COM3064
Automata Theory

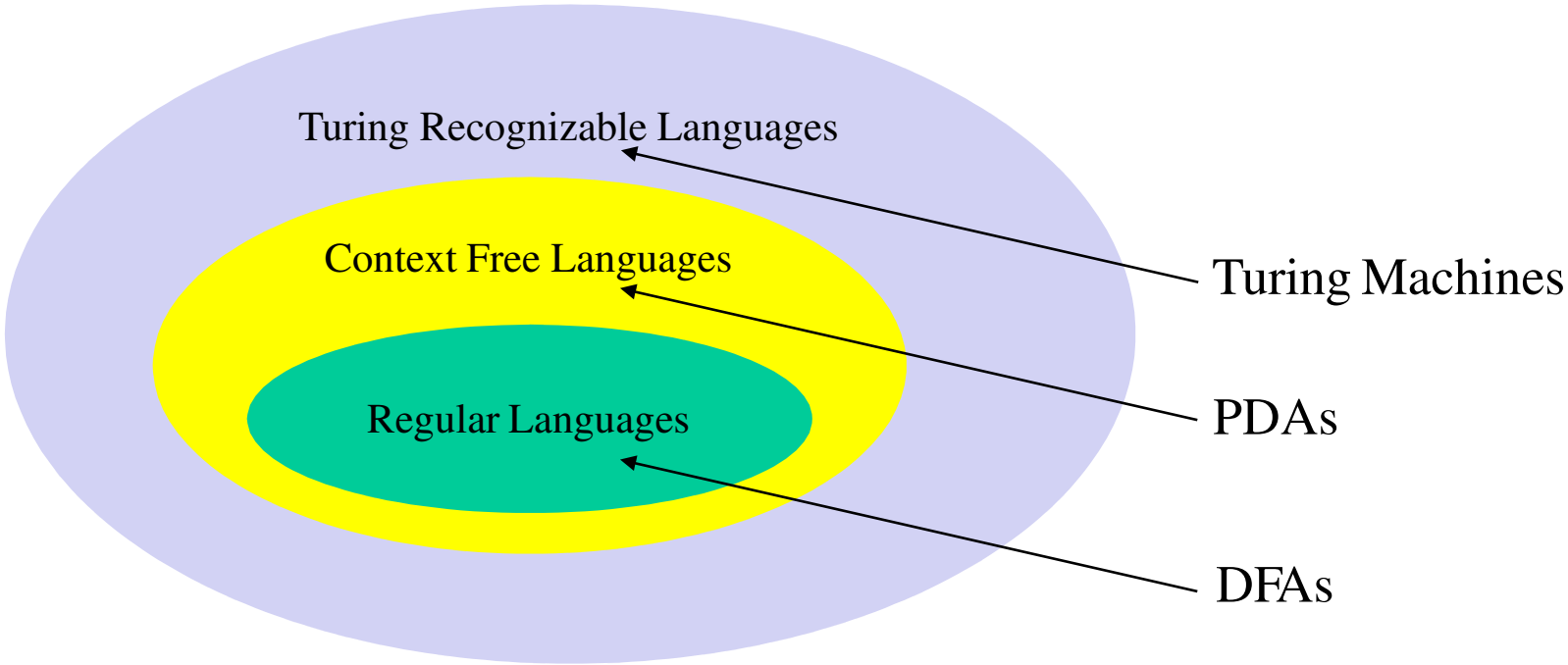
Week 10: Turing Machines

Lecturer: Dr. Sevgi YİĞİT SERT
Spring 2023

Resources: Introduction to The Theory of Computation, M. Sipser,
Introduction to Automata Theory, Languages, and Computation, J.E. Hopcroft, R. Motwani, and J.D. Ullman
BBM401 Automata Theory and Formal Languages, İlyas Çiçekli
CENG280 Formal Languages and Abstract Machines, Halit Oğuztüzün

Turing Machines

- DFAs recognize regular languages.
- PDAs recognize CFLs.
- There are languages that are NOT CFLs.



Turing Machines

Regular Language, CFL, Non-CFL

- $\{ 0^n 1^m \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n 1^n \mid n \geq 0 \}$
- $\{ 0^n 1^m 0^n \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n 1^n 0^n \mid n \geq 0 \}$
- $\{ 0^n 1^{2n} \mid n \geq 0 \}$
- $\{ 0^n 1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n 1^n 2^n \mid n \geq 0 \}$
- $\{ 0^i 1^j 2^k \mid k \geq j \geq i \geq 0 \}$
- $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$

Turing Machines

Regular Language, CFL, Non-CFL

- $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$ **regular language**
- $\{ 0^n1^n \mid n \geq 0 \}$
- $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n1^n0^n \mid n \geq 0 \}$
- $\{ 0^n1^{2n} \mid n \geq 0 \}$
- $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n1^n2^n \mid n \geq 0 \}$
- $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$
- $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$

Turing Machines

Regular Language, CFL, Non-CFL

- $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$ **regular language**
- $\{ 0^n1^n \mid n \geq 0 \}$ **CFL**
- $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n1^n0^n \mid n \geq 0 \}$
- $\{ 0^n1^{2n} \mid n \geq 0 \}$
- $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n1^n2^n \mid n \geq 0 \}$
- $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$
- $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$

Turing Machines

Regular Language, CFL, Non-CFL

- $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$ **regular language**
- $\{ 0^n1^n \mid n \geq 0 \}$ **CFL**
- $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$ **CFL**
- $\{ 0^n1^n0^n \mid n \geq 0 \}$
- $\{ 0^n1^{2n} \mid n \geq 0 \}$
- $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n1^n2^n \mid n \geq 0 \}$
- $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$
- $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$

Turing Machines

Regular Language, CFL, Non-CFL

- $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$ **regular language**
- $\{ 0^n1^n \mid n \geq 0 \}$ **CFL**
- $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$ **CFL**
- $\{ 0^n1^n0^n \mid n \geq 0 \}$ **non CFL**
- $\{ 0^n1^{2n} \mid n \geq 0 \}$
- $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n1^n2^n \mid n \geq 0 \}$
- $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$
- $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$

Turing Machines

Regular Language, CFL, Non-CFL

- $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$ **regular language**
- $\{ 0^n1^n \mid n \geq 0 \}$ **CFL**
- $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$ **CFL**
- $\{ 0^n1^n0^n \mid n \geq 0 \}$ **non CFL**
- $\{ 0^n1^{2n} \mid n \geq 0 \}$ **CFL**
- $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$
- $\{ 0^n1^n2^n \mid n \geq 0 \}$
- $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$
- $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$

Turing Machines

Regular Language, CFL, Non-CFL

• $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$	**regular language**
• $\{ 0^n1^n \mid n \geq 0 \}$	**CFL**
• $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$	**CFL**
• $\{ 0^n1^n0^n \mid n \geq 0 \}$	**non CFL**
• $\{ 0^n1^{2n} \mid n \geq 0 \}$	**CFL**
• $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$	**regular language**
• $\{ 0^n1^n2^n \mid n \geq 0 \}$	
• $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$	
• $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$	

Turing Machines

Regular Language, CFL, Non-CFL

• $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$	**regular language**
• $\{ 0^n1^n \mid n \geq 0 \}$	**CFL**
• $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$	**CFL**
• $\{ 0^n1^n0^n \mid n \geq 0 \}$	**non CFL**
• $\{ 0^n1^{2n} \mid n \geq 0 \}$	**CFL**
• $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$	**regular language**
• $\{ 0^n1^n2^n \mid n \geq 0 \}$	**non CFL**
• $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$	
• $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$	

Turing Machines

Regular Language, CFL, Non-CFL

• $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$	**regular language**
• $\{ 0^n1^n \mid n \geq 0 \}$	**CFL**
• $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$	**CFL**
• $\{ 0^n1^n0^n \mid n \geq 0 \}$	**non CFL**
• $\{ 0^n1^{2n} \mid n \geq 0 \}$	**CFL**
• $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$	**regular language**
• $\{ 0^n1^n2^n \mid n \geq 0 \}$	**non CFL**
• $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$	**non CFL**
• $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$	

Turing Machines

Regular Language, CFL, Non-CFL

- $\{ 0^n1^m \mid n \geq 0 \text{ and } m \geq 0 \}$ **regular language**
- $\{ 0^n1^n \mid n \geq 0 \}$ **CFL**
- $\{ 0^n1^m0^n \mid n \geq 0 \text{ and } m \geq 0 \}$ **CFL**
- $\{ 0^n1^n0^n \mid n \geq 0 \}$ **non CFL**
- $\{ 0^n1^{2n} \mid n \geq 0 \}$ **CFL**
- $\{ 0^n1^{2m} \mid n \geq 0 \text{ and } m \geq 0 \}$ **regular language**
- $\{ 0^n1^n2^n \mid n \geq 0 \}$ **non CFL**
- $\{ 0^i1^j2^k \mid k \geq j \geq i \geq 0 \}$ **non CFL**
- $\{ w\#w \mid w \text{ is in } \{0,1\}^* \}$ **non CFL**

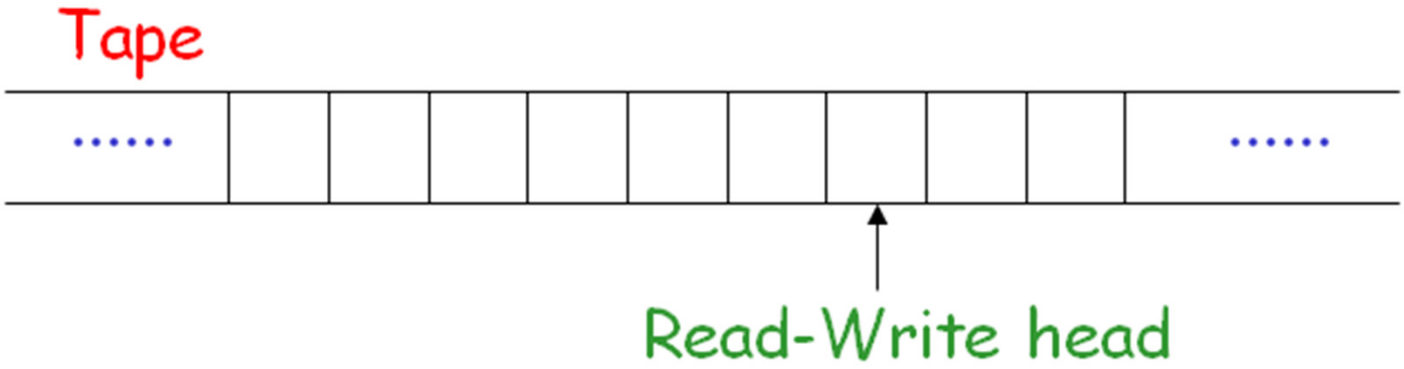
Turing Machines

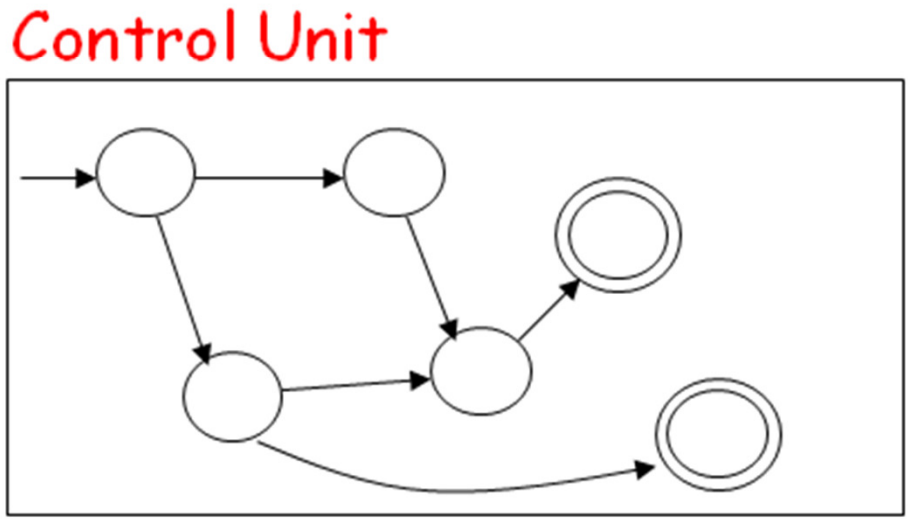
- There is a much more powerful model of computation called **Turing Machines (TM).**
  - It is first proposed by Alan Turing in 1936

Alan Turing
1912-1954

- TMs are similar to finite automata, but a TM has an **unlimited memory**.
- A Turing machine is a much more accurate model of a general purpose computer.
- **A Turing machine can do everything that a real computer can do**.
- **Even a TM can not solve certain problems.**
  - Such problems are beyond theoretical limits of computation.

