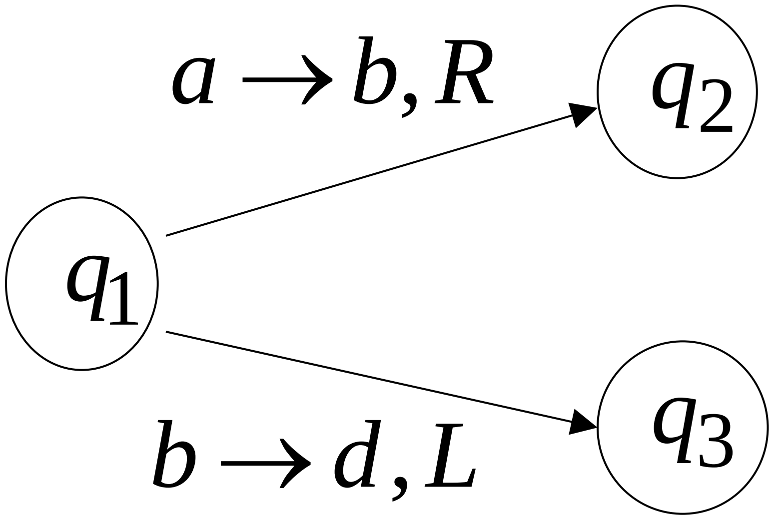
Example:



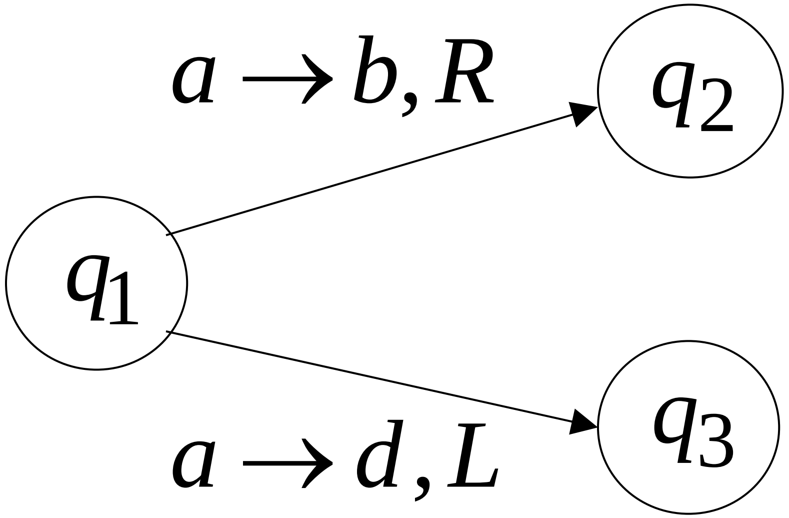
Determinism

Turing Machines are deterministic

Allowed



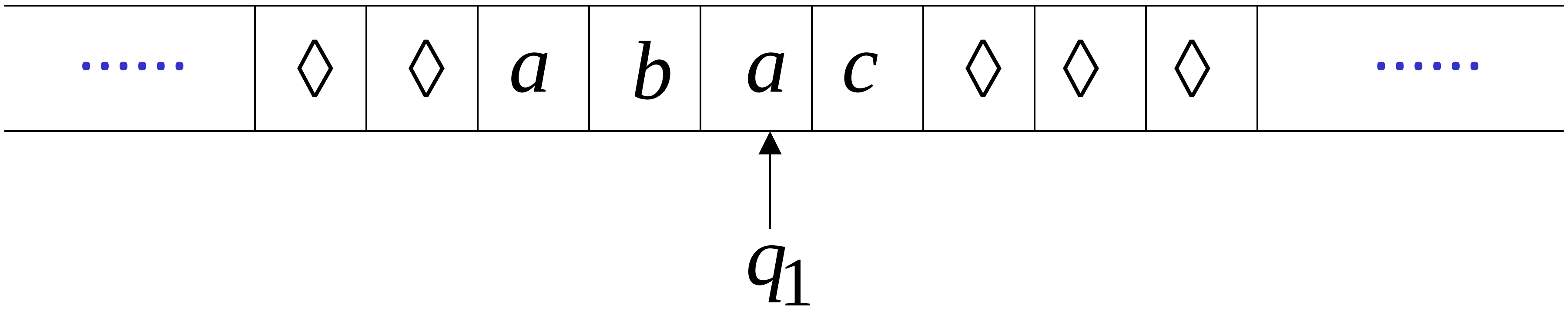
Not Allowed

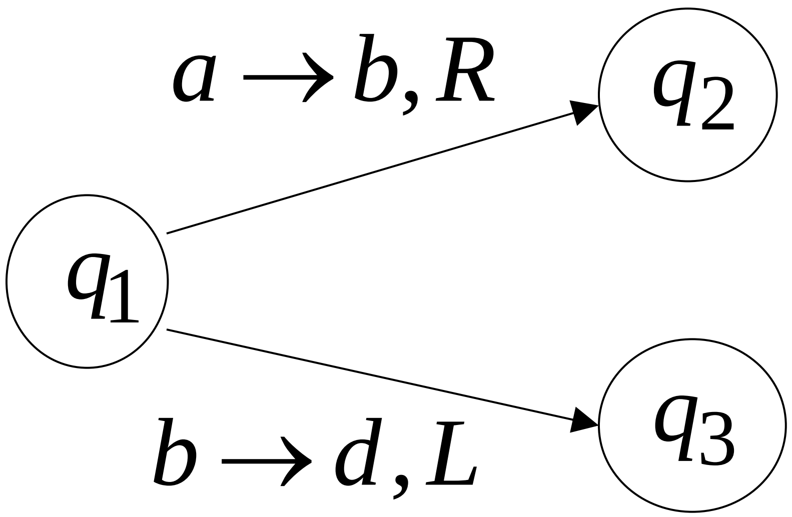


No lambda transitions allowed

Partial Transition Function

Example:





Allowed:

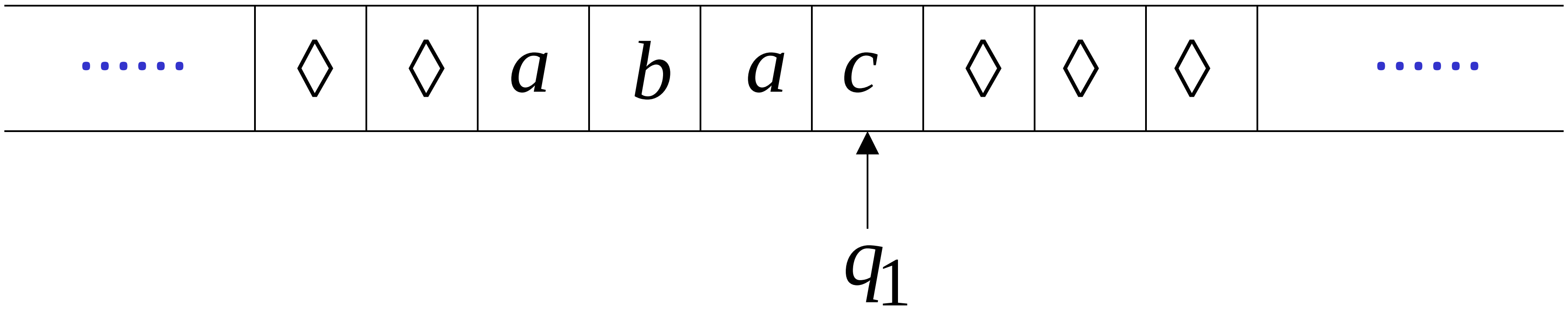
No transition

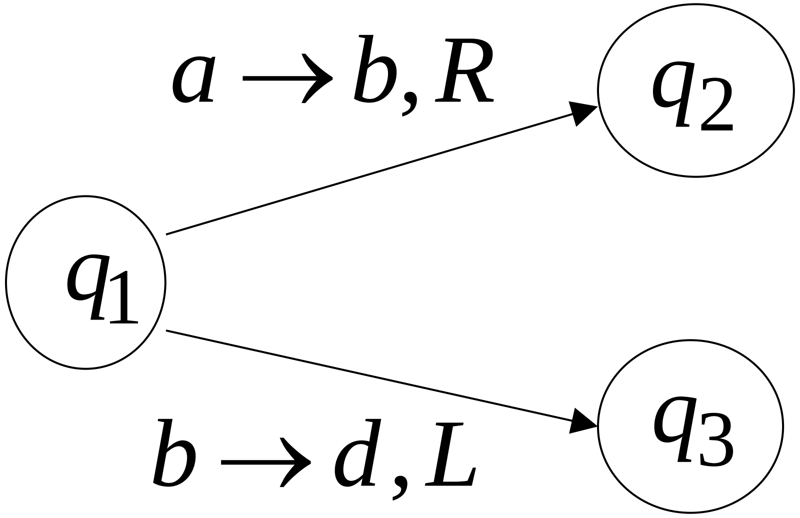
for input symbol c

Halting

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

Example:



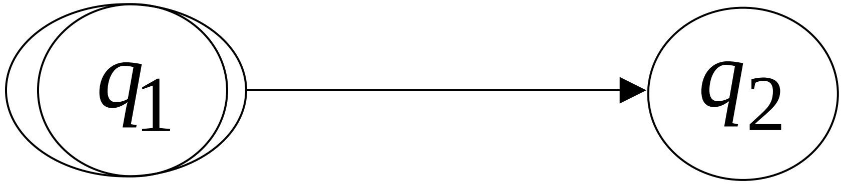


No possible transition

HALT!!!

Final States

Allowed



Not Allowed

- Final states have no outgoing transitions

- In a final state the machine halts

Acceptance

Accept Input ⇒

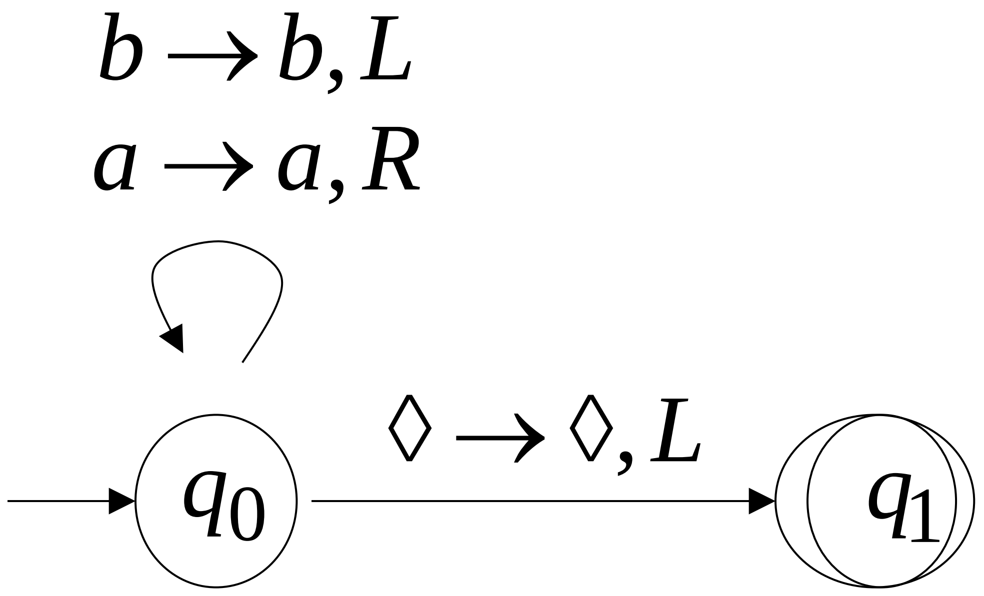
If machine halts
in a final state

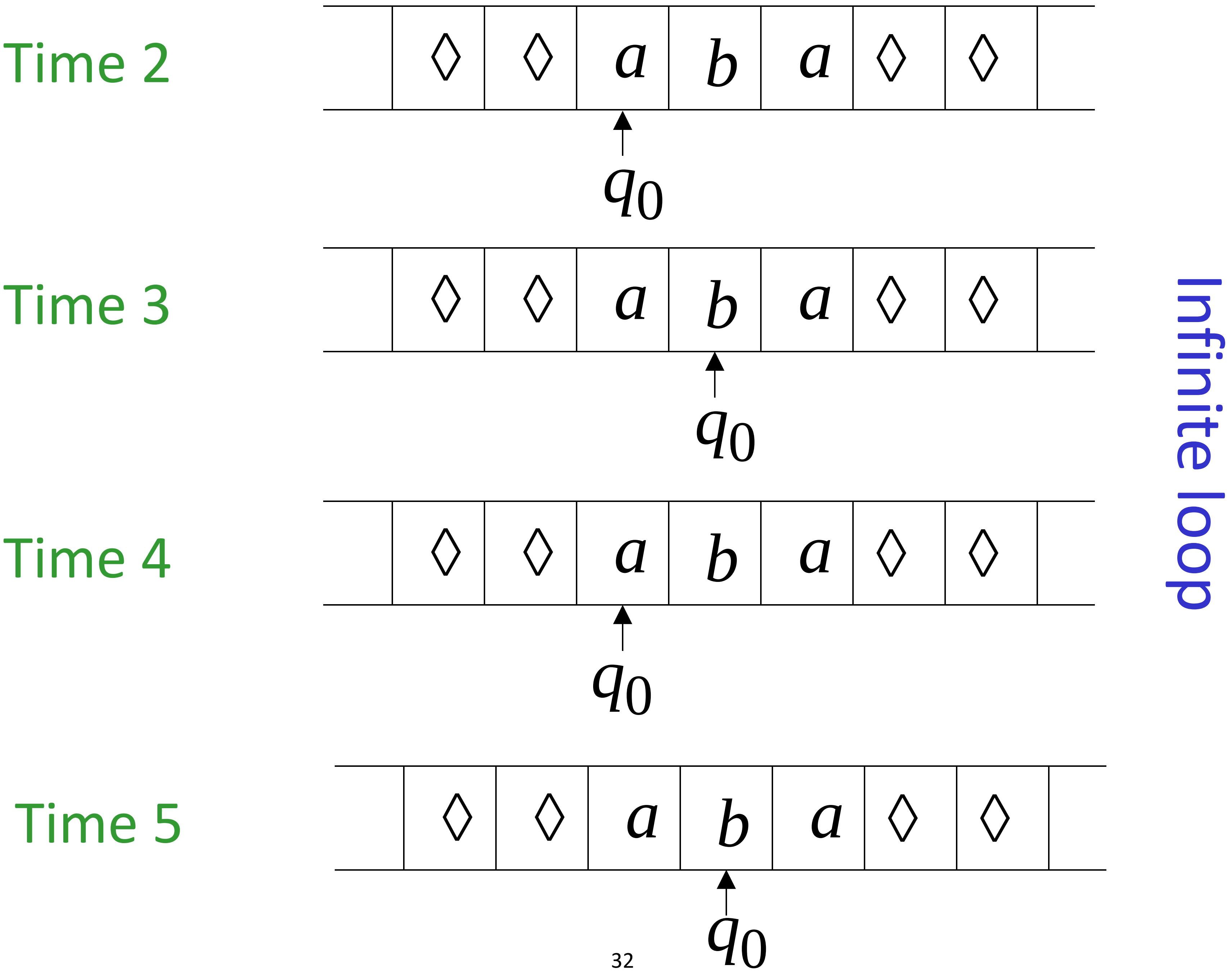
Reject Input ⇒

If machine halts
in a non-final state
or
If machine enters
an *infinite loop*

Infinite Loop Example

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




Time 2
◊ ◊ a b a ◊ ◊
q0
Time 3
◊ ◊ a b a ◊ ◊
q0
Time 4
◊ ◊ a b a ◊ ◊
q0
Time 5
◊ ◊ a b a ◊ ◊
q0
Infinite loop

Because of the infinite loop:

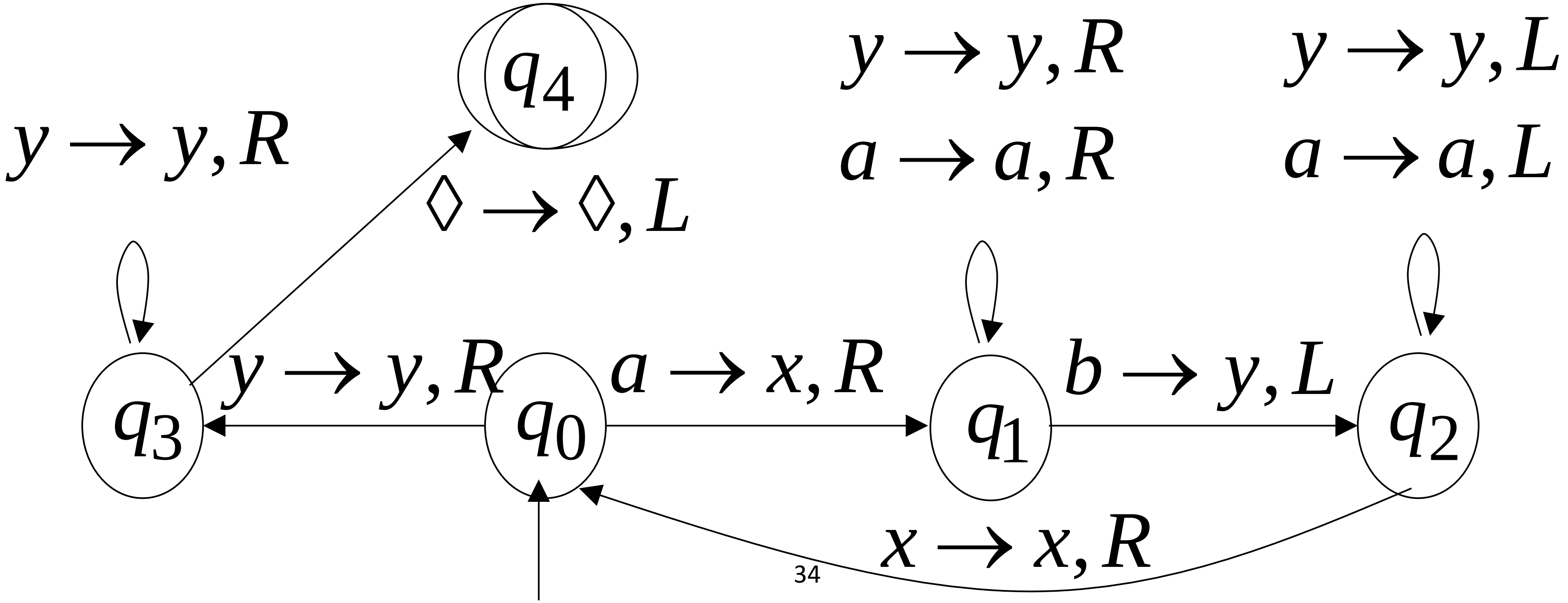
- The final state cannot be reached

- The machine never halts

- The input is not accepted

Another Turing Machine Example

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



Standard Turing Machine

The machine we described is the standard:

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

Outline

- Last week
- Formal Definition for Turing Machines
- Computing Functions with Turing Machines
- Turing’s Thesis
- Variations of the Turing Machine
- Universal Turing Machine
- Countable/uncountable Sets

