THEORY OF COMPUTATION CSC-251 Turing Machine(TM)

DWIT

Sailesh.bajracharya@gmail.com

9841594548

Turing Machine

- Alan Turing(1936)
- Very powerful (abstract) machines that could simulate any modern day computer (although very, very slowly!)
- Why design such a machine?
- If a problem cannot be "solved" even using a TM, then it implies that the problem is ***undecidable***

Devices of Increasing Computational Power

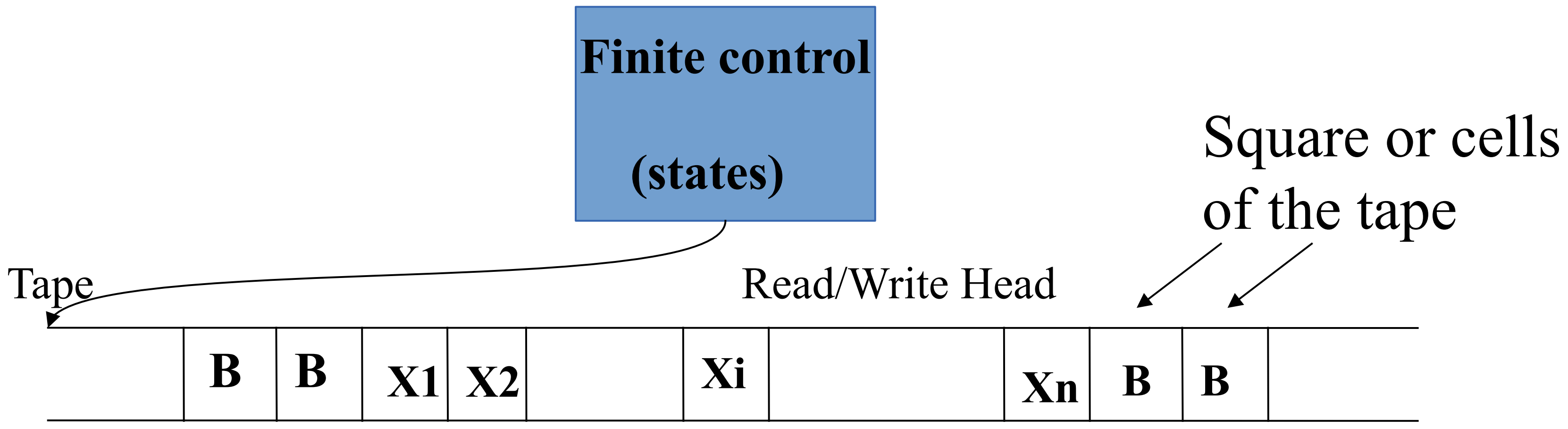
- Finite Automata – good for devices with small amounts of memory, relatively simple control
- Pushdown Automata – stack-based automata
- But both have limitations for even simple tasks, too restrictive as general purpose computers
- Enter the **Turing Machine**
    - More powerful than either of the above
    - Essentially a finite automaton but with unlimited memory
    - Although theoretical, can do everything a general purpose computer of today can do
    - If a TM can’t solve it, neither can a computer

Turing Machine

Finite automaton with an unlimited and unrestricted memory

A Turing machine is however a more accurate model of a general purpose computer

A Turing machine can do everything that a real computer can do



A move of a Turing machine (TM) is a function of the state of the finite control and the tape symbol just scanned.

Turing Machine

- All cells on the tape are originally filled with a special “blank” character “delta”
- Tape head is read/write
- Tape head can not move to the left of the start

of the tape

-If it tries, the machine “crashes”

Turing Machine

In one move, the Turing machine will:

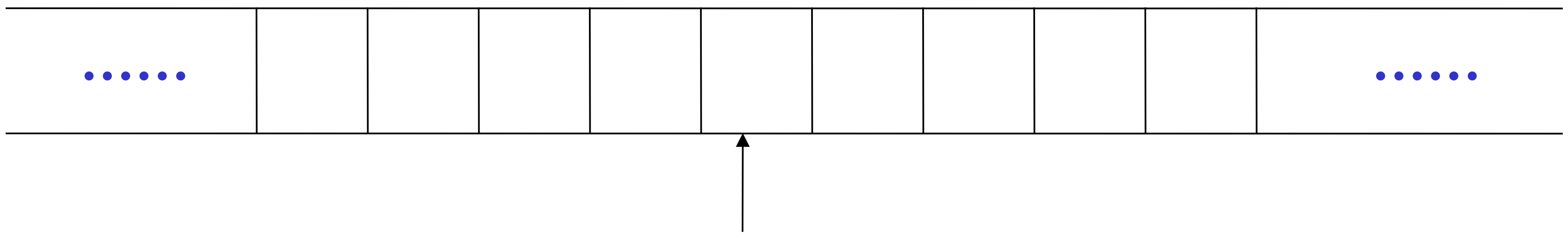
1. Change state.

2. Write a tape symbol in the cell scanned.

3. Move the tape head left or right.

The Tape

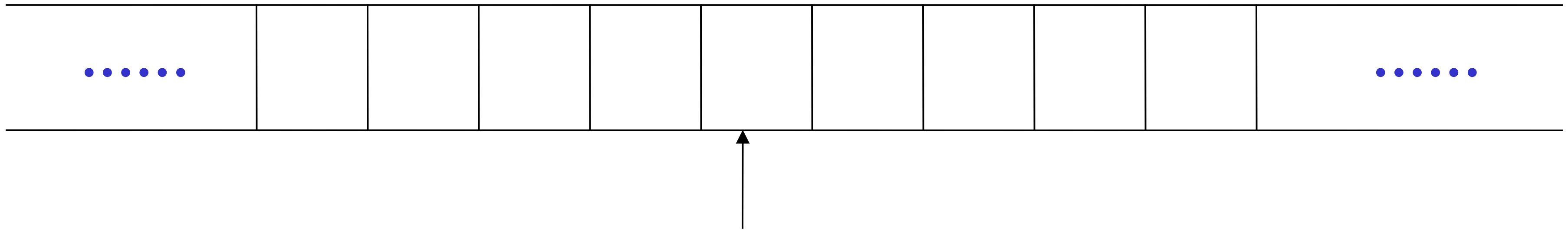
No boundaries -- infinite length



Read-Write head

The head moves Left or Right

The Tape



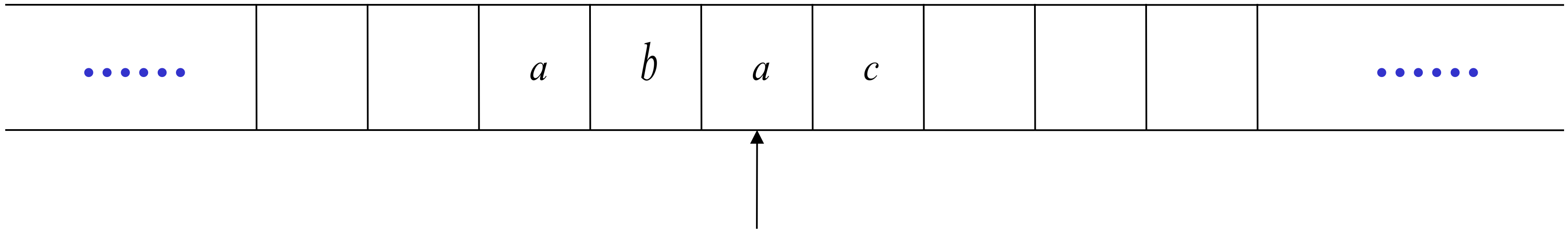
Read-Write head

The head at each transition (time step):

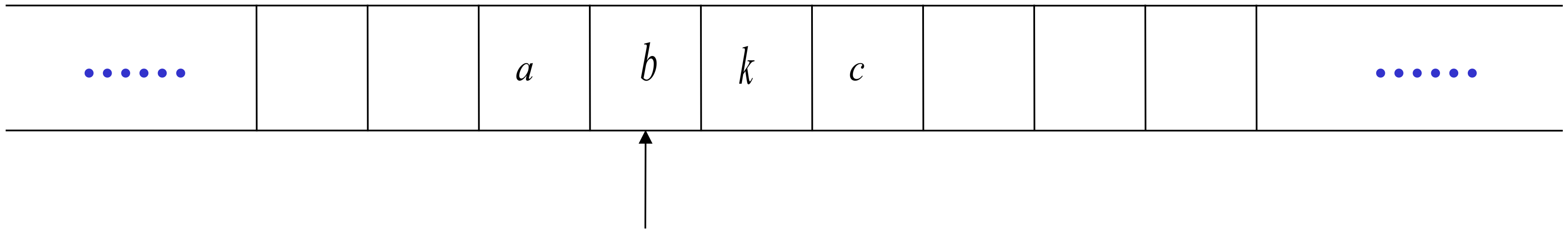
1. Reads a symbol
2. Writes a symbol
3. Moves Left or Right

Example:

Time 0

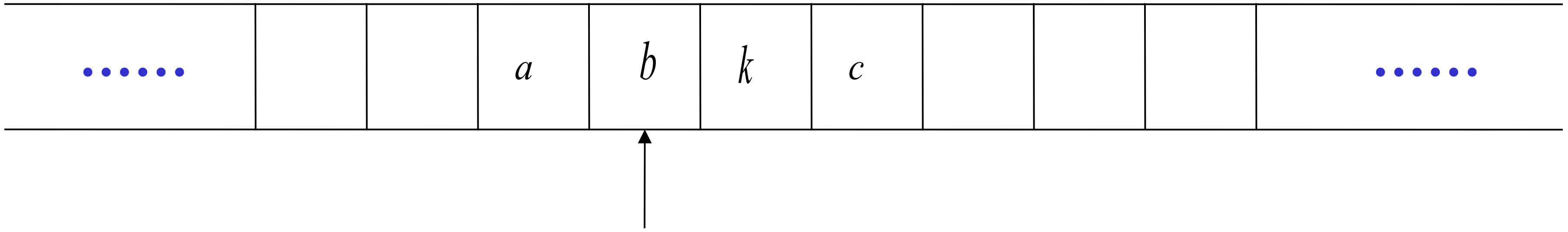


Time 1

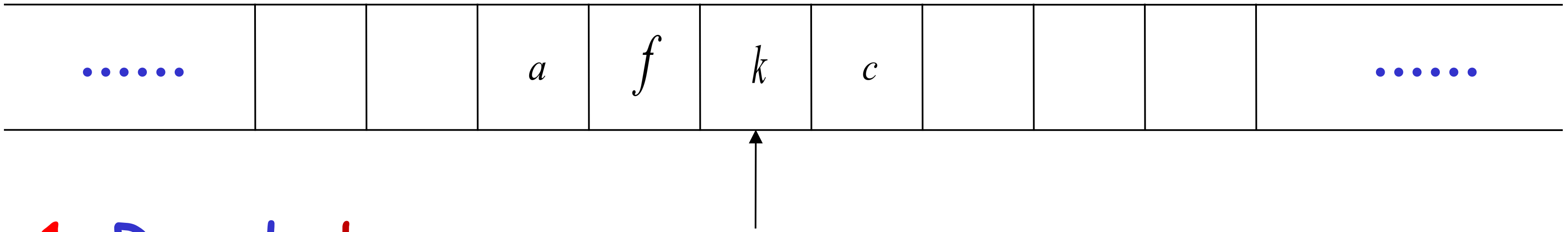


1. Reads a
2. Writes k
3. Moves Left

Time 1



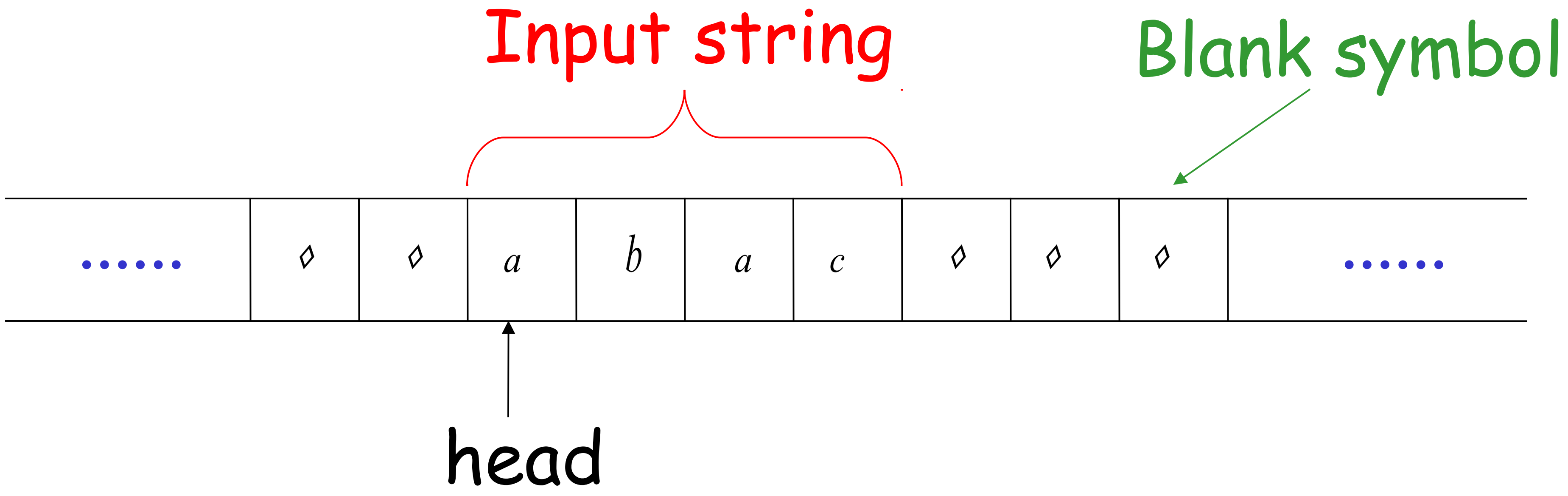
Time 2



1. Reads b

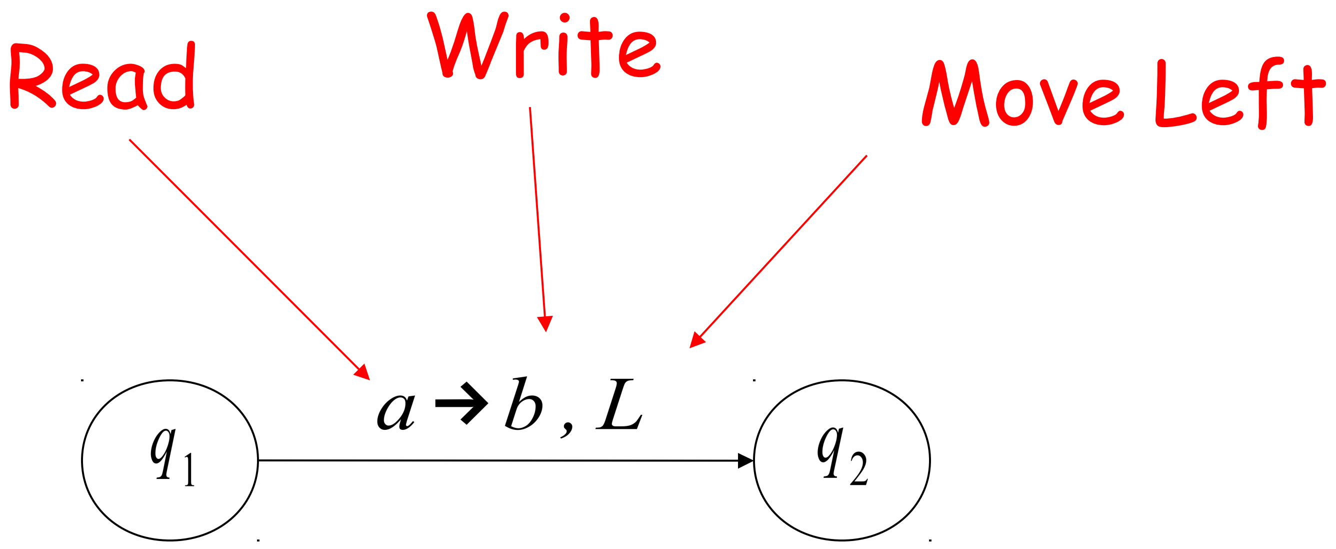
2. Writes f
3. Moves Right

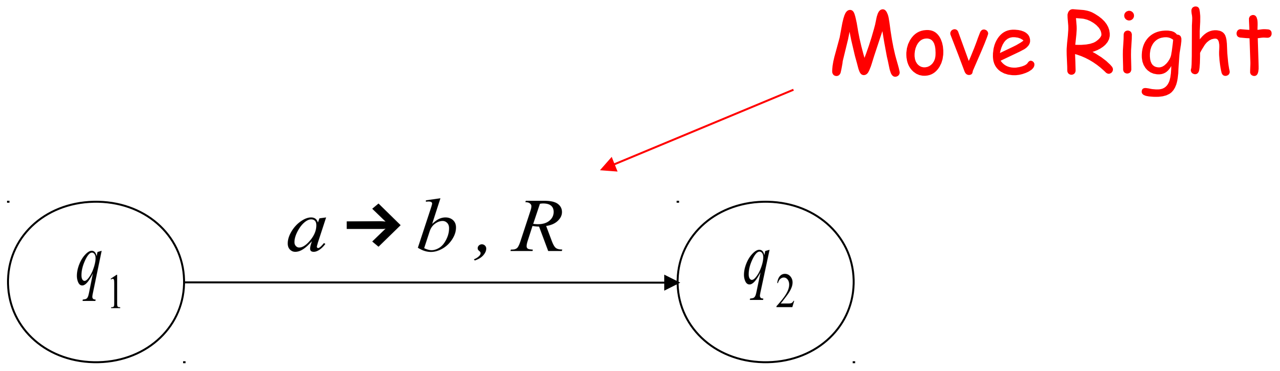
The Input String



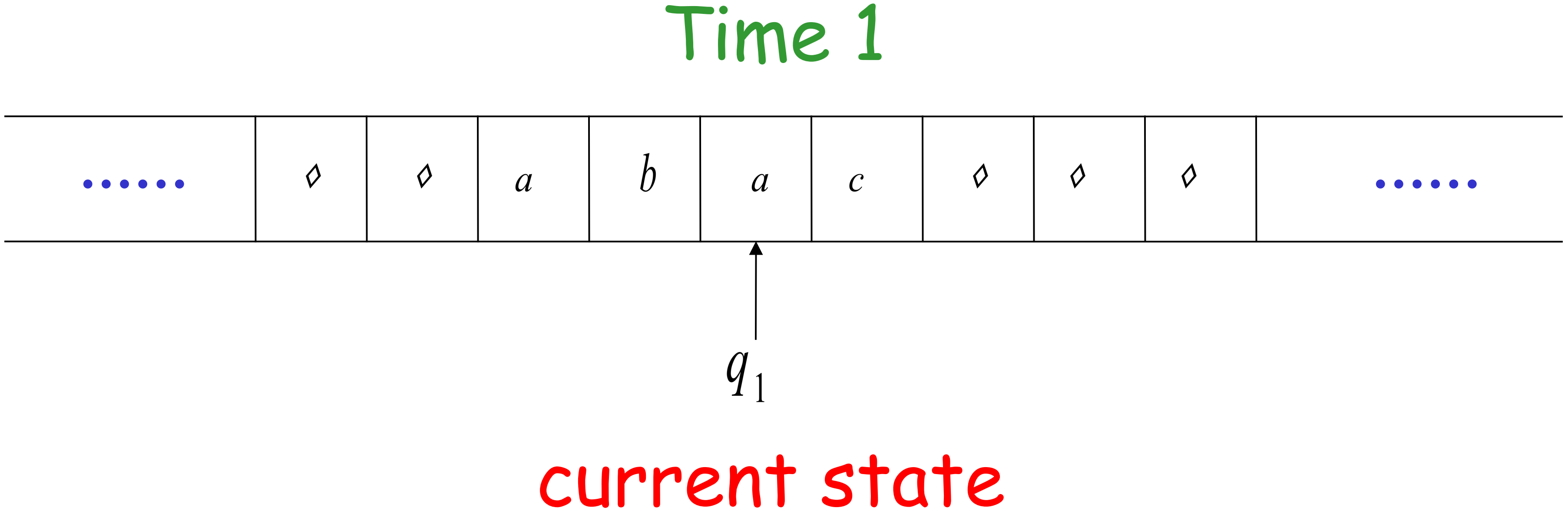
Head starts at the leftmost position of the input string