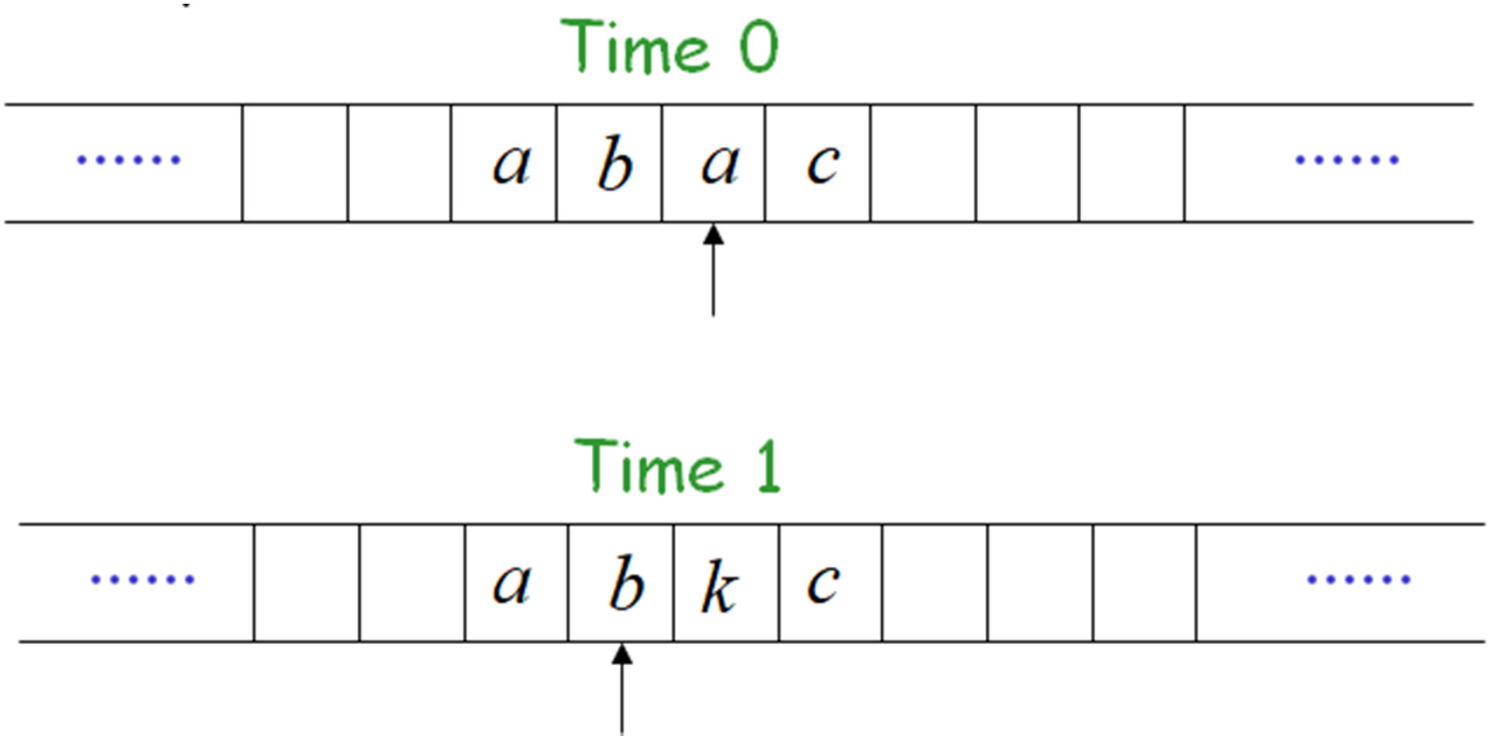
Turing Machines





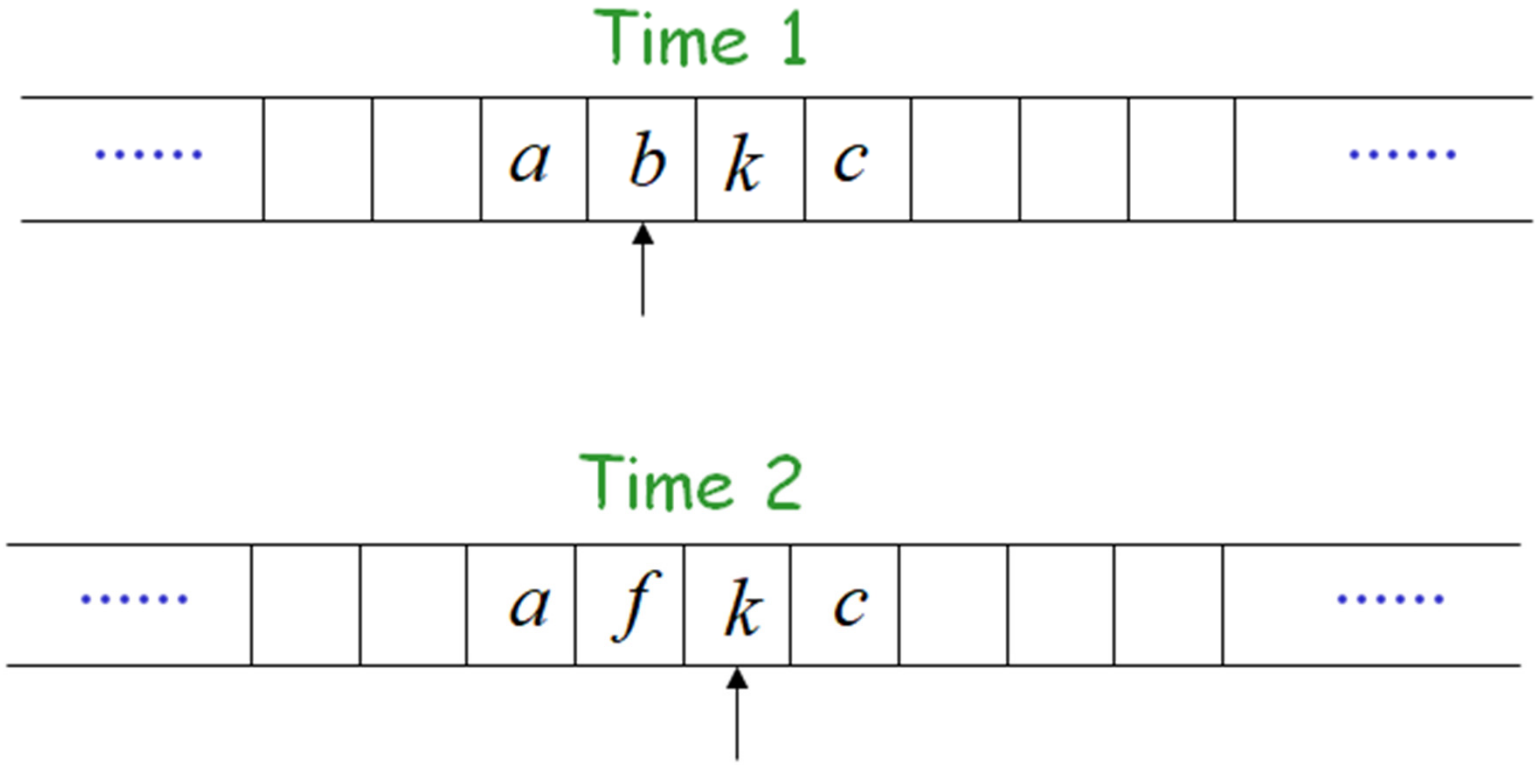
- The Turing machine model uses a **tape with infinite length** as its unlimited memory.
- It has a **tape head** that can read and write symbols on the tape.
- Tape head can move to Left or Right.
- The head at each time step:
  - Reads a symbol
  - Writes a symbol
  - Moves Left or Right

Turing Machines



- Reads a
- Writes k
- Moves Left

Turing Machines

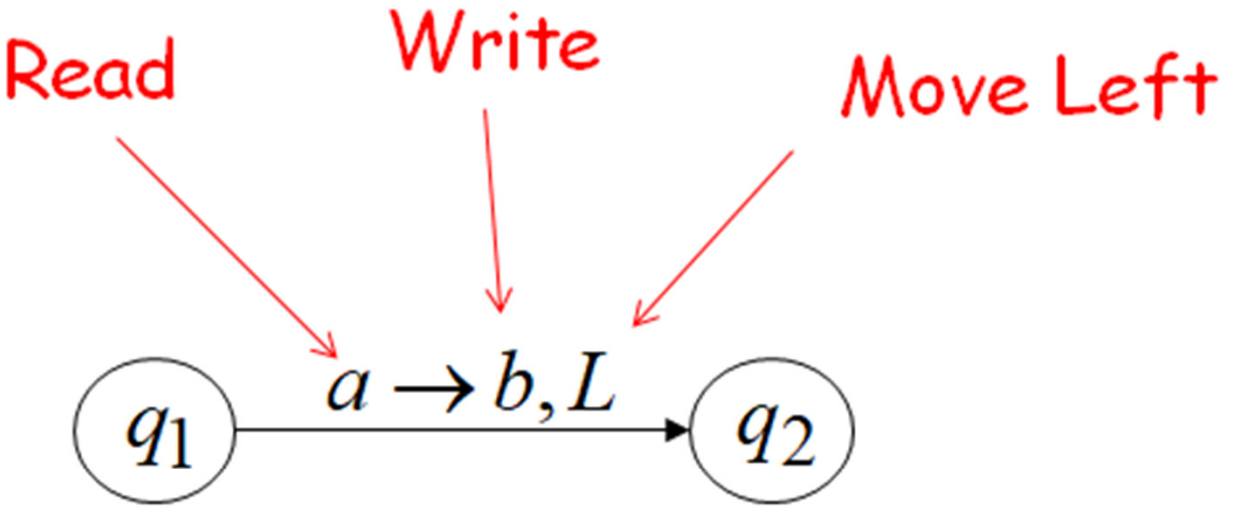


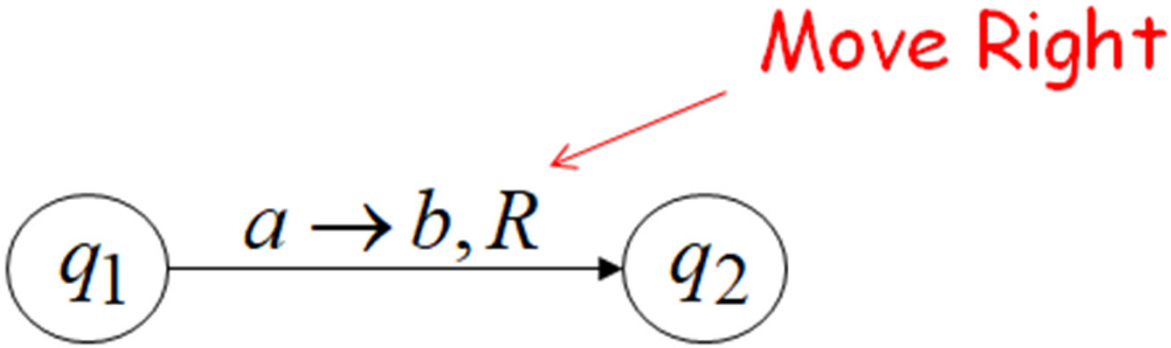
- Reads b
- Writes f
- Moves Right

Turing Machine Computation

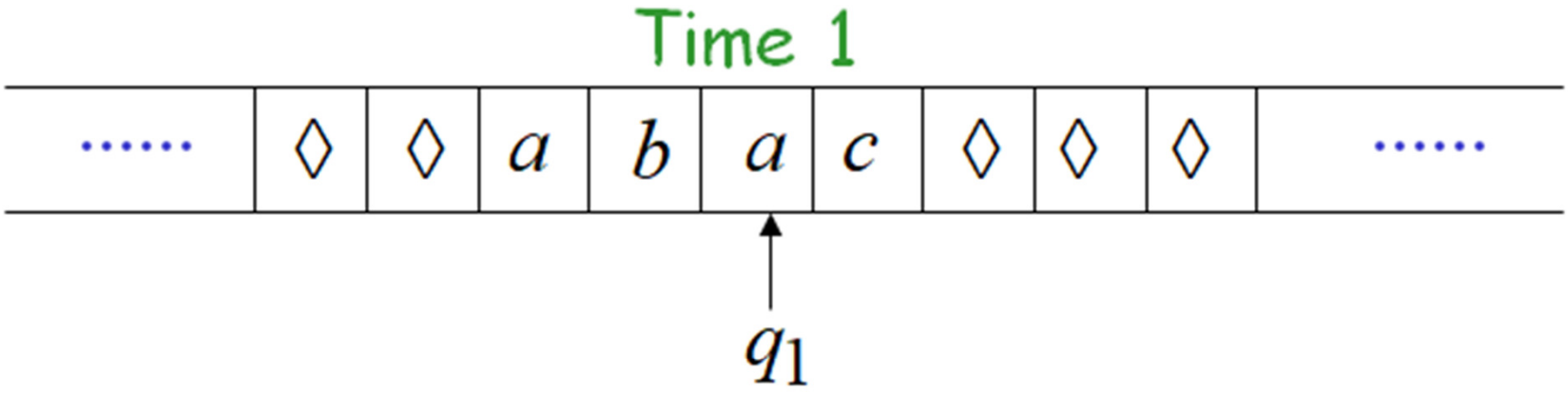
- Read/write head starts at leftmost position on tape
- Input string written on leftmost squares of tape, rest is **blank**
- Computation proceeds according to **transition function**:
  - Given ***current state of machine***, and ***current symbol being read***
  - the machine
    - transitions to new state
    - writes a symbol to its current position (overwriting existing symbol)
    - moves the tape head L or R