Transition Function

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

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

Transition Function

$$q_1 \xrightarrow{c \rightarrow d, L} q_2$$

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

Turing Machine:

Input alphabet

Tape alphabet

States

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

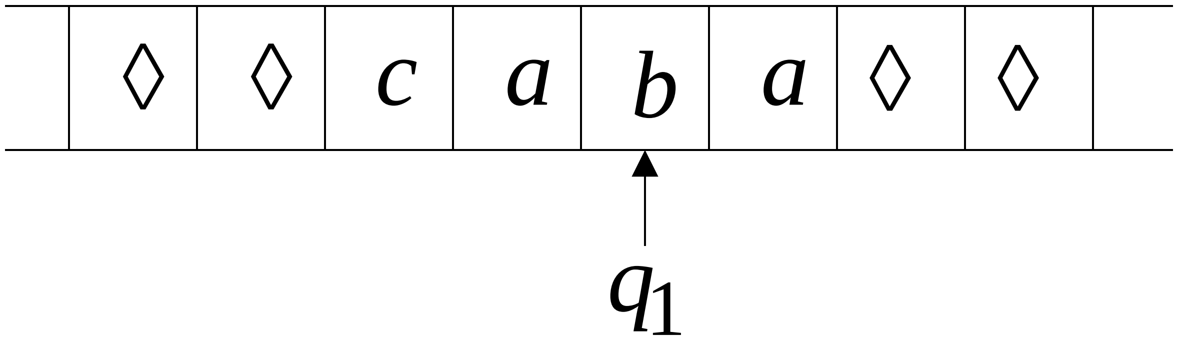
Transition function

Final states

Initial state

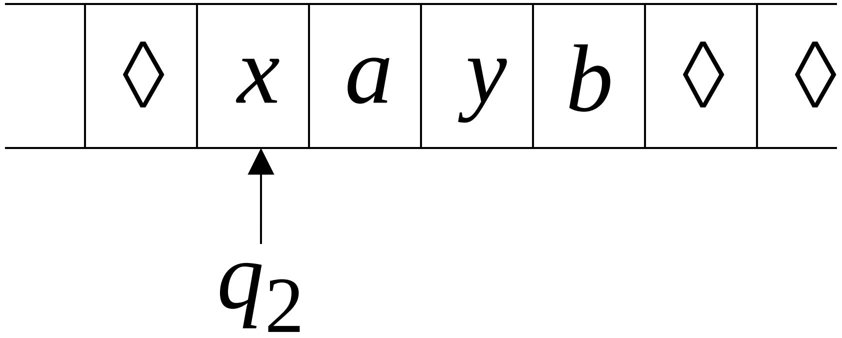
blank

Configuration

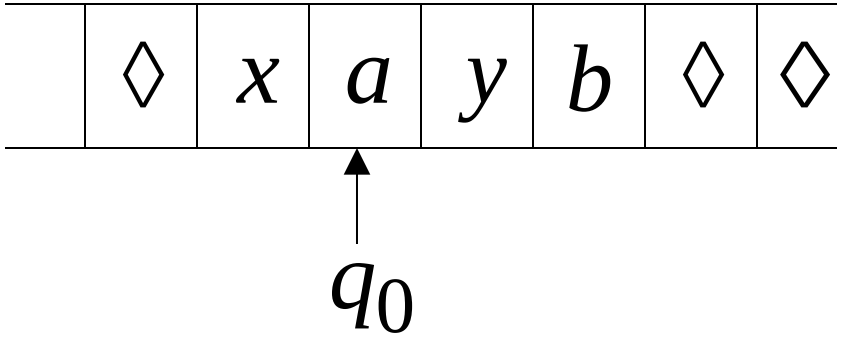


Instantaneous description: $ca\ q_1\ ba$

Time 4

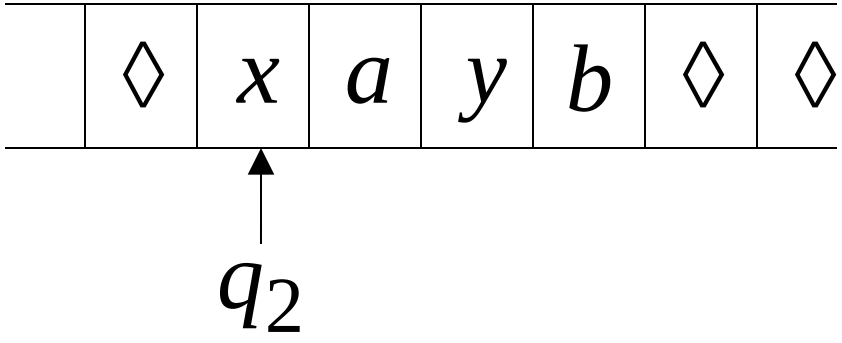


Time 5

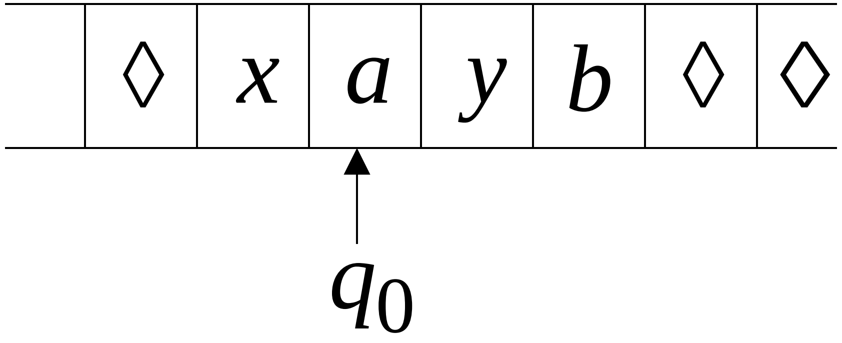


A Move: $q_2 \; xayb \succ x \; q_0 \; ayb$

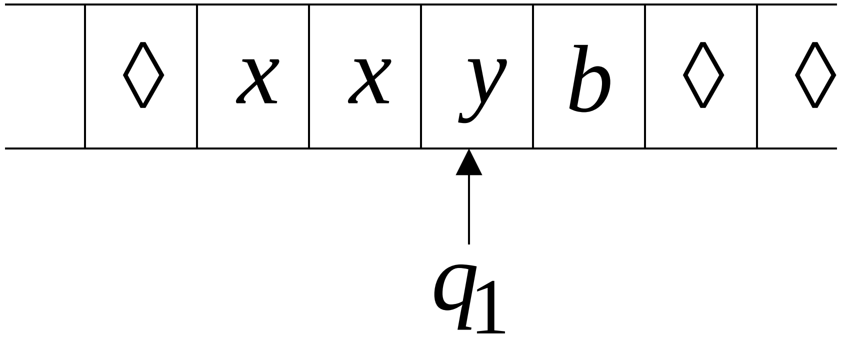
Time 4



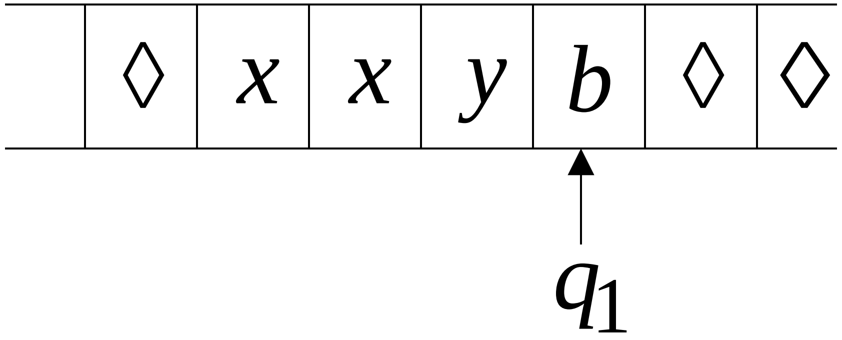
Time 5



Time 6



Time 7



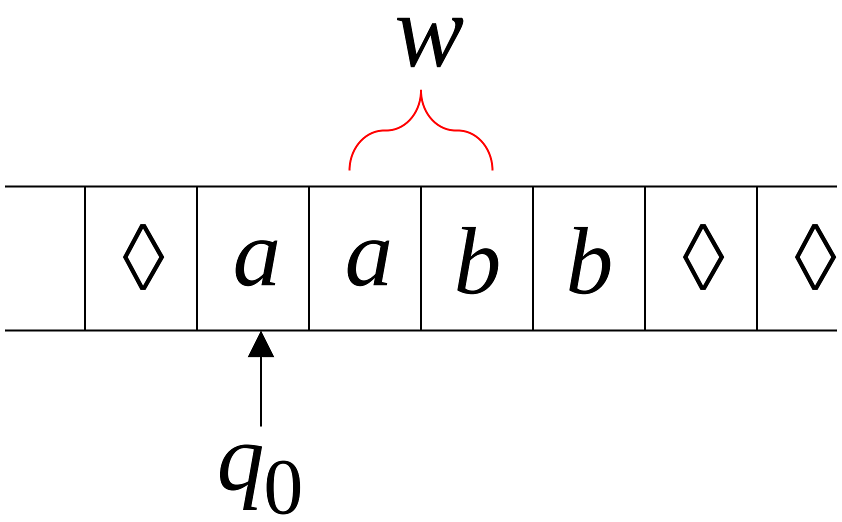
$$q_2\, xayb \succ x\, q_0\, ayb \succ xx\, q_1\, yb \succ xxy\, q_1\, b$$

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

Equivalent notation: $$q_2\ xayb \overset{*}{\succ} xxy\, q_1\, b$$

Initial configuration: $q_0 \; w$

Input string



The Accepted Language

For any Turing Machine M

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

Initial state

Final state

Outline

- Last week
- Formal Definition for Turing Machines
- Computing Functions with Turing Machines
- Turing's Thesis
- Variations of the Turing Machine
- Universal Turing Machine
- Countable/uncountable Sets

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 such that:

Initial configuration

	$\Diamond$	w	$\Diamond$	

q_0 initial state

Final configuration

	$\Diamond$	$f(w)$	$\Diamond$	

q_f final state

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

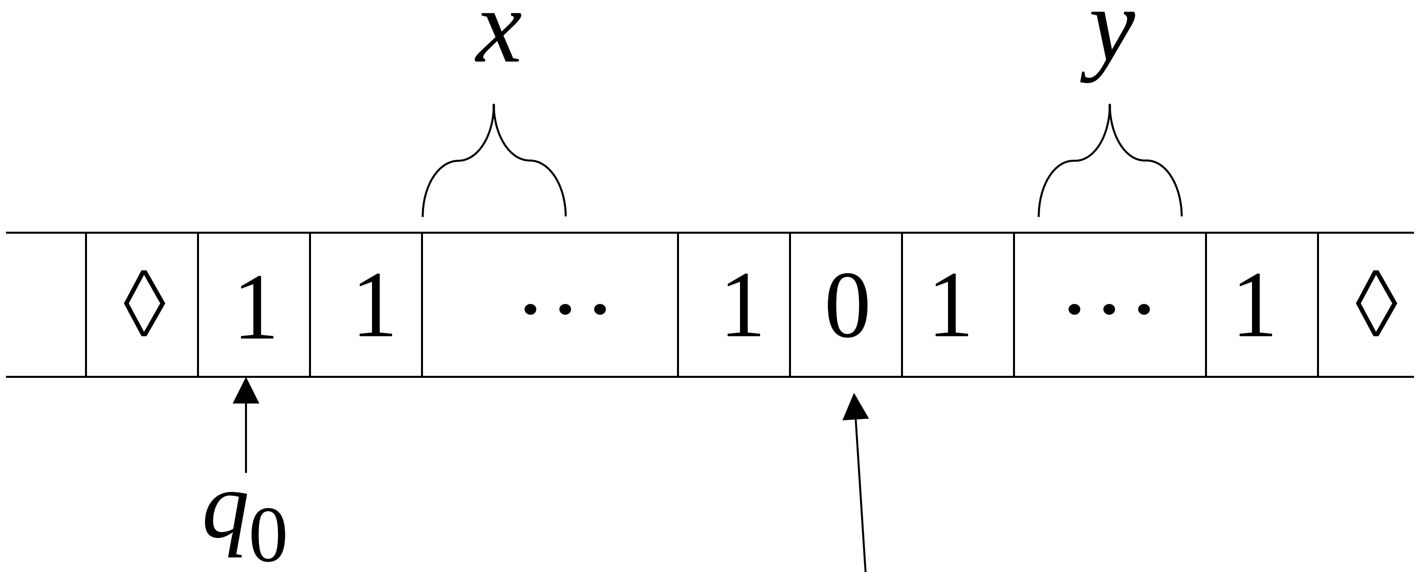
x, y are integers

Turing Machine:

Input string: $x0y$ unary

Output string: $xy0$ unary

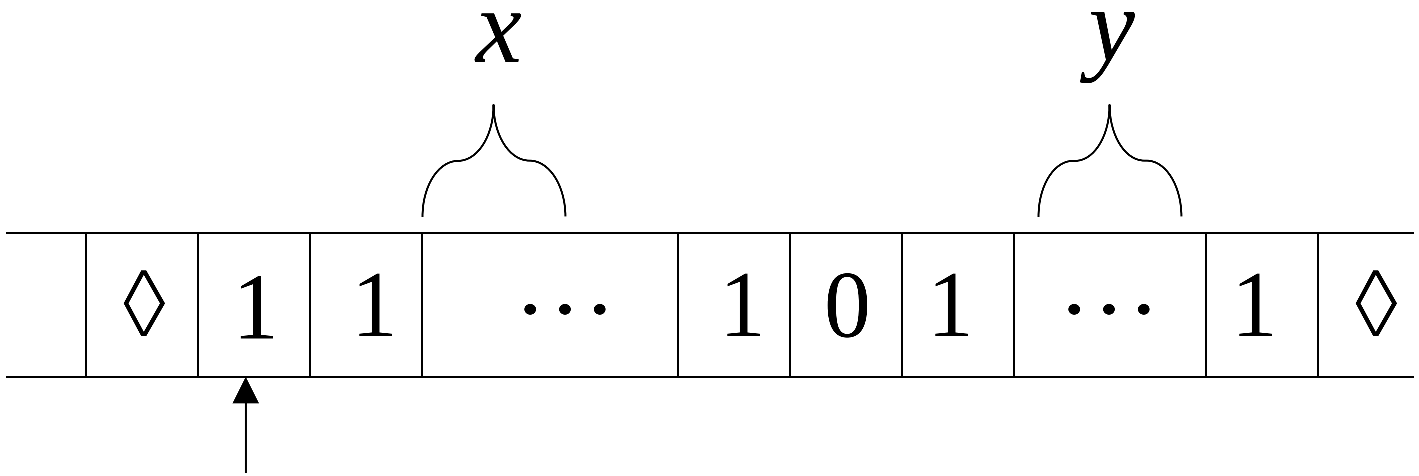
Start



initial state

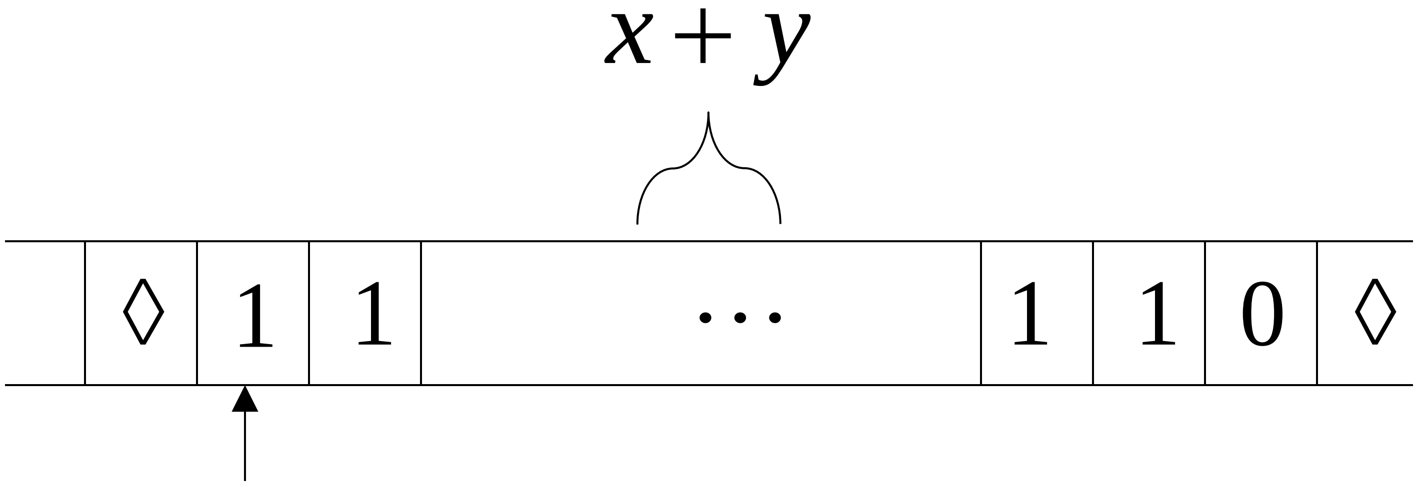
The 0 is the delimiter that separates the two numbers