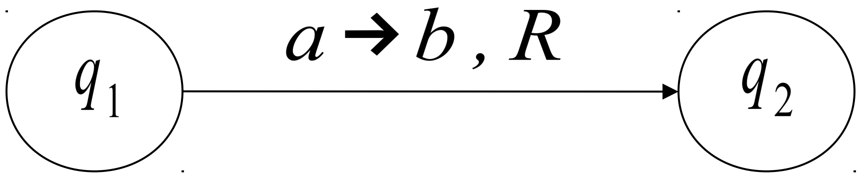
States & Transitions



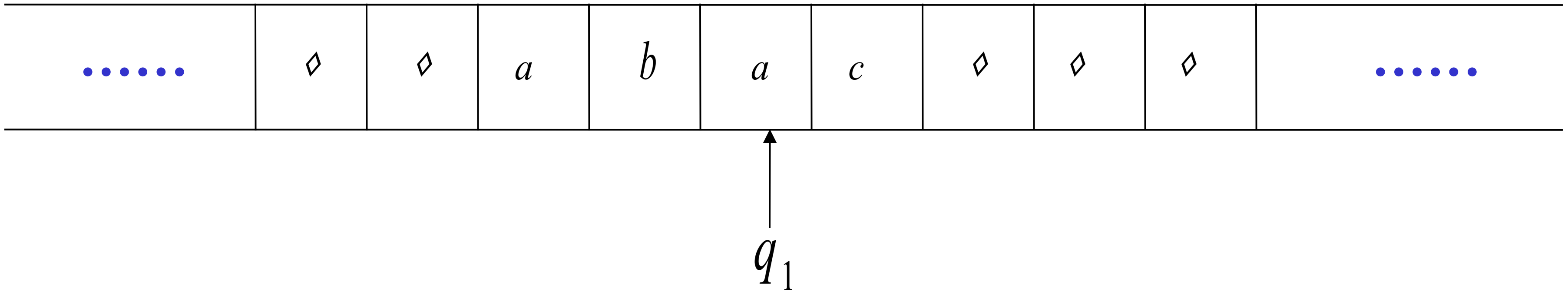


States & Transitions: Example 1

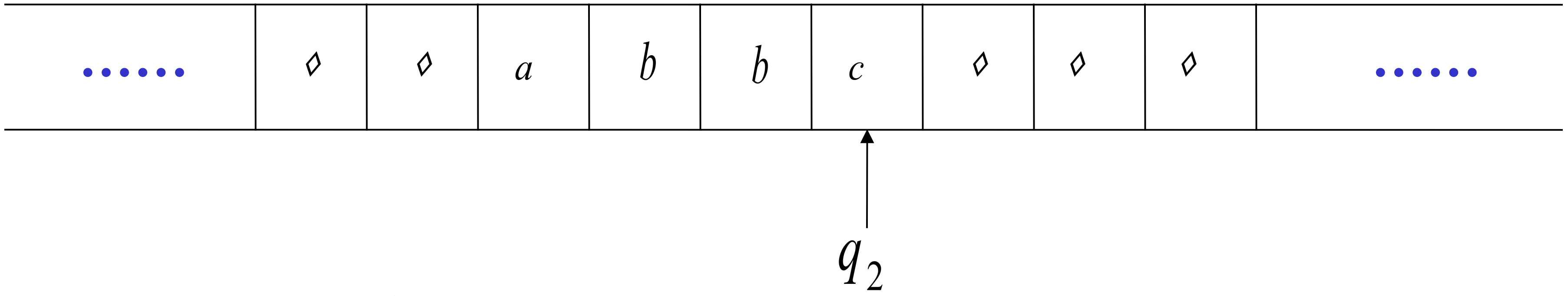


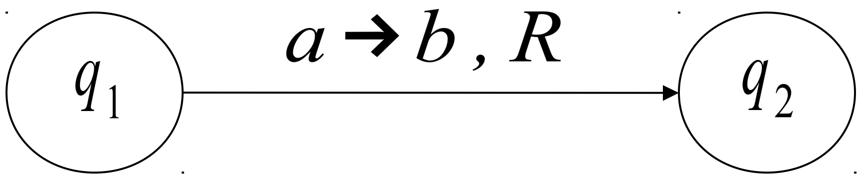


Time 1



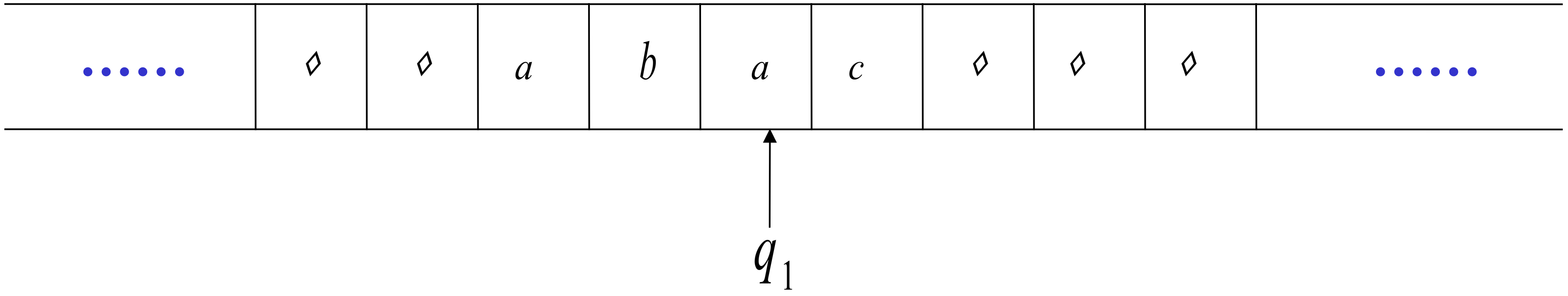
Time 2



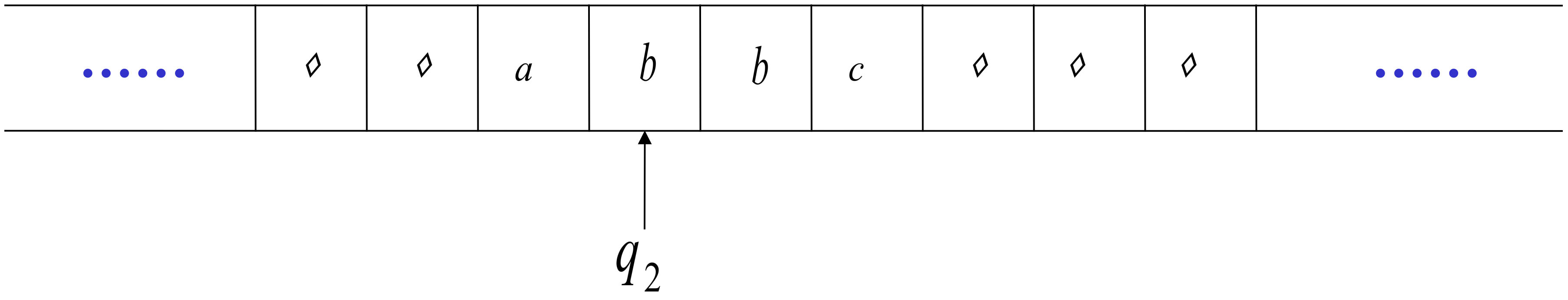


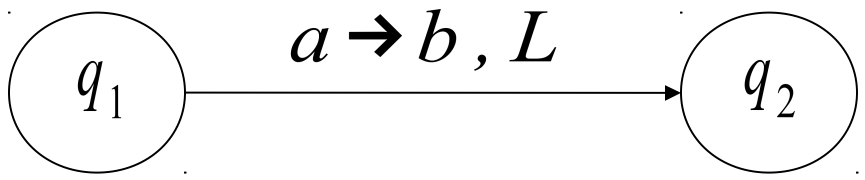
Example 2

Time 1



Time 2





Turing Machine: $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

- Q is the finite set of states of the finite control.
- Σ is the finite set of input symbols.
- Γ is the finite set of tape symbols ;$\Sigma \subset \Gamma$.
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function, which is a partial function.
- $q_0 \in Q$ is the start state.
- $B \in \Gamma$ is the blank symbol;
- $F \subseteq Q$ is the set of final or accepting states.

TM versus FA

1. A TM can both write on the tape and read from it; *a FA can only read its input*
2. The read/write head of a TM can move both to the left and to the right; *a FA can move in one direction only*
3. The tape of a TM is infinite; *the input of a FA is finite*
4. The special states of a TM for rejecting and accepting the input take immediate effect; *FA terminates when input is entirely consumed*

Transition Function: δ

$$\delta(q,x) \longrightarrow (p,Y,D)$$

Where,

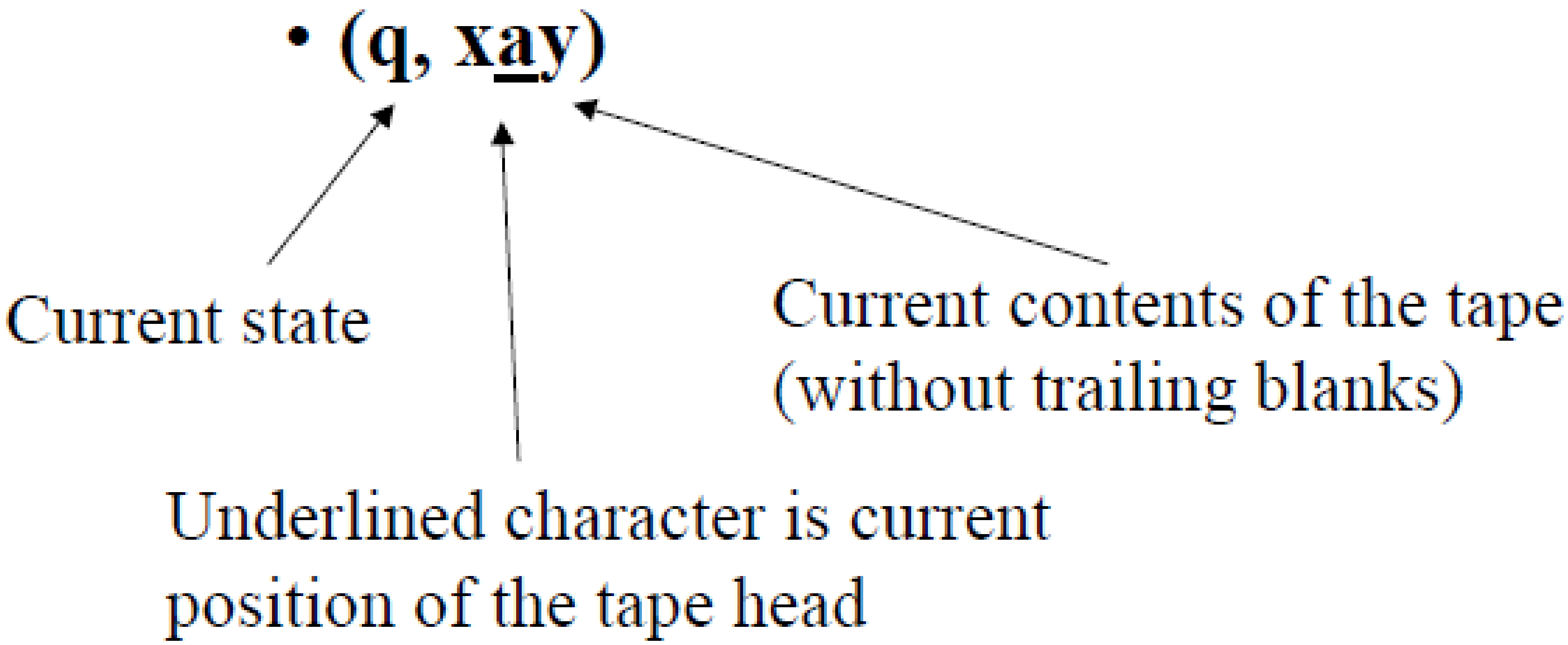
p = next state in Q

Y =replaced cell symbol

D = direction of the head , either L or R

Configuration of a TM

Configuration of a TM
– Gives the current “configuration” of a TM


• (q, x$\underline{a}$y)
Current state
Current contents of the tape (without trailing blanks)
Underlined character is current position of the tape head

Configuration of a TM

We indicate

- $(q, x\underline{a}y) \mapsto (p, u\underline{b}v)$

– If you can go from one configuration to another on a single move, and…

- $(q, x\underline{a}y) \mapsto^* (p, u\underline{b}v)$

– If you can go from one configuration to another in zero or more moves.

Instantaneous Descriptions for TMs

A Turing machine changes its configuration upon each move.

We use instantaneous descriptions (IDs) for describing such configurations.

An instantaneous description is a string of the form

$X_1X_2\cdots X_{i-1}\, qX_i\, X_{i+1}\cdots X_n$ where

1.q is the state of the Turing machine.

2. The tape head is scanning the ith symbol from the left.

3.$X_1X_2\cdots X_n$ is the portion of the tape between the leftmost and rightmost non blanks

The Moves and Language of a TM

We use $\underset{M}{\vdash}$ to designate a move of a Turing machine M from one ID to another.

If $\delta(q, X_i) = (p, Y, L)$, then:

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \underset{M}{\vdash}$$
$$X_1 X_2 \cdots X_{i-2} p X_{i-1} Y X_{i+1} \cdots X_n$$

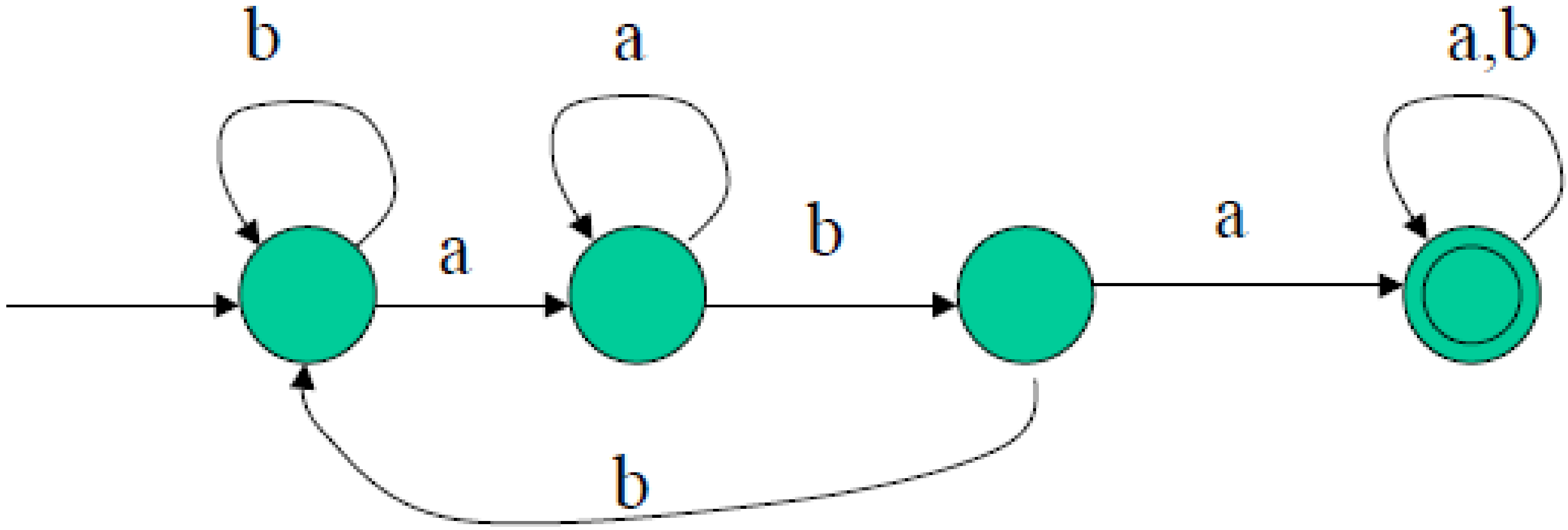
If $\delta(q, X_i) = (p, Y, R)$, then:

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \underset{M}{\vdash}$$
$$X_1 X_2 \cdots X_{i-1} Y p X_{i+1} \cdots X_n$$

The reflexive-transitive closure of $\underset{M}{\vdash}$ is denoted by $\overset{*}{\underset{M}{\vdash}}$.

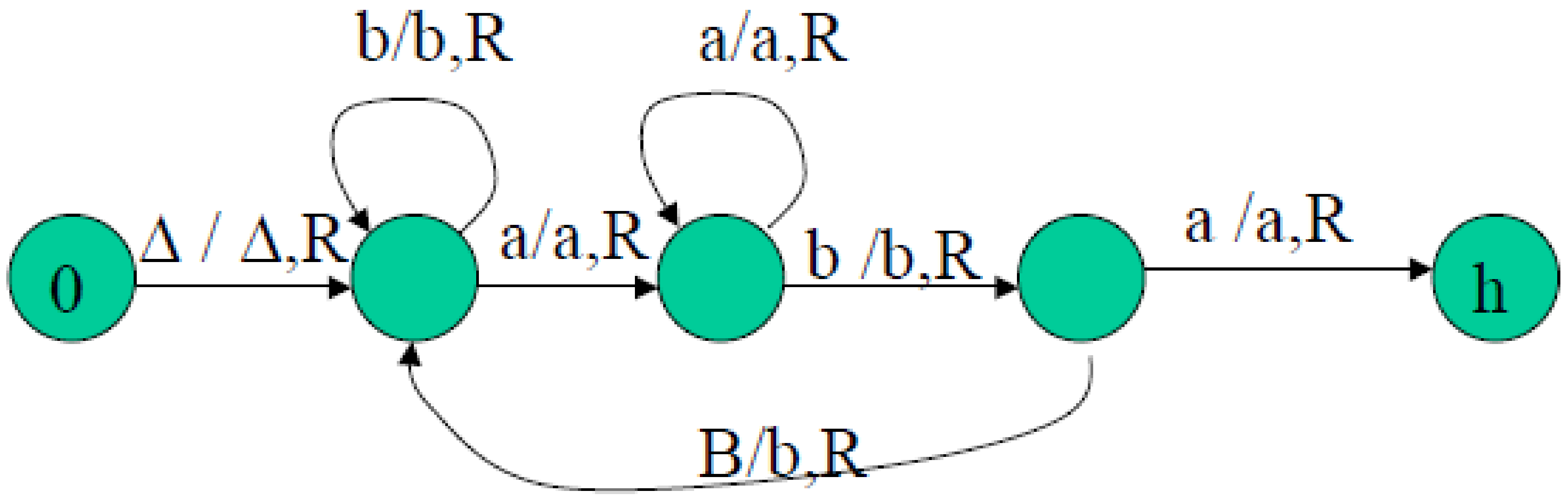
TMs and Regular Languages

L = { x ∈ { a, b }* | x contains the substring aba }



TMs and Regular Languages

L = { x ∈ { a, b }* | x contains the substring aba }

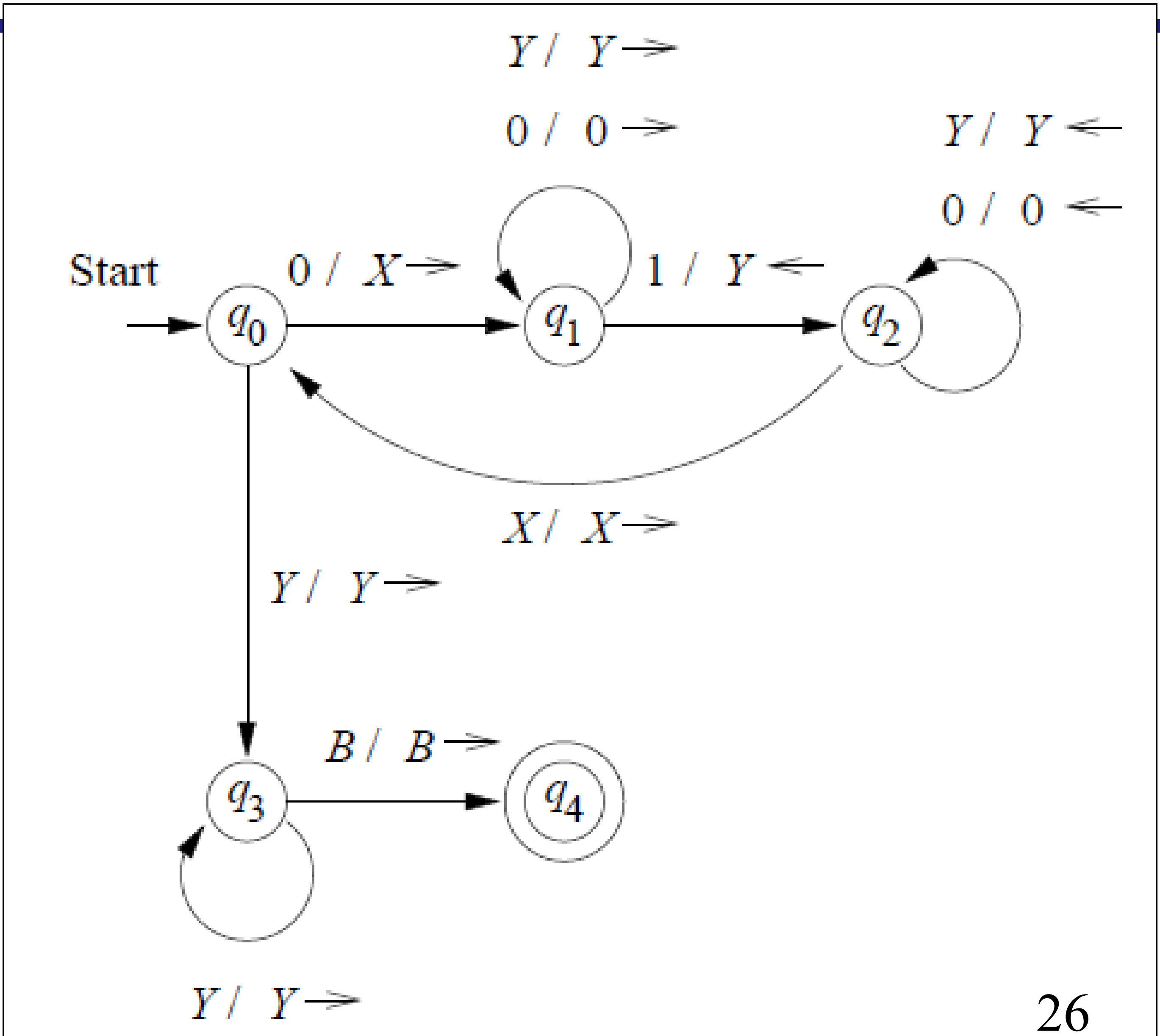


Example: A TM for $\{0^n1^n : n \geq 1\}$

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$
where δ is given by the following table:

	0	1	X	Y	B
$\rightarrow q_0$	(q_1, X, R)			(q_3, Y, R)	
q_1	$(q_1, 0, R)$	(q_2, Y, L)		(q_1, Y, R)	
q_2	$(q_2, 0, L)$		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(q_4, B, R)
$\star\ q_4$					

Transition Diagram: A TM for $\{0^n 1^n : n \geq 1\}$



Design a TM for 0011

$$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 0Y1 \vdash q_2 X0Y1 \vdash$$
$$Xq_0 0Y1 \vdash XXq_1 Y1 \vdash XXYq_1 1 \vdash XXq_2 YY \vdash Xq_2 XYY \vdash$$
$$XXq_0 YY \vdash XXYq_3 Y \vdash XXYYq_3 B \vdash XXYYBq_4 B$$

0010---not accepted

$$q_0 0010 \vdash Xq_1 010 \vdash X0q_1 10 \vdash Xq_2 0Y0 \vdash q_2 X0Y0 \vdash$$
$$Xq_0 0Y0 \vdash XXq_1 Y0 \vdash XXYq_1 0 \vdash XXY0q_1 B$$

Language of TM

Let M= (Q,Σ,Γ, δ, q_0, B, F) be a Turing Machine.