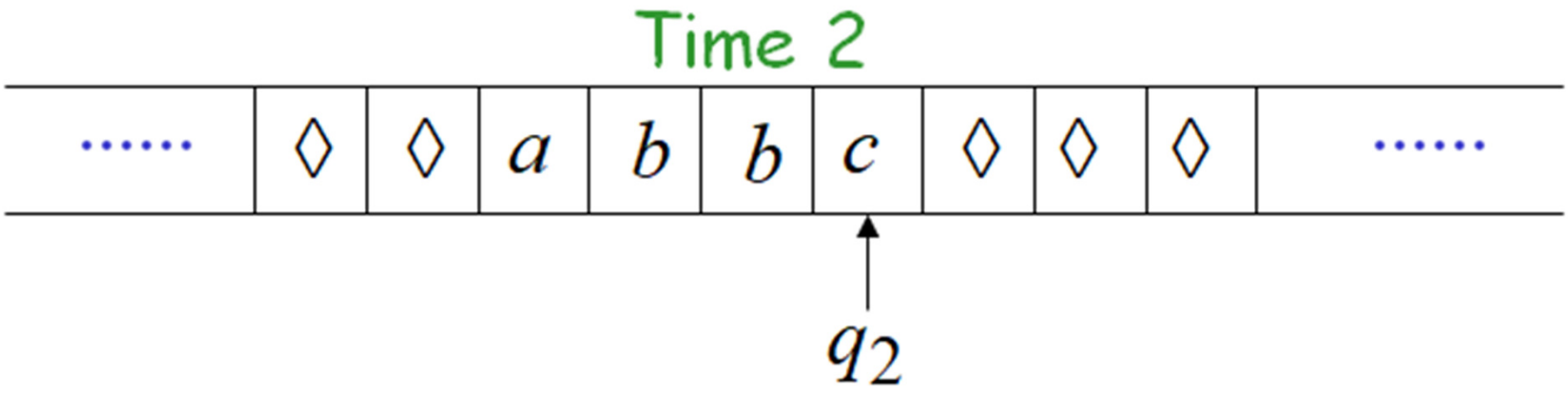
States & Transitions

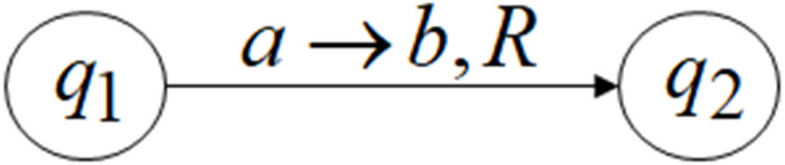




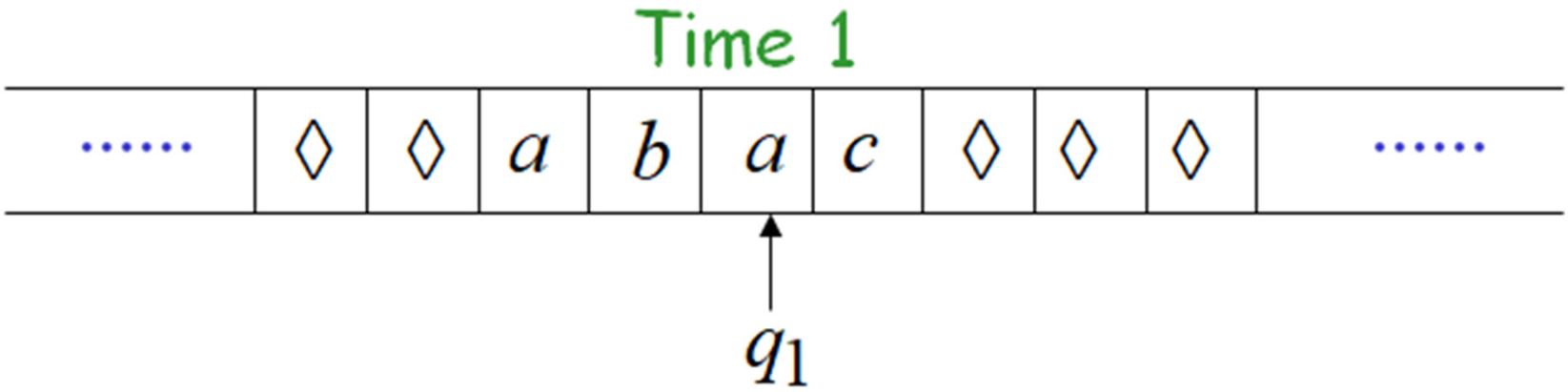
States & Transitions

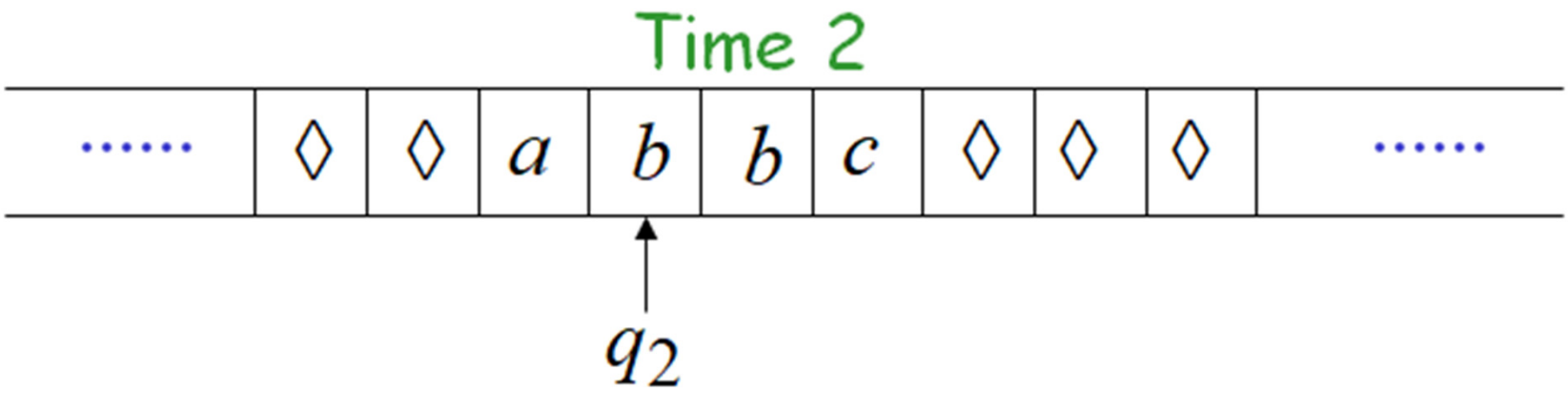


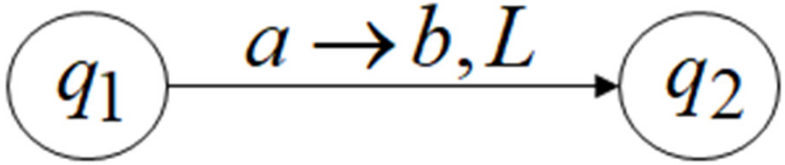




States & Transitions

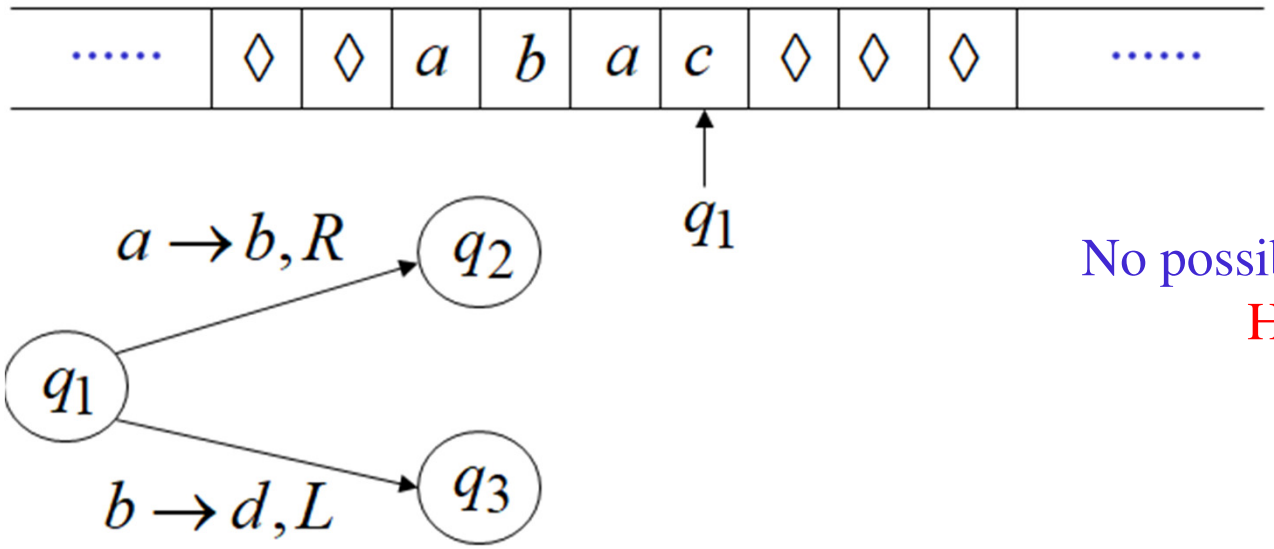






Turing Machine Computation

- The machine halts if there are no possible transitions to follow
- Computation ends when it enters either the **accept** or the **reject** state.
- The machine accepts if machine halts in an accept state.
- The machine rejects the input if it halts in a non-accepting state or if machine enters an infinite loop.