Start



q_0 initial state

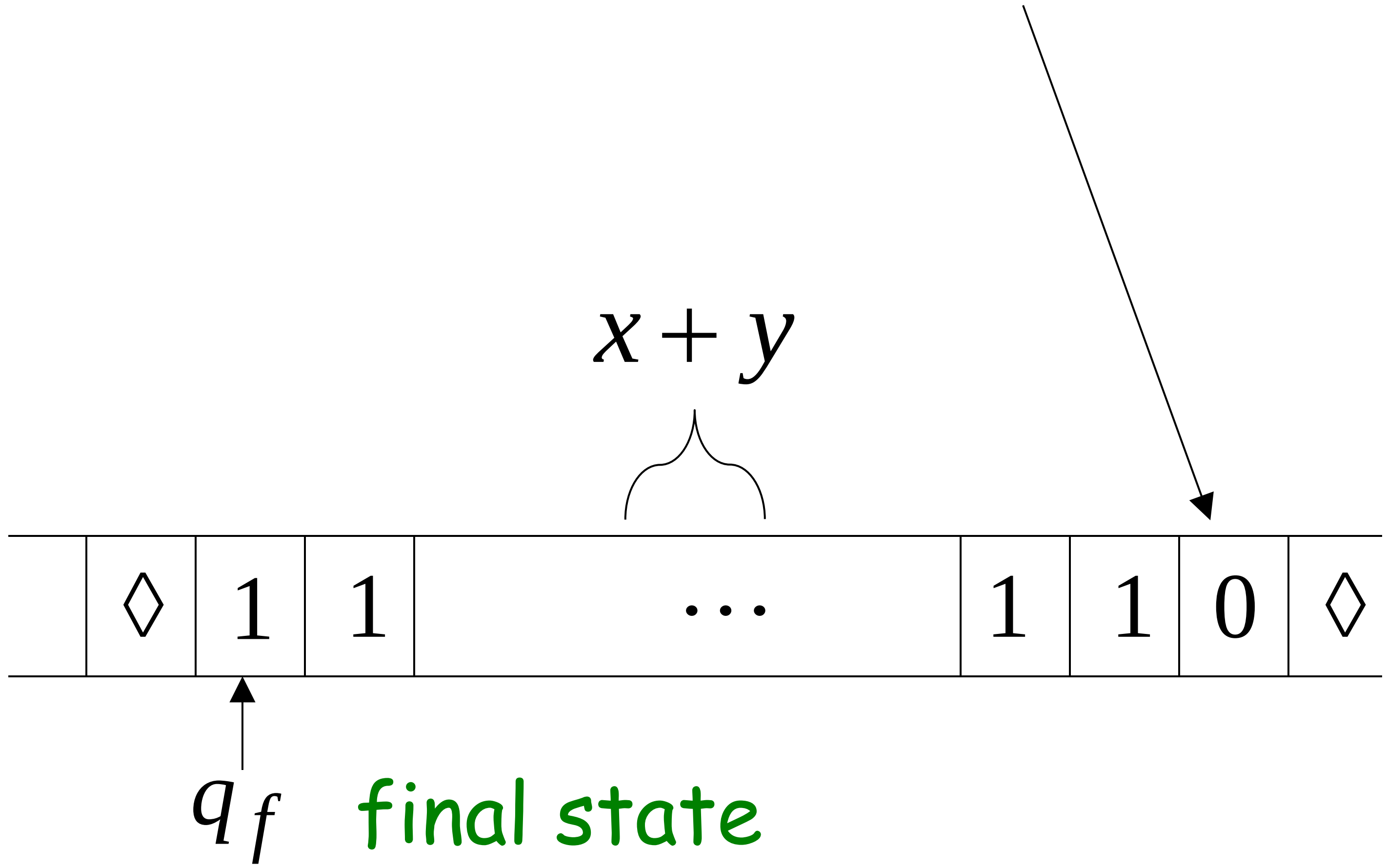
Finish



q_f final state

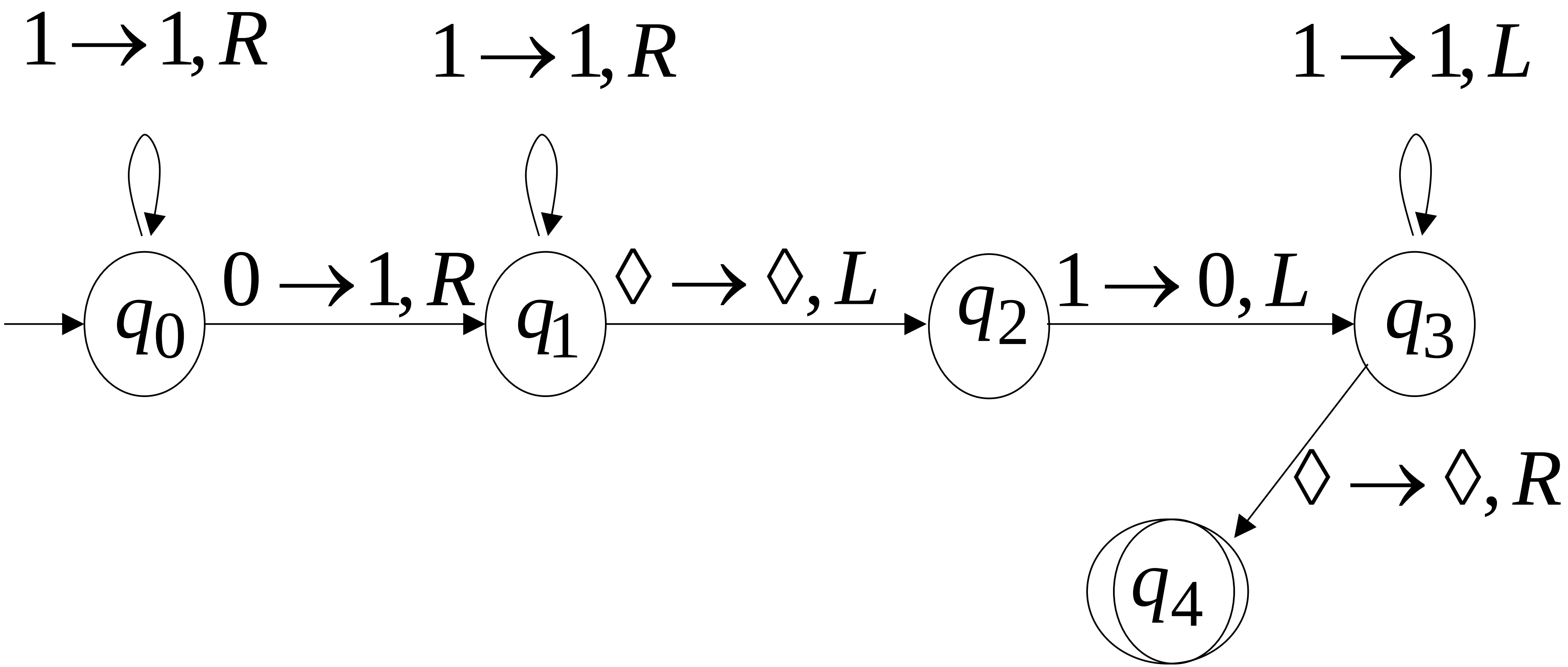
The 0 helps when we use
the result for other operations

Finish



final state

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

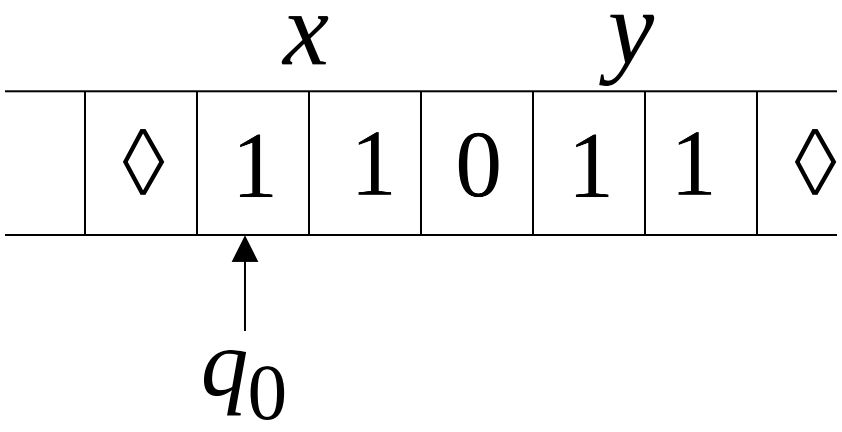


Execution Example:

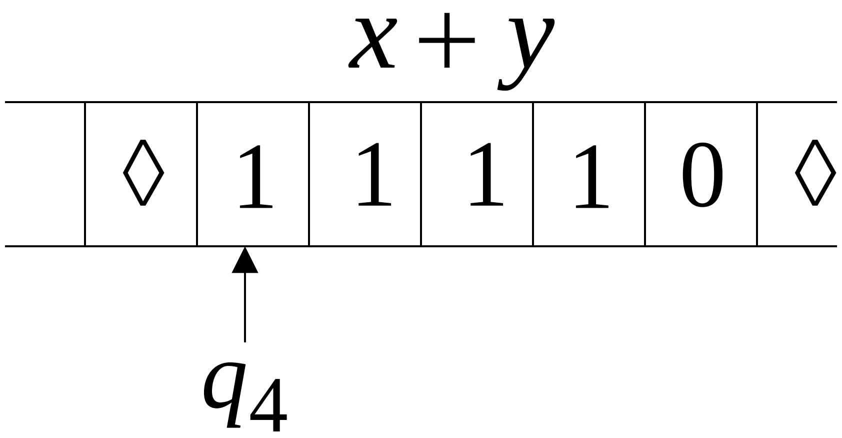
$x = 11$ (2)

$y = 11$ (2)

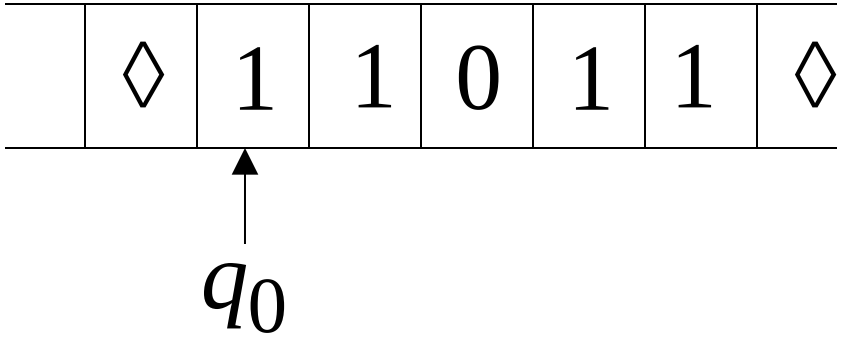
Time 0



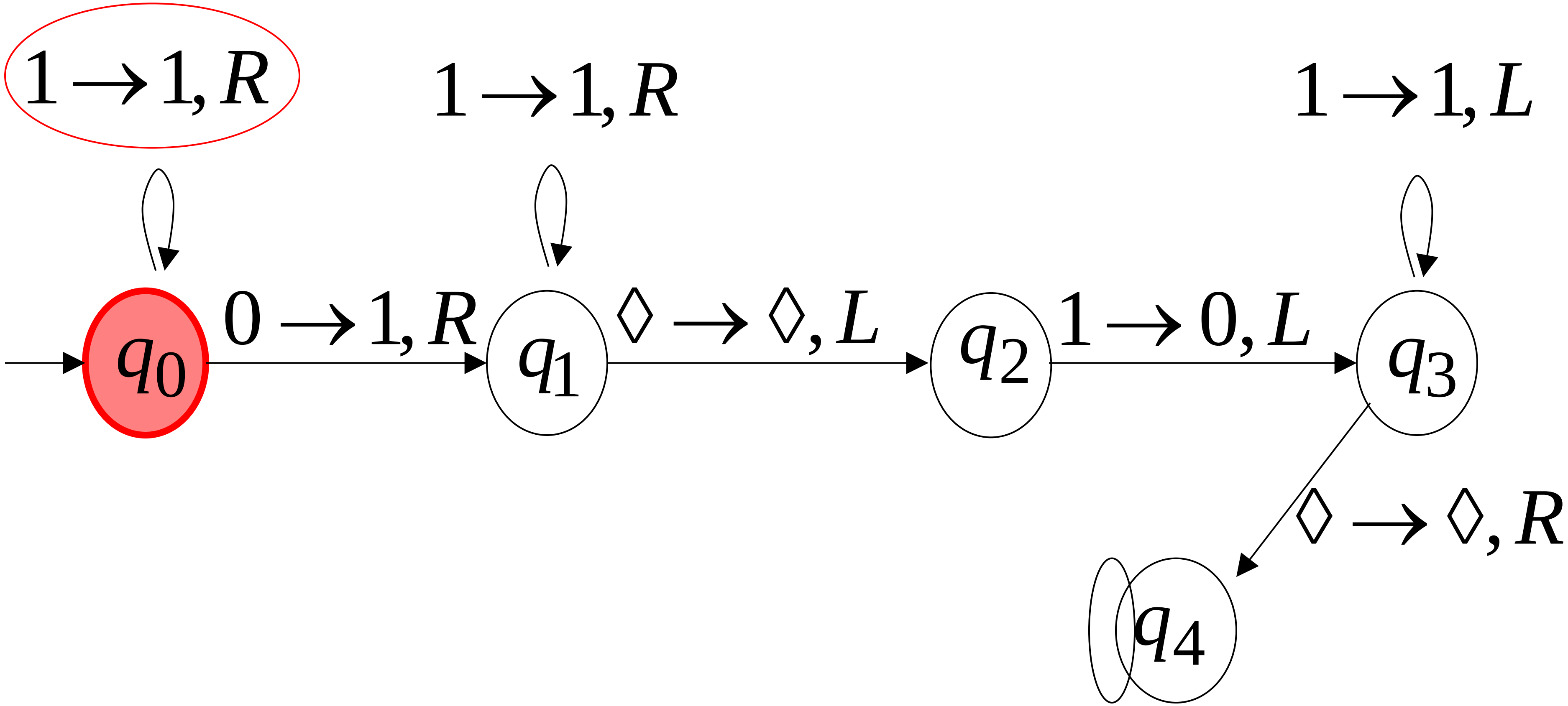
Final Result



Time 0

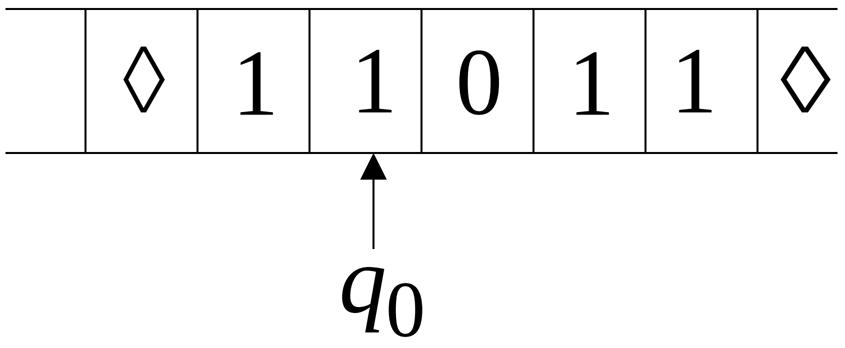


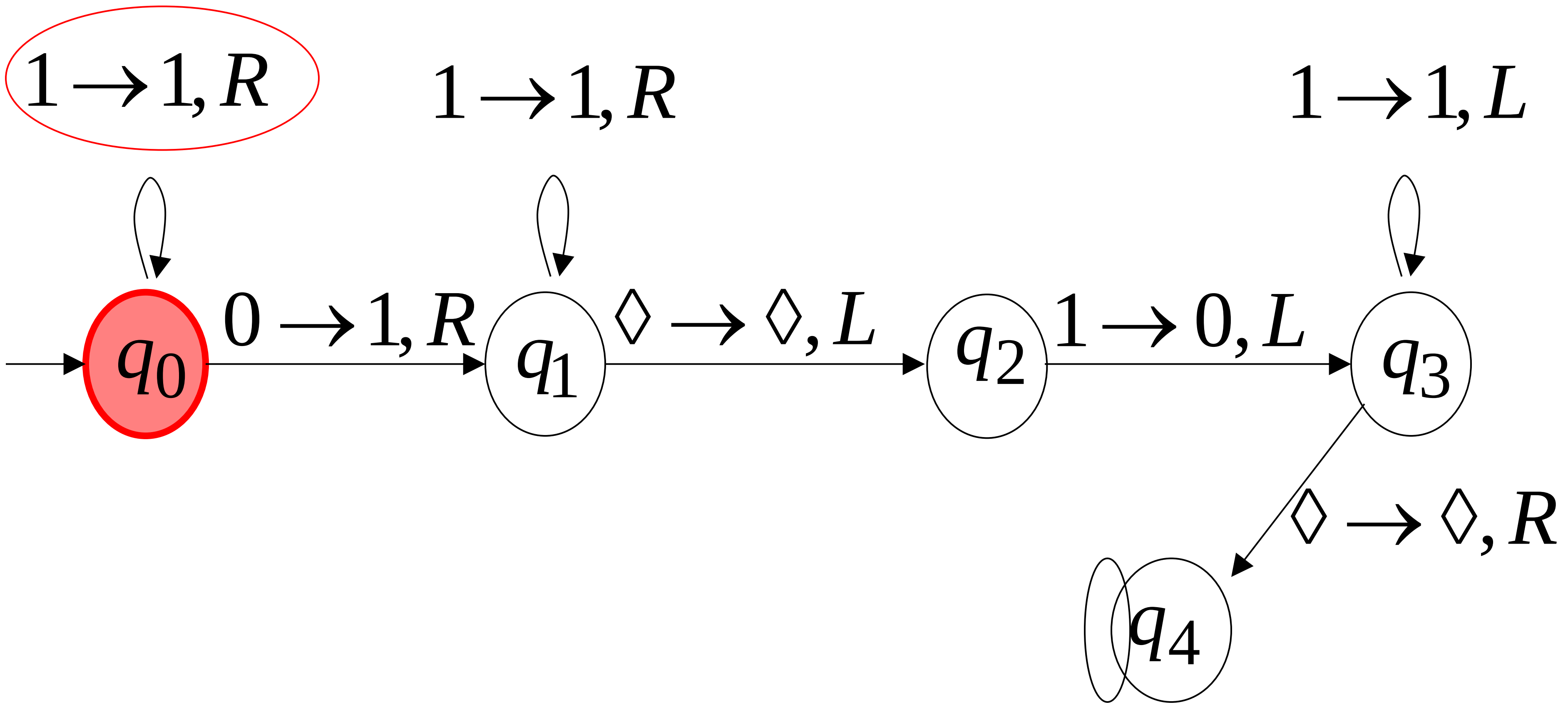
◊ 1 1 0 1 1 ◊
q_0



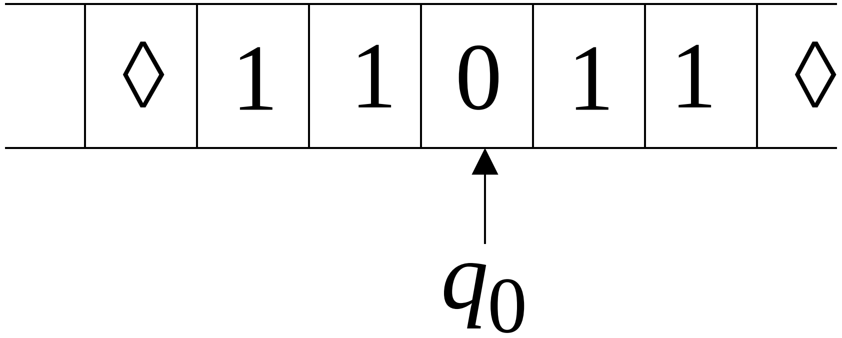
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
q_0
$0 \to 1, R$
q_1
$\Diamond \to \Diamond, L$
q_2
$1 \to 0, L$
q_3
$\Diamond \to \Diamond, R$
q_4