Then L(M) is the set of strings w in Σ* such that

$\mathbf{q_0\, w \vdash^* \alpha p \beta}$ for some state p in F and any tape strings α and ß

Recursive language(Turing Decidable): there is a TM, corresponding to the concept of algorithm, that halts eventually, whether it accepts or not

Recursively enumerable language: there is a TM that halts if the string is accepted

Acceptance by Halting

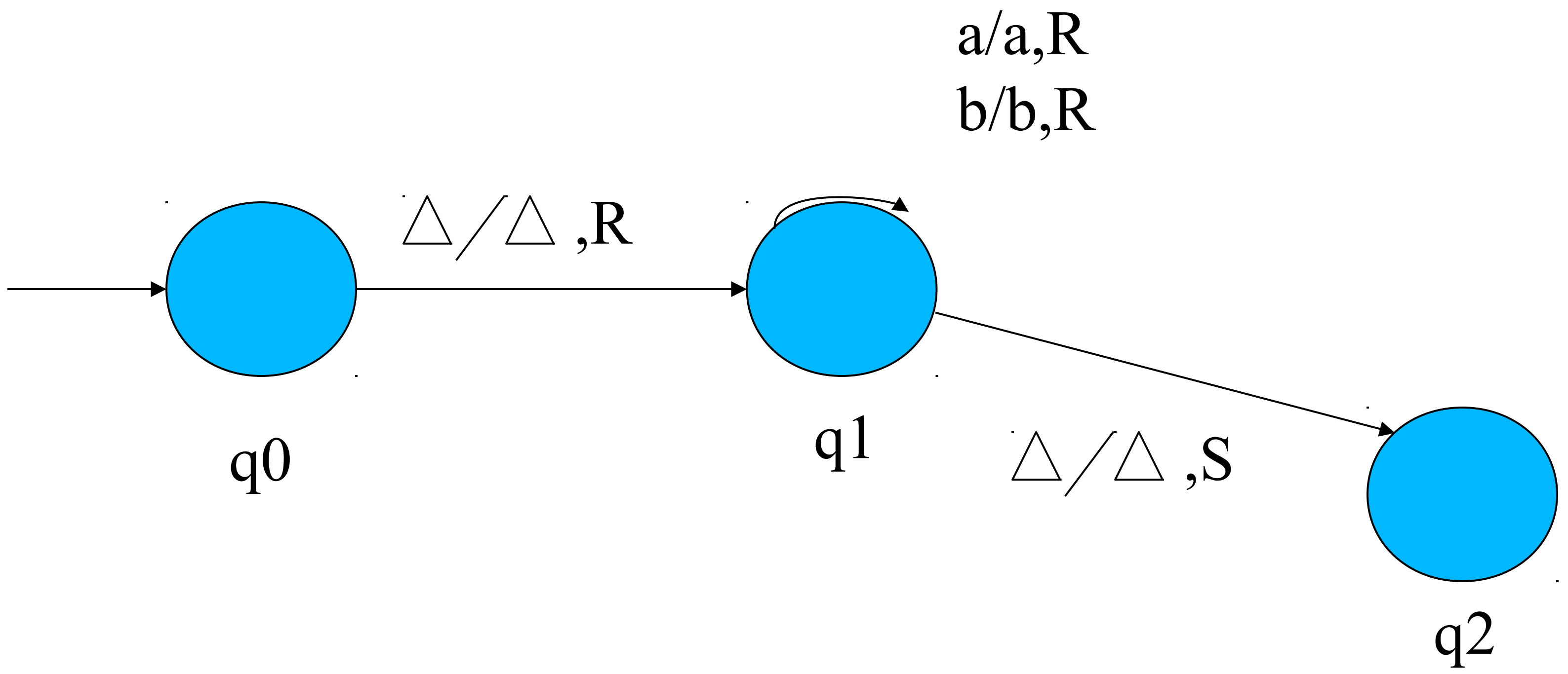
- A Turing machine halts if it enters a state q, scanning a tape symbol X, and there is no move in this situation, i.e.,$\delta(q, X)$ is undefined.
- We can always assume that a Turing machine halts if it accepts, as we can make $\delta(q, X)$ un-defined whenever q is an accepting state.
- Unfortunately, it is not always possible to require that a Turing machine halts even if it does not accept

The execution of a TM

- The machine “halts” (ends up in the halting state)--ACCEPT
- The machine has nowhere to go (at a state, reading a symbol where no transition is defined)--REJECT
- The machine “crashes” (tries to move the tape head to before the start of the tape)--REJECT
- The machine goes into an “infinite loop” (never halts) --REJECT

TM: example 1

…BBBababbaBBB…

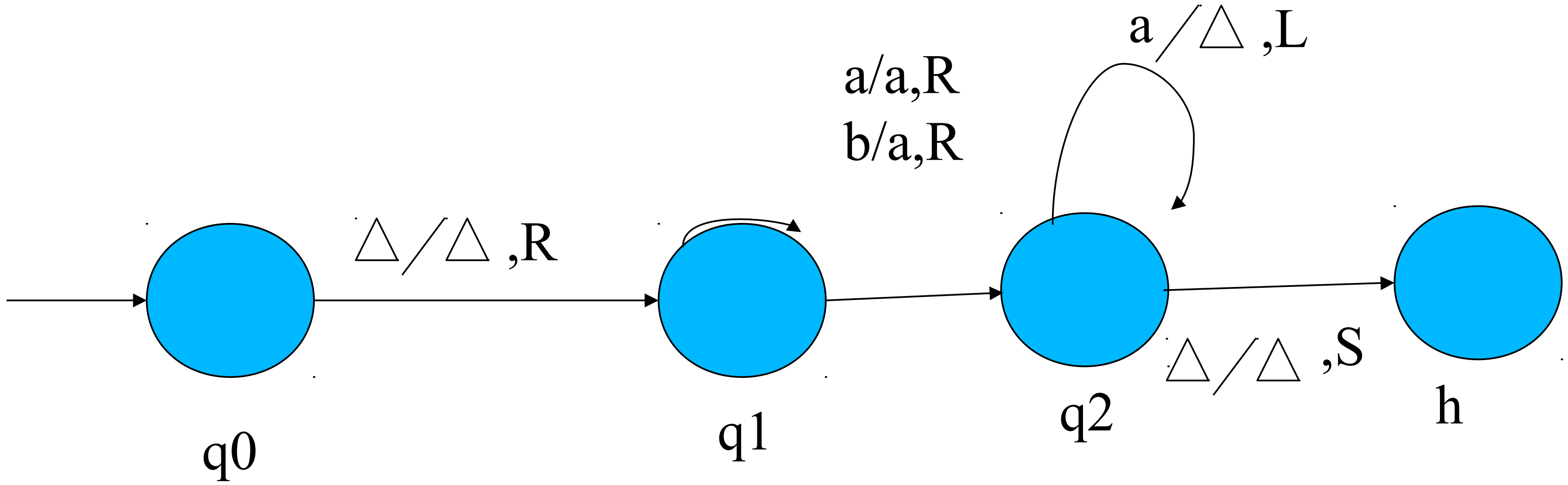


You can use S or L to stop

TM: example 2

Erase the string on the tape and move the head to left end

…BBBababbaBBB…

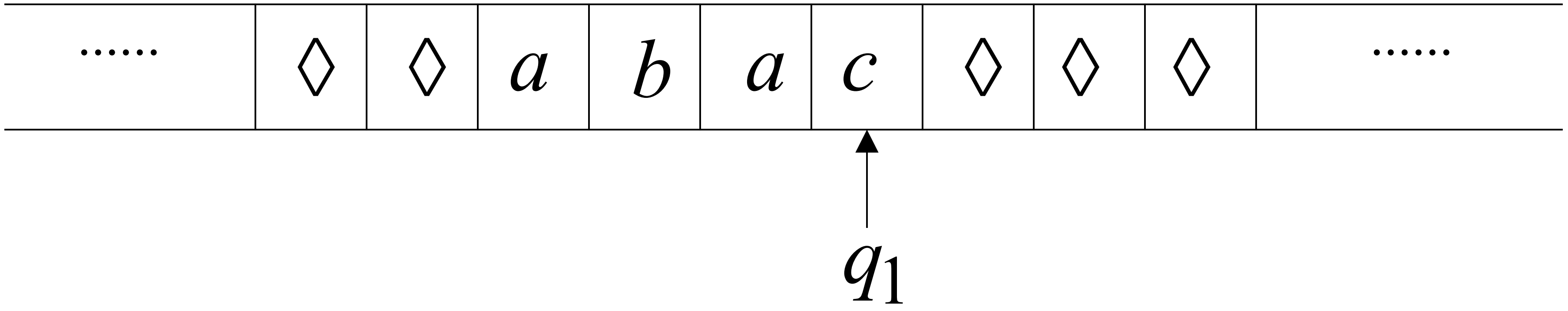


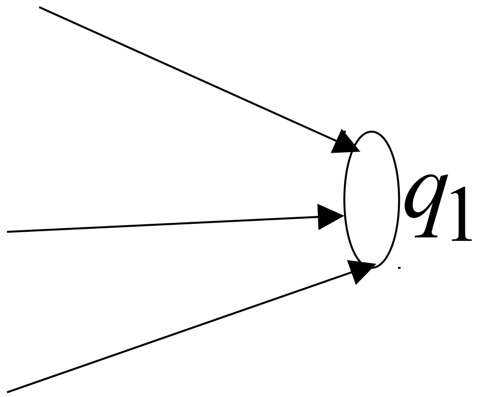
You can use S or L to stop

Halting

The machine ***halts*** in a state if there is no transition to follow

Halting Example 1:

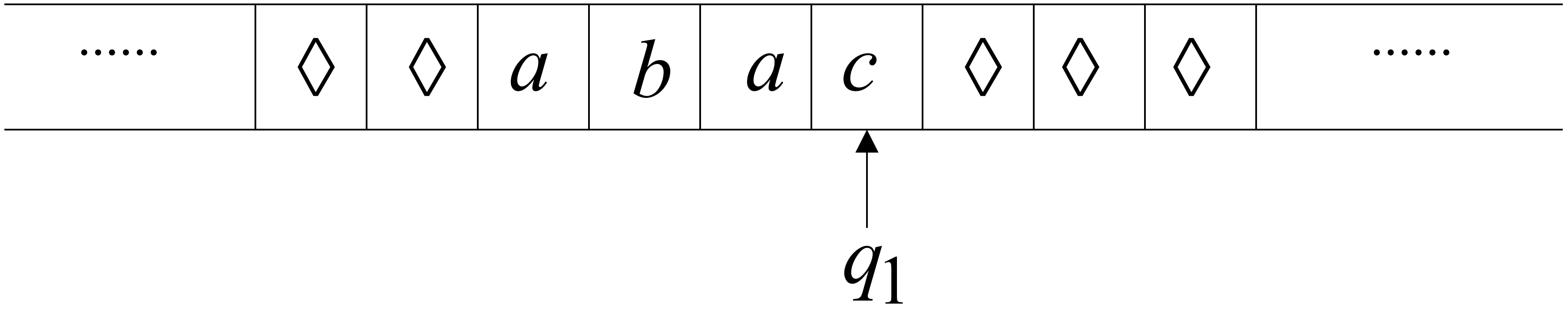




No transition from q_1

HALT!!!

Halting Example 2:



$$\begin{tikzcd}
 & q_2 \\
\arrow[r] & q_1 \arrow[u, "a \rightarrow b, R"] \arrow[d, "b \rightarrow d, L"'] \\
 & q_3
\end{tikzcd}$$

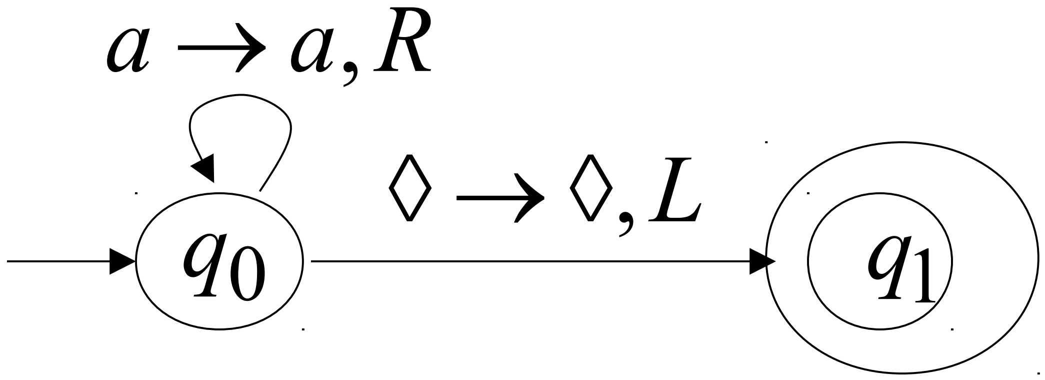
No possible transition from q_1 and symbol c

HALT!!!