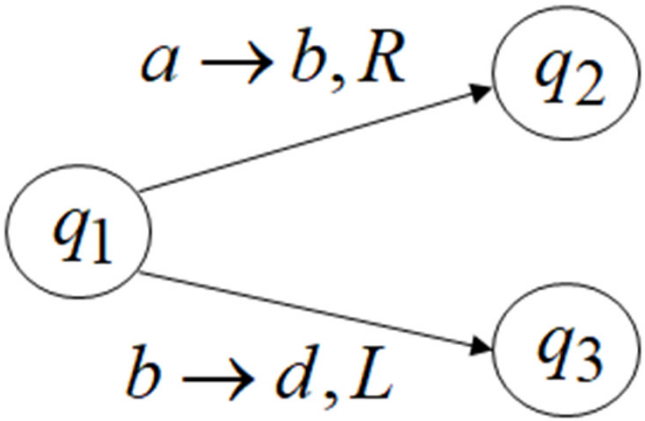


No possible transition
HALT

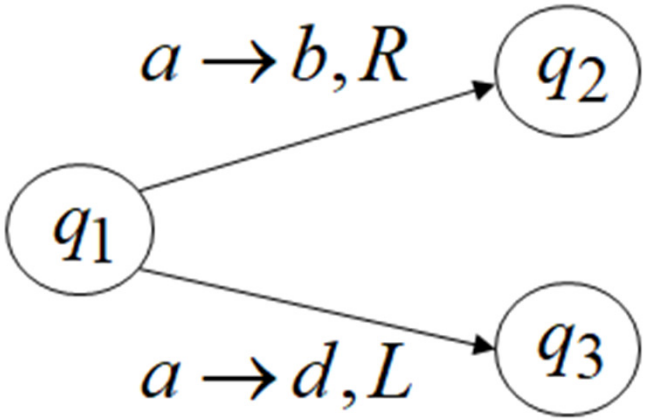
Turing Machine Computation

- Turing Machines are deterministic
- No lambda transitions are allowed

Allowed



Not Allowed

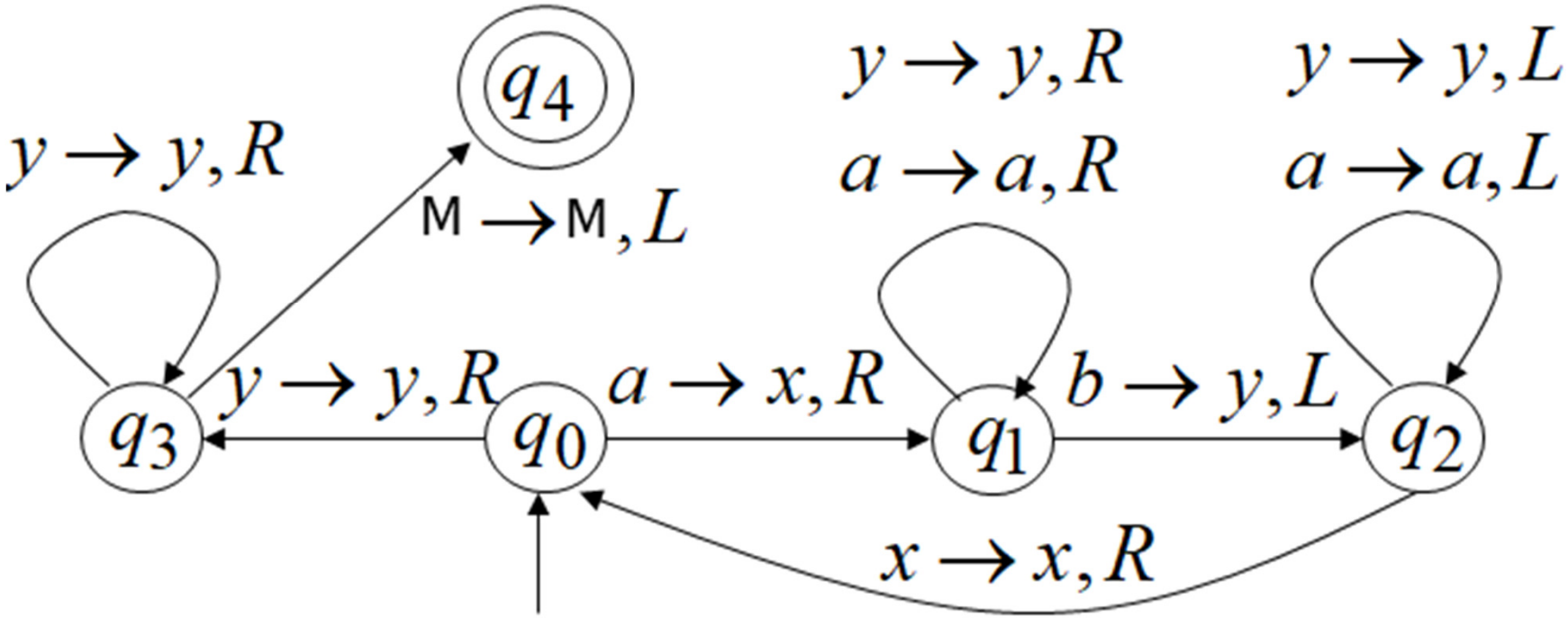


Turing Machine - Example

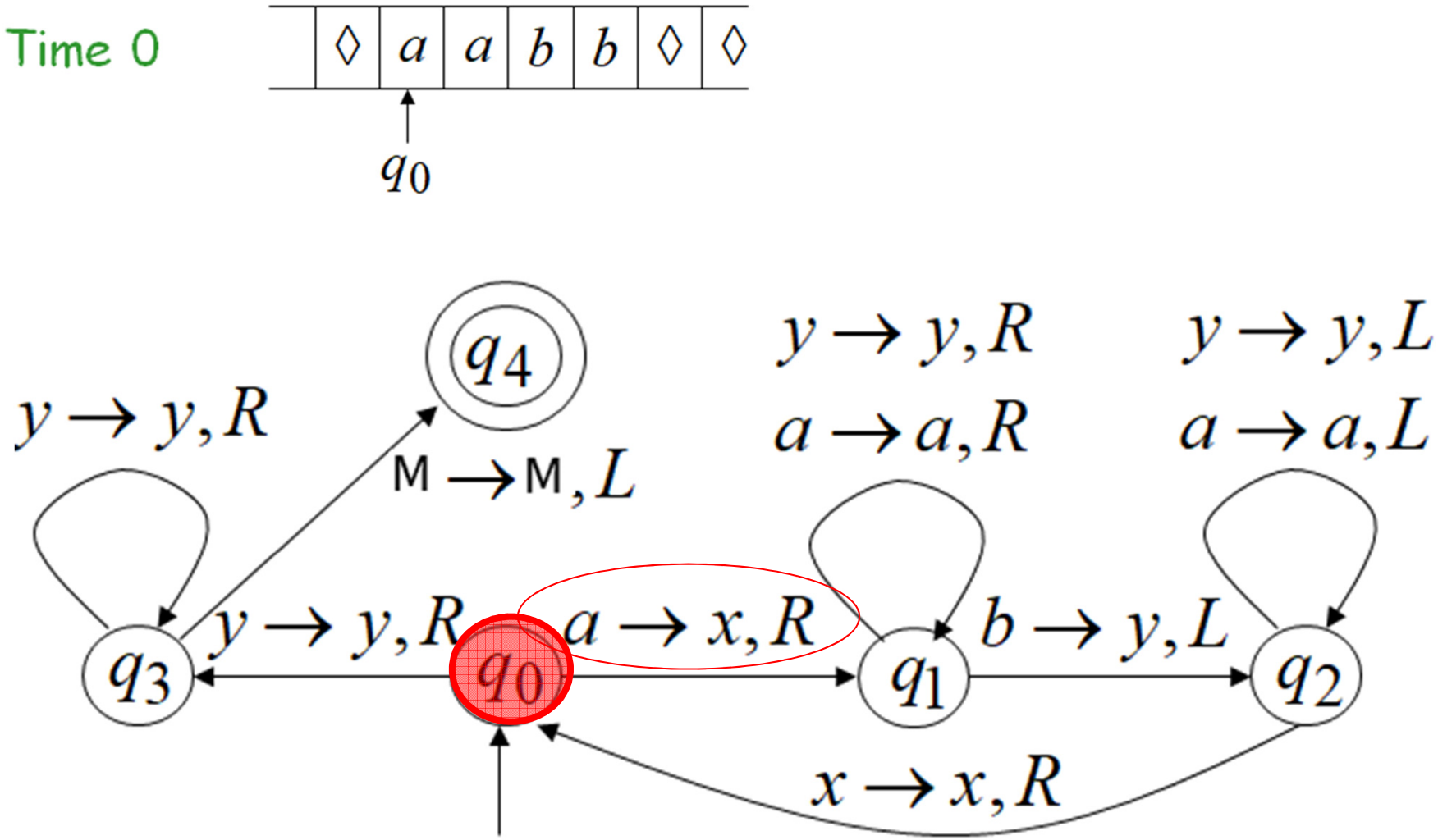
- L = $\{a^n b^n \mid n \geq 0\}$

Idea for Turing machine

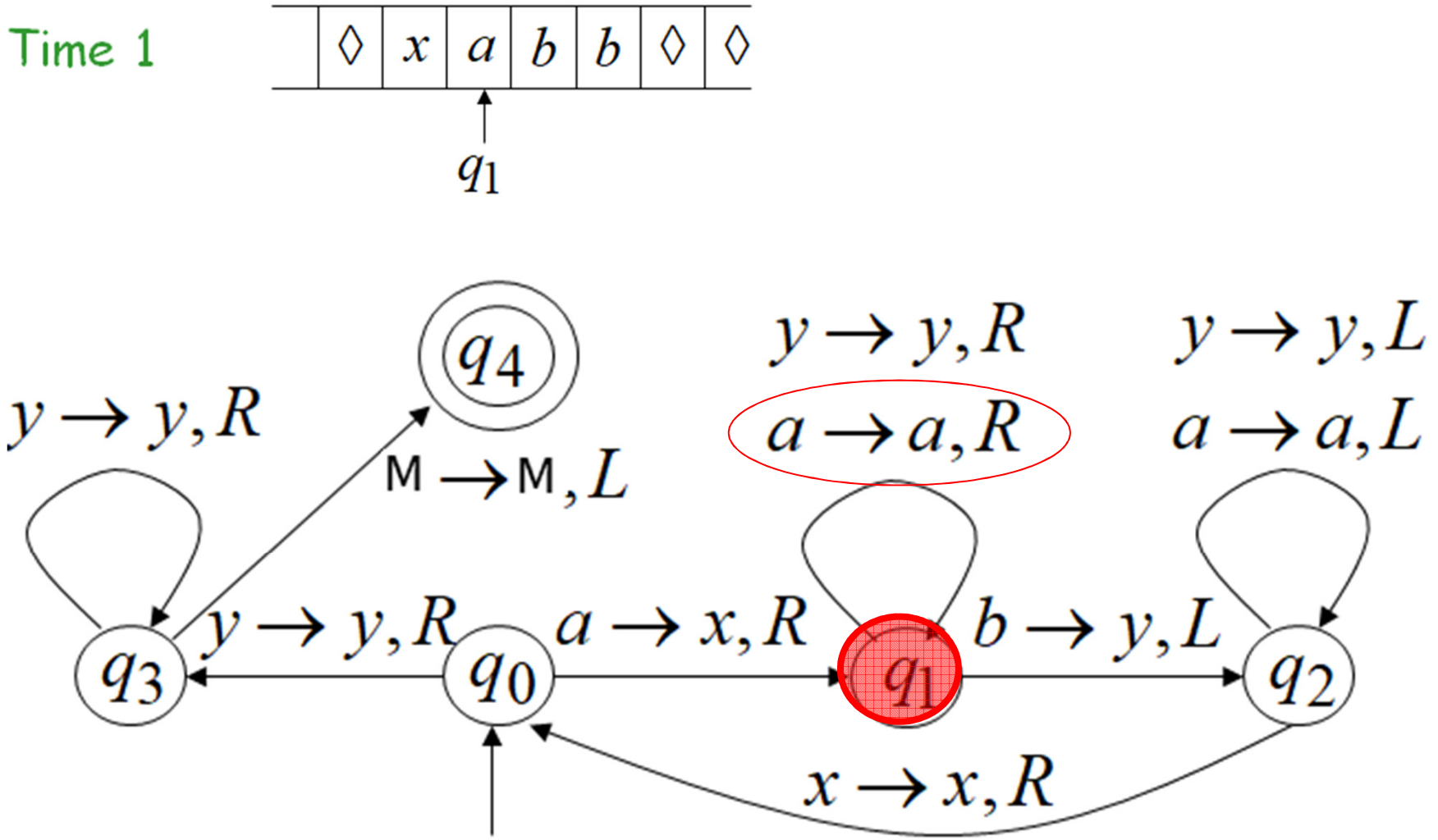
- Zig-zag across tape to match the number of a's and b's
- Replace current a with x and move right to find corresponding b and change b to y to identify matched b's
- When a reaches to end :
  - if there are any b's, ***reject***
  - if there aren't any b's, ***accept***.