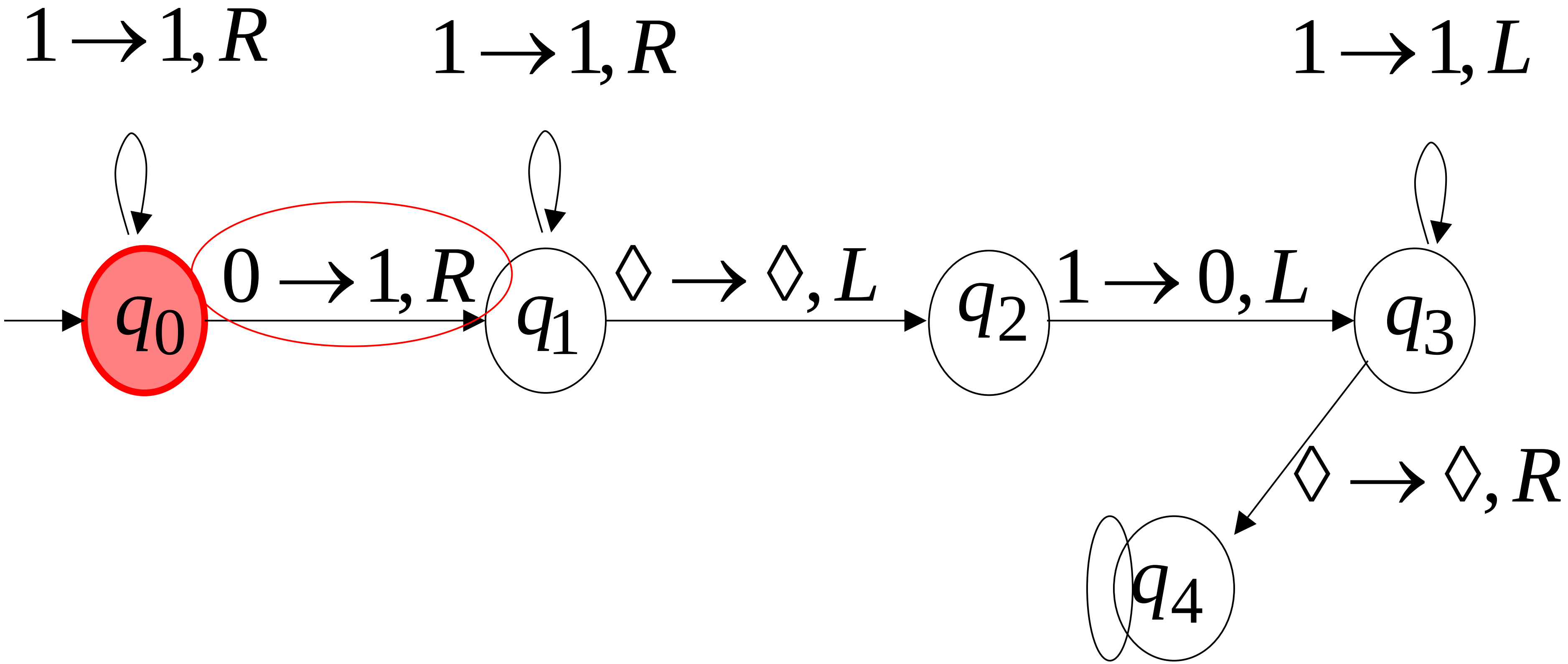
Time 1


◊
1
1
0
1
1
◊
q_0

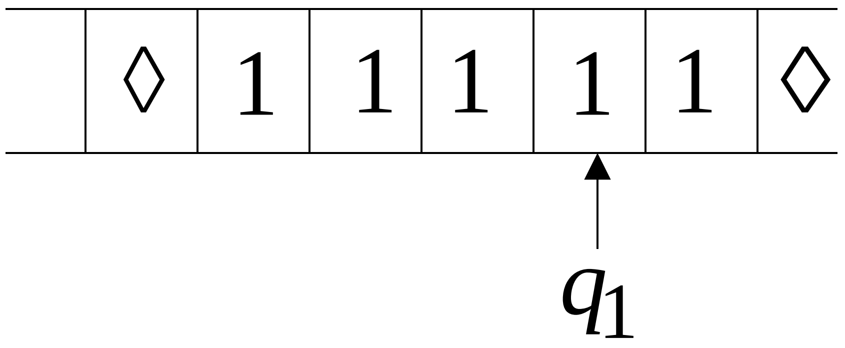

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
q_0
$0 \to 1, R$
q_1
$\Diamond \to \Diamond, L$
q_2
$1 \to 0, L$
q_3
$\Diamond \to \Diamond, R$
q_4

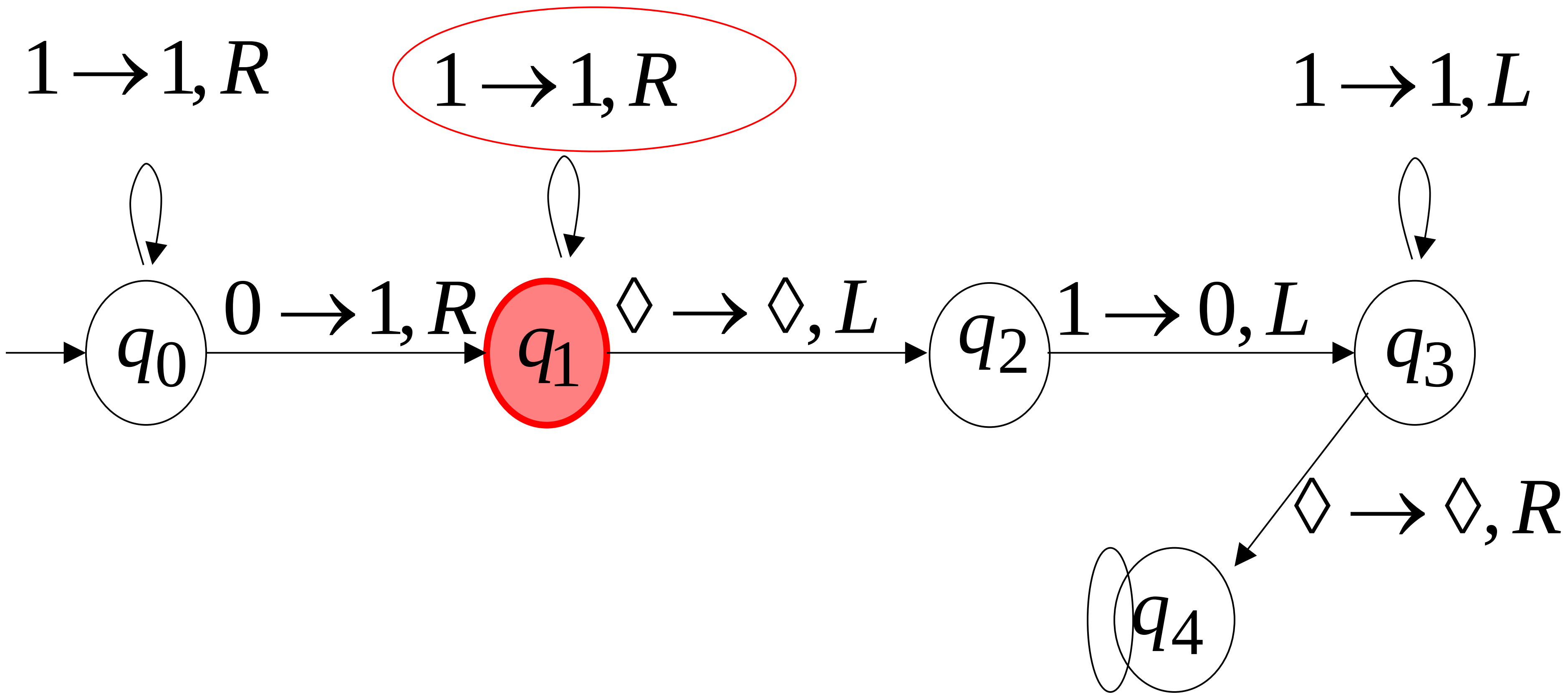
Time 2


◊ 1 1 0 1 1 ◊
q_0

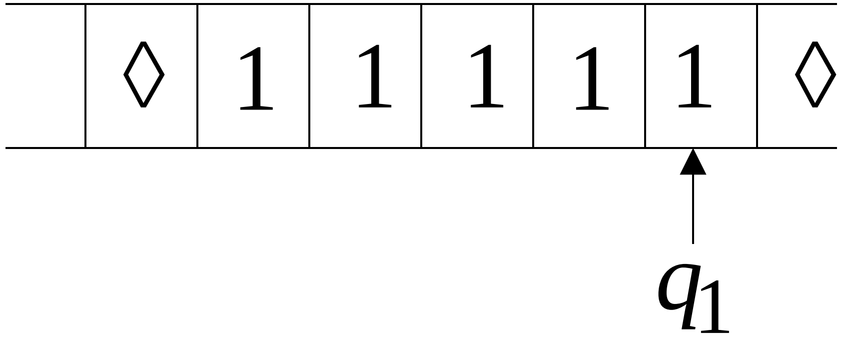

$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
q_0
$0 \to 1, R$
q_1
$\Diamond \to \Diamond, L$
q_2
$1 \to 0, L$
q_3
$\Diamond \to \Diamond, R$
q_4

Time 3

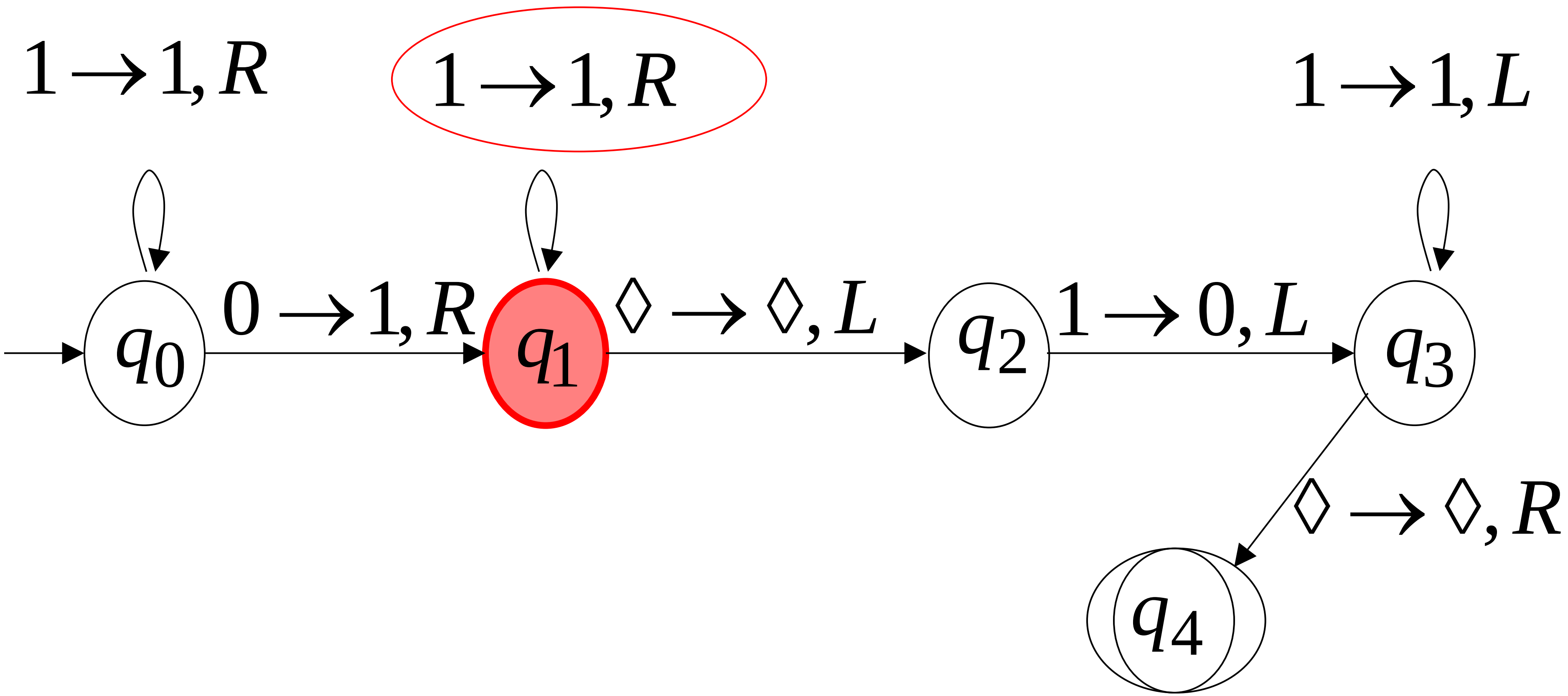

◊
1
1
1
1
1
◊
q_1


$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
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 4

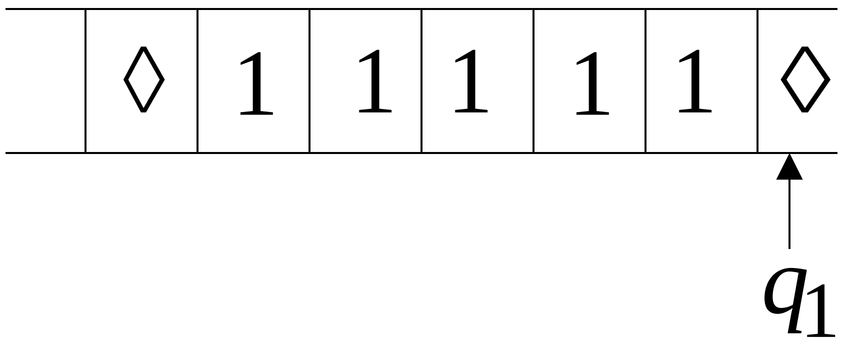


◊
1
1
1
1
1
◊
q_1

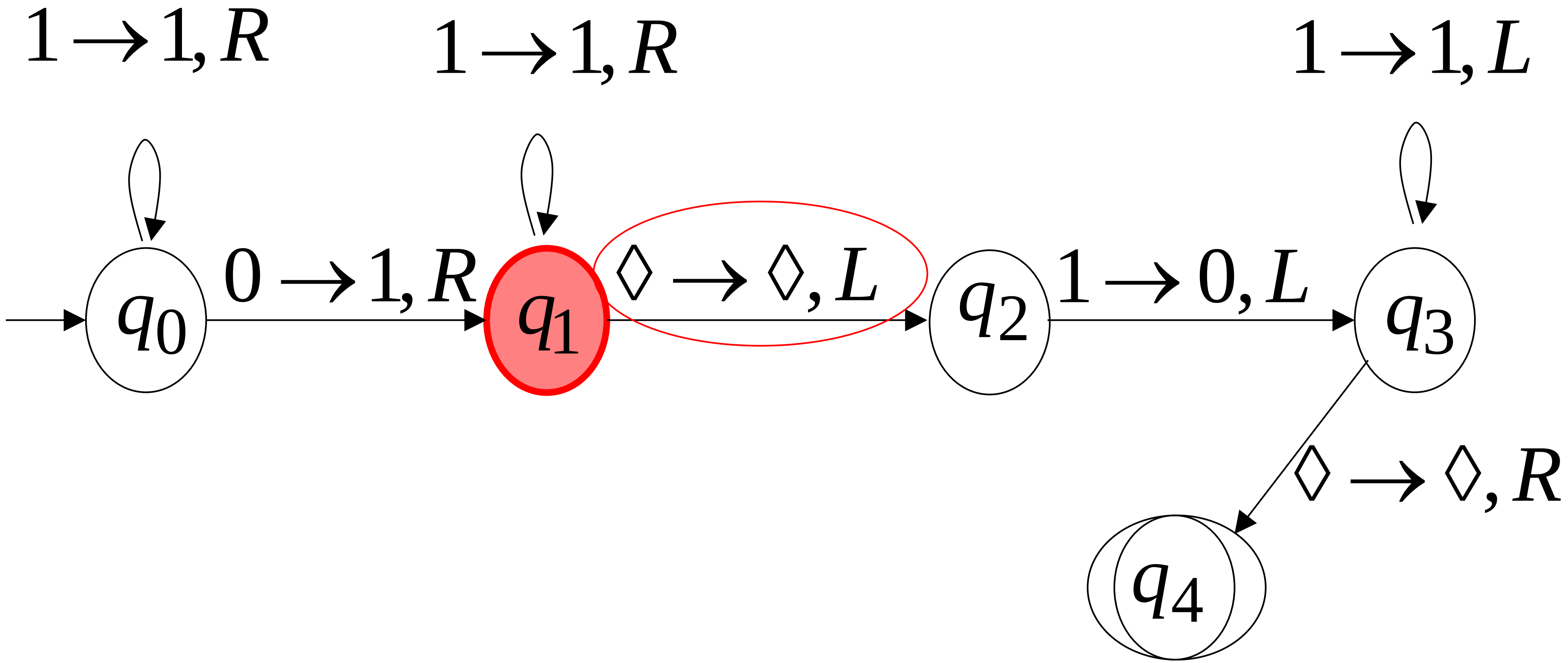


$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
q_0
$0 \rightarrow 1, R$
q_1
$\Diamond \rightarrow \Diamond, L$
q_2
$1 \rightarrow 0, L$
q_3
$\Diamond \rightarrow \Diamond, R$
q_4