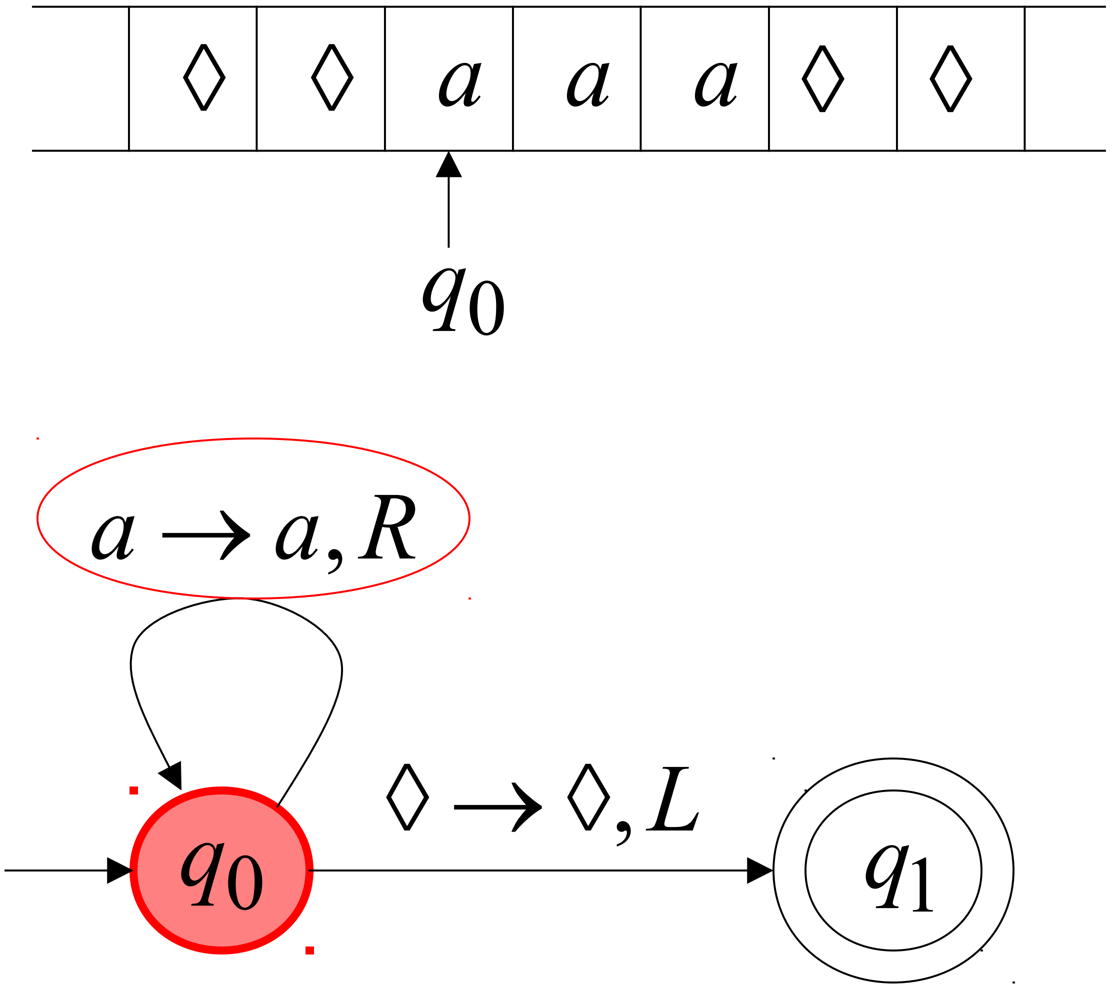
Turing Machine Example

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

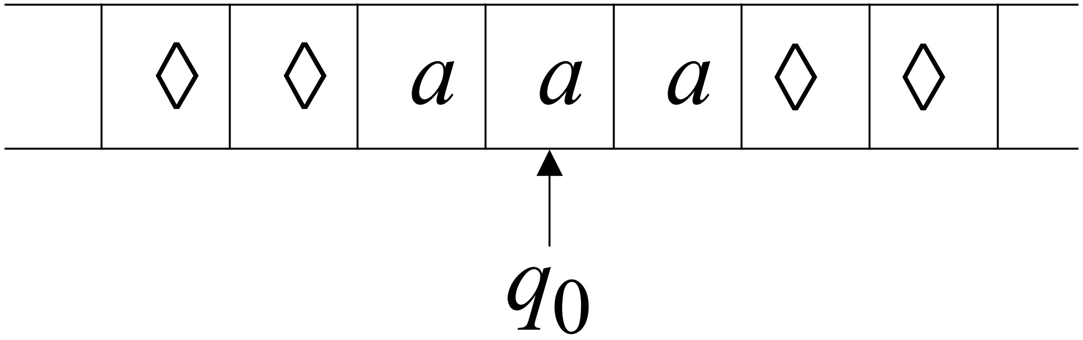
Accepts the language: a^*

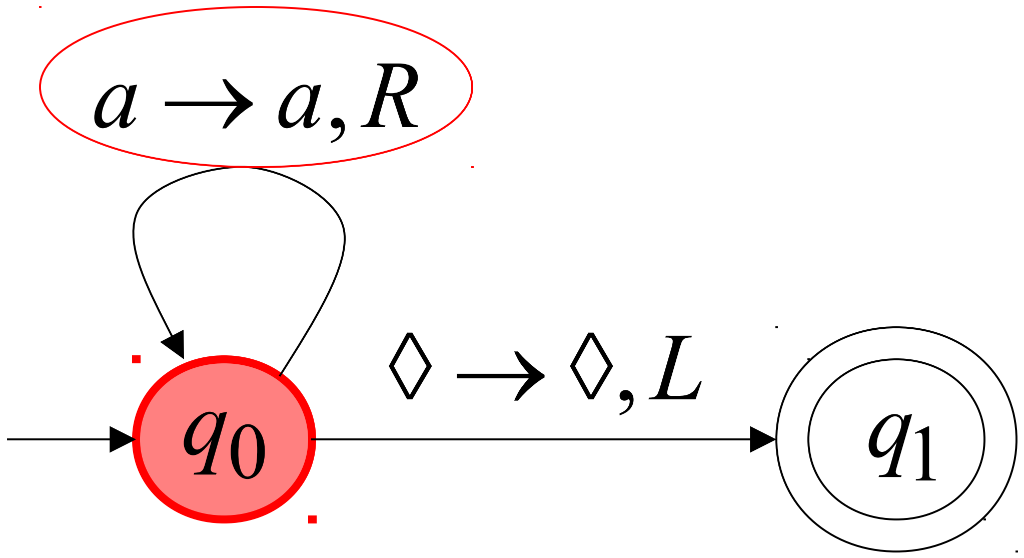


Time 0

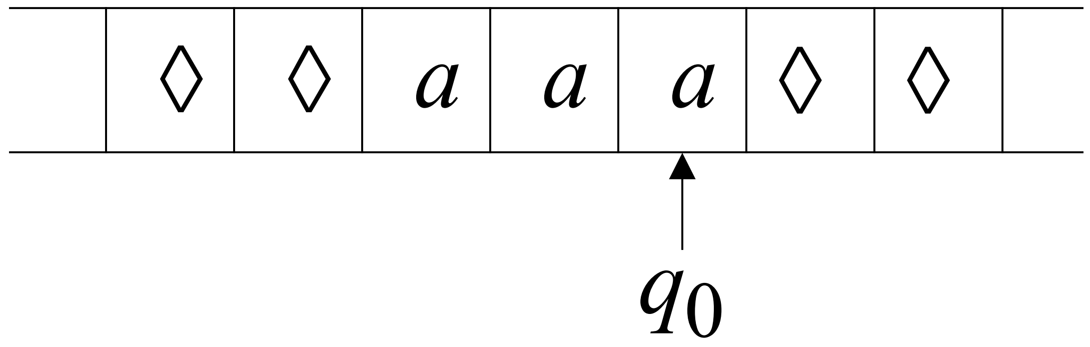


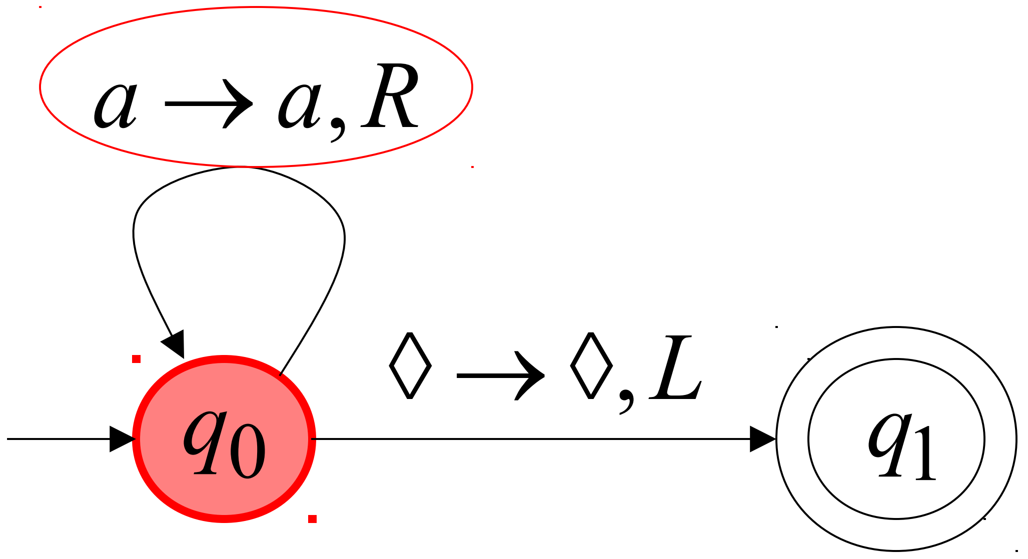
Time 1



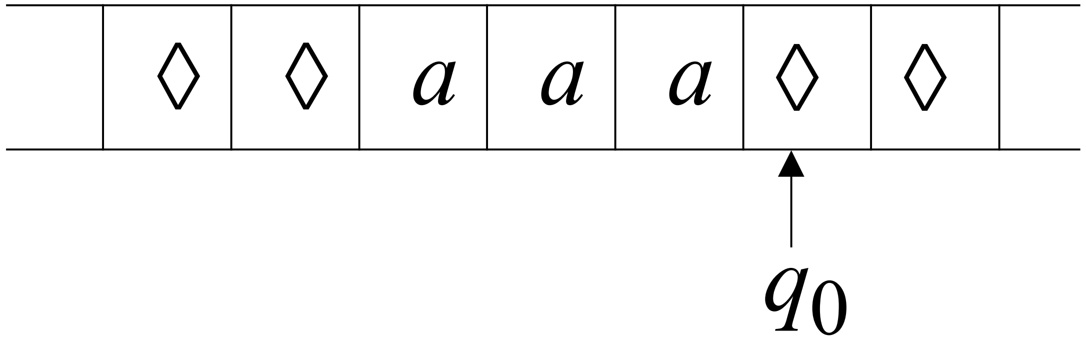


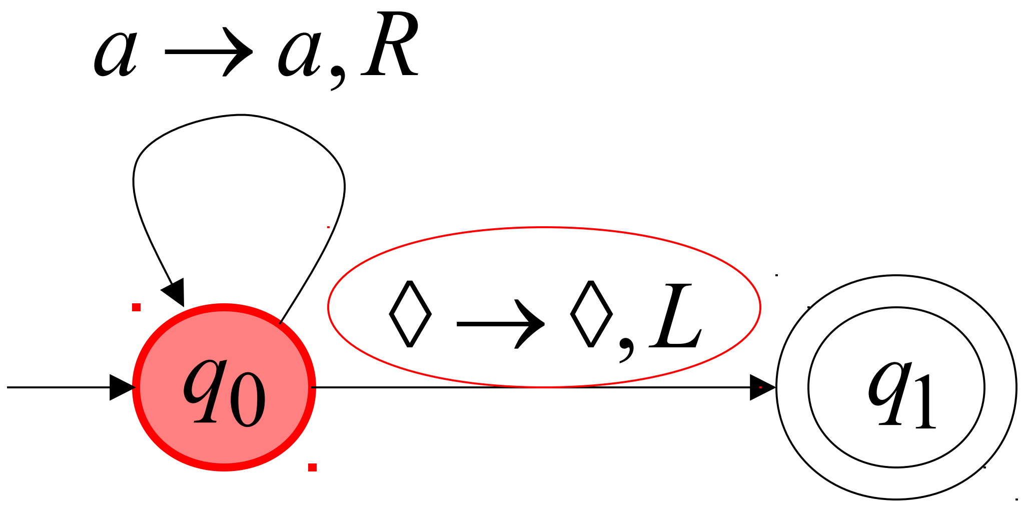
Time 2



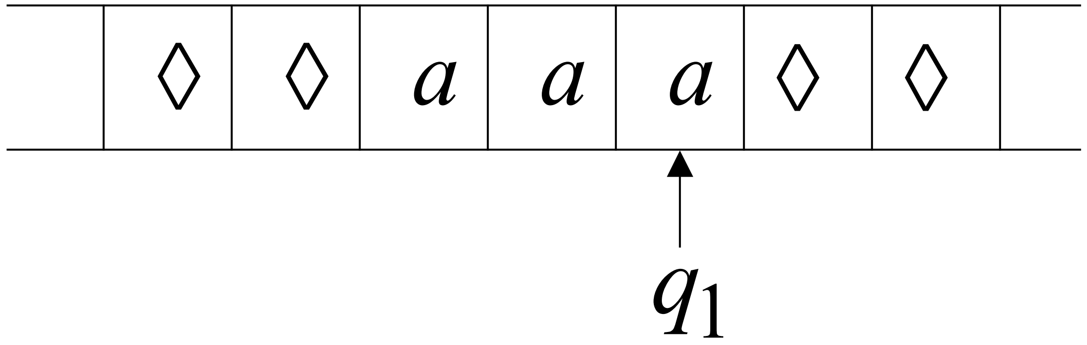


Time 3

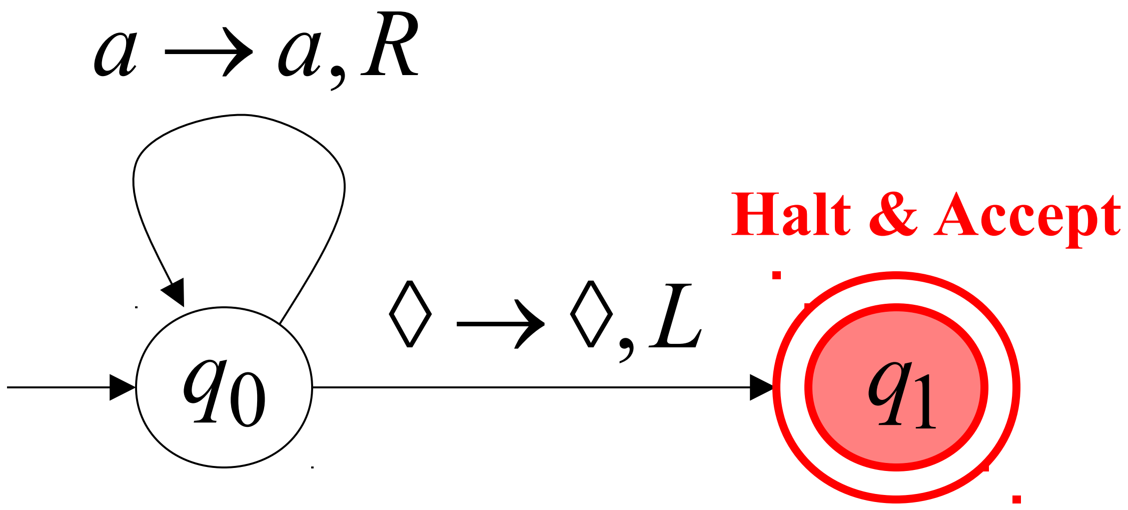




Time 4



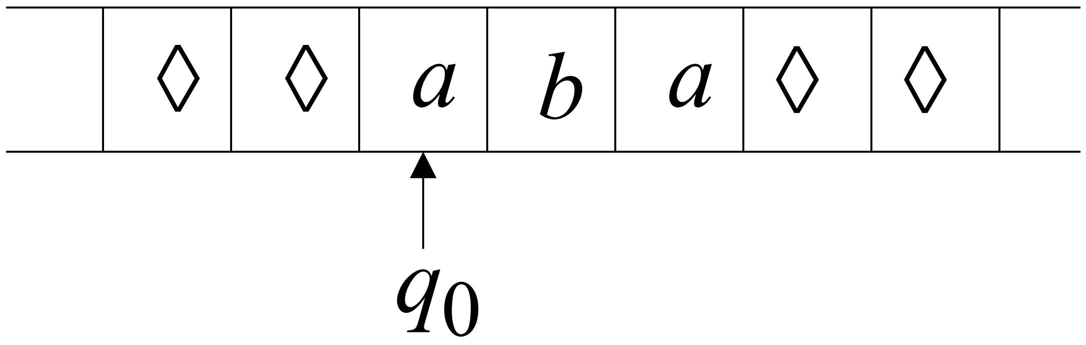
$\Diamond$ $\Diamond$ a a a $\Diamond$ $\Diamond$
q_1

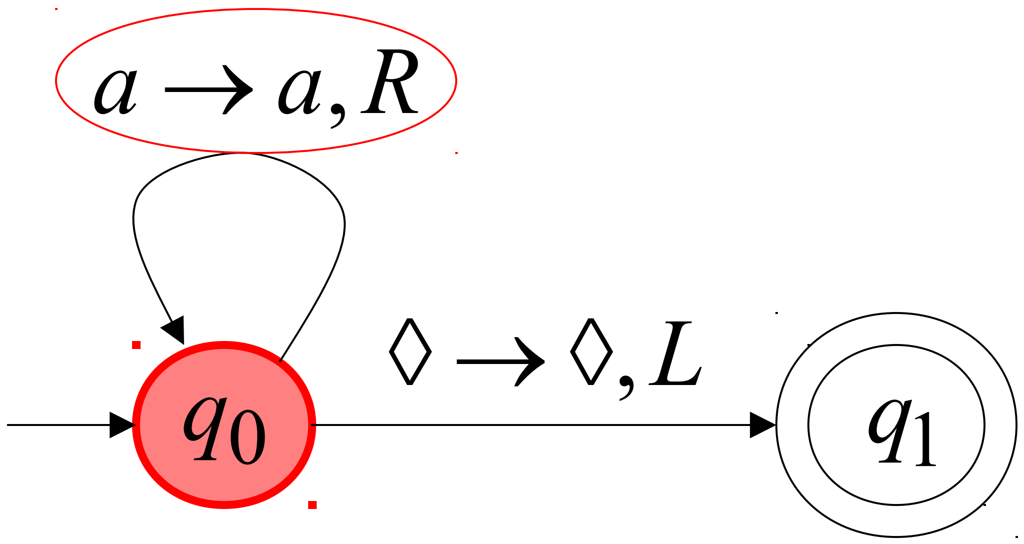


$a \rightarrow a, R$
$\Diamond \rightarrow \Diamond, L$
q_0
q_1
Halt & Accept

Rejection Example

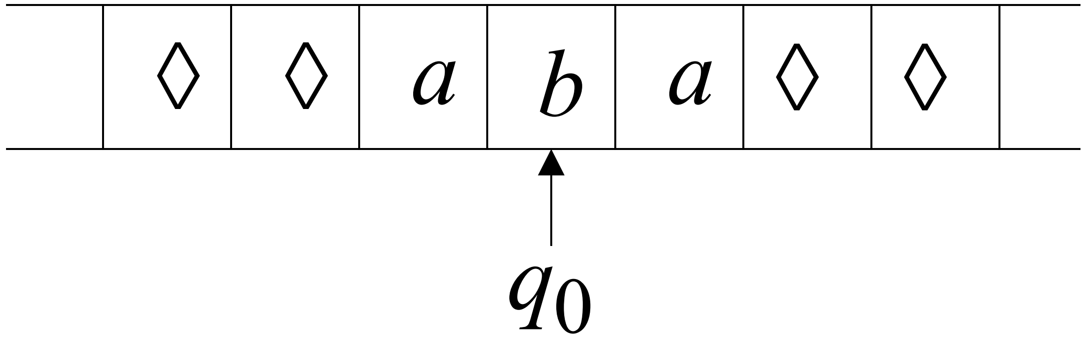
Time 0



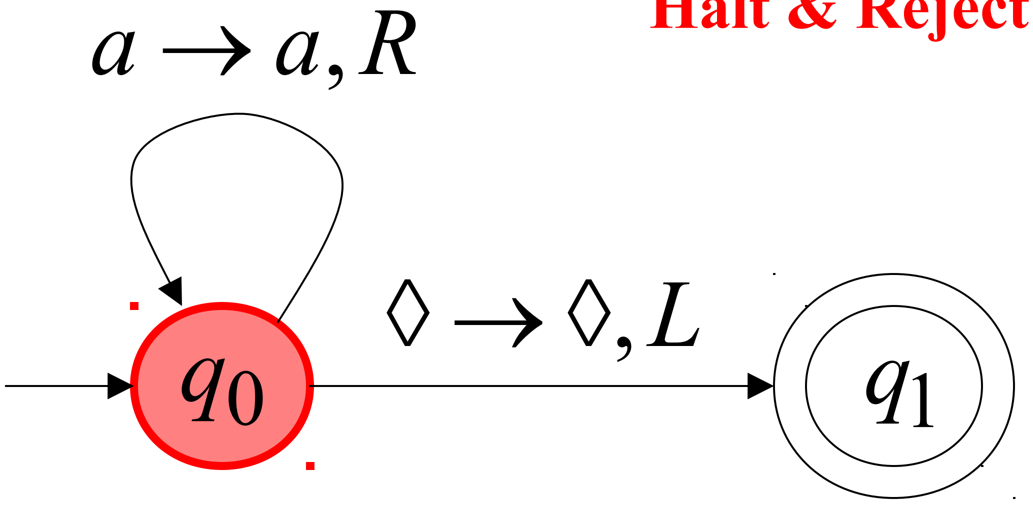


Rejection Example

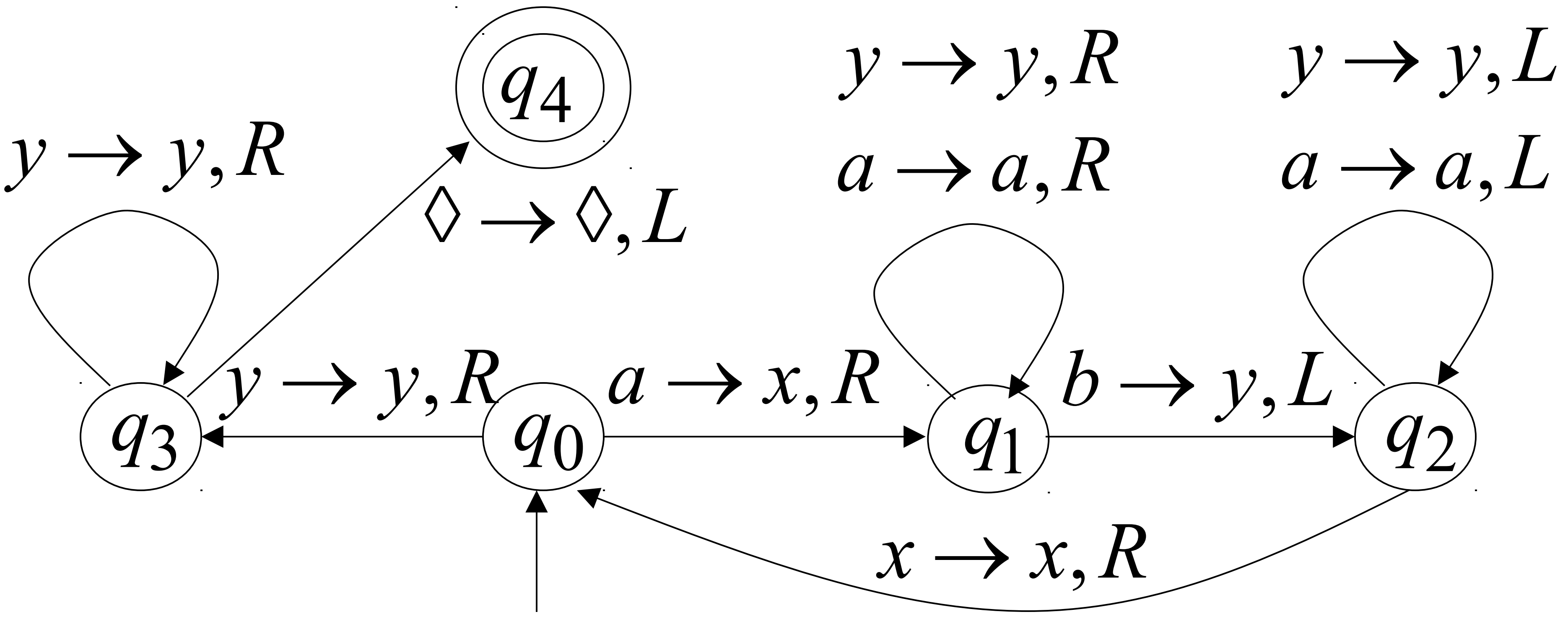
Time 1



Halt & Reject



Turing machine for the language $\{a^n b^n\}$ $n \geq 1$



Turing machine for the language $\{a^n b^n\} \quad n \geq 1$

Basic Idea:

Match a's with b's:

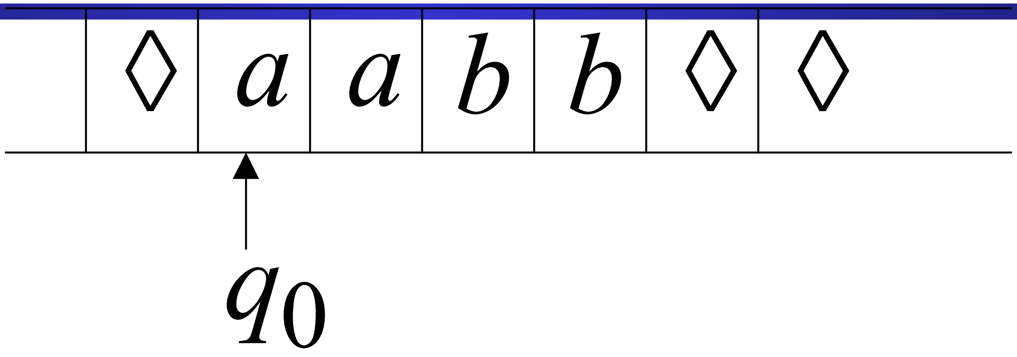
Repeat:

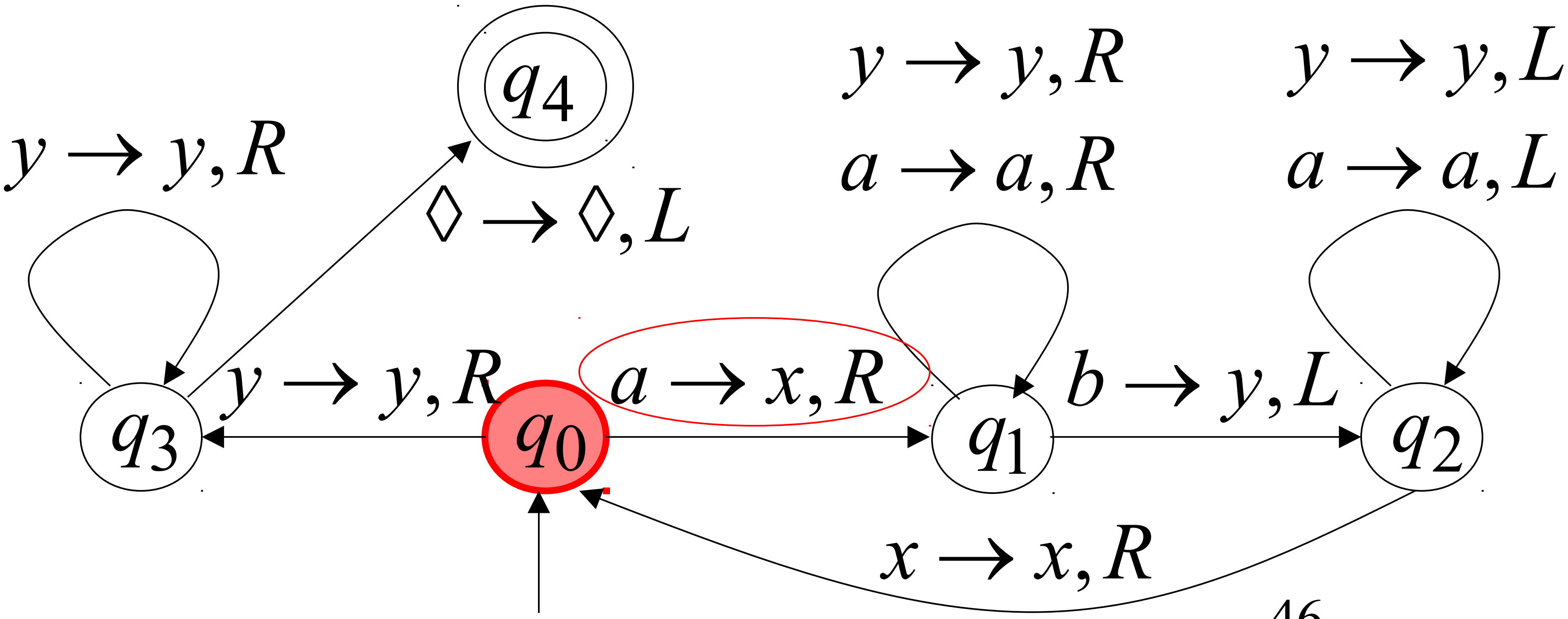
- replace leftmost a with x
- find leftmost b and replace it with y