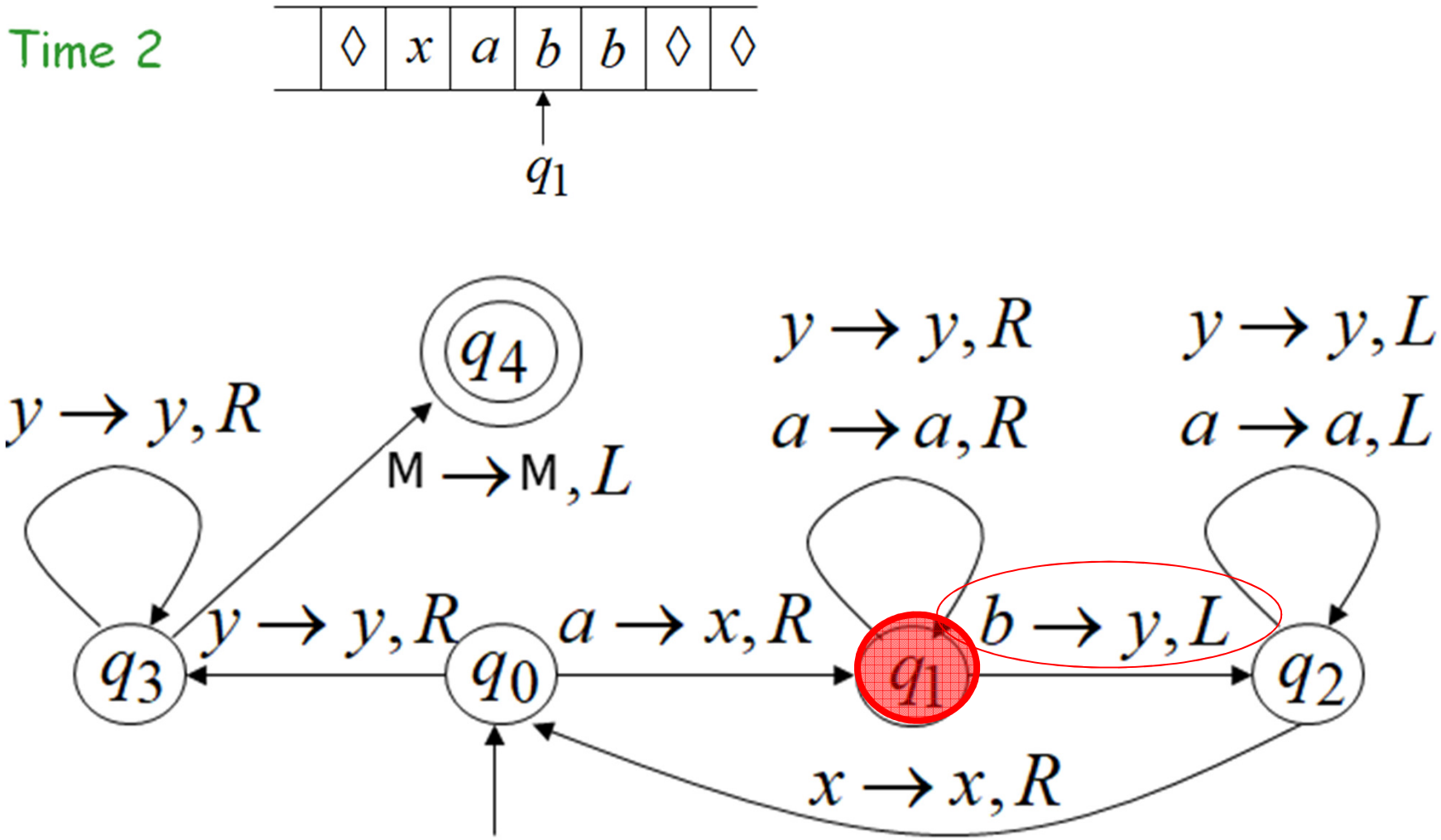
Turing Machine - Example



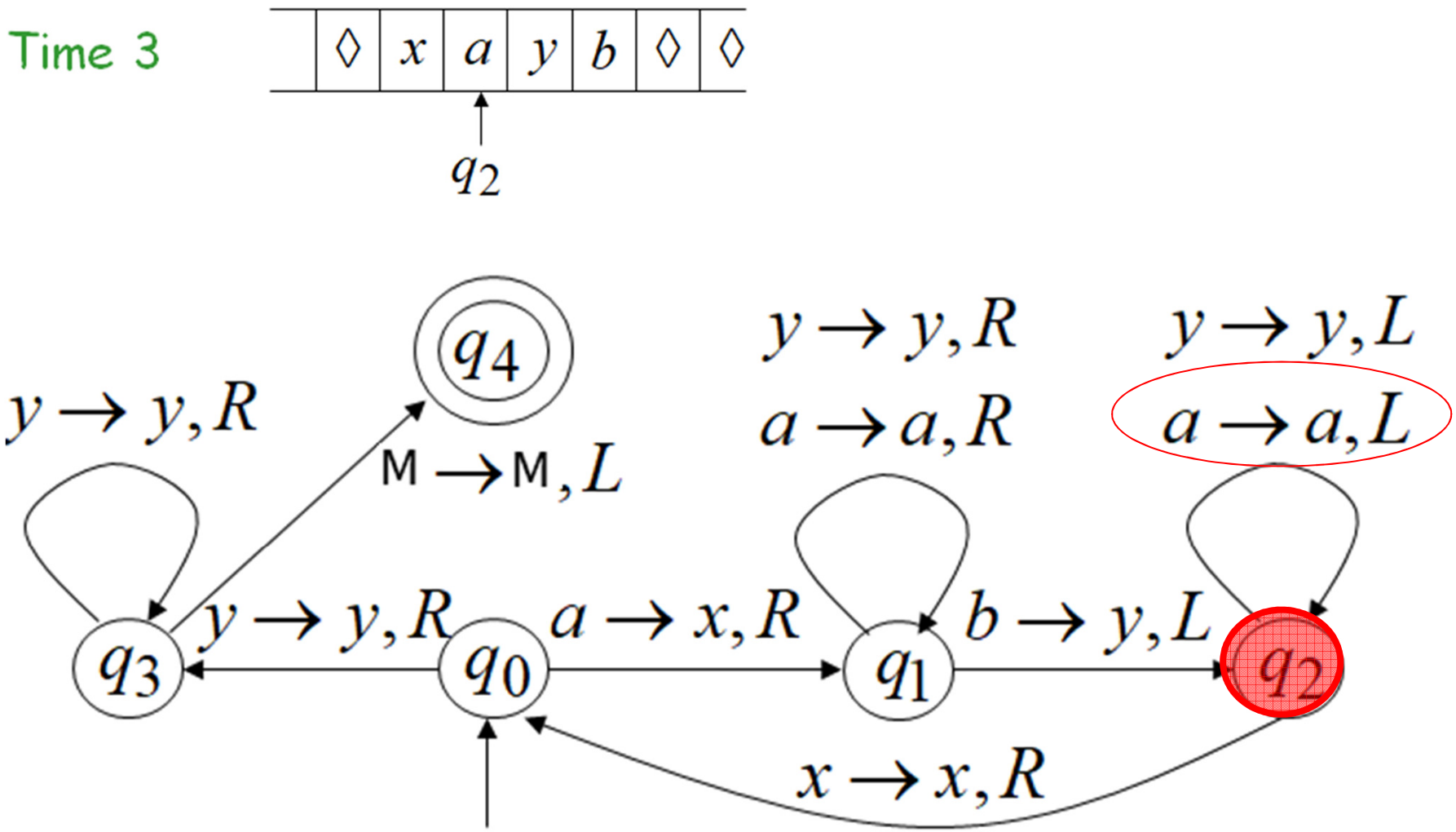
Turing Machine - Example



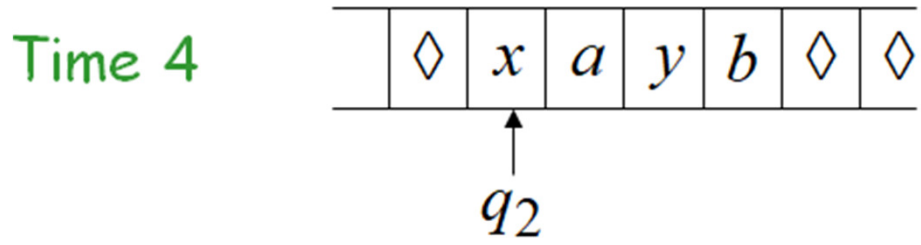
Turing Machine - Example

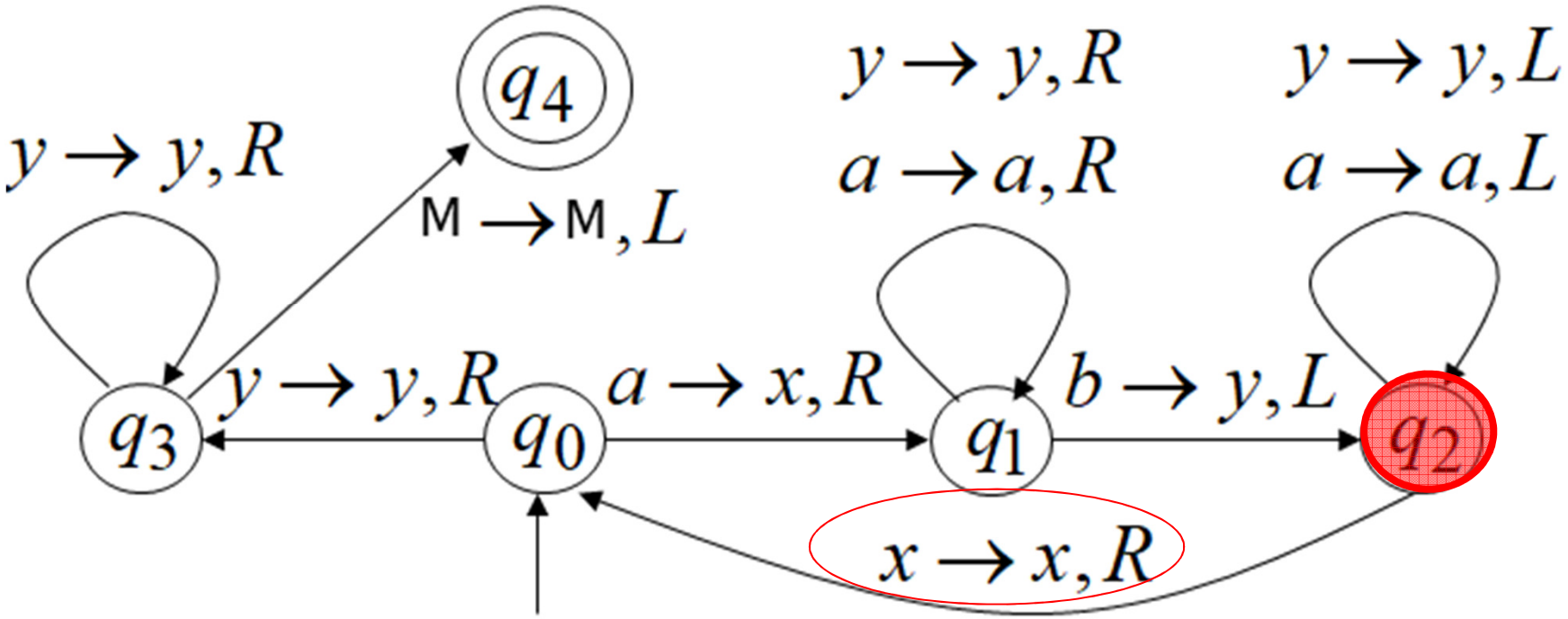


Turing Machine - Example

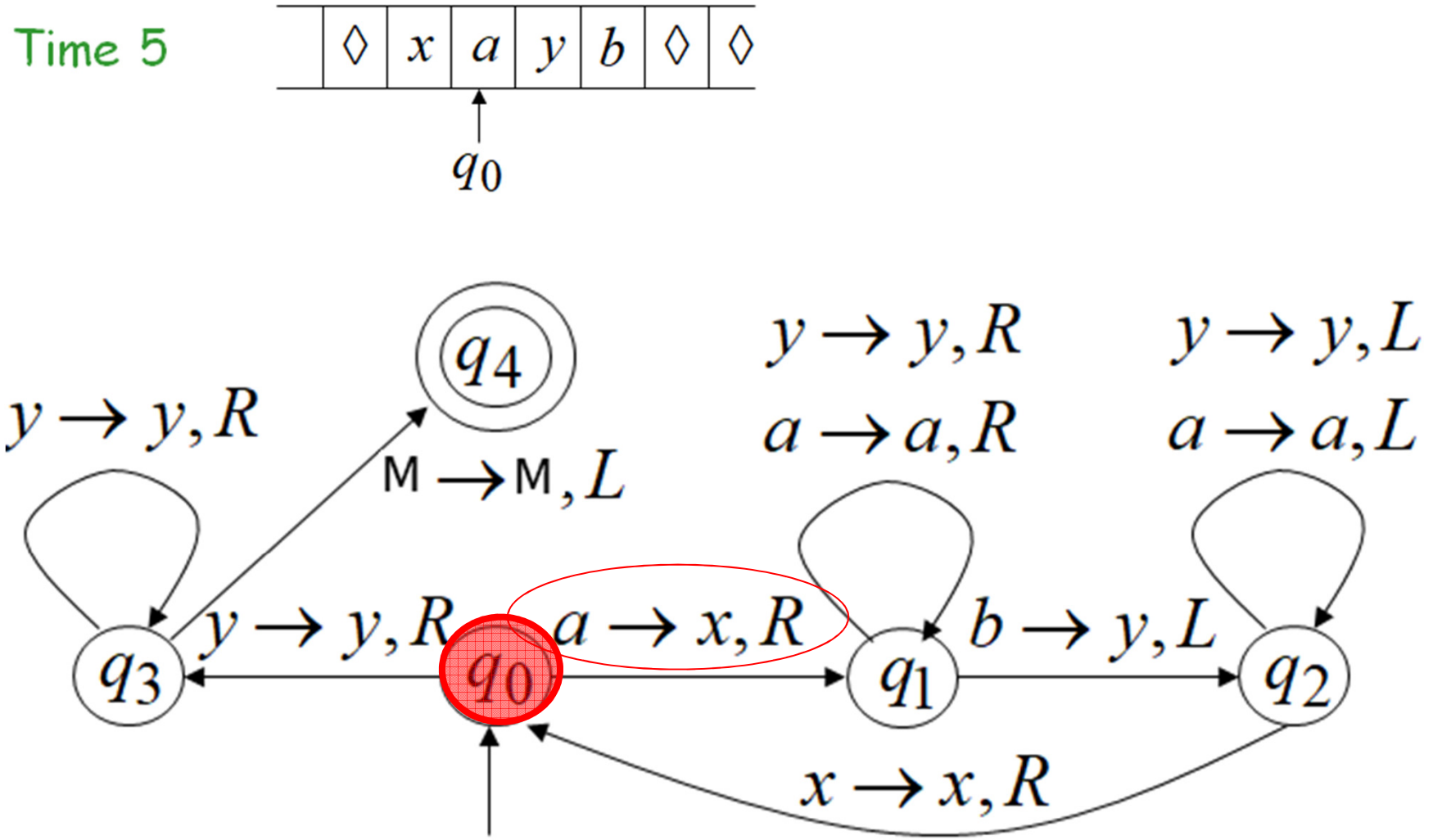


Turing Machine - Example

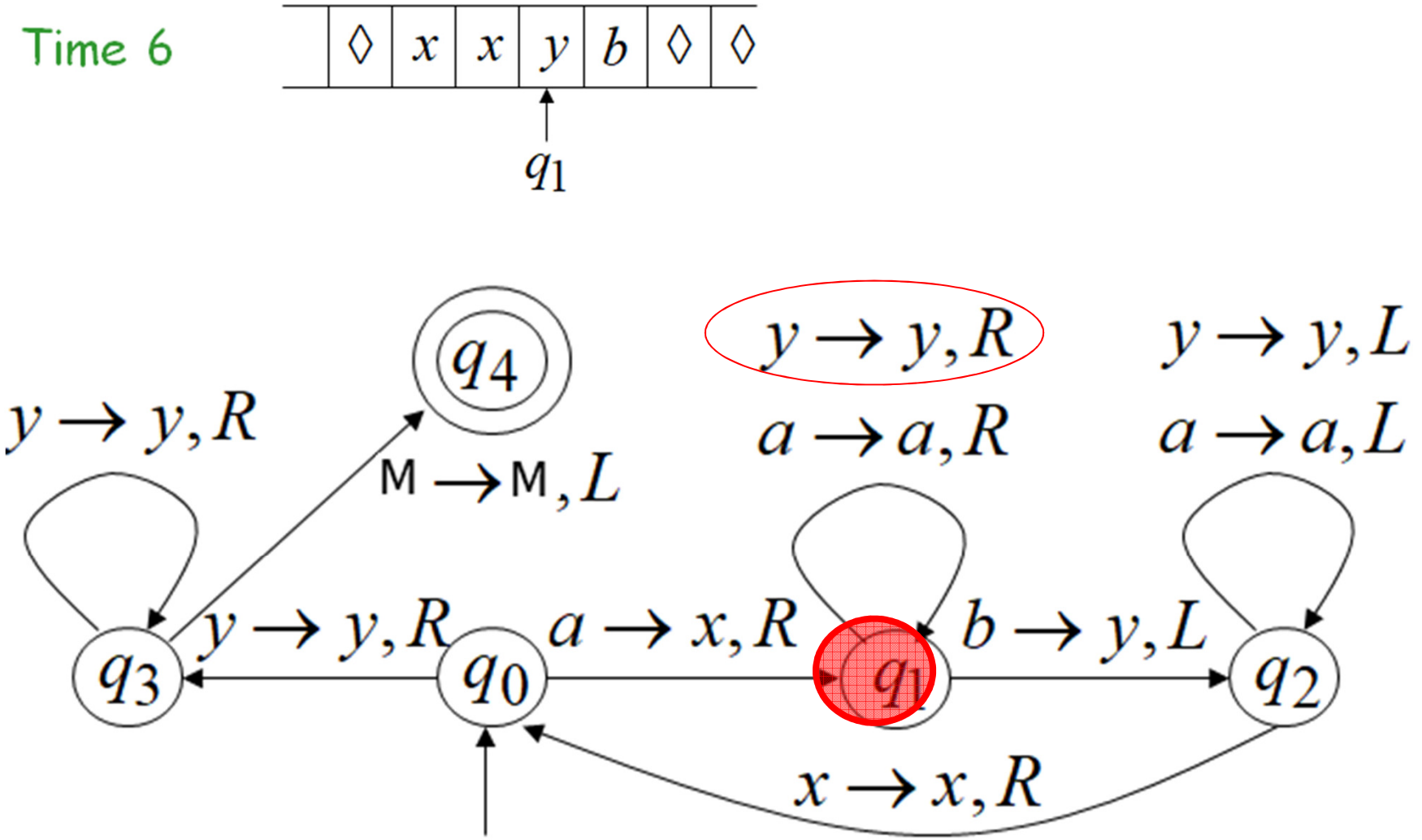




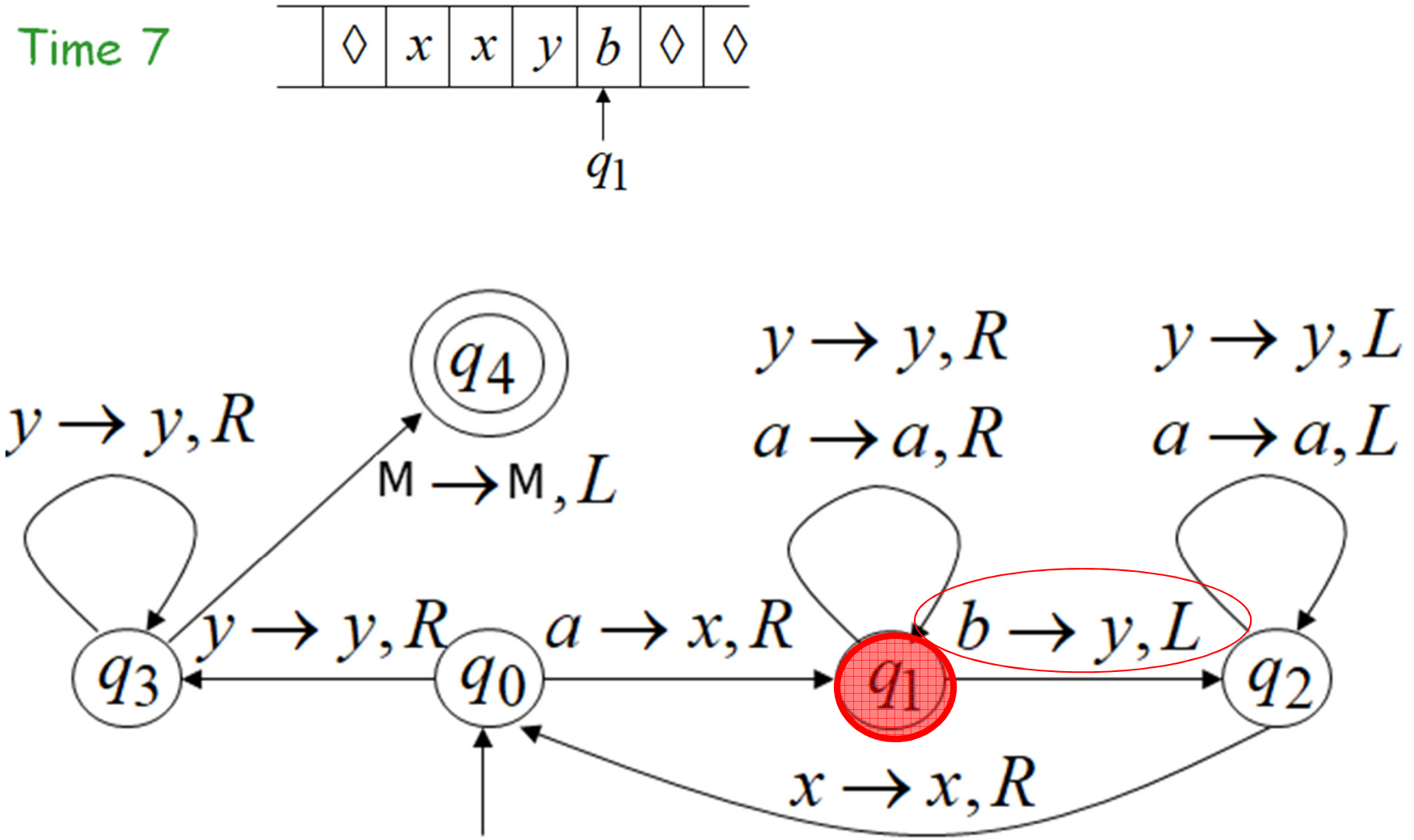
Turing Machine - Example



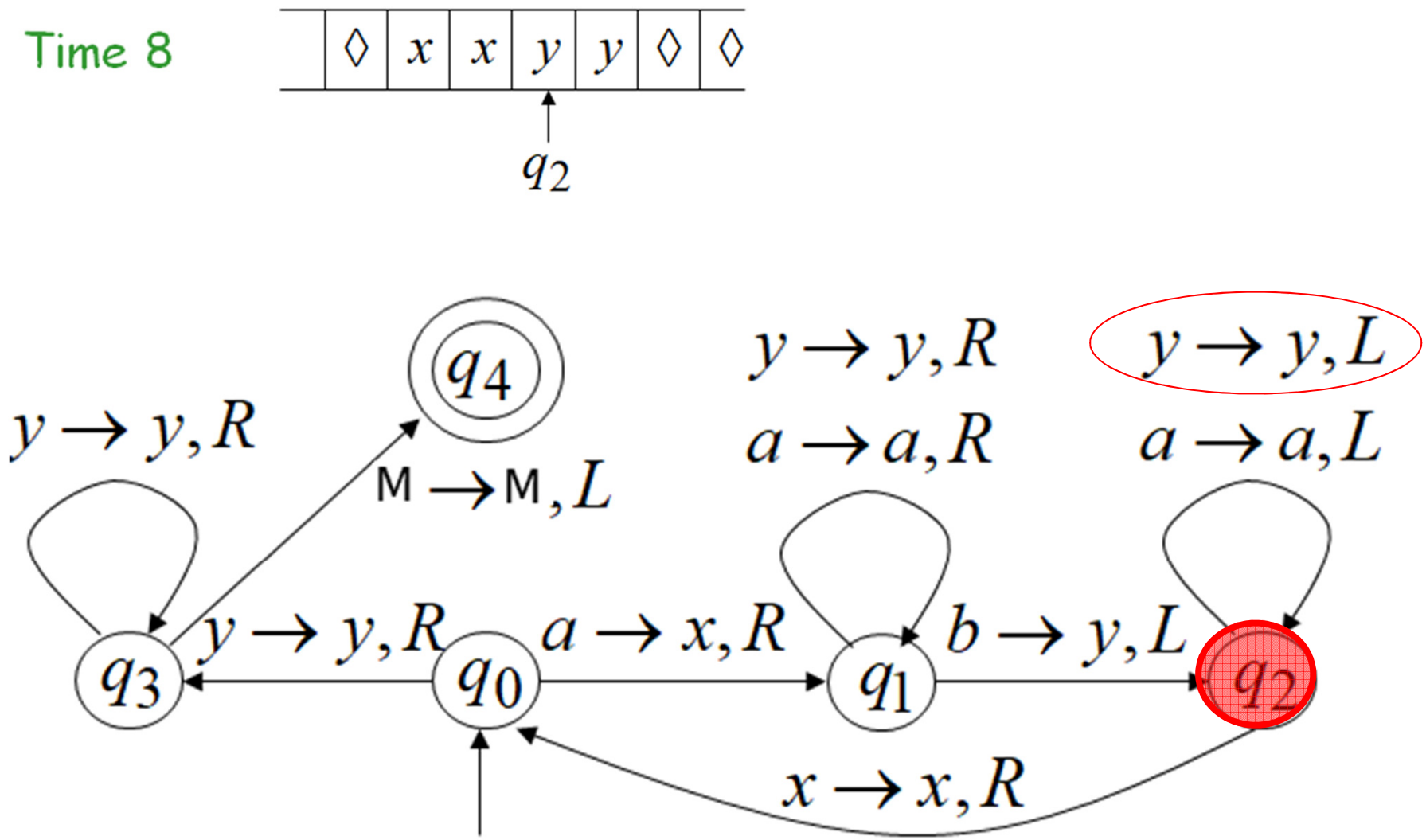
Turing Machine - Example



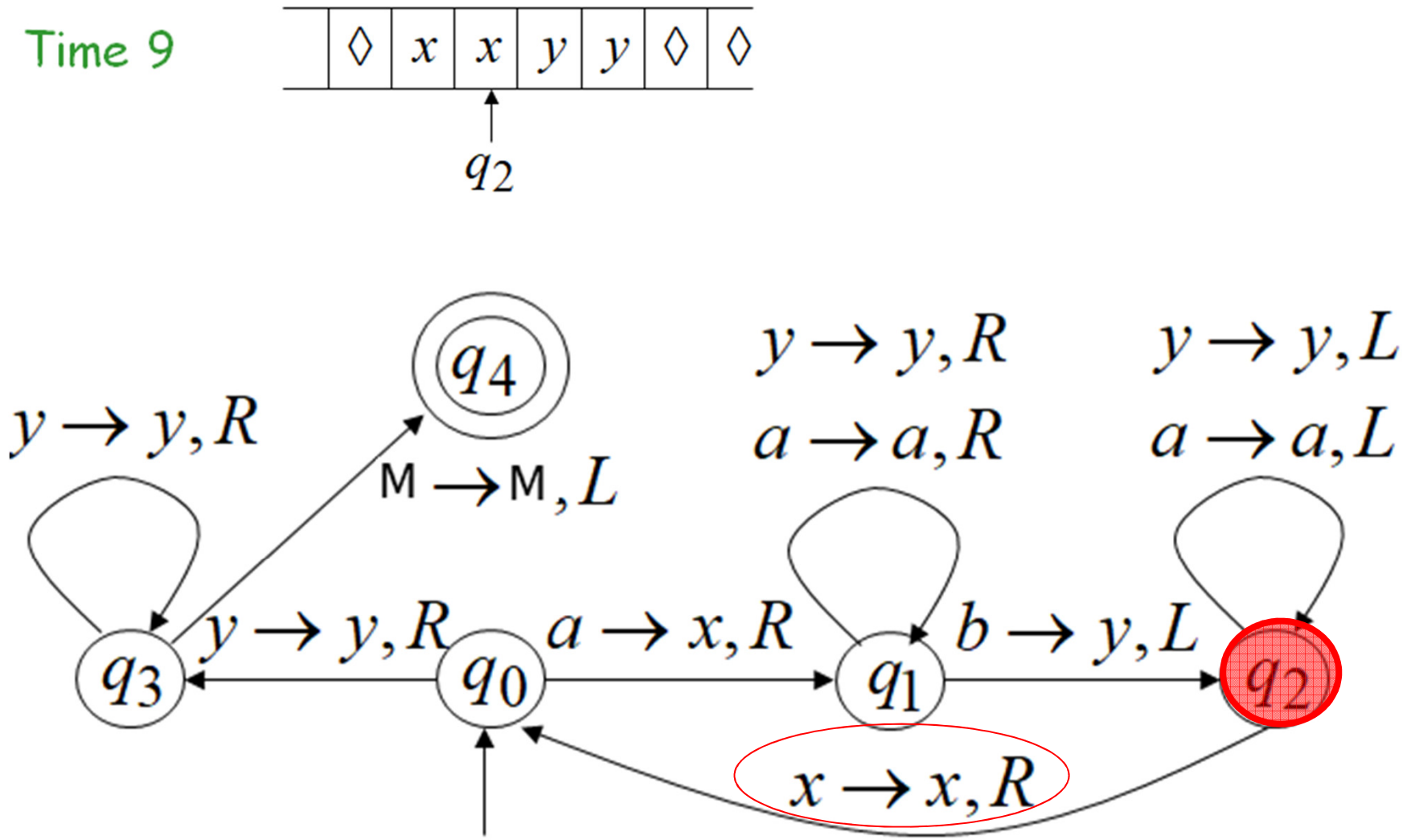
Turing Machine - Example



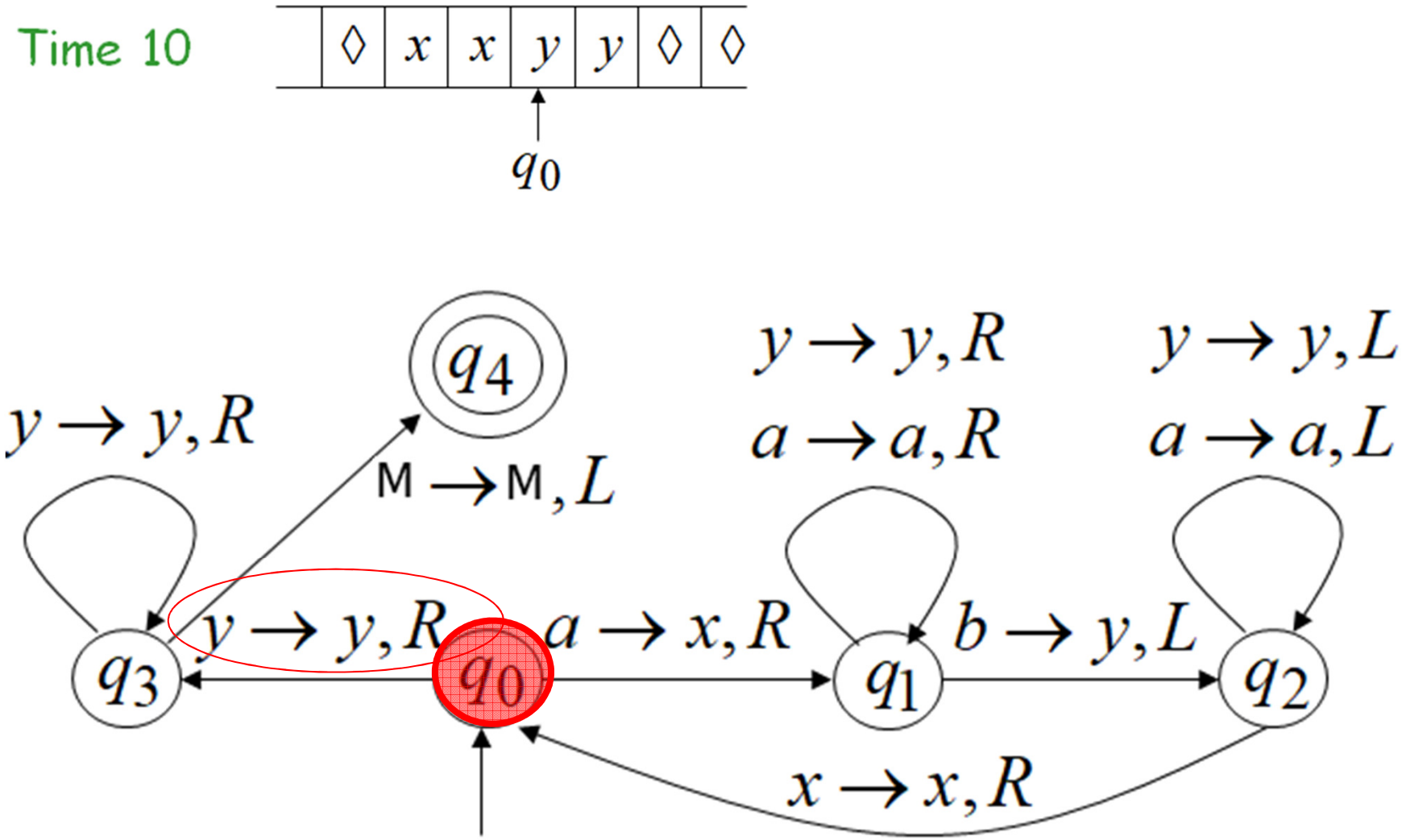
Turing Machine - Example



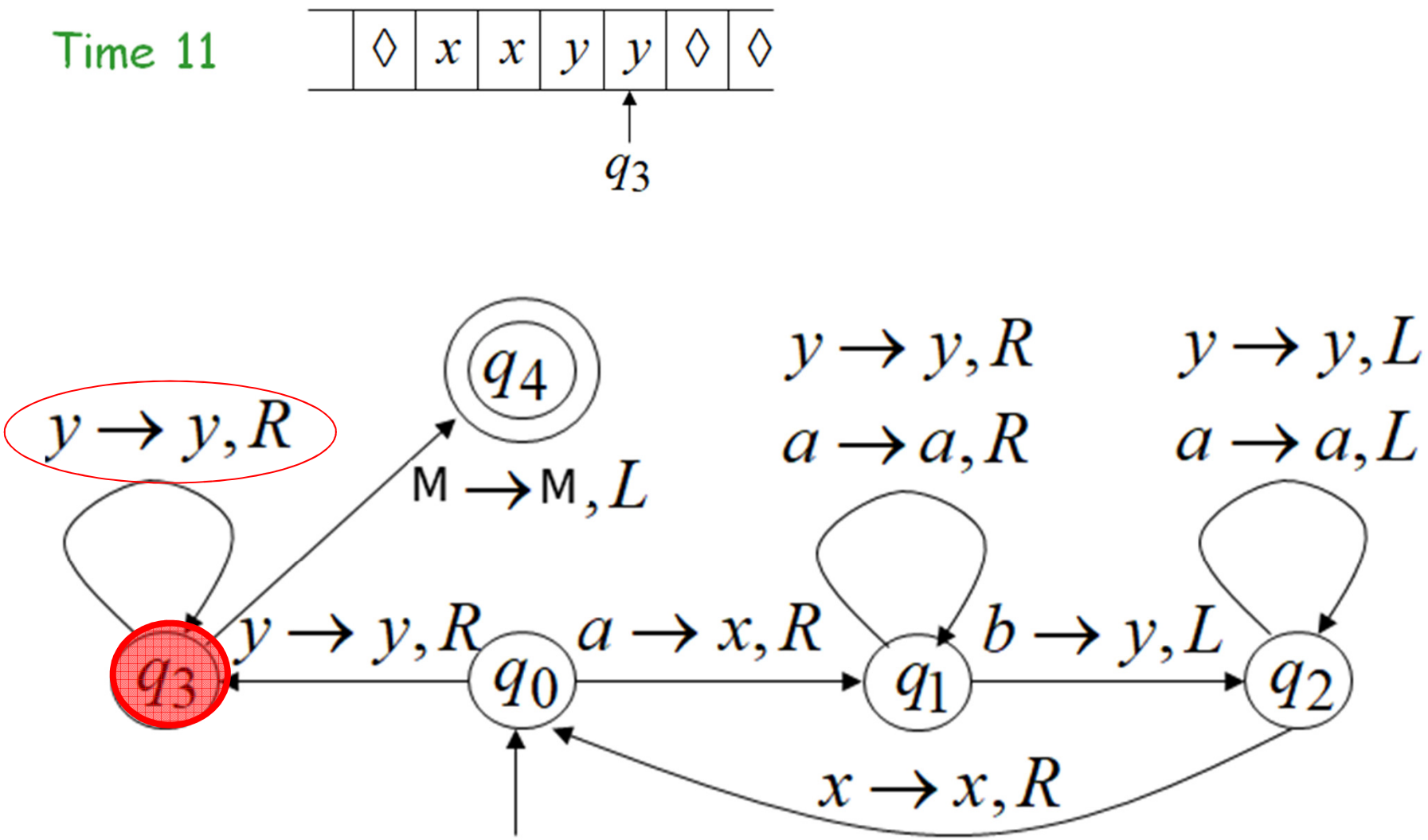
Turing Machine - Example



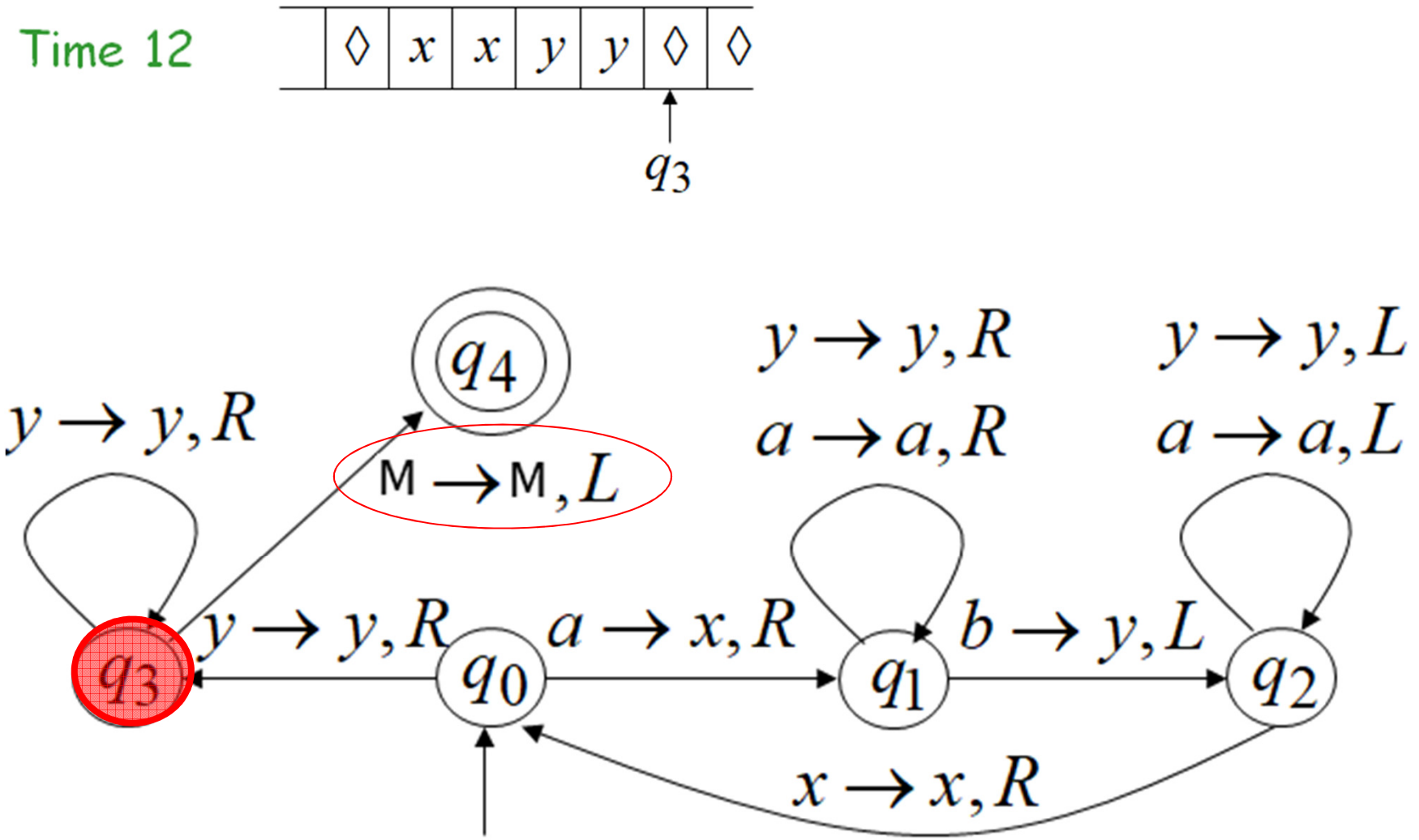
Turing Machine - Example



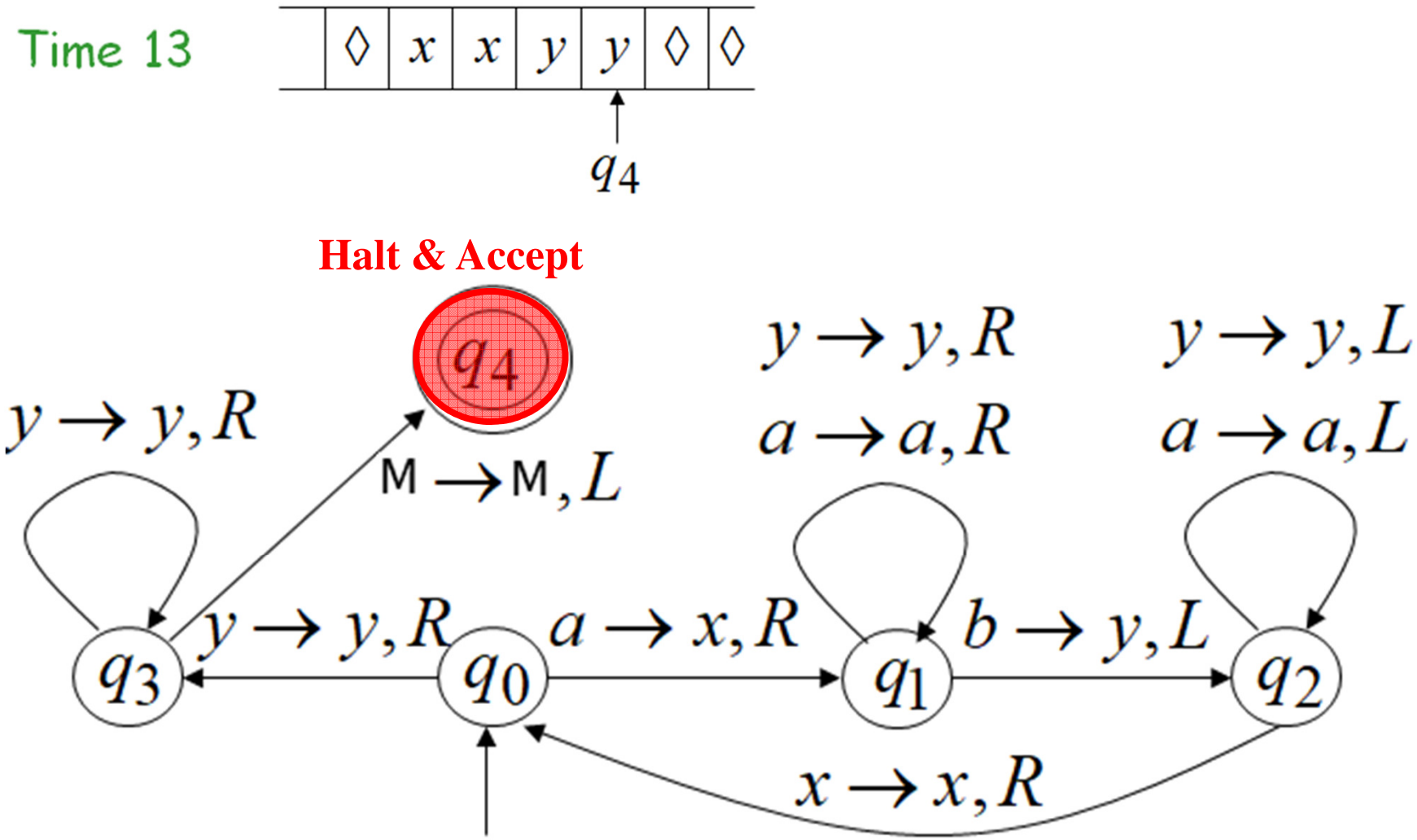
Turing Machine - Example



Turing Machine - Example



Turing Machine - Example