Time 5

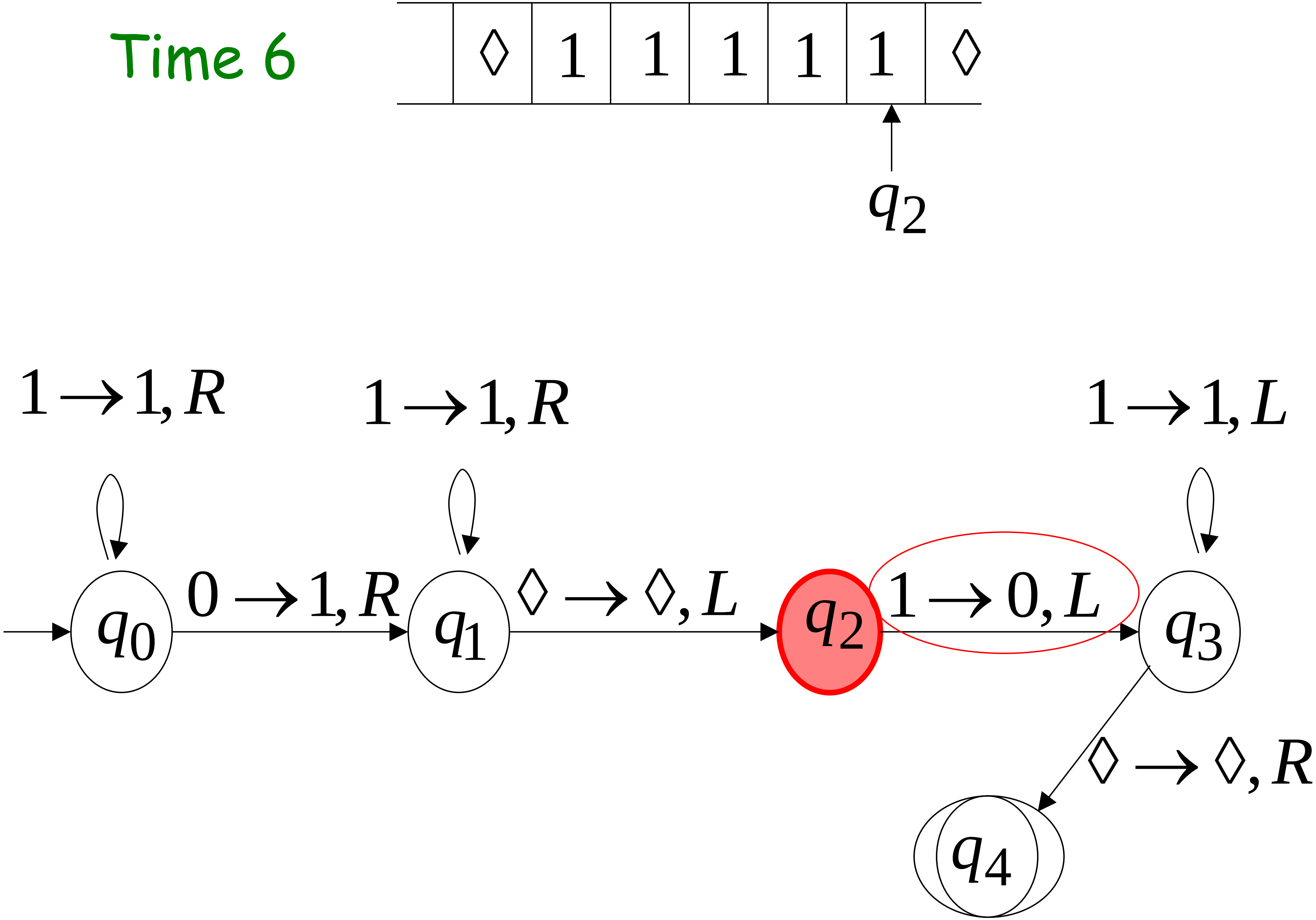


◊ 1 1 1 1 1 ◊
q_1



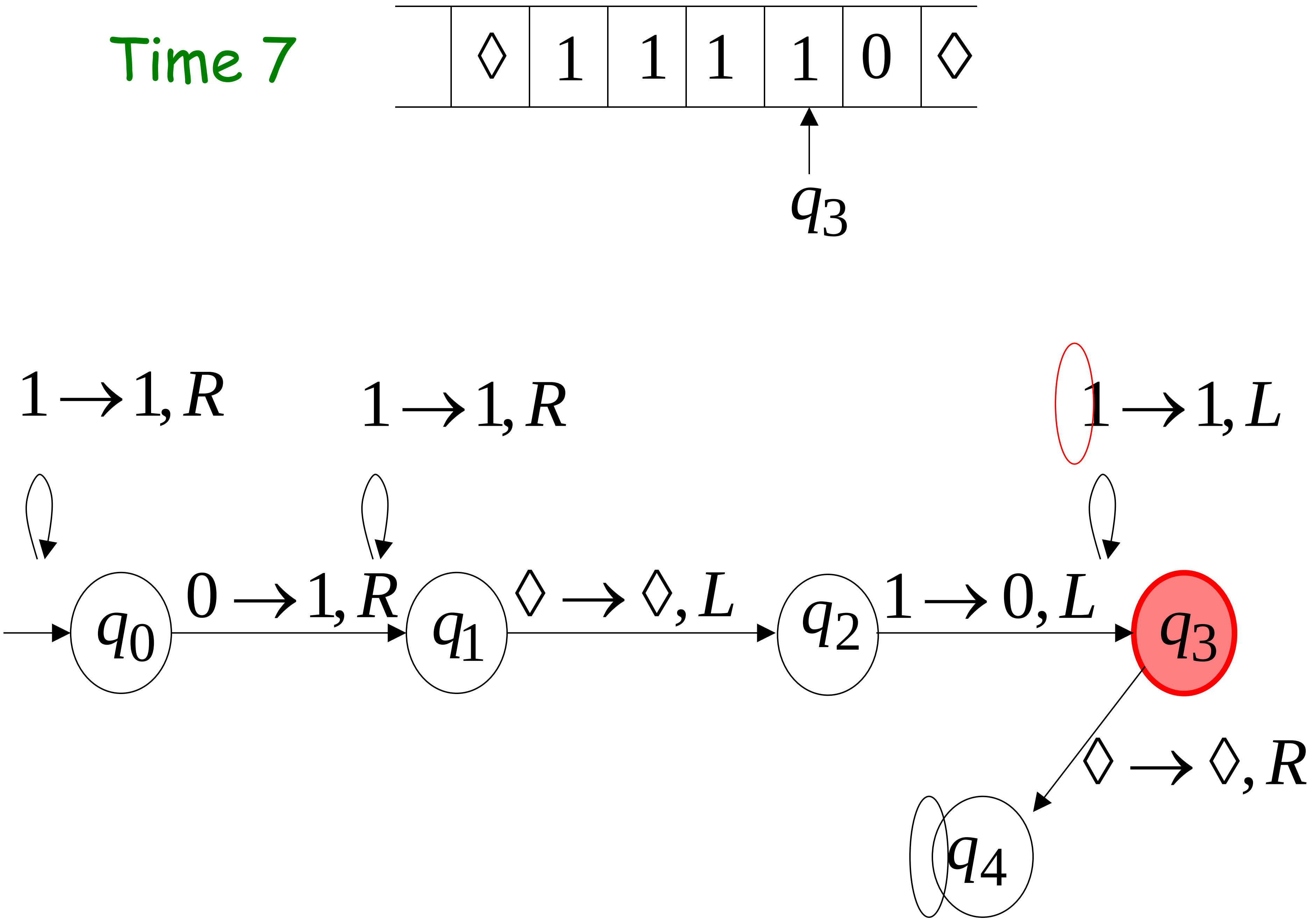
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$\lozenge \to \lozenge, L$
$1 \to 0, L$
$\lozenge \to \lozenge, R$
q_0
q_1
q_2
q_3
q_4

Time 6

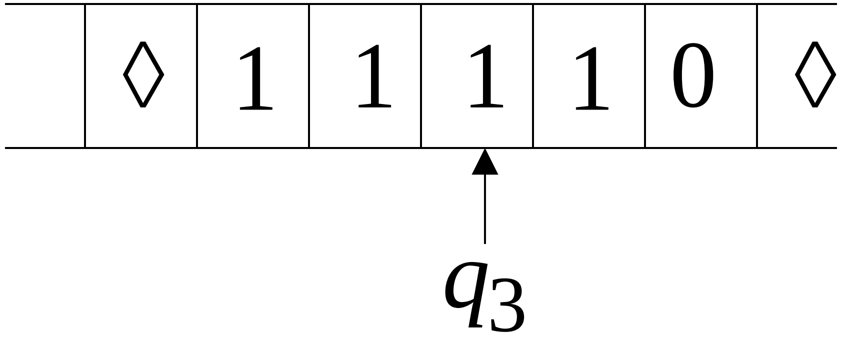


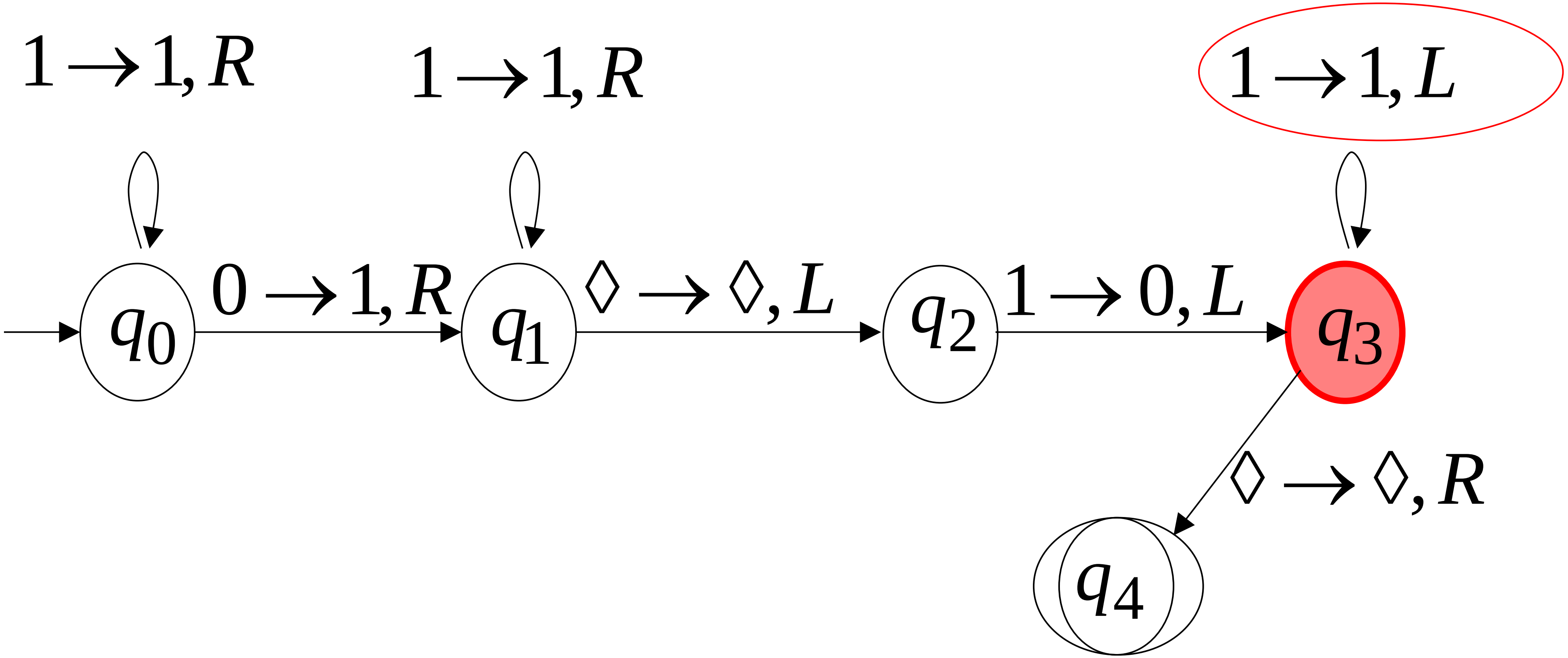
◊
1
1
1
1
1
◊
q_2
$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
q_0
$0 \rightarrow 1, R$
q_1
$\Diamond \rightarrow \Diamond, L$
q_2
$1 \rightarrow 0, L$
q_3
$\Diamond \rightarrow \Diamond, R$
q_4

Time 7

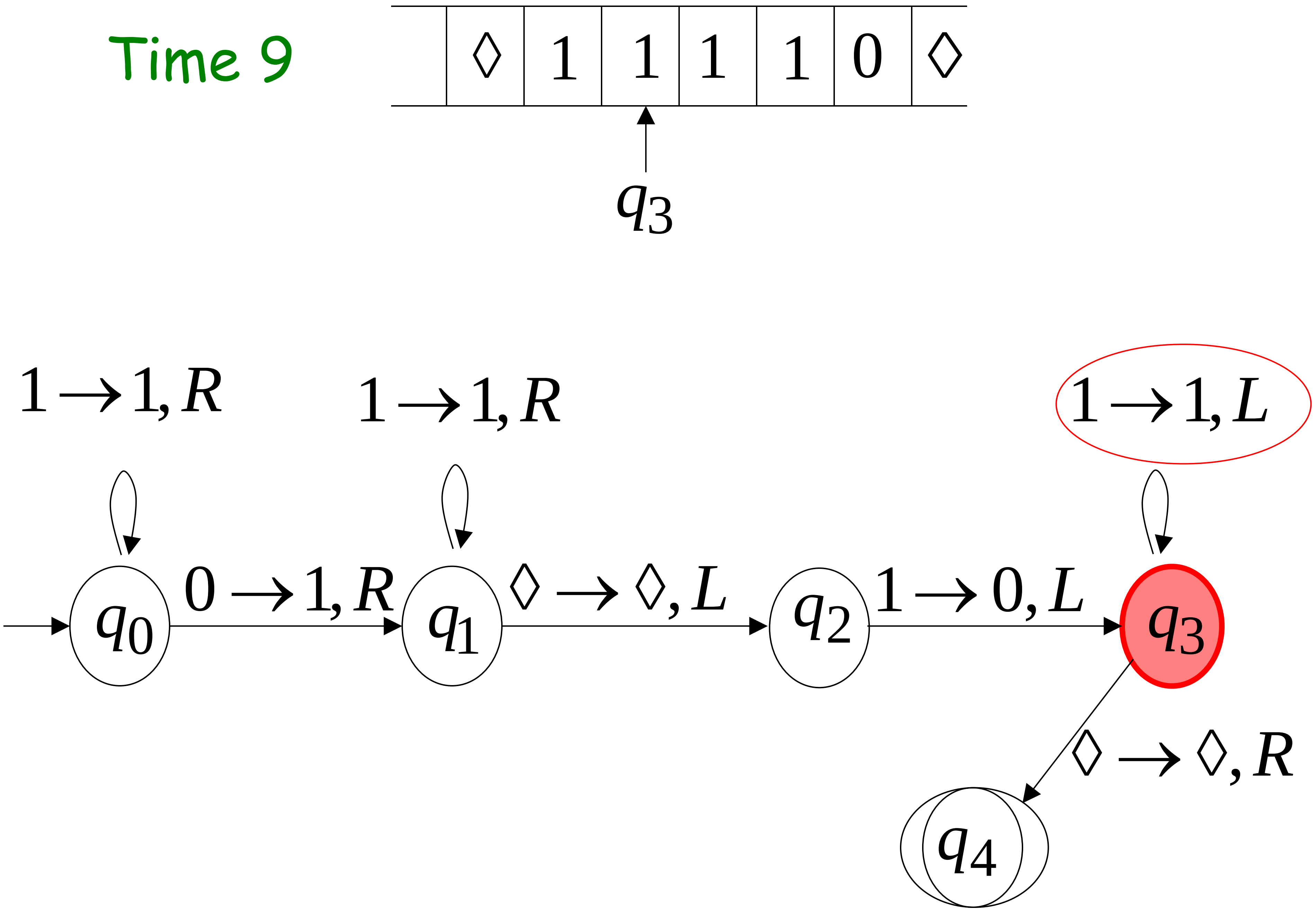


Time 8

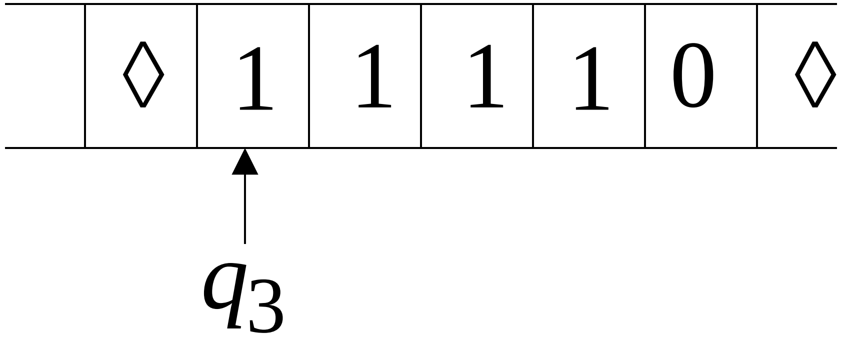

◊
1
1
1
1
0
◊
q_3


$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
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

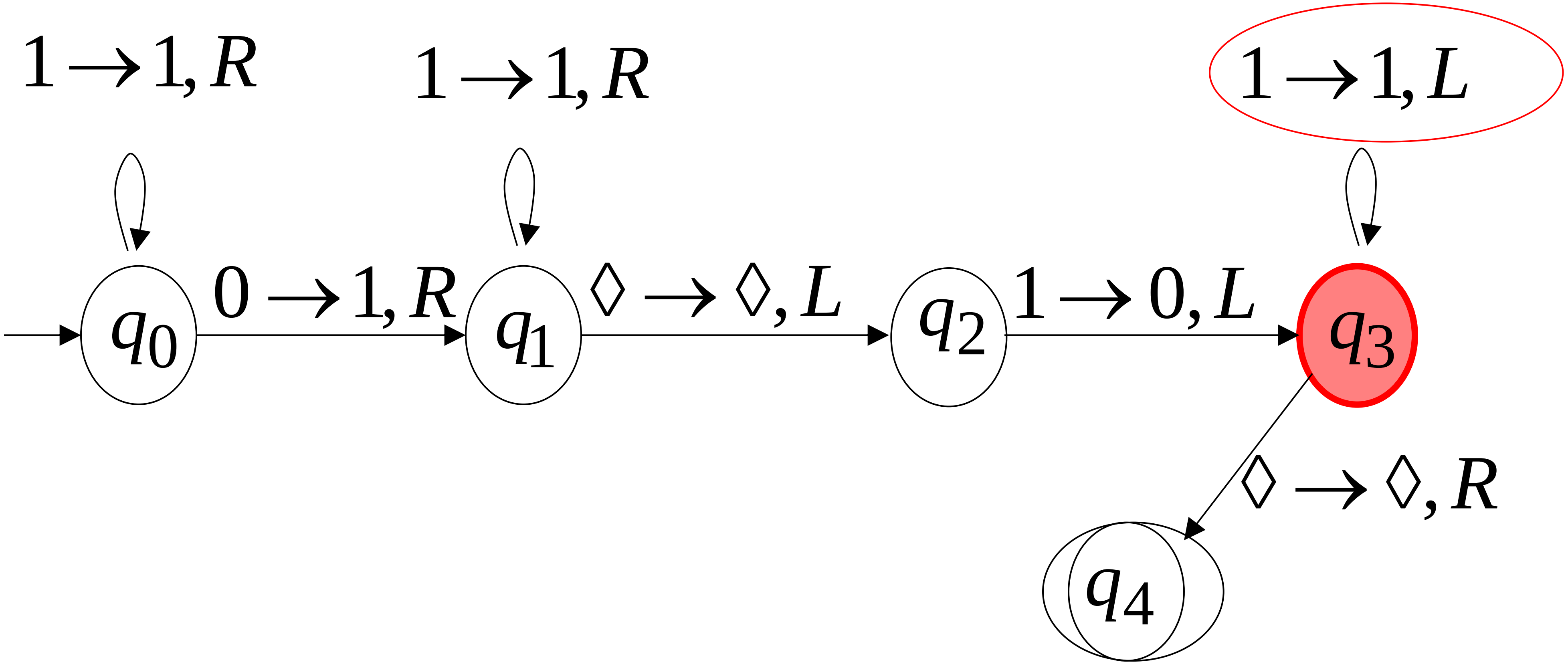
Time 9


◊ 1 1 1 1 0 ◊
q_3
$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
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 10