Until there are no more a's or b's

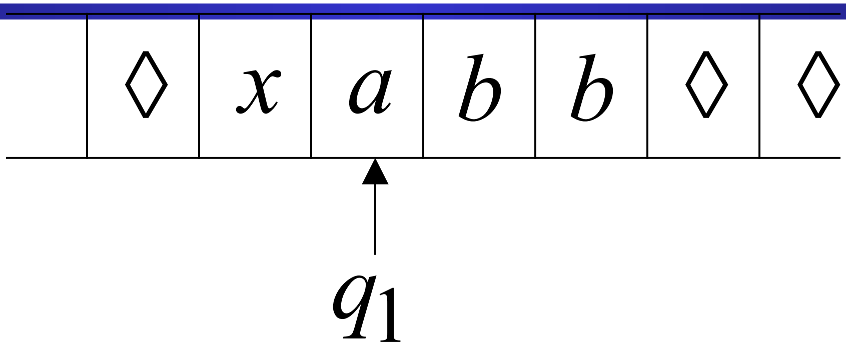
If there is a remaining a or b, reject

Time 0

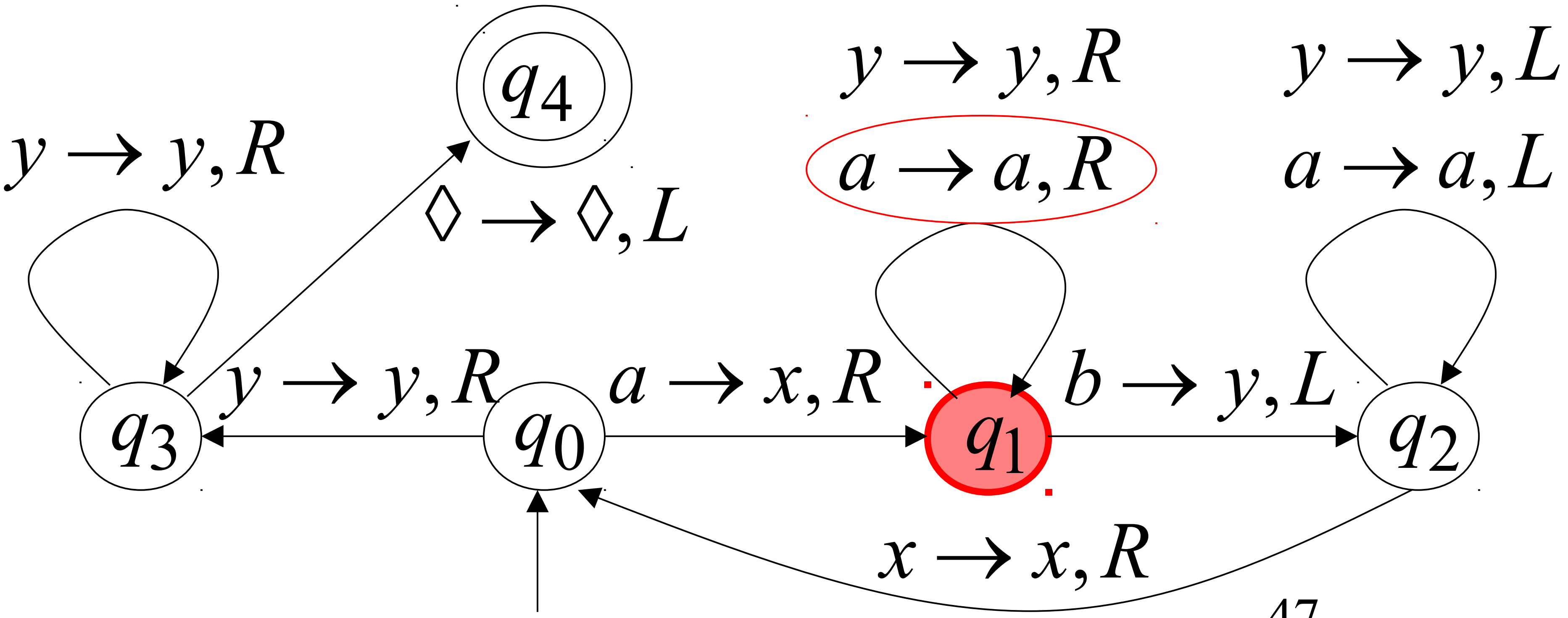

◊
a
a
b
b
◊
◊
q_0


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

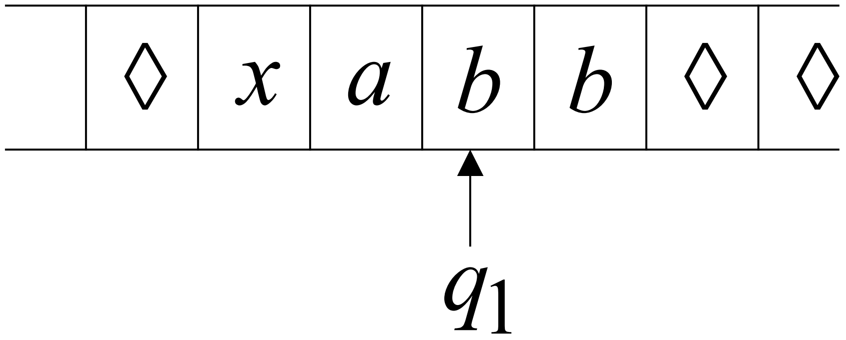


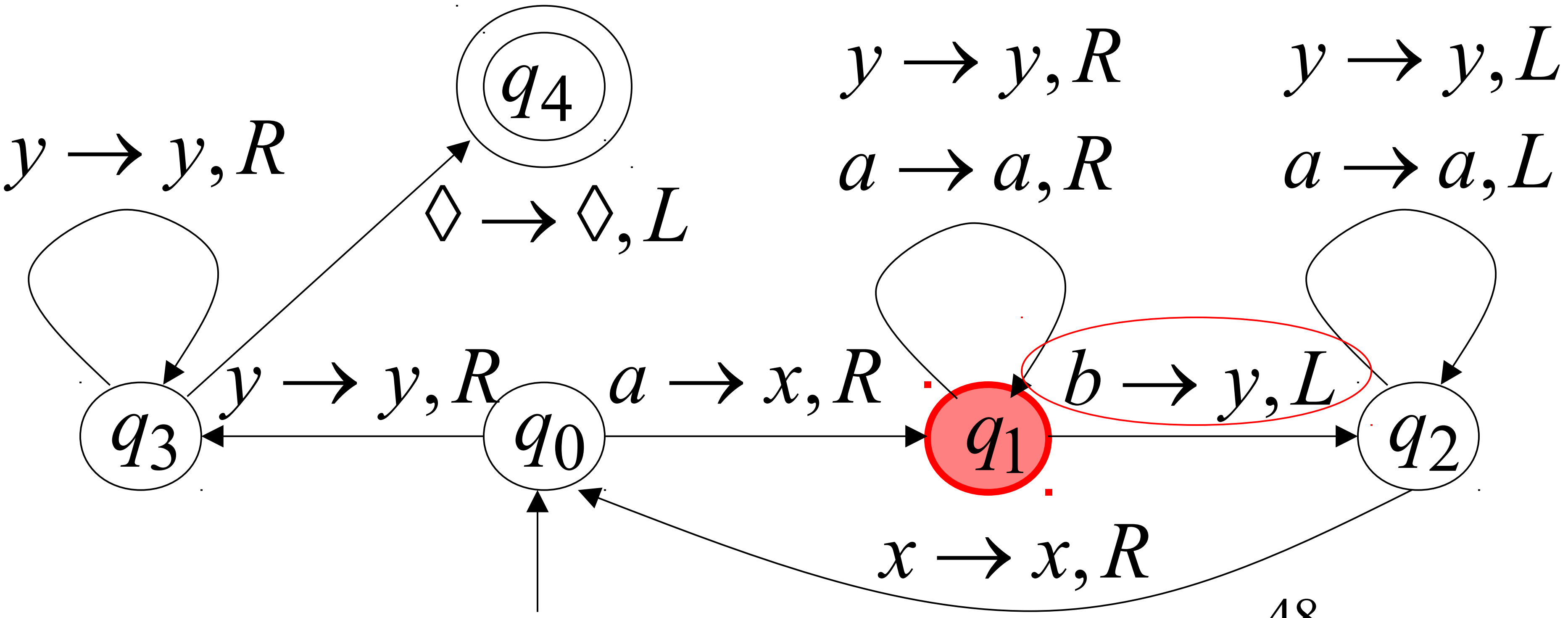
◊ x a b b ◊ ◊
q_1



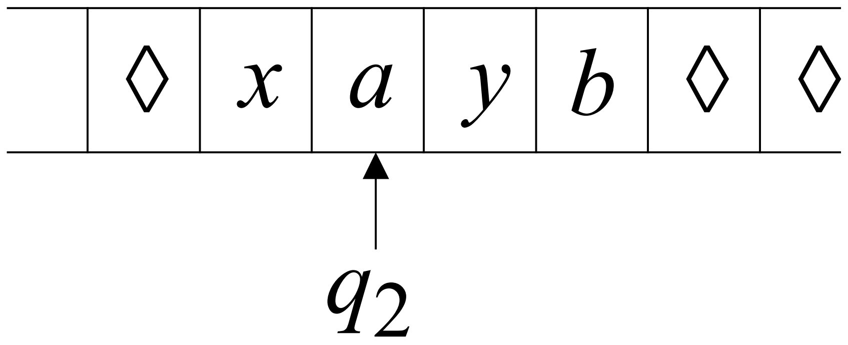
q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

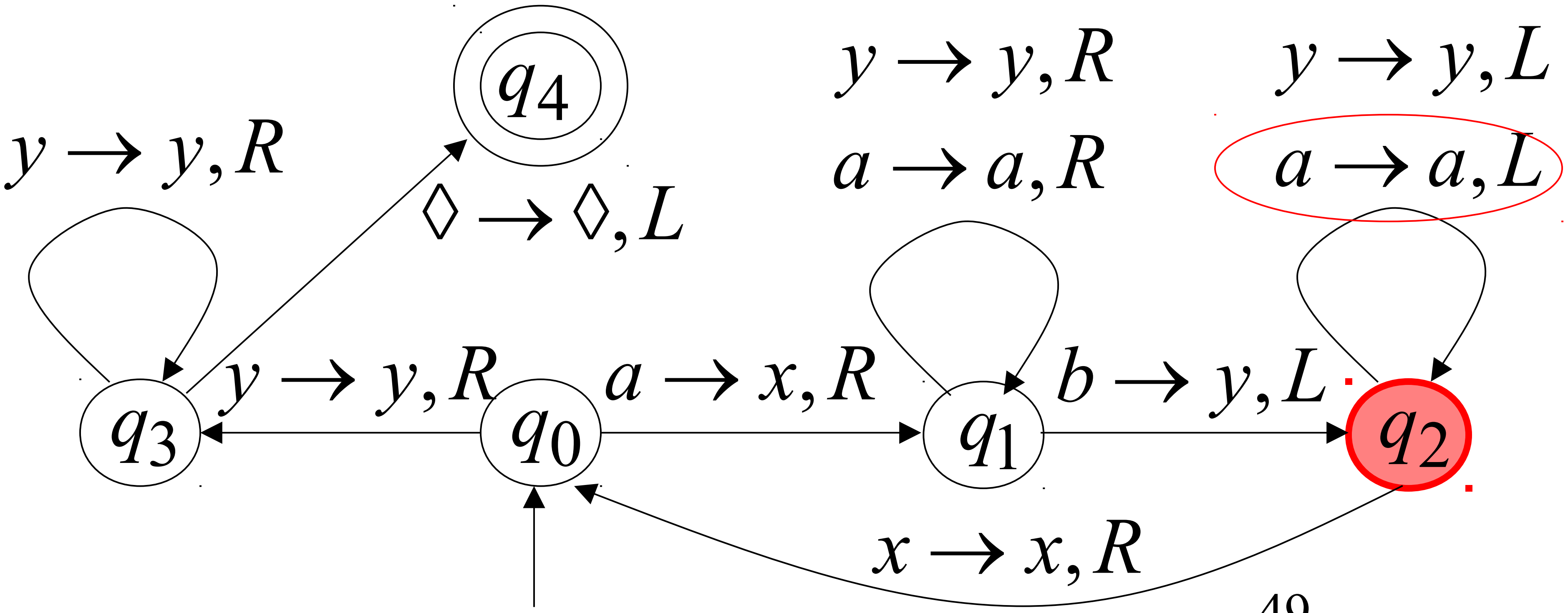
Time 2


◊
x
a
b
b
◊
◊
q_1

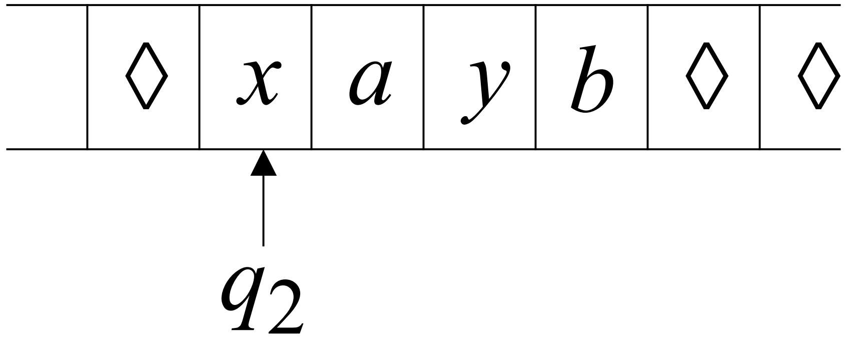

q_4
q_3
q_0
q_1
q_2
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
$x \rightarrow x, R$

Time 3

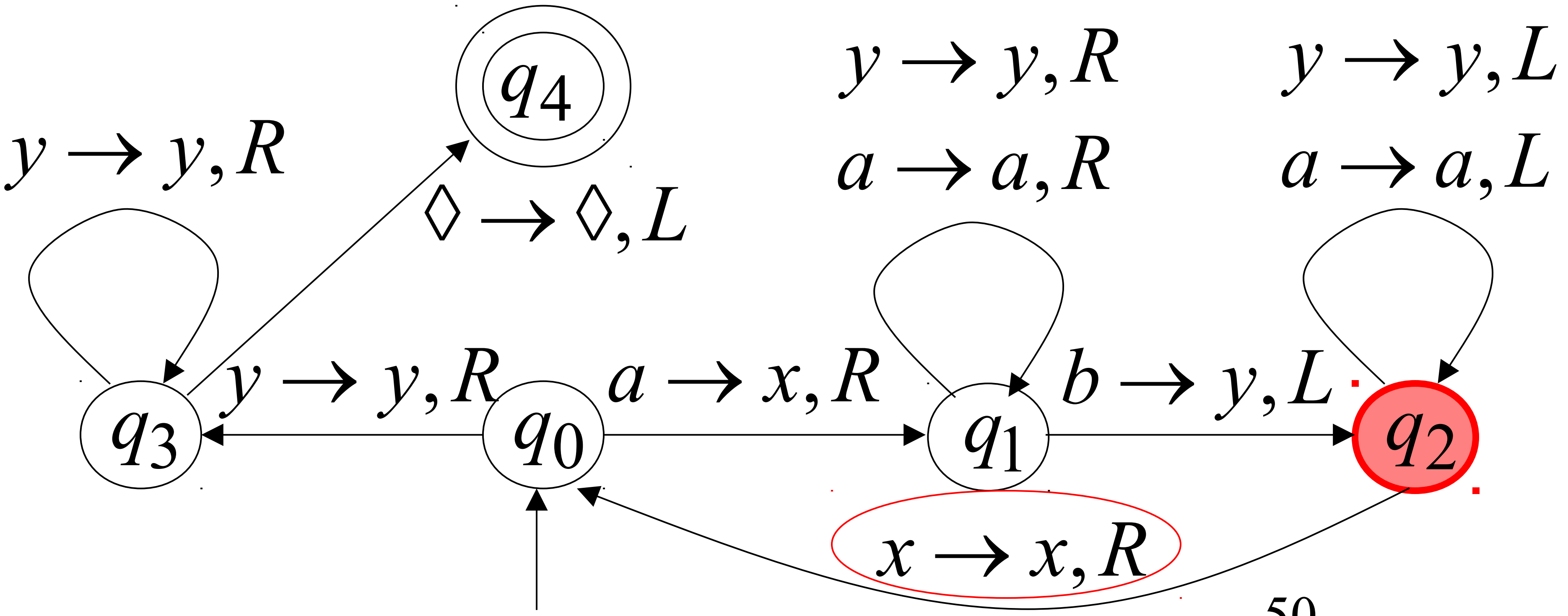

◊
x
a
y
b
◊
◊
q_2


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

Time 4

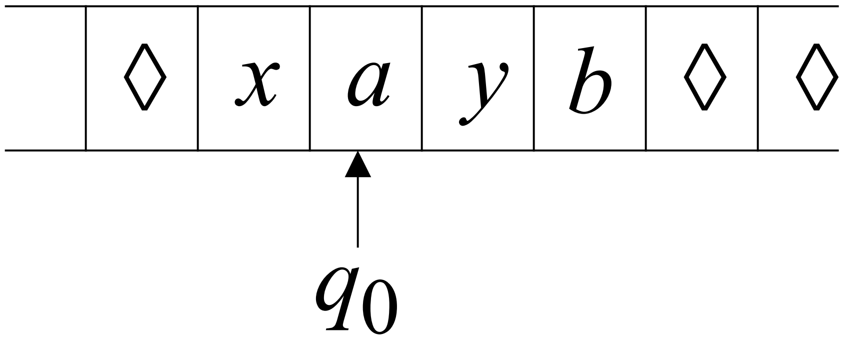


◊
x
a
y
b
◊
◊
q_2

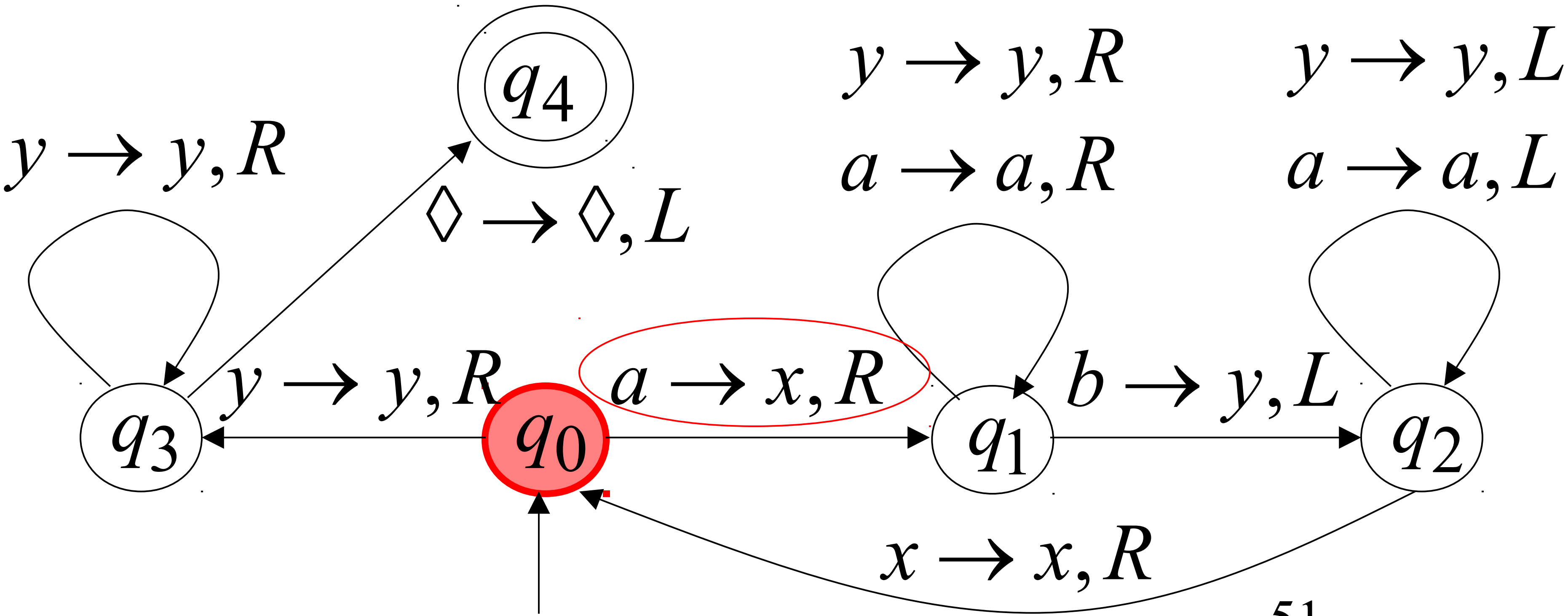


$y \to y, R$
$\lozenge \to \lozenge, L$
q_4
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

Time 5

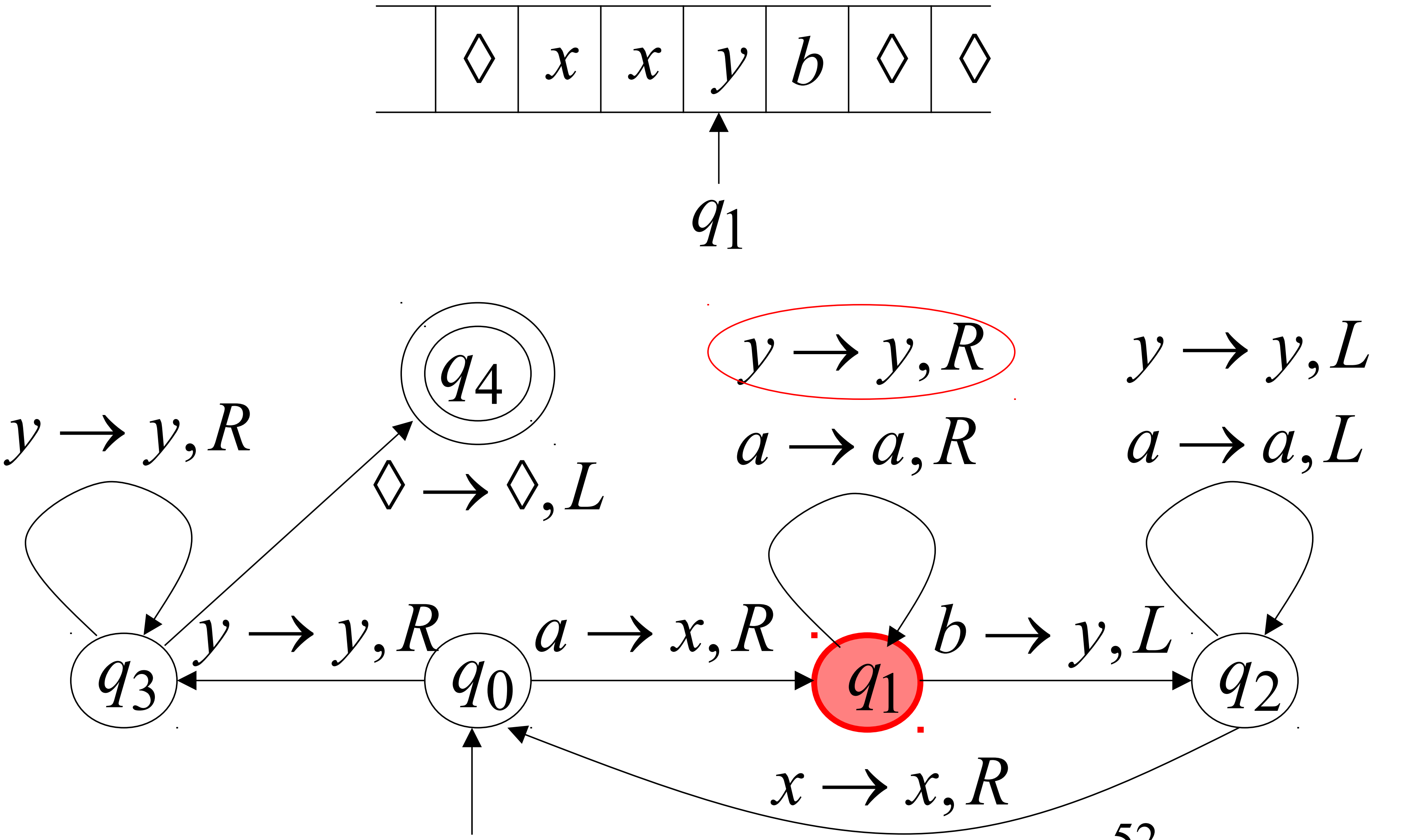


◊
x
a
y
b
◊
◊
q_0

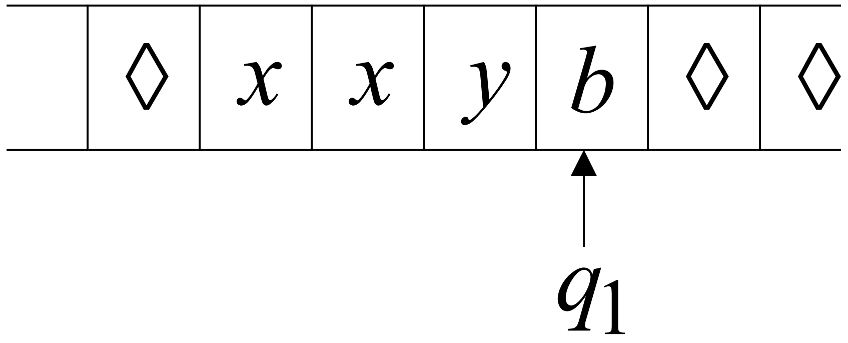


q_4
$y \rightarrow y, R$
$\lozenge \rightarrow \lozenge, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