Another Turing Machine Example - 1

- $L = \{ w\#w \mid w \in \{0,1\}^* \}$

Idea for Turing machine

- Zig-zag across tape to corresponding positions on either side of **'#'** to check whether these positions agree.
  - If they do not, or if there is no **'#'**, ***reject***.
  - If they do, cross them off.
- Once all symbols to the left of the **'#'** are crossed off, check for any symbols to the right of **'#'**:
  - if there are any, ***reject***;
  - if there aren't, ***accept***.

Another Turing Machine Example - 1

- $L = \{ w\#w \mid w \in \{0,1\}^* \}$
- The TM starts with the input on the tape.

0 1 1 0 0 0 # 0 1 1 0 0 0 M M M cross the symbol, move to R

X 1 1 0 0 0 # 0 1 1 0 0 0 M M M

→→ . . . move to R until first un-crossed symbol after #, it must be 0

X 1 1 0 0 0 # X 1 1 0 0 0 M M M

←← . . . move to L until first crossed symbol after #

X 1 1 0 0 0 # X 1 1 0 0 0 M M M move to R

X X 1 0 0 0 # X 1 1 0 0 0 M M M

→→ . . . move to R until first un-crossed symbol after #, it must be 1

⋮ Repeat ziz-zag operations

X X X X X X # X X X X X X M M M move to R

X X X X X X # X X X X X X M M M

→→ . . . move to R until first un-crossed symbol, it must be blank

X X X X X X # X X X X X X M M M **ACCEPT**

Formal Definition of a Turing Machine

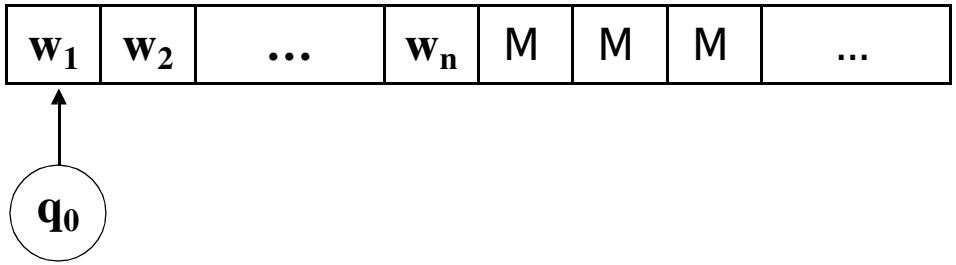
- **A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and**

  1. Q **is the set of states,**
  2. Σ **is the input alphabet not containing the** ***blank symbol*** M**,**
  3. Γ **is the tape alphabet, where** $M \in \Gamma$ and $\Sigma \subset \Gamma$**,**
  4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ **is the transition function,**
  5. q_0 **is the start state,**
  6. q_{accept} **is the accept state, and**
  7. q_{reject} **is the reject state, where** $q_{reject} \neq q_{accept}$

How does a Turing Machine Compute?