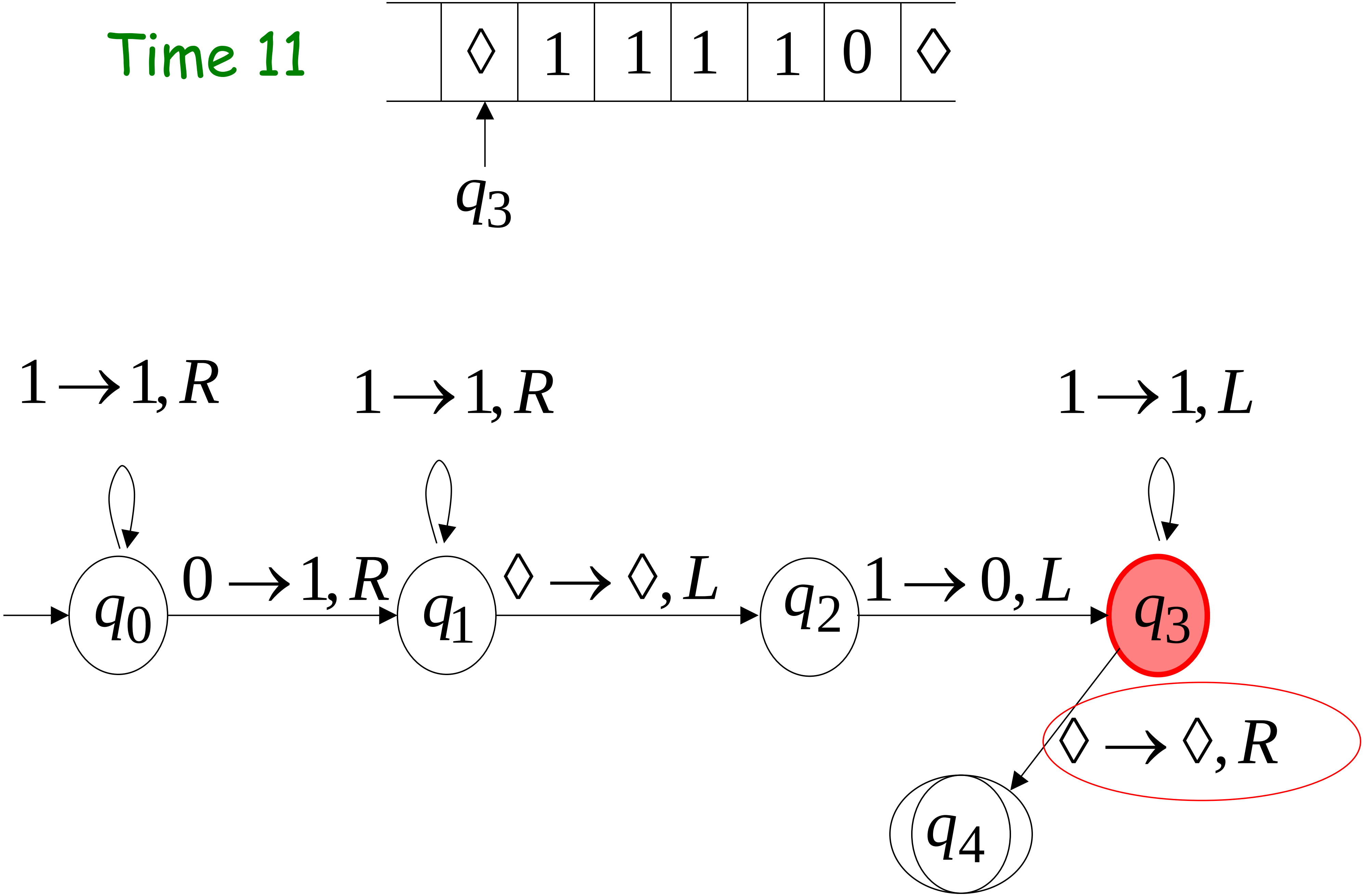


◊ 1 1 1 1 0 ◊
q_3

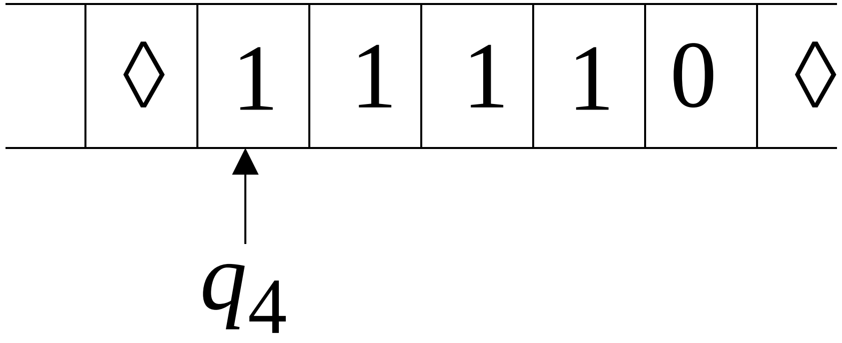


$1 \rightarrow 1, R$
$1 \rightarrow 1, R$
$1 \rightarrow 1, L$
q_0
$0 \rightarrow 1, R$
q_1
$\Diamond \rightarrow \Diamond, L$
q_2
$1 \rightarrow 0, L$
q_3
$\Diamond \rightarrow \Diamond, R$
q_4

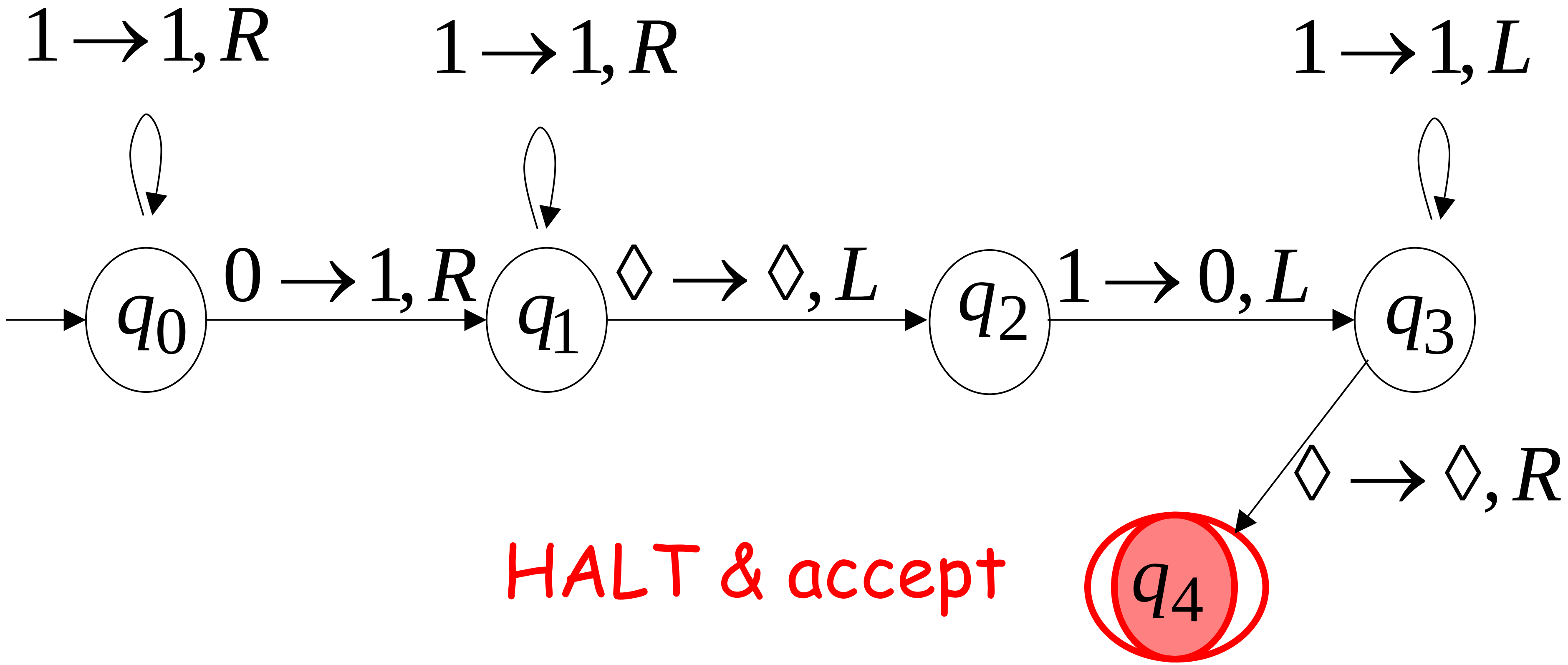
Time 11


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

Time 12



◊
1
1
1
1
0
◊
q_4



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

Another Example

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

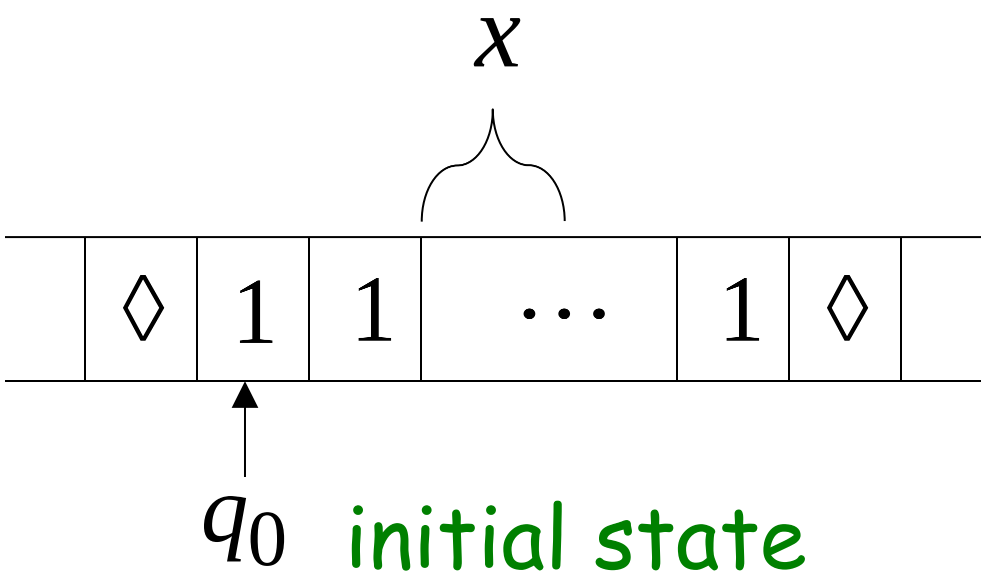
x is integer

Turing Machine:

Input string: x unary

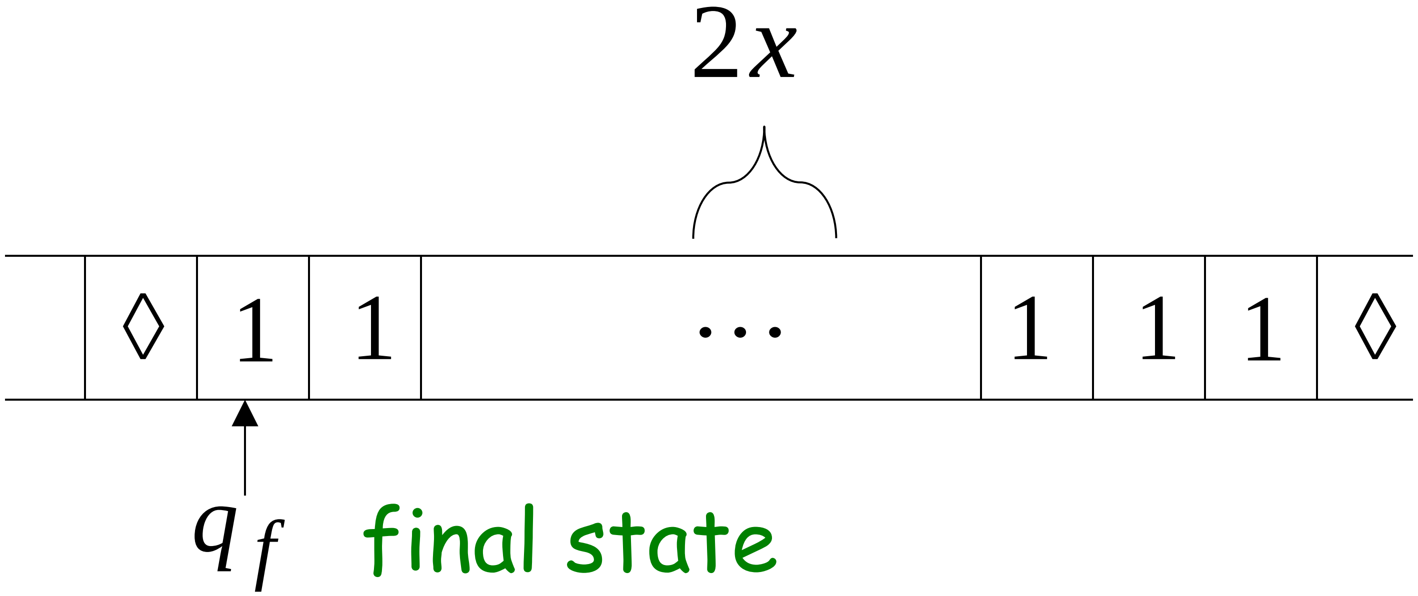
Output string: xx unary

Start



initial state

Finish



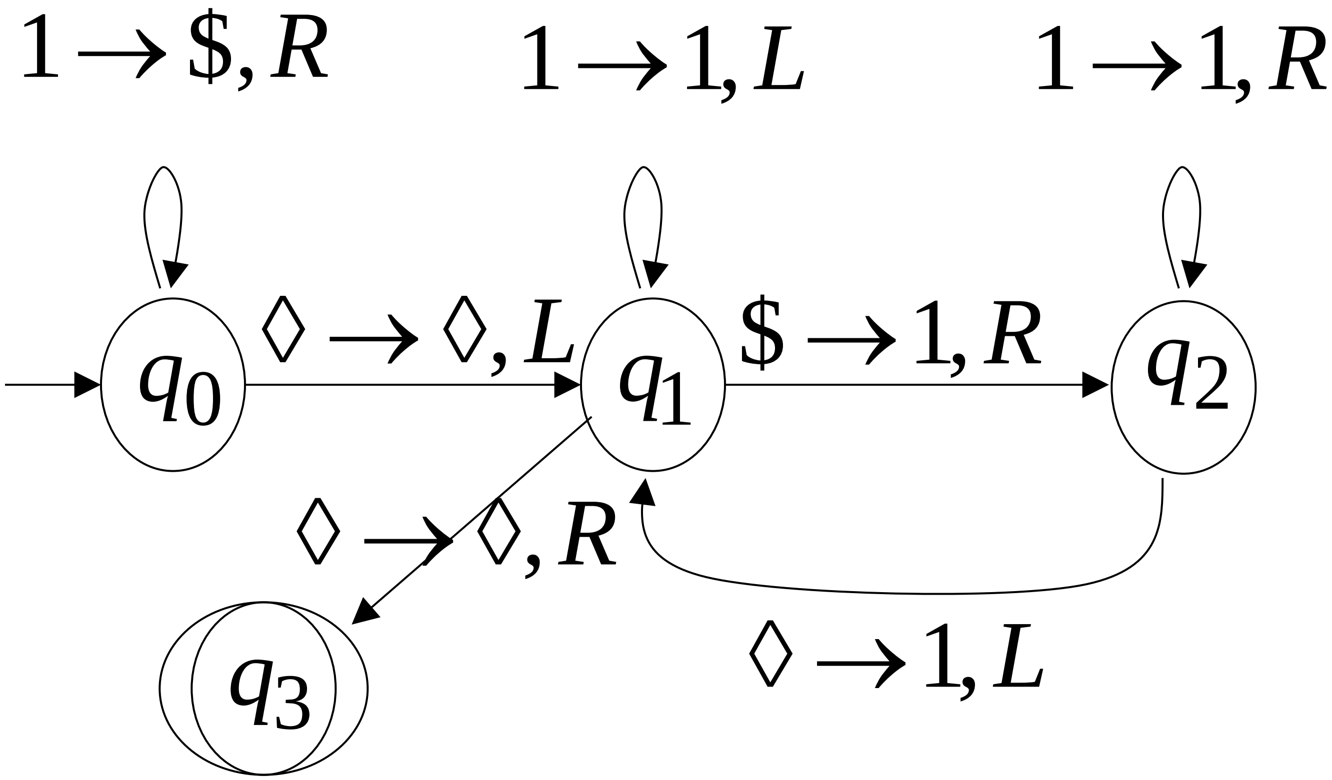
final state

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

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

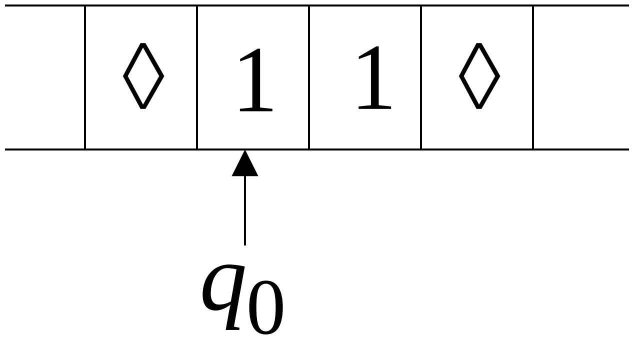
  Until no more $ remain

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

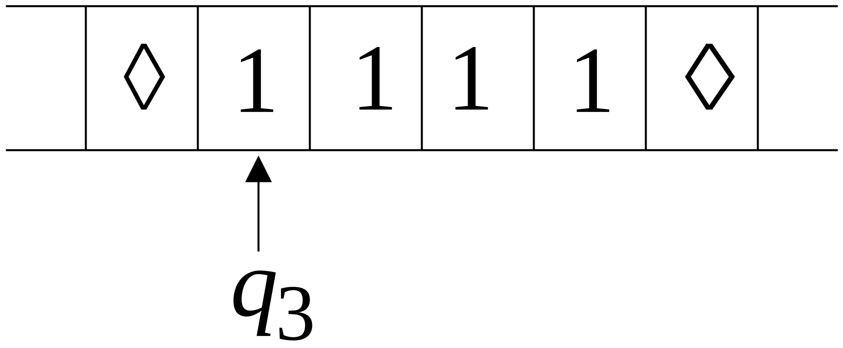


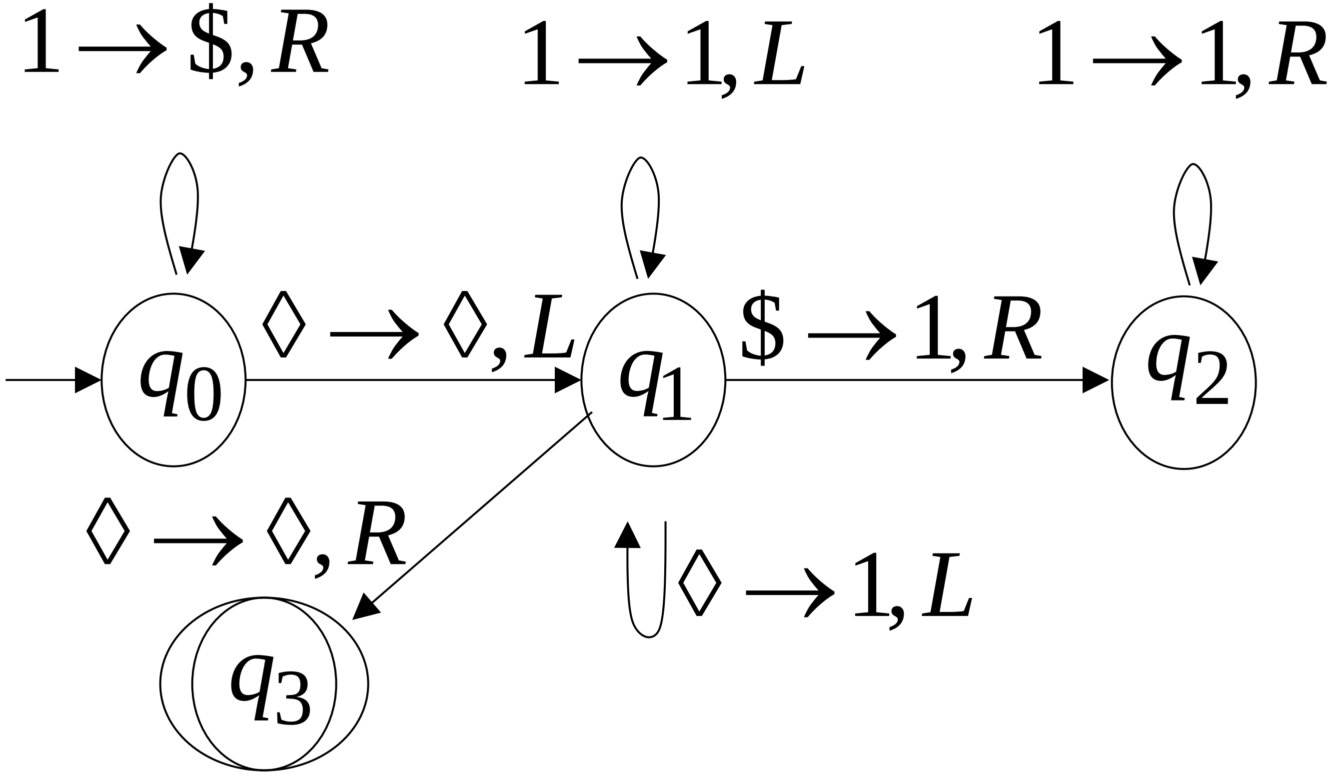
Example

Start



Finish





Another Example

The function

is computable

$$f(x,y)=\begin{cases} 1 & \text{if } x>y \\ 0 & \text{if } x\le y \end{cases}$$

Turing Machine for

$$f(x,y) = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{if } x \le y \end{cases}$$