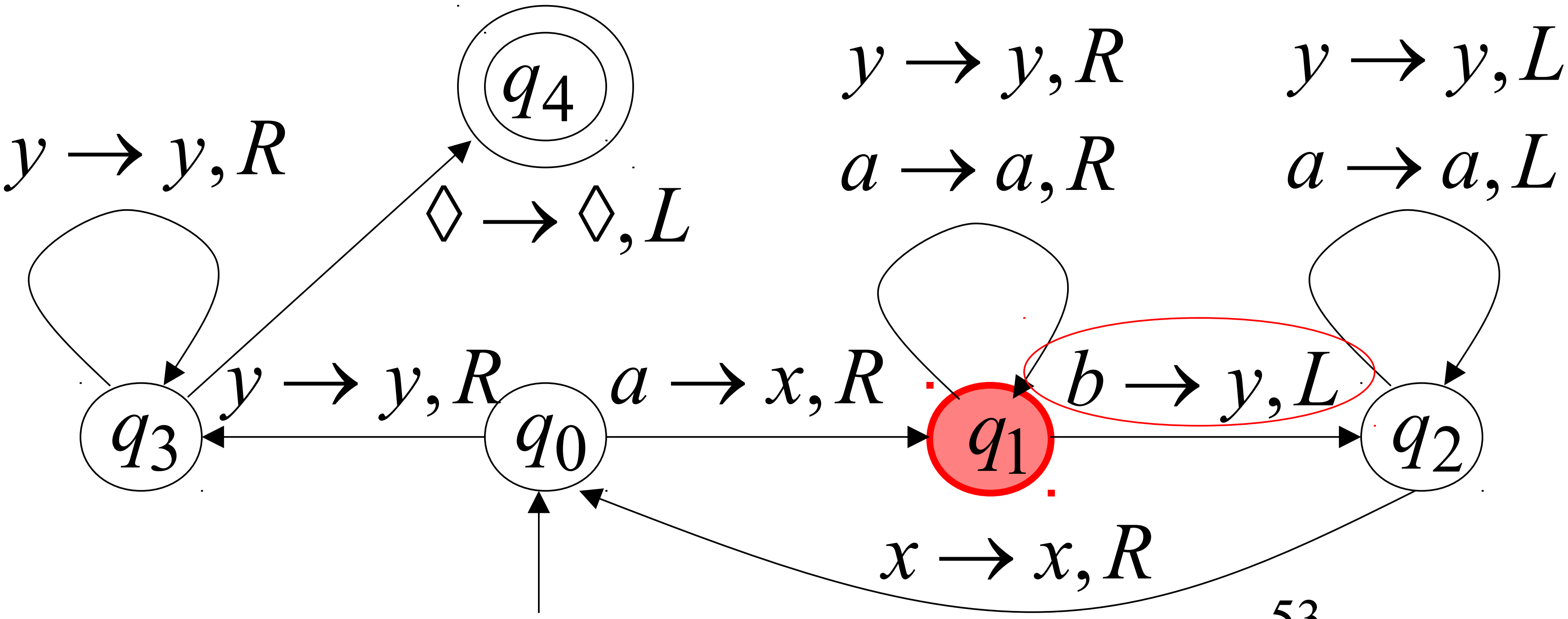
Time 6


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

Time 7

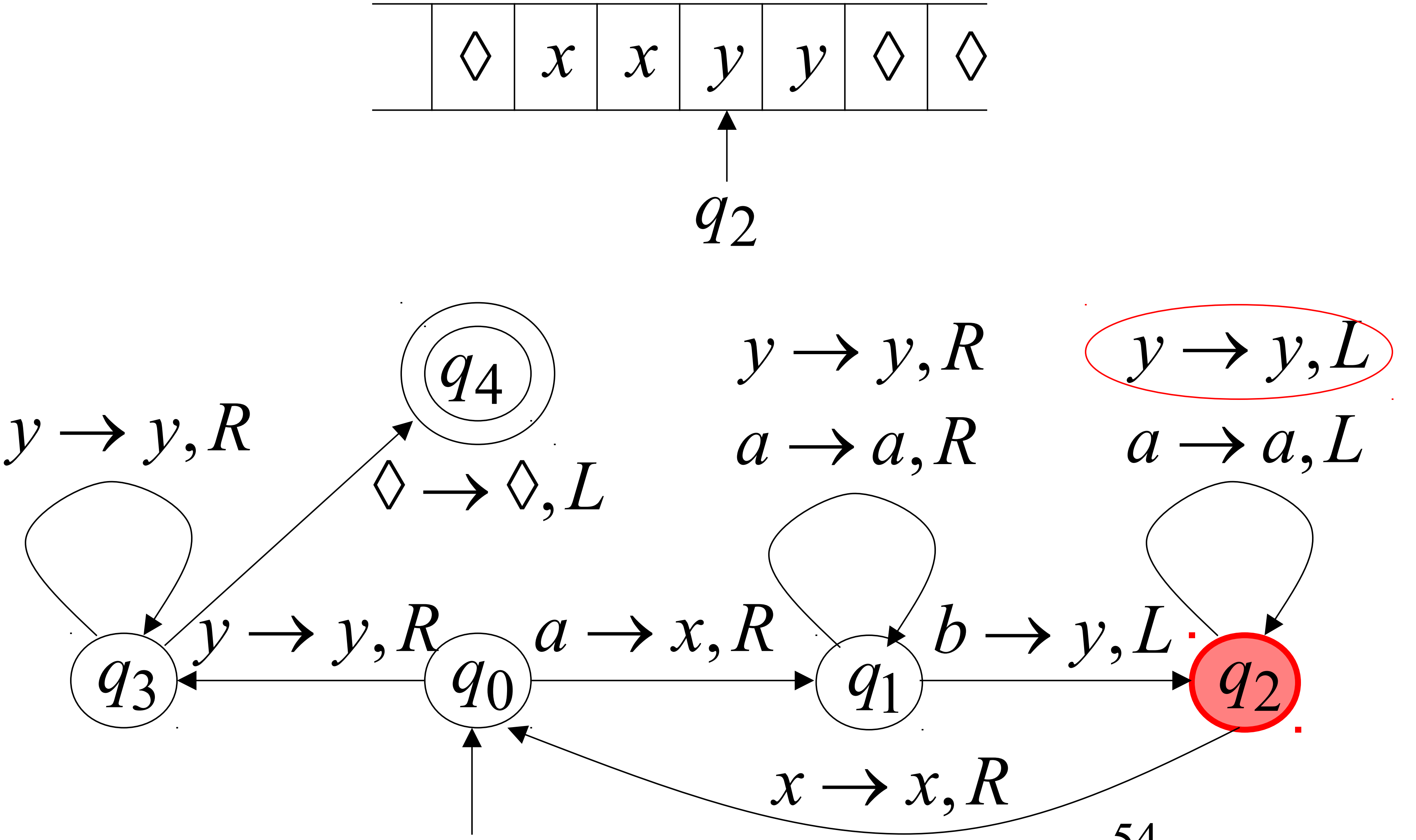


◊ x x y b ◊ ◊
q_1



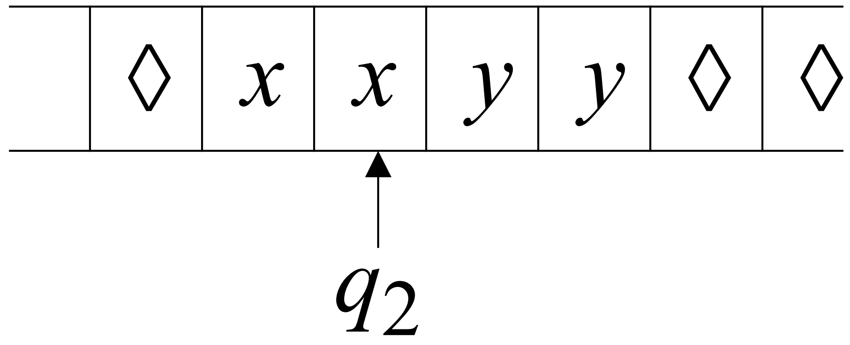
q_4
$y \to y, R$
$\lozenge \to \lozenge, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

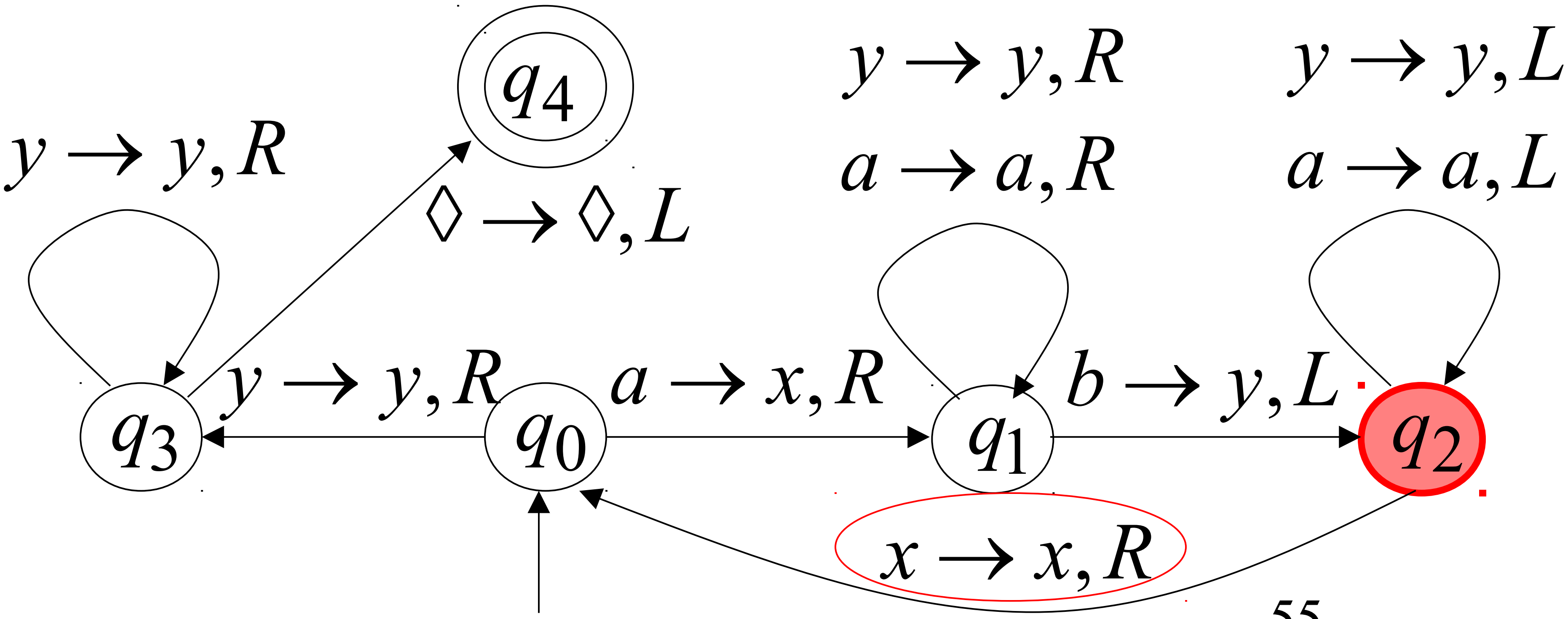
Time 8



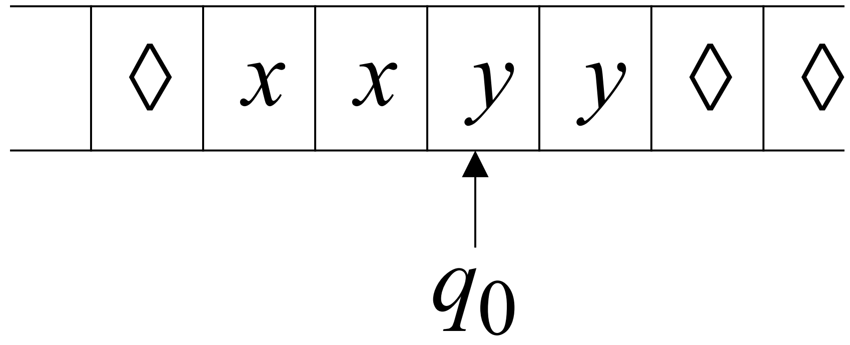
$\Diamond$
x
x
y
y
$\Diamond$
$\Diamond$
q_2
q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 9

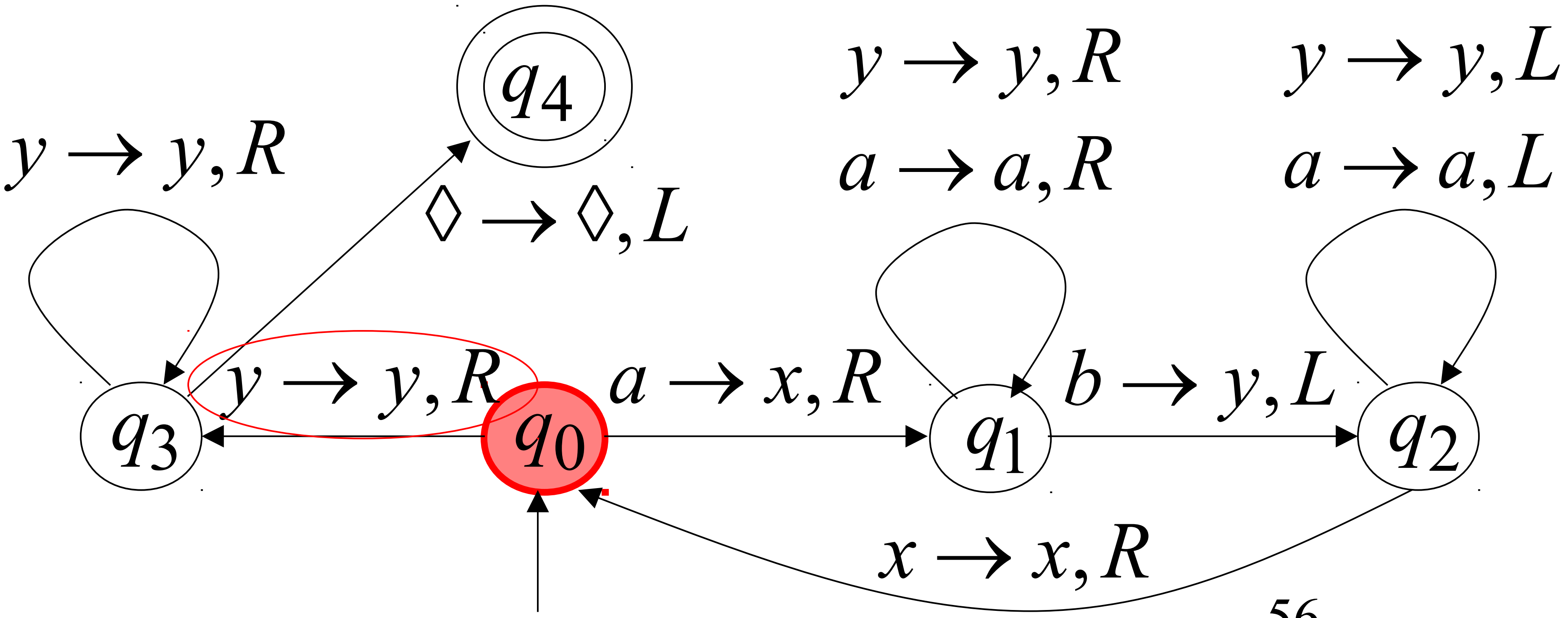

◊
x
x
y
y
◊
◊
q_2


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

Time 10

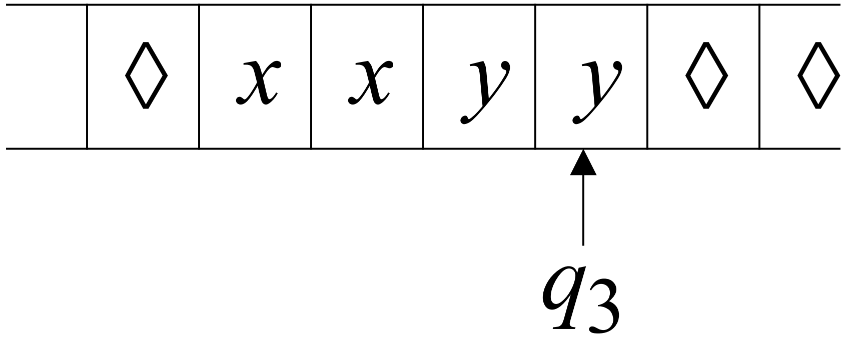


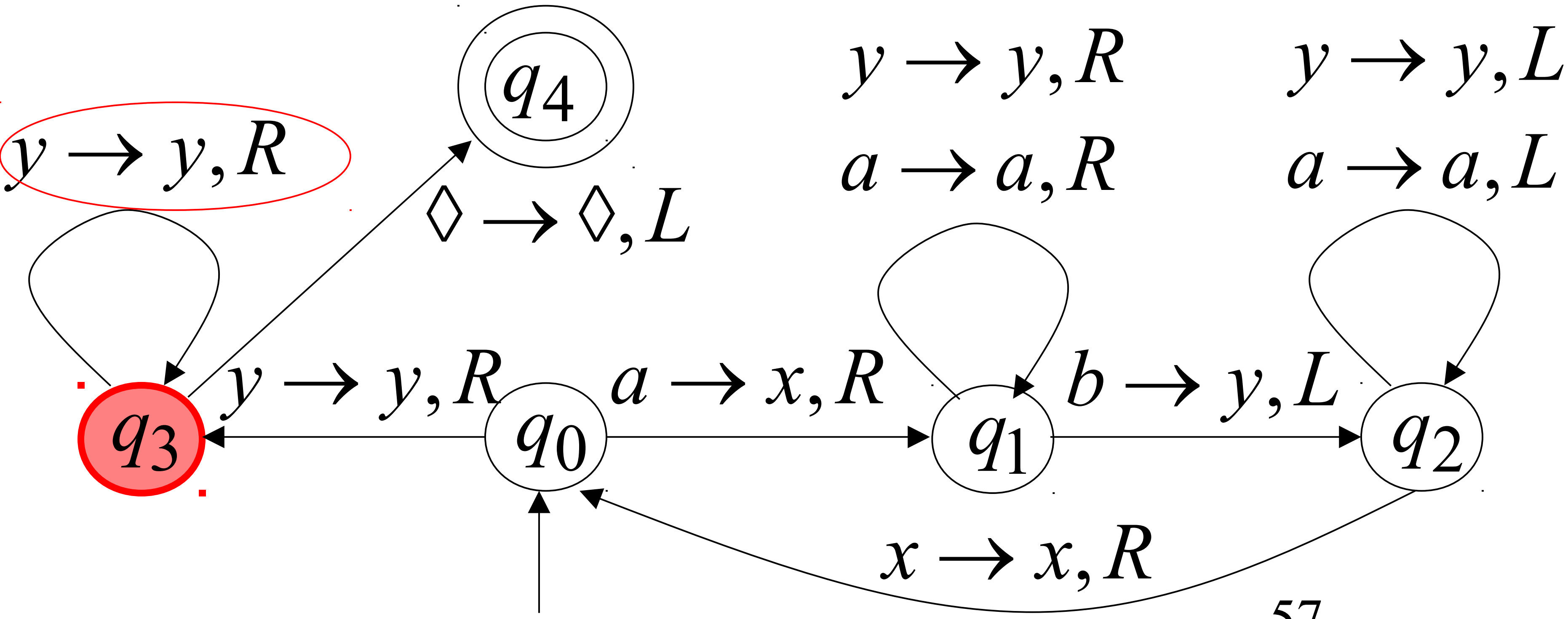
◊
x
x
y
y
◊
◊
q_0



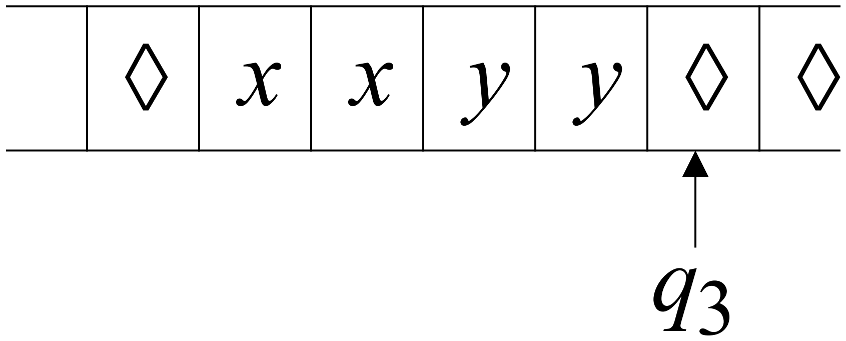
q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

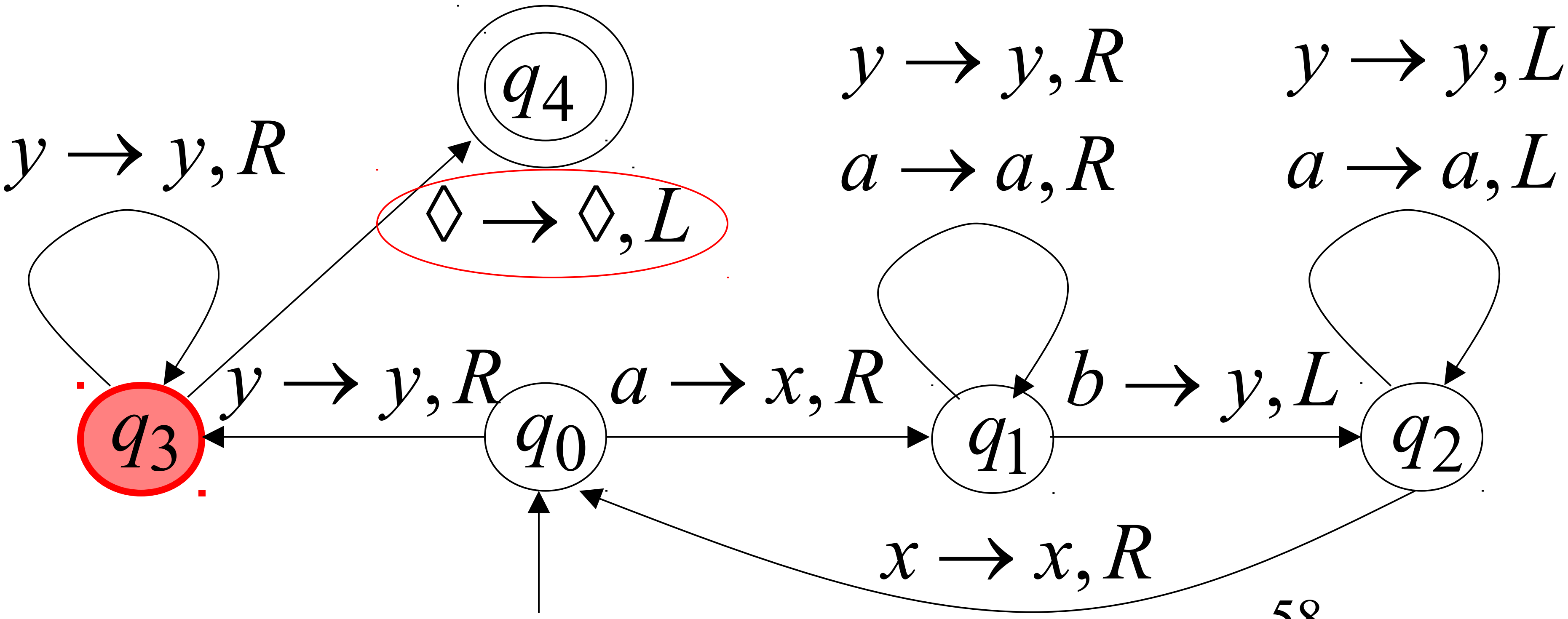
Time 11


◊
x
x
y
y
◊
◊
q_3

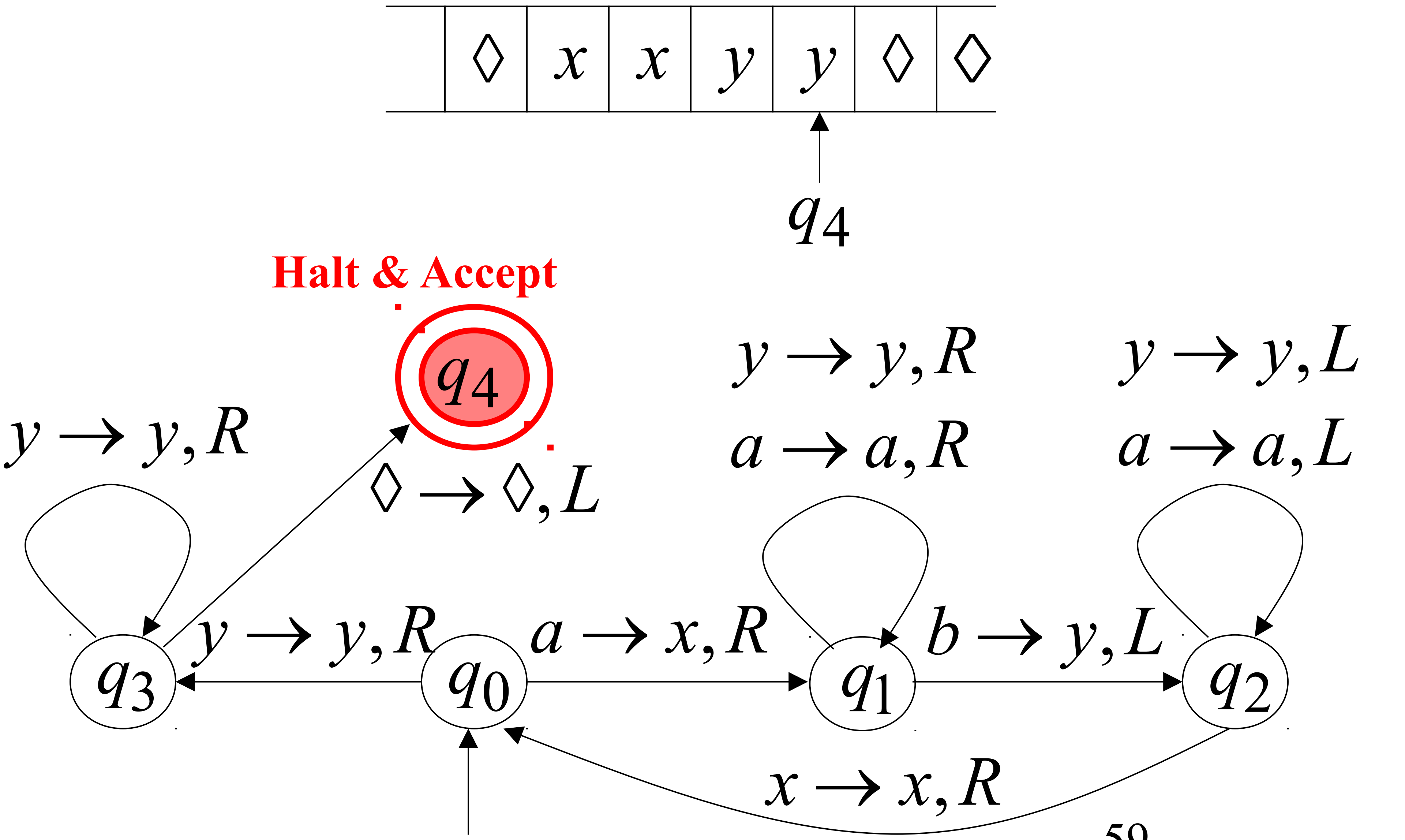

q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 12


◊
x
x
y
y
◊
◊
q_3


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

Time 13


◊ x x y y ◊ ◊
q_4
Halt & Accept
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
$x \rightarrow x, R$

Turing machine $M = (Q, \Gamma, \Sigma, \delta, s, B, F)$ with

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$,
- $\Gamma = \{a, b, X, Y, \#\}$,
- $\Sigma = \{a, b\}$,
- $s = q_0$,
- $B = \#$,
- δ given by

	a	b	X	Y	$\#$
q_0	(q_1, X, R)	$-$	$-$	(q_3, Y, R)	$-$
q_1	(q_1, a, R)	(q_2, Y, L)	$-$	(q_1, Y, R)	$-$
q_2	(q_2, a, L)	$-$	(q_0, X, R)	(q_2, Y, L)	$-$
q_3	$-$	$-$	$-$	(q_3, Y, R)	$(q_4, \#, R)$
q_4	$-$	$-$	$-$	$-$	$-$

M accepts $\{a^n b^n\}$, For example, its execution on aaabbb is

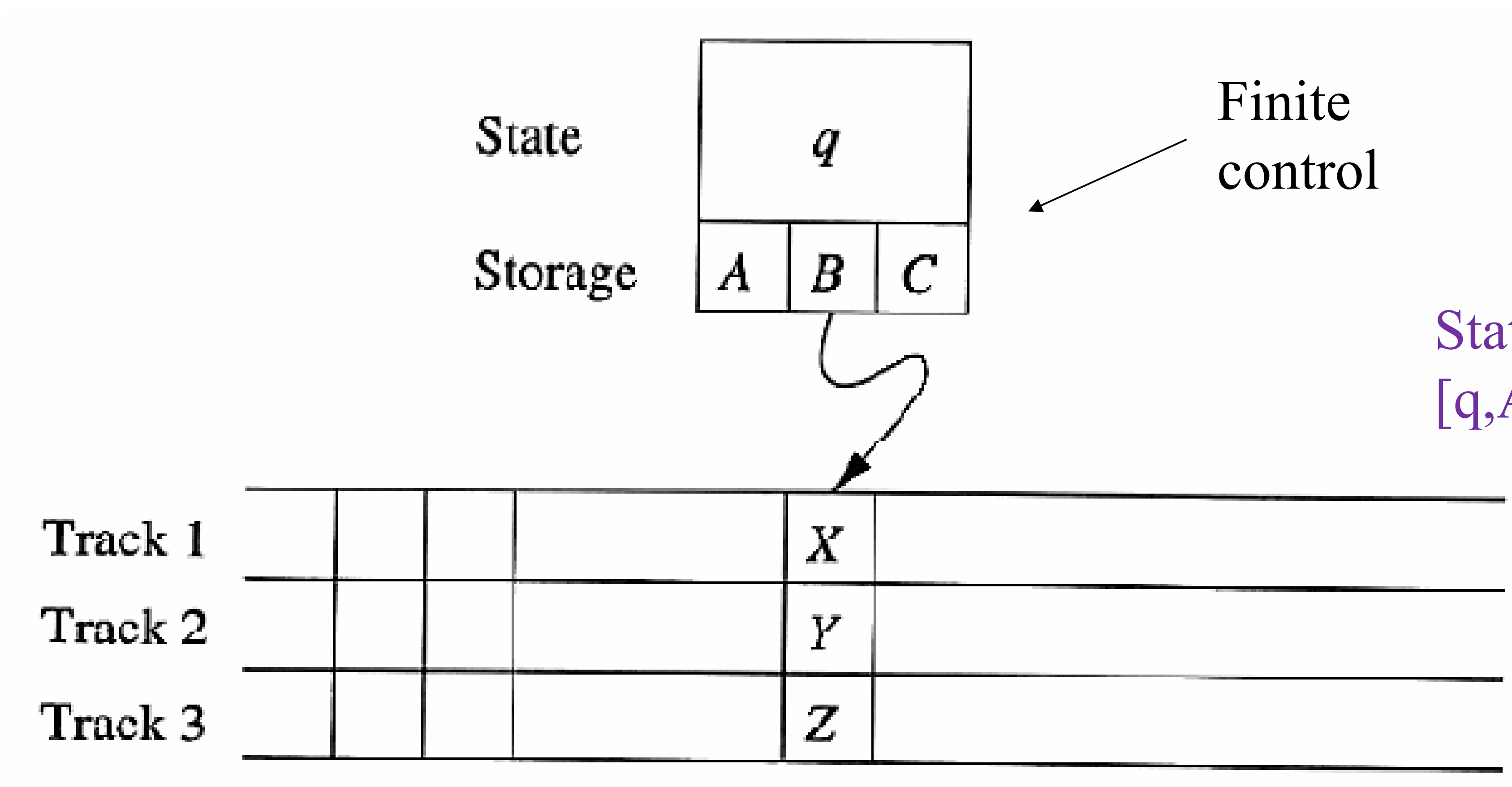
	$\vdots$
$(q_0, \varepsilon, aaabbb)$	$(q_1, XXXYY, b)$
$(q_1, X, aabbb)$	$(q_2, XXXY, YY)$
$(q_1, Xa, abbb)$	(q_2, XXX, YYY)
(q_1, Xaa, bbb)	$(q_2, XX, XYYY)$
$(q_2, Xa, aYbb)$	(q_0, XXX, YYY)
$(q_2, X, aaYbb)$	$(q_3, XXXY, YY)$
$(q_2, \varepsilon, XaaYbb)$	$(q_3, XXXYY, Y)$
$(q_0, X, aaYbb)$	$(q_3, XXXYYY, \varepsilon)$
$(q_1, XX, aYbb)$	$(q_4, XXXYYY\#, \varepsilon)$

Computation by Turing Machines

How can a TM be used to compute like a computer?

- **Programming Techniques or Alternate models of TM**
  - Storage in a state
  - TM with multiple tracks
  - Subroutines

Storage in a state

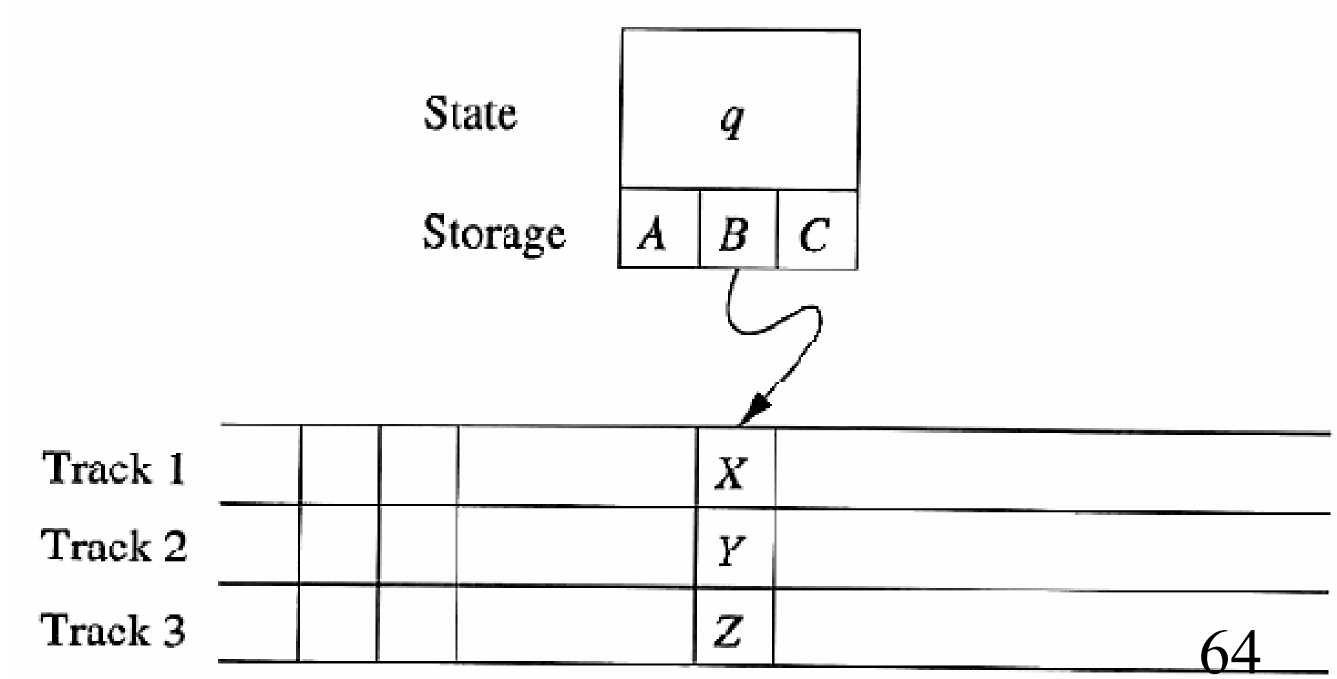


States as [q,A,B,C]

Fig: A TM viewed as having finite control storage and multiple tracks

TM with Multiple Tracks

- Tape composed of several TRACKS
- Each Track can hold one symbol
- Tape alphabet of TM consists of tuples, with one component for each "track"
- Cell scanned by the tape head contains the symbol [X,Y,Z]



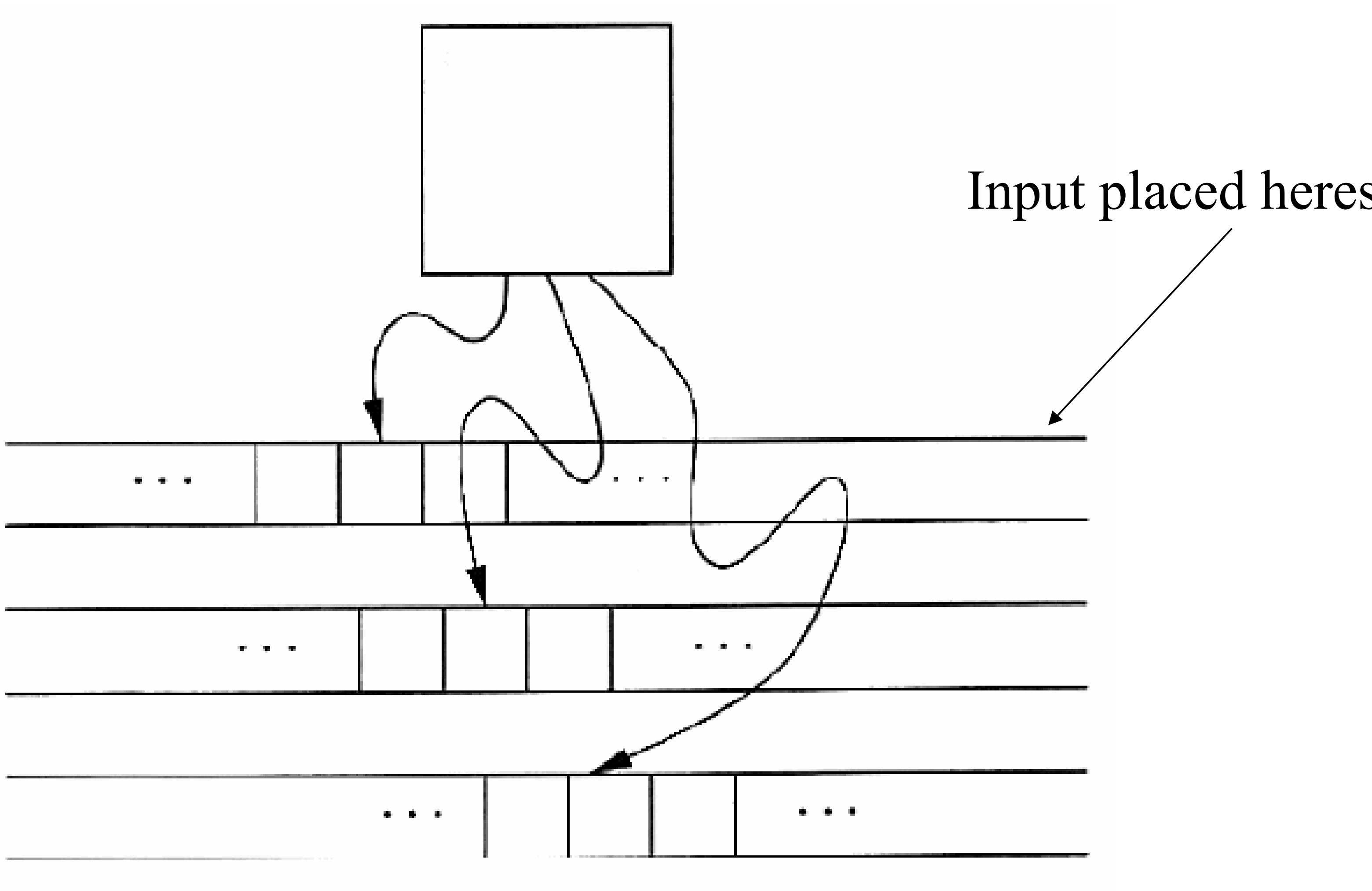
Subroutines

- Set of states that perform some useful process.
- Set of states include
    - Start state
    - Another state
        - Has no moves
        - Serves as the “return” state to pass control to whatever other set of states called the subroutine
- “Call” of a subroutine occurs whenever there is a transition to its initial state
- Use copies of subroutine

Variants of TM (Extentions to the basic TM)

- Multi-Tape Turing Machine
- Non-Deterministic Turing Machine

Multi-Tape Turing Machine



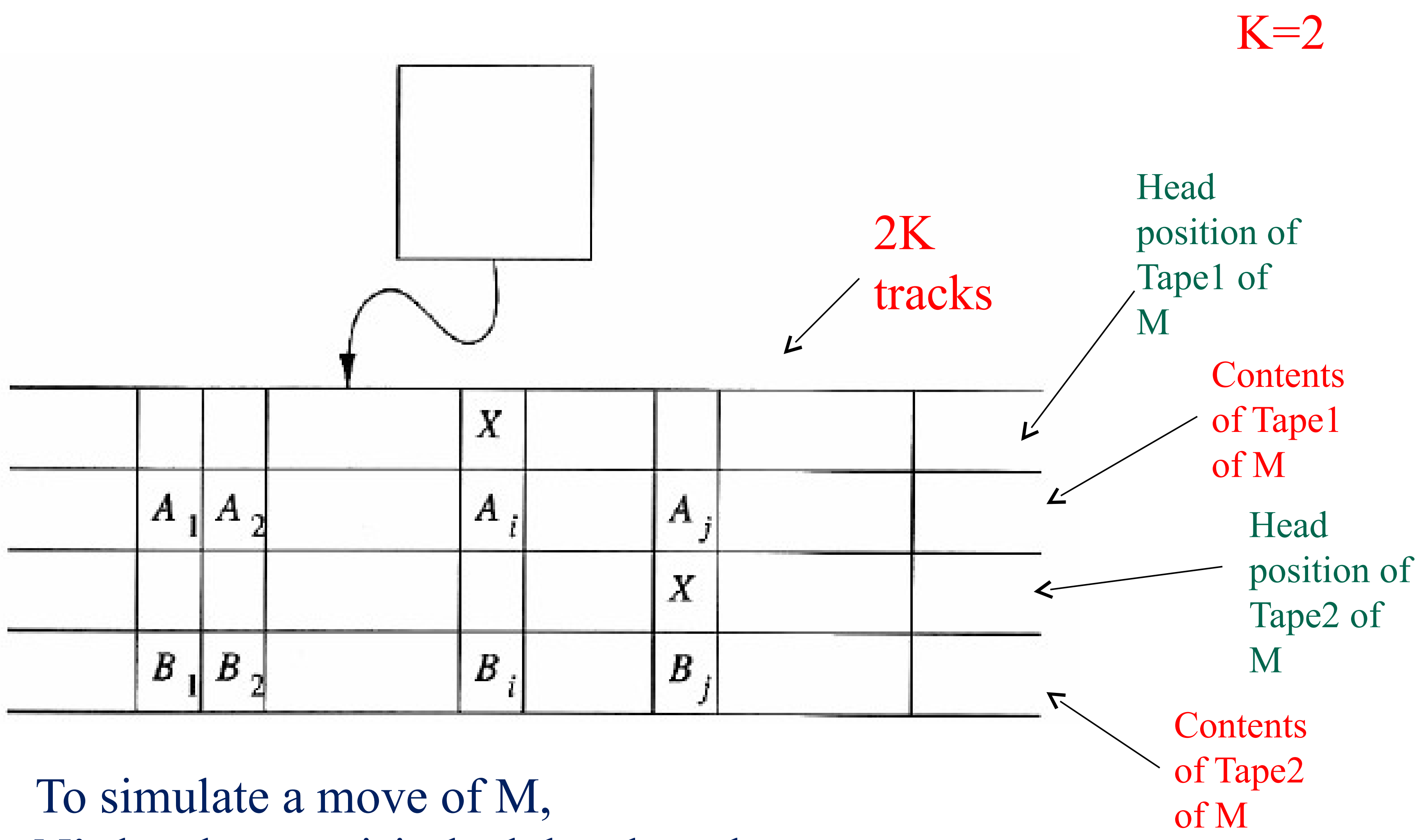
Move of Multi-Tape Turing Machine

- Depends on
  - State
  - Symbol scanned by each of the tape heads
- In one move, the multitape TM does the following:
  - Control enters a new state
  - A new tape symbol written on the cell scanned on each tape
  - Each tape head moves a move(left, right, or stationary(S))

Equivalence of One-Tape and MultiTape TMs

- One Tape TM accepts Recursively enumerable languages
- MultiTape TM doesnot accept languages that are not enumerable
- **Theorem: Every language accepted by a multitape TM is recursively enumerable**

Simulation of a 2 Tape TM(M) by a one-tape TM(N)



To simulate a move of M,
N’s head must visit the k head markers

Non-Deterministic TM

- Differs from deterministic by transition function δ
- For each state q and tape symbol X, $\delta(q,X)$ is a set of triples
  $\{(q_{1,}Y_{1,}D_1),(q_{2,}Y_{2,}D_2),\ldots\ldots,(q_{k,}Y_{k,}D_k)\}$