Input: $x0y$

Output: 1 or 0

Turing Machine Pseudocode:

- Repeat

  Match a 1 from x with a 1 from y

  Until all of x or y is matched

- If a 1 from x is not matched

  erase tape, write 1 $(x > y)$

  else

  erase tape, write 0 $(x \leq y)$


Erase
1
State
2
Position
7
Step

Outline

- Last week
- Formal Definition for Turing Machines
- Computing Functions with Turing Machines
- Turing's Thesis
- Variations of the Turing Machine
- Universal Turing Machine
- Countable/uncountable Sets

Turing's thesis:

Any computation carried out
by mechanical means
can be performed by a Turing Machine

(1930)

Computer Science Law:

A computation is mechanical
if and only if
it can be performed by a Turing Machine

There is no known model of computation
more powerful than Turing Machines

Definition of Algorithm:

An algorithm for function $f(w)$

is a

Turing Machine which computes $f(w)$

Algorithms are Turing Machines

When we say:

There exists an algorithm

We mean:

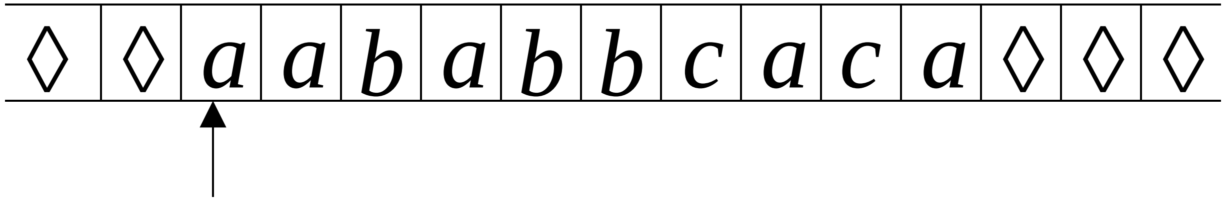
There exists a Turing Machine
that executes the algorithm

Outline

- Last week
- Formal Definition for Turing Machines
- Computing Functions with Turing Machines
- Turing’s Thesis
- Variations of the Turing Machine
- Universal Turing Machine
- Countable/uncountable Sets

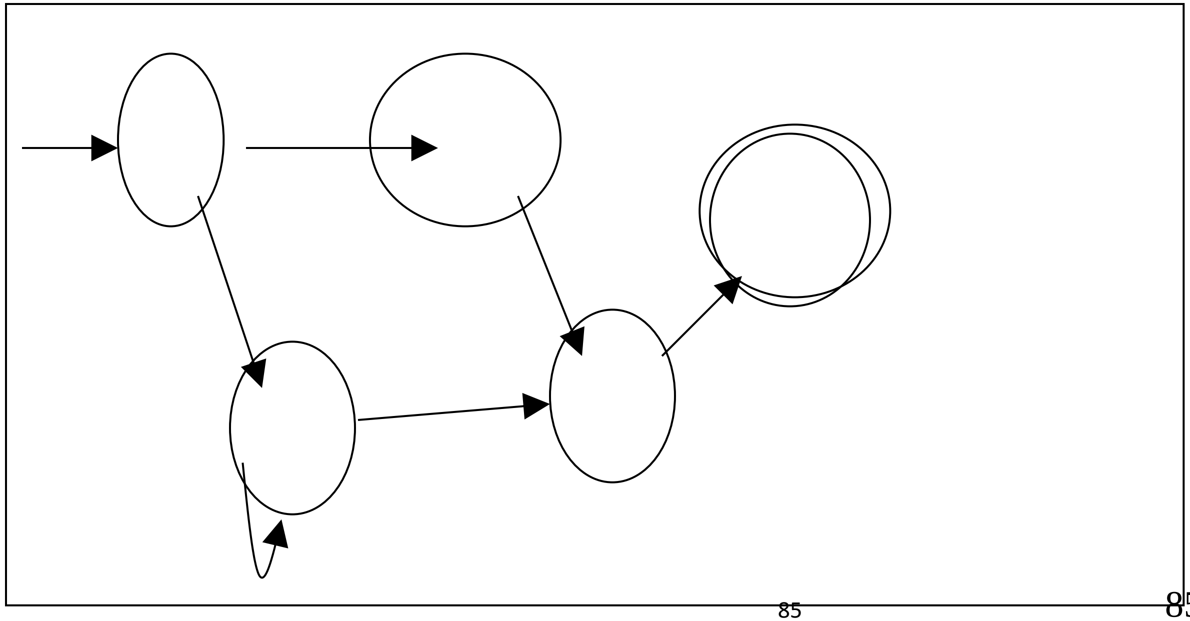
The Standard Model

Infinite Tape



Read-Write Head (Left or Right)

Control Unit



Deterministic

85

Variations of the Standard Model

Turing machines with:

- Stay-Option
- Semi-Infinite Tape
- Off-Line
- Multitape
- Multidimensional

The variations form different Turing Machine **Classes**

Each **Class** has the same power with the Standard Model

Same Power of two classes means:

For any machine M_1 of first class

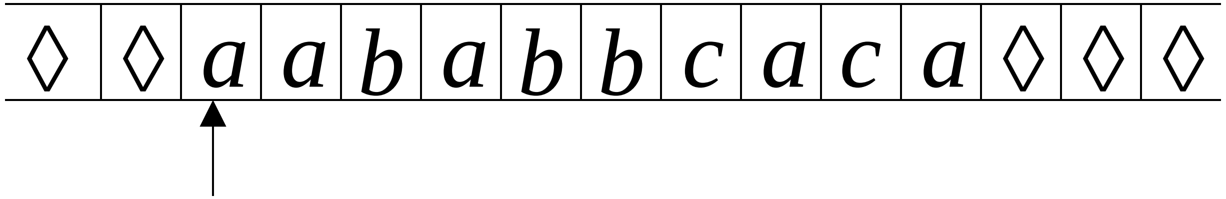
there is a machine M_2 of second class

such that: $L(M_1) = L(M_2)$

And vice-versa

Turing Machines with Stay-Option

The head can stay in the same position



Left, Right, Stay

L,R,S: moves

Example:

Time 1

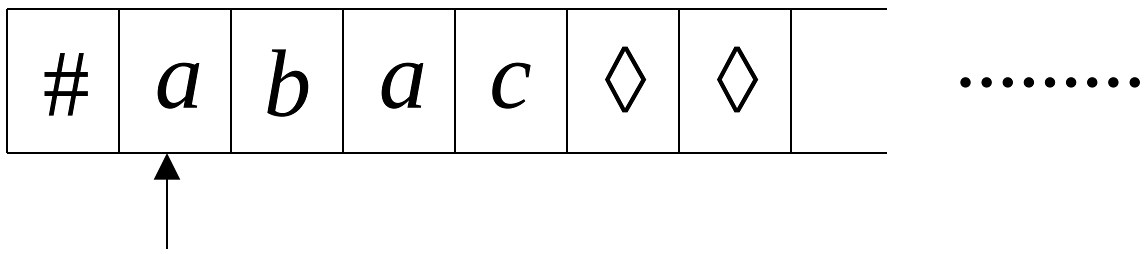
◊	◊	a	a	b	a	b	b	c	a	c	a	◊	◊	◊
		$\uparrow$												
		q_1												

Time 2

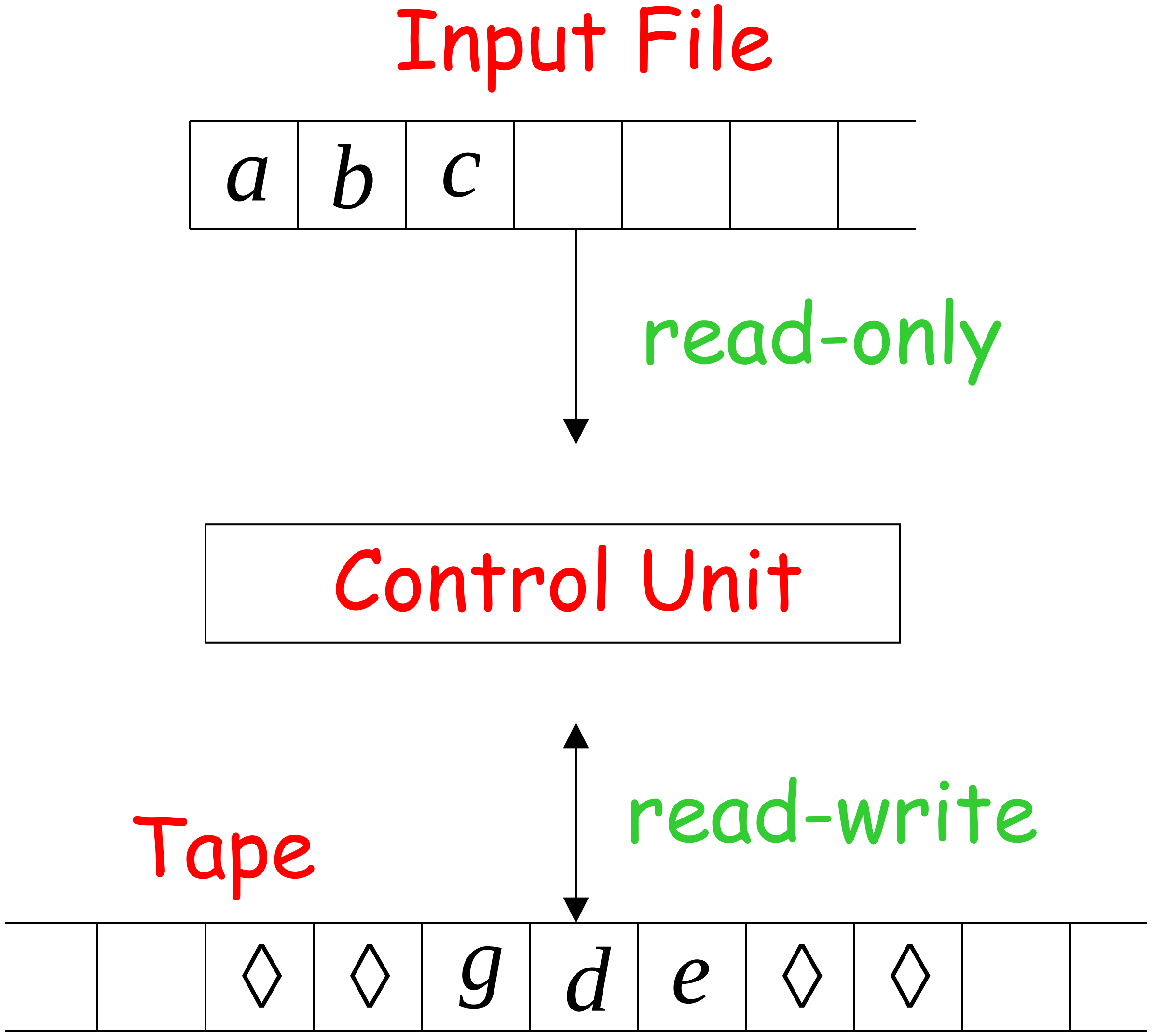
◊	◊	b	a	b	a	b	b	c	a	c	a	◊	◊	◊
		$\uparrow$												
		q_2												

$$q_1 \xrightarrow{a \to b, S} q_2$$

Semi-Infinite Tape



The Off-Line Machine



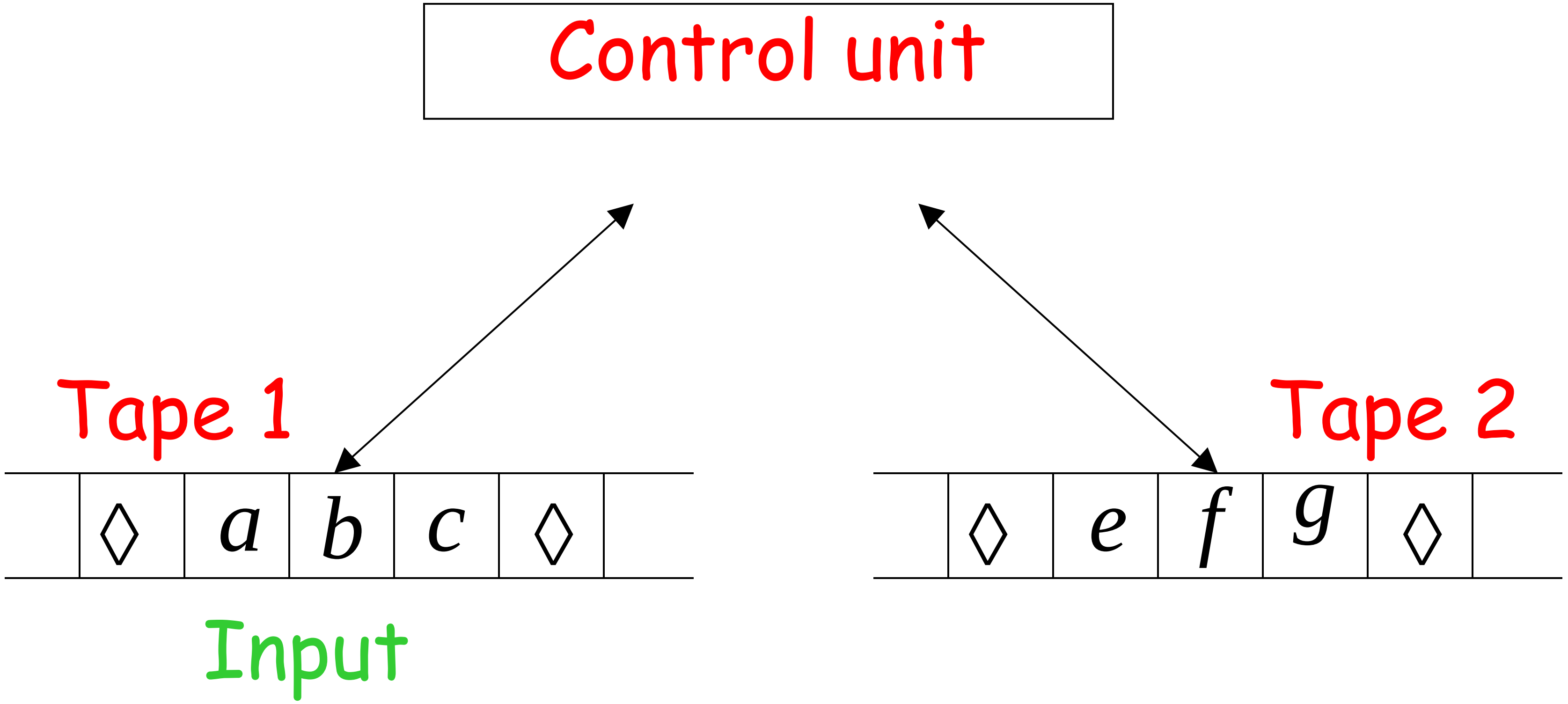
Off-line machines simulate Standard Turing Machines:

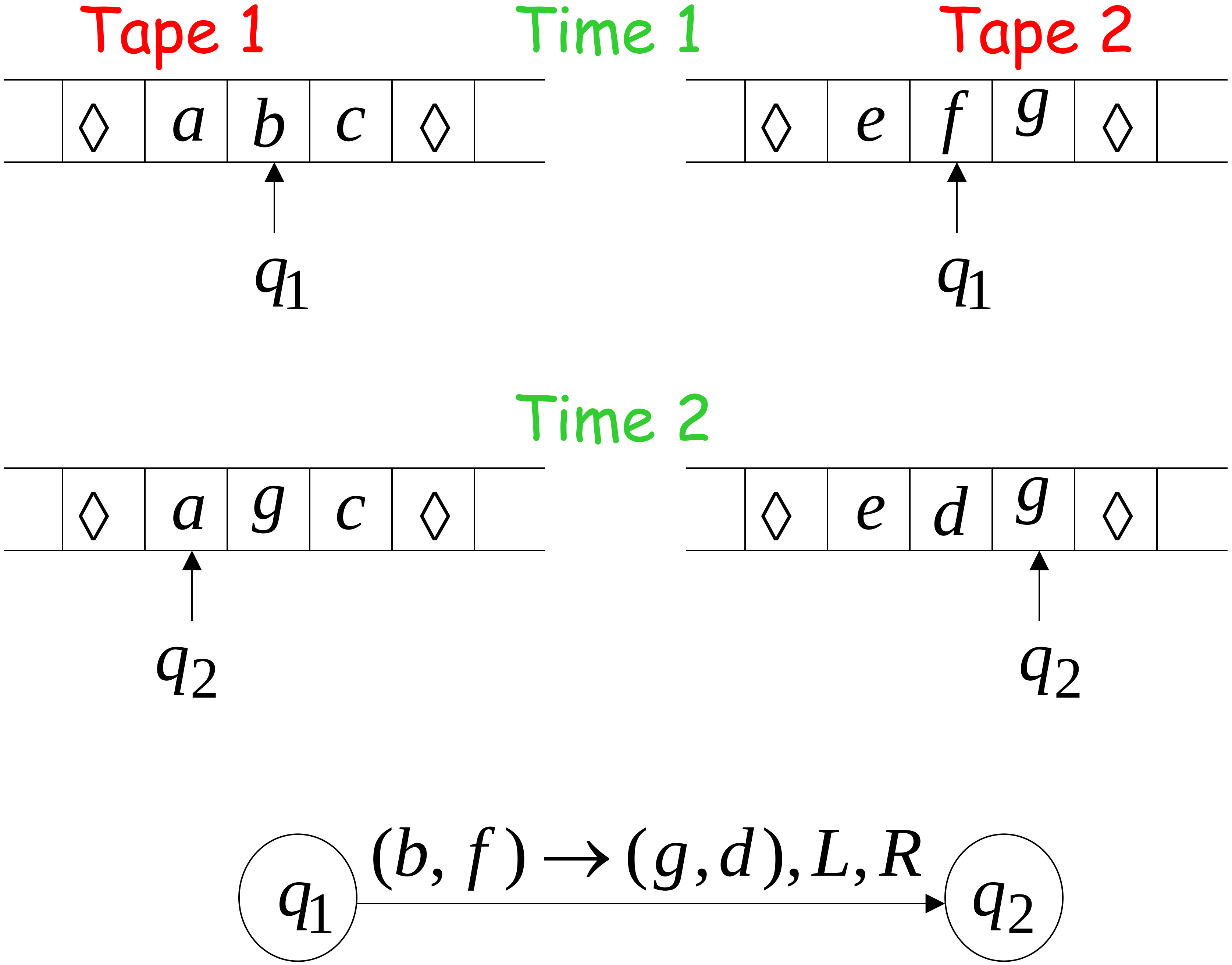
Off-line machine:

1. Copy input file to tape

2. Continue computation as in Standard Turing machine

Multitape Turing Machines




Tape 1
Time 1
Tape 2
◊ a b c ◊
◊ e f g ◊
q_1
q_1
Time 2
◊ a g c ◊
◊ e d g ◊
q_2
q_2
q_1
$(b, f) \to (g, d), L, R$
q_2

Outline

- Last week
- Formal Definition for Turing Machines
- Computing Functions with Turing Machines
- Turing’s Thesis
- Variations of the Turing Machine
- Universal Turing Machine
- Countable/uncountable Sets

A limitation of Turing Machines:

Turing Machines are "hardwired"

they execute
only one program

Real Computers are re-programmable

Solution: Universal Turing Machine

Attributes:

- Reprogrammable machine
- Simulates any other Turing Machine

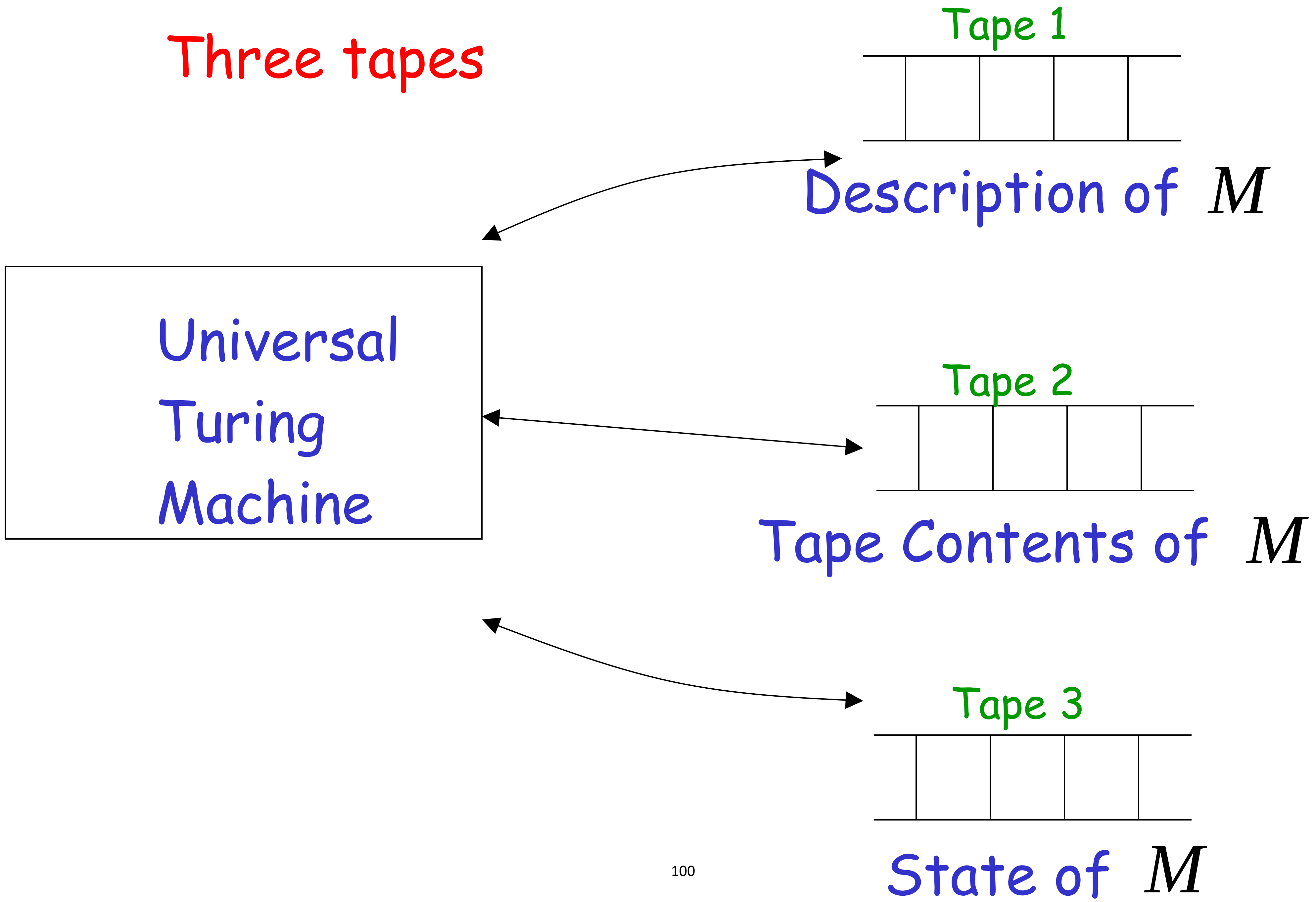
Universal Turing Machine

simulates any other Turing Machine M

Input of Universal Turing Machine:

Description of transitions of M

Initial tape contents of M



Three tapes
Tape 1
Description of M
Universal Turing Machine
Tape 2
Tape Contents of M
Tape 3
State of M

Tape 1

Description of M

We describe Turing machine M
as a string of symbols:

We encode M as a string of symbols

Alphabet Encoding

Symbols:	a	b	c	d	$\cdots$
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
Encoding:	1	11	111	1111	

State Encoding

States: q_1 q_2 q_3 q_4 $\cdots$

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$

Encoding: 1 11 111 1111

Head Move Encoding

Move: L R

$\downarrow$ $\downarrow$

Encoding: 1 11

Transition Encoding

Transition: $\delta(q_1, a) = (q_2, b, L)$

Encoding: $1\,0\,1\,0\,11\,0\,11\,0\,1$

separator

Machine Encoding

Transitions:

$\delta(q_1, a) = (q_2, b, L)$ $\qquad$ $\delta(q_2, b) = (q_3, c, R)$

Encoding:

$10101101101\ 00\ 1101101110111011$

separator

Tape 1 contents of Universal Turing Machine:

encoding of the simulated machine M
as a binary string of 0's and 1's

A Turing Machine is described
with a binary string of 0's and 1's

Therefore:

The set of Turing machines forms a language:

each string of the language is
the binary encoding of a Turing Machine

Language of Turing Machines

L = { 010100101, (Turing Machine 1)

00100100101111, (Turing Machine 2)

111010011110010101, ……

…… }

Outline

- Last week
- Formal Definition for Turing Machines
- Computing Functions with Turing Machines
- Turing’s Thesis
- Variations of the Turing Machine
- Universal Turing Machine
- Countable/uncountable Sets

Infinite sets are either:

Countable

or

Uncountable

Countable set:

Any finite set

or

Any Countably infinite set:

There is a one to one correspondence

between

elements of the set

and

Natural numbers

Example: The set of even integers is countable

Even integers: $0,\ 2,\ 4,\ 6,\ \ldots$

Correspondence: $\downarrow\ \downarrow\ \downarrow\ \downarrow$

Positive integers: $1,\ 2,\ 3,\ 4,\ \ldots$

$2n$ corresponds to $n+1$

Example: The set of rational numbers is countable

Rational numbers: $\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$

Naïve Proof

Rational numbers: $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \ldots$

Correspondence: $\downarrow \quad \downarrow \quad \downarrow$

Positive integers: $1, \; 2, \; 3, \; \ldots$

Doesn't work:

we will never count
numbers with nominator 2: $\frac{2}{1}, \frac{2}{2}, \frac{2}{3}, \ldots$

Better Approach

$$\frac{1}{1} \quad \frac{1}{2} \quad \frac{1}{3} \quad \frac{1}{4} \quad \cdots$$

$$\frac{2}{1} \quad \frac{2}{2} \quad \frac{2}{3} \quad \cdots$$

$$\frac{3}{1} \quad \frac{3}{2} \quad \cdots$$

$$\frac{4}{1} \quad \cdots$$

$$\frac{1}{1} \longrightarrow \frac{1}{2} \qquad \frac{1}{3} \qquad \frac{1}{4} \quad \cdots$$

$$\frac{2}{1} \qquad \frac{2}{2} \qquad \frac{2}{3} \quad \cdots$$

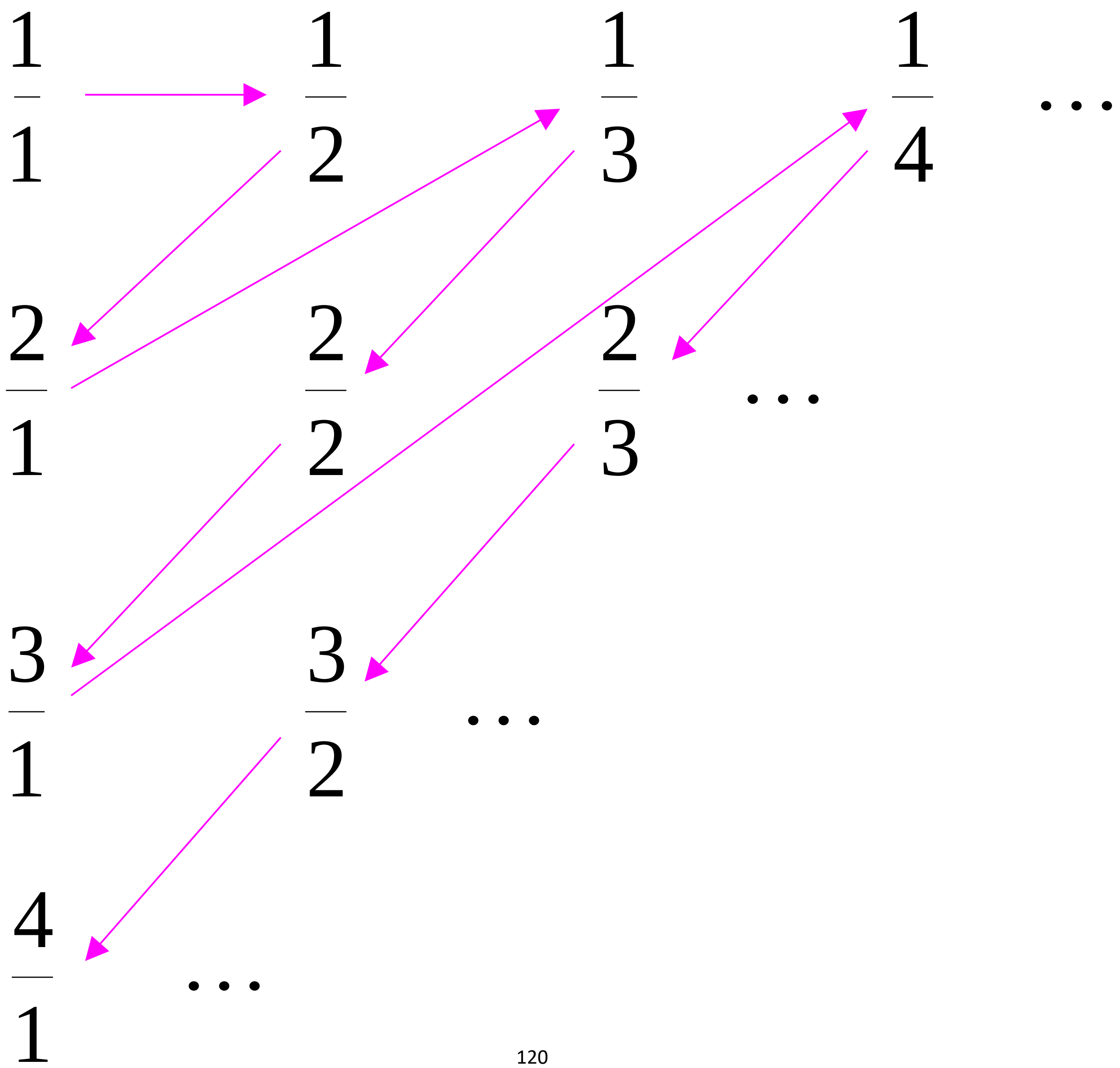
$$\frac{3}{1} \qquad \frac{3}{2} \quad \cdots$$

$$\frac{4}{1} \quad \cdots$$

$$
\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}
$$

$$
\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}
$$

$$\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}$$



$\frac{1}{1}$ $\frac{1}{2}$ $\frac{1}{3}$ $\frac{1}{4}$ $\cdots$
$\frac{2}{1}$ $\frac{2}{2}$ $\frac{2}{3}$ $\cdots$
$\frac{3}{1}$ $\frac{3}{2}$ $\cdots$
$\frac{4}{1}$ $\cdots$

Rational Numbers: $\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{2}{2}, \cdots$

Correspondence: $\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

Positive Integers: $1, \; 2, \; 3, \; 4, \; 5, \; \ldots$

We proved:

the set of rational numbers is countable
by describing an enumeration procedure

Definition

Let S be a set of strings

An **enumeration procedure** for S is a Turing Machine that generates all strings of S one by one

and

each string is generated in finite time

strings $s_1, s_2, s_3, \ldots \in S$

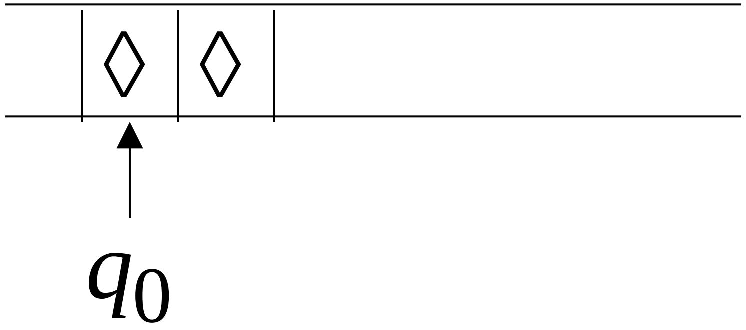
$$\boxed{\text{Enumeration Machine for } S} \xrightarrow[\text{(on tape)}]{\text{output}} s_1, s_2, s_3, \ldots$$

Finite time: $t_1, t_2, t_3, \ldots$

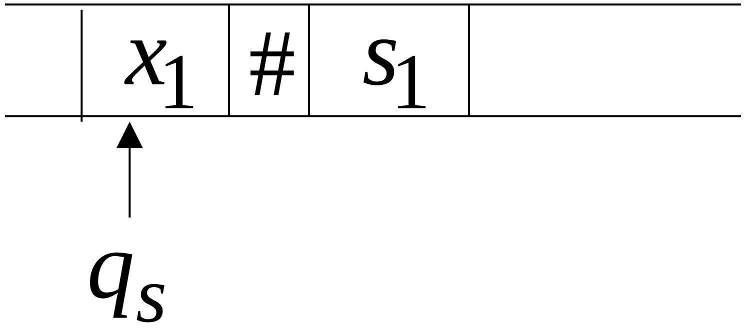
Enumeration Machine

Configuration

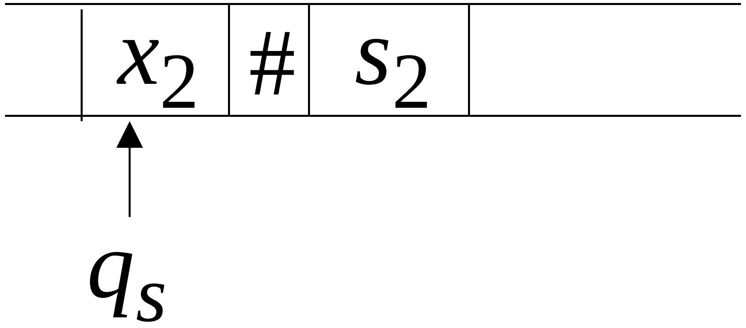
Time 0



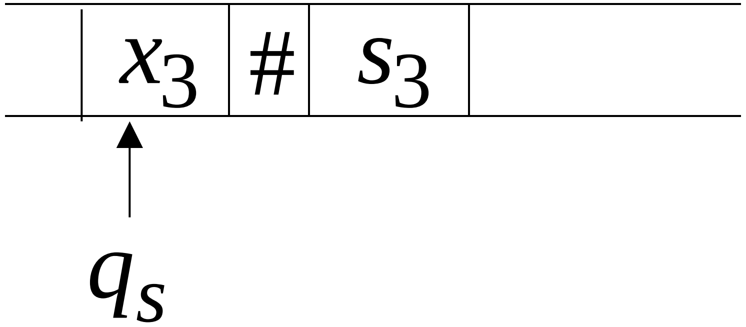
Time t_1



Time t_2



Time t_3



Observation:

If for a set there is an
enumeration procedure,
then the set is countable

Example:

The set of all strings $\{a,b,c\}^{+}$ is countable

Proof:

We will describe an enumeration procedure

Naive procedure:

Produce the strings in lexicographic order:

$$a$$

$$aa$$

$$aaa$$

$$aaaa$$

$$\ldots\ldots$$

Doesn't work:

strings starting with b will never be listed

(violates the generation in finite time rule)

Better procedure: **Proper Order**

1. Produce all strings of length 1

2. Produce all strings of length 2

3. Produce all strings of length 3

4. Produce all strings of length 4

....

Produce strings in **Proper Order:**

a
b
c

length 1

aa
ab
ac
ba
bb
bc
ca
cb
cc

length 2

aaa
aab
aac

length 3

$\cdots\cdots$

Theorem: The set of all Turing Machines is countable

Proof: Any Turing Machine can be encoded with a binary string of 0's and 1's

Find an enumeration procedure for the set of Turing Machine strings

Enumeration Procedure:

Repeat

1. Generate the next binary string of 0's and 1's in proper order

2. Check if the string describes a Turing Machine

   if **YES:** print string on output tape

   if **NO:** ignore string

Definition: A set is uncountable if it is not countable