Where k=any finite integer

-NTM can choose at each step any of triples to be the next move

-Cannot pick a state from one, a tape symbol from another and direction from yet another

NTM and DTM equivalence($L(M_N)=L(M_D)$))

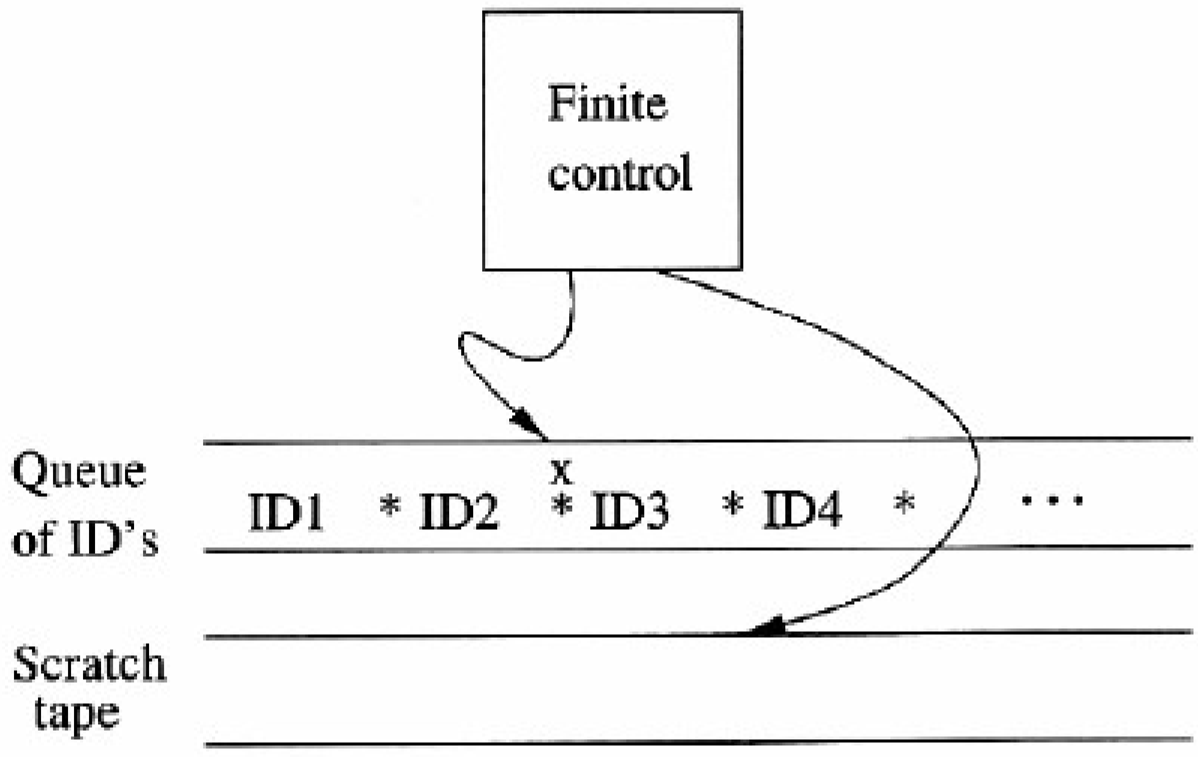


Figure 8.18: Simulation of an NTM by a DTM

1. If state in current ID is accepting, M_D accepts and simulates M_N no further.
2. Else, M_D uses 2nd tape to copy the ID and then make k moves of that ID at the end of the seq. of ID's on Tape1
3. M_D modifies each of those k ID's a/c to a different k choices of move that M_N has from its current ID
4. M_D returns to marked, current ID , erases the mark and moves the mark to next ID to the right.
5. Repeat with step1

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

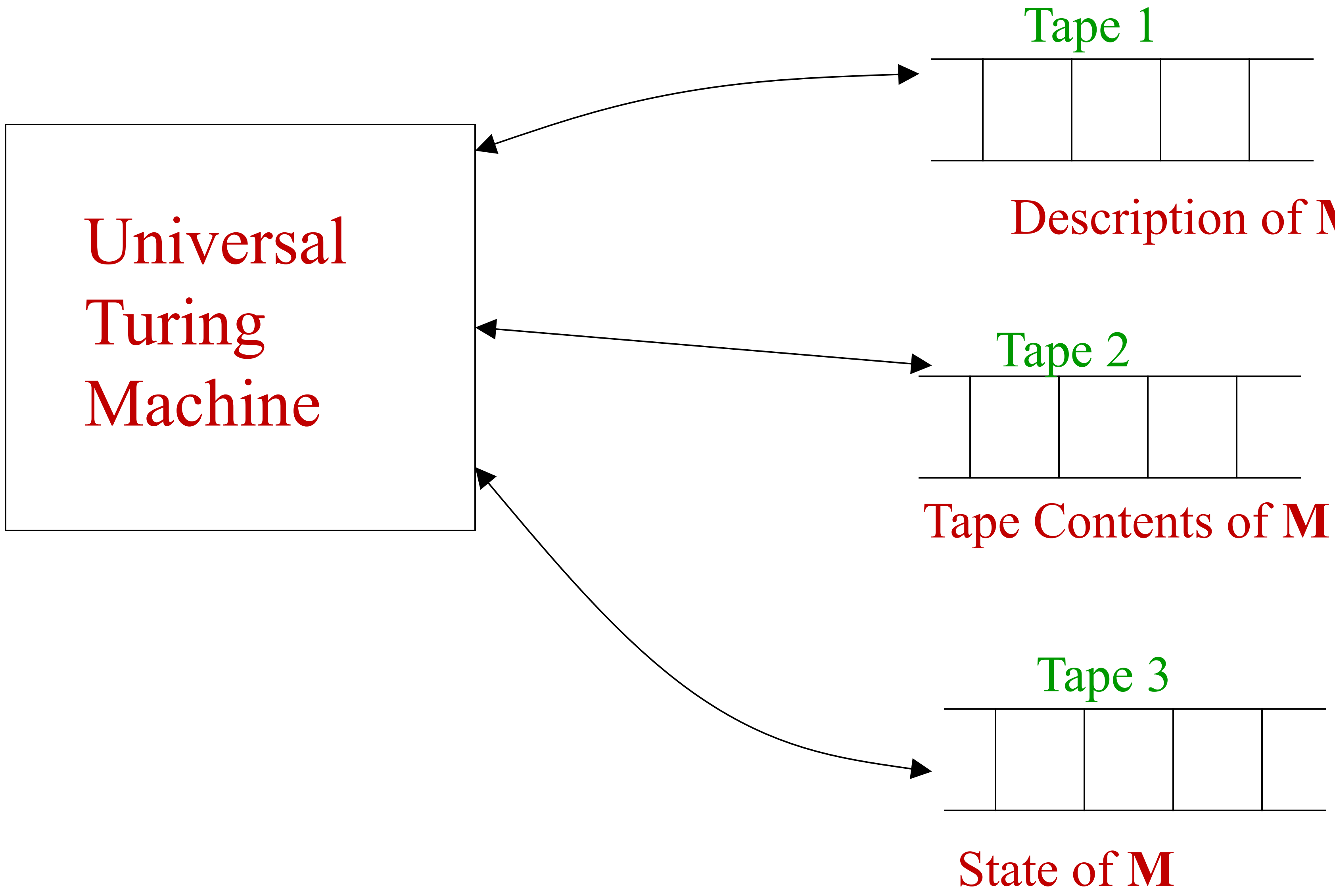
Universal Turing Machine

simulates any Turing Machine M

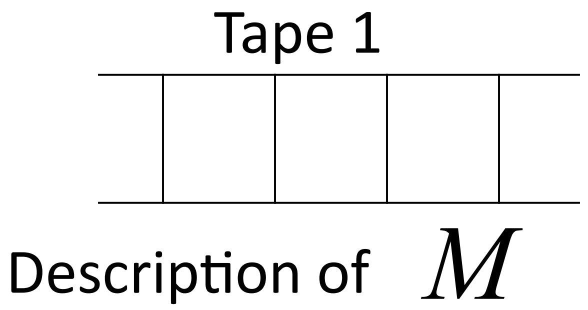
<u>Input of Universal Turing Machine:</u>

1. Description of transitions of M
2. Input string of M

Universal Turing Machines



Universal Turing Machines



We describe Turing machine M as a string of symbols:

We encode M as a string of symbols

Universal Turing Machines: Alphabet Encoding

$$\begin{tikzcd}
\text{Symbols:} & a \arrow[d] & b \arrow[d] & c \arrow[d] & d \arrow[d] & \cdots \\
\text{Encoding:} & 1 & 11 & 111 & 1111 &
\end{tikzcd}$$

Universal TM: State Encoding

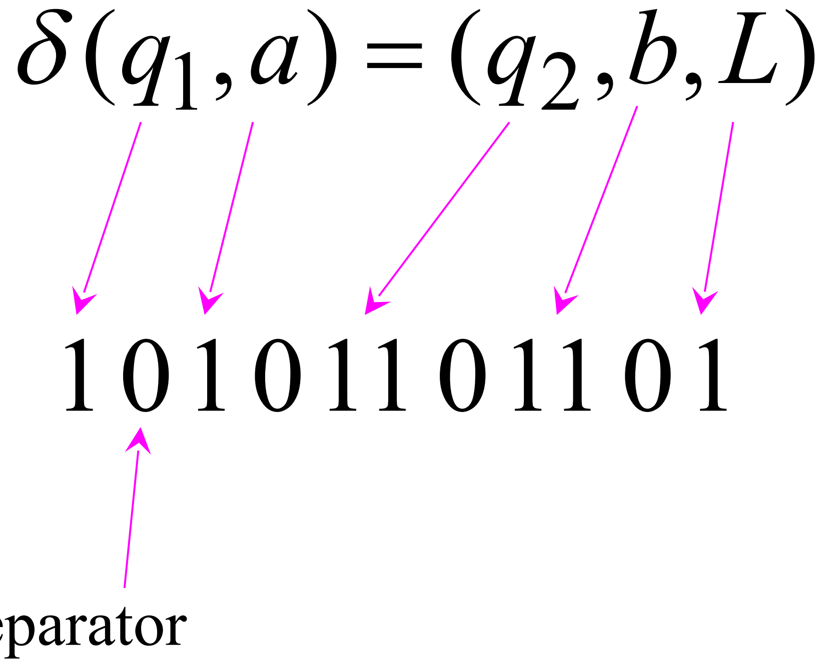
$$\begin{tikzcd}
\textbf{States:} & q_1 \arrow[d] & q_2 \arrow[d] & q_3 \arrow[d] & q_4 \arrow[d] \\
\textbf{Encoding:} & 1 & 11 & 111 & 1111
\end{tikzcd}$$

Universal Turing Machines

Head Move Encoding

$$\begin{tikzcd}
\textbf{Move:} & L \arrow[d] & R \arrow[d] \\
\textbf{Encoding:} & 1 & 11
\end{tikzcd}$$

Universal TM: Transition Encoding



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

Encoding: 1 0 1 0 11 0 11 0 1

separator

Turing Machine Encoding

Transitions:

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

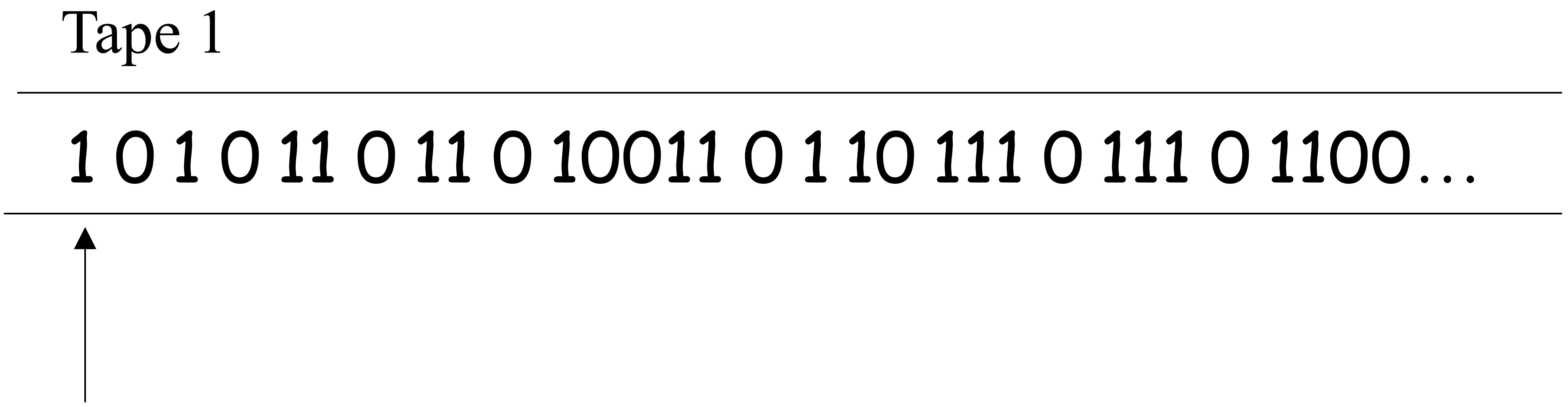
Encoding:

$$1\,0\,1\,0\,11\,0\,11\,0\,1 \quad \underset{\uparrow}{00} \quad 11\,0\,11\,0\,111\,0\,111\,0\,11$$

separator

Tape 1 contents of Universal Turing Machine:

binary encoding
of the simulated machine M



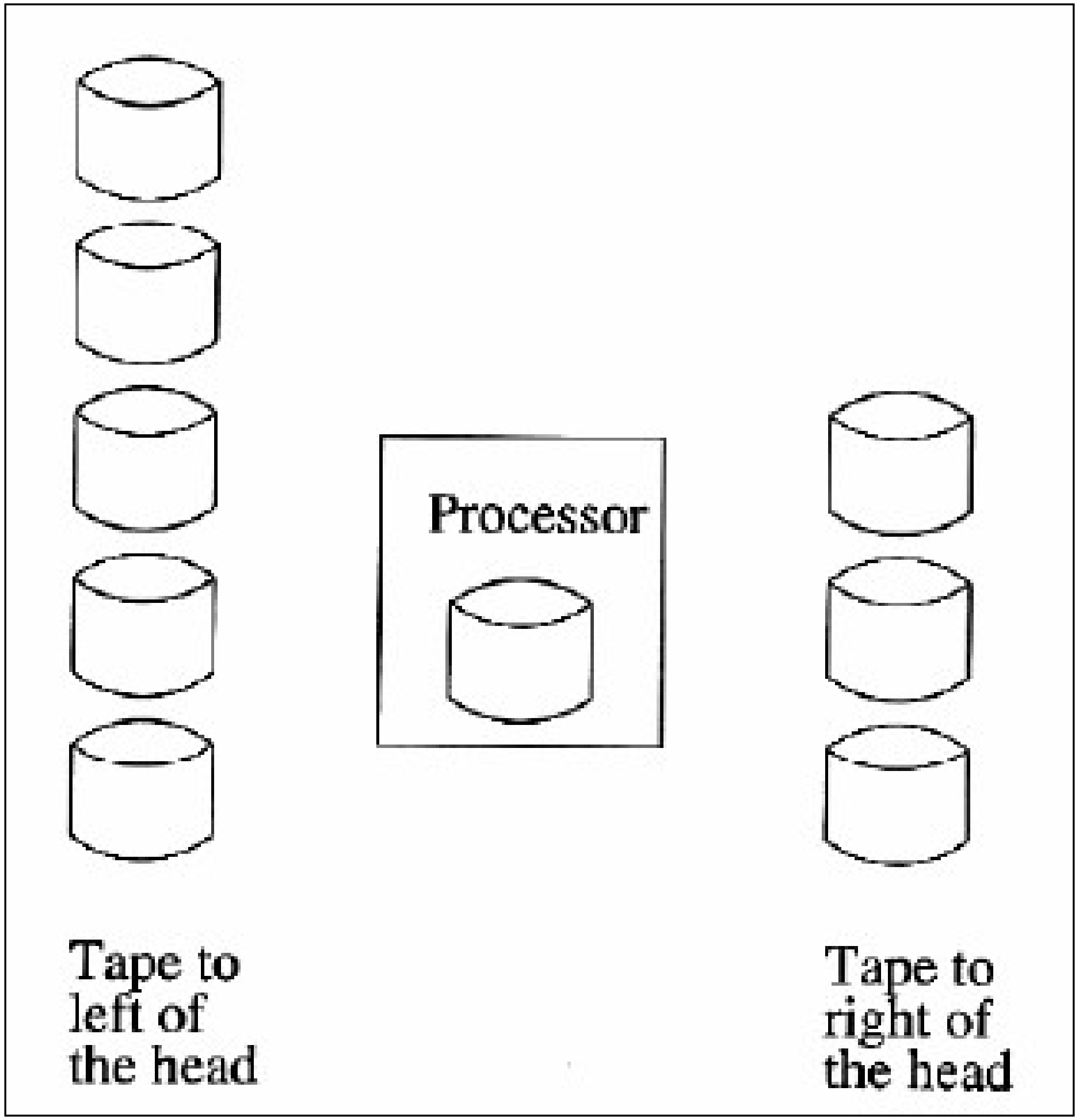
THE CHURCH-TURING THESIS

Anything that can be computed by algorithm can be computed by a Turing Machine.

Turing Machines and computers

- **Simulating a TM by a computer**
- **Simulating a computer by a TM**

Simulating a TM by a common computer



Can we simulate infinite tape with finite memory?

Assume as many disks as computer needs is available
--Swappable storage-

-The further down the stacks,
the further away from the tape head the data is

Swap left
Swap right

Simulating a TM by a common computer

Swap left

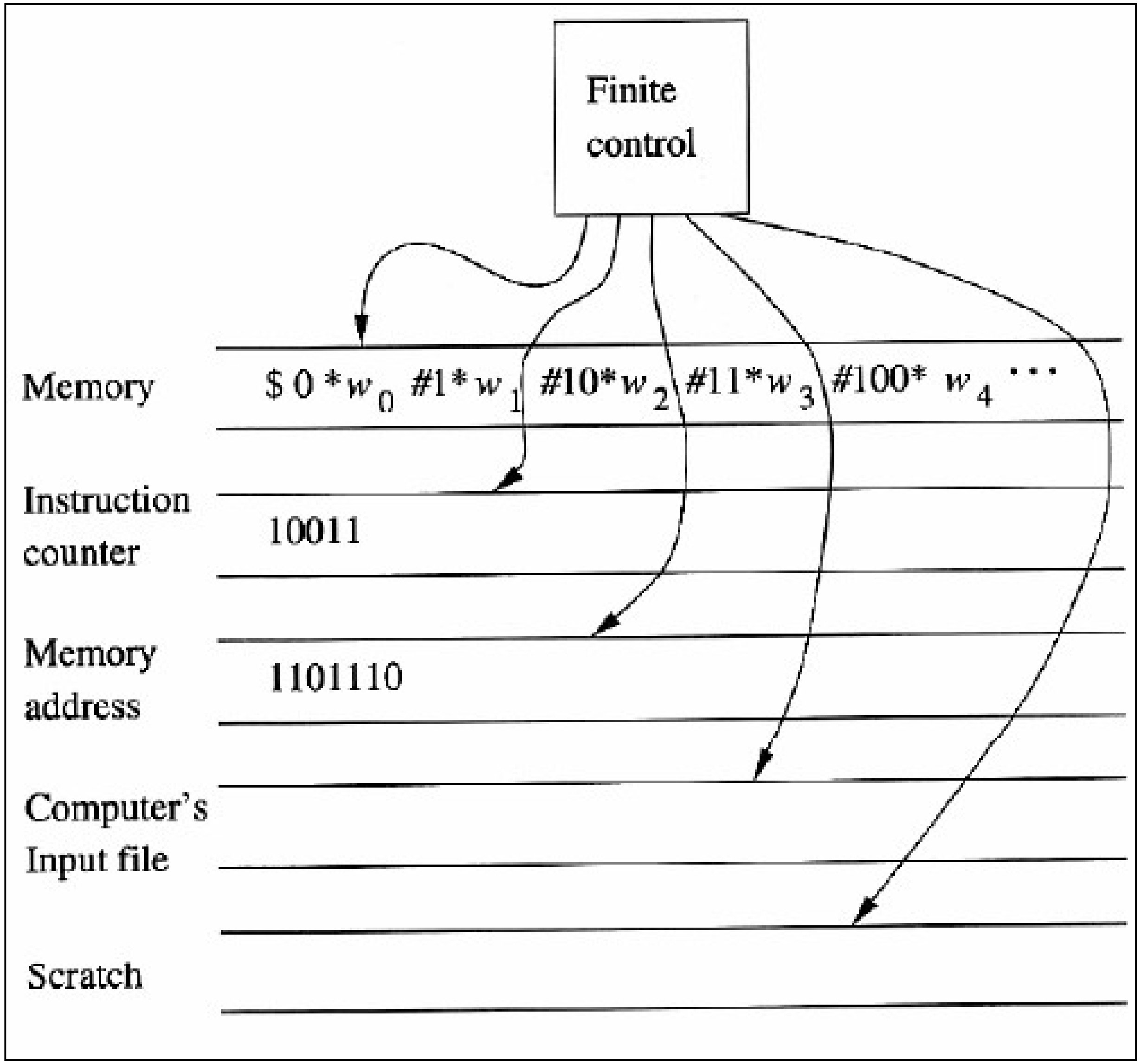
If the Tape head of TM moves sufficiently far to the left that it reaches cells that are not represented by the disk currently mounted in computer, then it prints “swap left”

-The currently mounted disk is removed and placed on the top of the right stack

- disk on top of left stack is mounted in the computer and computation resumes

Swap right

Simulating a Computer by a TM



* and # represent end of address and contents of memory words(binary string)
$ = begin of seq of address o& contents

Simulating a Computer by a TM

- First Tape== memory
- Second Tape= instruction counter
  - Holds one integer in binary, represent one of the memory locations on Tape1
  - Value stored = next computer instruction to be executed
- Third Tape
  - memory address or
  - Contents of that address after the address has been located on tape 1.
  - To execute an instruction, TM must find the contents of one or more memory address that hold data involved in the computation
  - Desired address is copied to tape 3 and compared with the address on tape 1 , until a match is found.

Simulating a Computer by a TM

- Fourth Tape
    - Simulated input to the computer
- last tape
    - Scratch tape
    - Compute arithmetic operations such as multiplication

Please check

Properties of Recursive language