Initial Configuration:

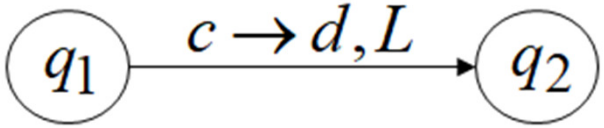
- A Turing Machine M receives its input $w = w_1 w_2 \ldots w_n$ on the leftmost **n** squares on the tape. The rest of the tape is blank.
  - The head starts on the leftmost square on the tape.
  - The first blank symbol on the tape marks the end of the input.
- The initial state is the start state $\mathbf{q_0}$.



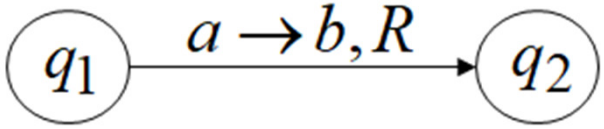
How does a Turing Machine Compute?

Transition:

- The computation proceeds according to the **transition function** δ.
  - If the current state is $\mathbf{q_i}$, the current tape symbol **a**, and $\delta(\mathbf{q_i}, \mathbf{a}) = (\mathbf{q_j}, \mathbf{b}, \mathbf{D})$ where **a** and **b** are tape symbols (they can be the same symbol) and **D** is **L** or **R**, then
    - Machine M goes from the state $\mathbf{q_i}$ to state $\mathbf{q_j}$.
    - Machine M writes **b** onto the current tape position
    - Tape head moves to **Left** (if **D** is **L**) or moves to **Right** (if **D** is **R**)



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



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

- The head of Machine M never moves left of the beginning of the tape.
  - **If Machine M is on the leftmost square, it stays there!**

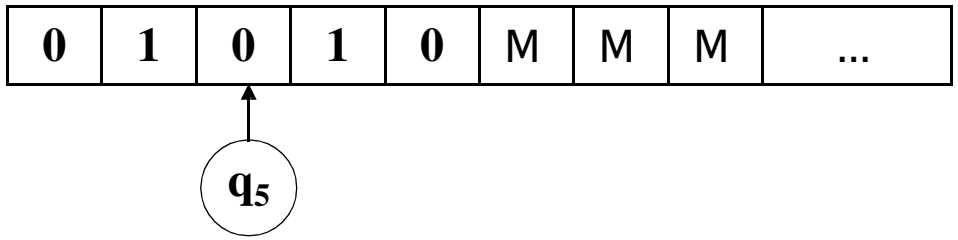
Configuration of a Turing Machine

- As a Turing machine computes, changes occur in the current state, the current tape contents, and the current head location.
- **Each step of a TM computation can be captured by the notion of a configuration.**

Configuration:

- **For a state q and two strings u and v over the tape alphabet Γ, we write uqv for the configuration where the current state is q, the current tape contents is uv, and the current head location is the first symbol of v.**
- **The tape contains only blanks following the last symbol of v.**

Configuration of a Turing Machine



- The current state is q_5,
- **u = 01** is **to the left of the head**,
- **v = 010** is **under and to the right of the head**.
- Tape has **uv = 01010** on it.

- **We represent this configuration by $01q_5010$**

start configuration : q_0 w

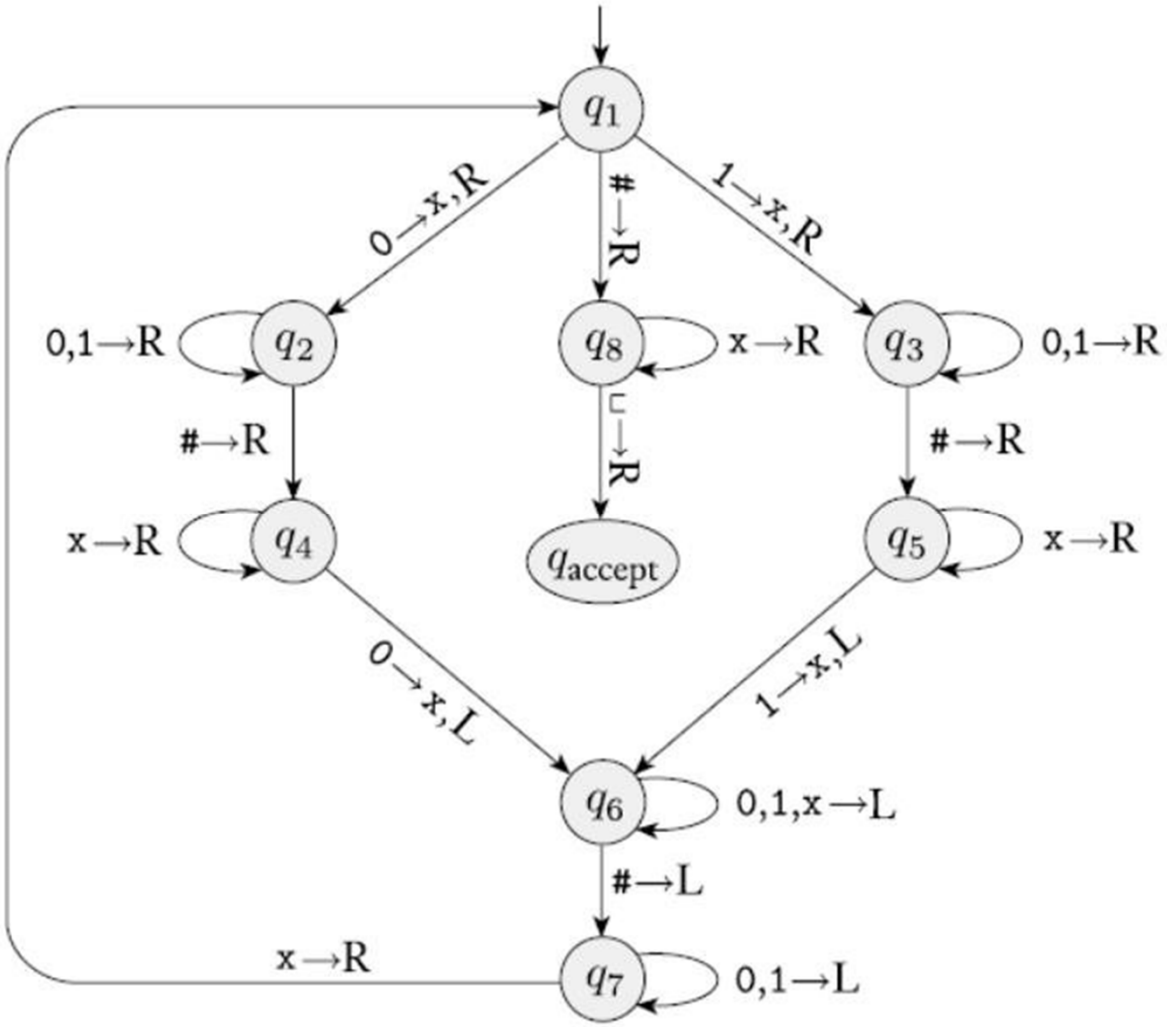
accepting configuration, the state of the configuration is q_{accept}

rejecting configuration, the state of the configuration is q_{reject}

Turing Machine - Example

- $L = \{ w\#w \mid w \in \{0,1\}^* \}$
- A formal description of a TM which decides L is $(Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$ where
  - $Q = \{q_1,\ldots,q_8, q_{accept}, q_{reject} \}$
  - $\Sigma = \{0,1,\#\}$
  - $\Gamma = \{0,1,\#,x,M\}$
  - The start, accept, and reject states are q_1, q_{accept}, and q_{reject}, respectively.
  - We describe δ with a state diagram.
    - On the given state diagram, q_{reject} is missing and some transitions are also missing.
    - We assume that all missing transitions goes to state q_{reject} without changing the current tape symbol and head moves to right.

Turing Machine - Example



- q_{reject} is missing and all missing transitions goes to state q_{reject} .

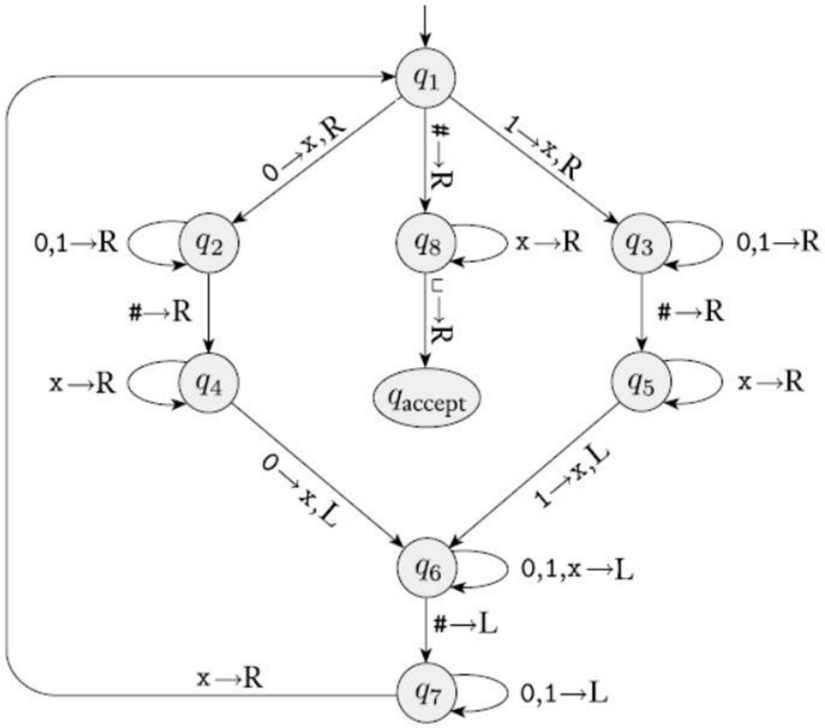
Arc label meanings:

- $0 \rightarrow x,R$ from q_1 to q_2 means: $\delta(q_1,0)=(q_2,x,R)$
- $\# \rightarrow R$ from q_1 to q_8 means: $\delta(q_1,\#)=(q_8,\#,R)$
- $0,1 \rightarrow R$ from q_3 to q_3 means: $\delta(q_3,0)=(q_3,0,R)$ and $\delta(q_3,1)=(q_3,1,R)$
- $0,1,x \rightarrow L$ from q_6 to q_6 means: $\delta(q_6,0)=(q_6,0,L)$ $\delta(q_6,1)=(q_6,1,L)$ and $\delta(q_6,x)=(q_6,x,L)$

Missing arcs from q_8 means:
$\delta(q_8,0)=(q_{reject},0,R)$
$\delta(q_8,1)=(q_{reject},1,R)$
$\delta(q_8,\#)=(q_{reject},\#,R)$

Turing Machine - Example

An accepting configuration



The computation of 01#01

q_1 01#01 ⇒ x q_2 1#01 ⇒ x1 q_2 #01 ⇒ x1# q_4 01

⇒ x1 q_6 #x1 ⇒ x q_7 1#x1 ⇒ q_7 x1#x1 ⇒ x q_1 1#x1

⇒ xx q_3 #x1 ⇒ xx# q_5 x1 ⇒ xx#x q_5 1 ⇒ xx# q_6 xx

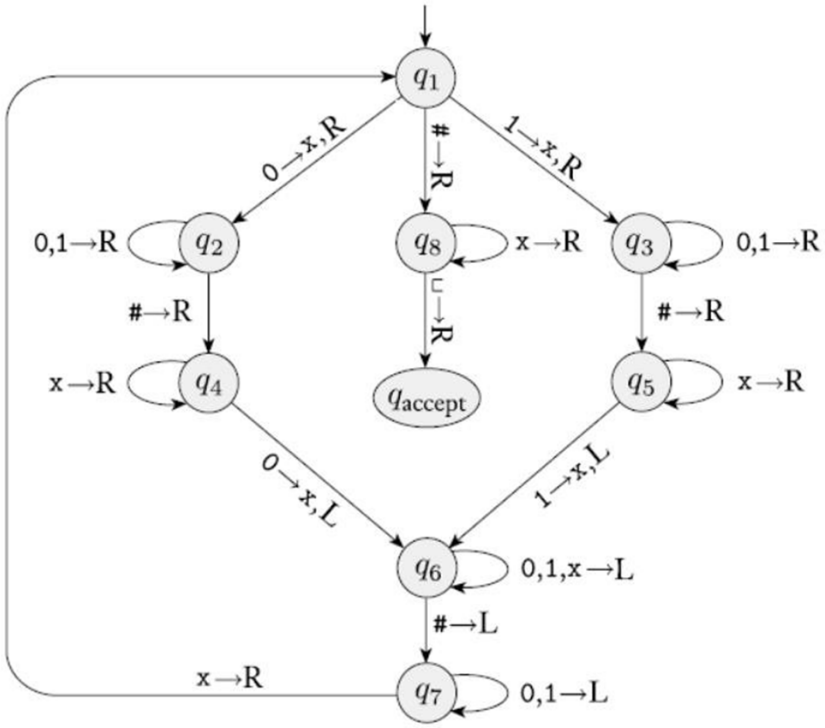
⇒ xx q_6 #xx ⇒ x q_7 x#xx ⇒ xx q_1 #xx ⇒ xx# q_8 xx

⇒ xx#x q_8 x ⇒ xx#xx q_8 M ⇒ **xx#xx**M $\mathbf{q_{accept}}$ M

ACCEPT

Turing Machine - Example

An rejecting configuration



The computation of 01#00

q_1 01#00 ⇒ x q_2 1#00 ⇒ x1 q_2 #00 ⇒ x1# q_4 00

⇒ x1 q_6 #x0 ⇒ x q_7 1#x0 ⇒ q_7 x1#x0 ⇒ x q_1 1#x0

⇒ xx q_3 #x0 ⇒ xx# q_5 x0 ⇒ xx#x **q_5 0**

⇒ **xx#x0 q_{reject}** M

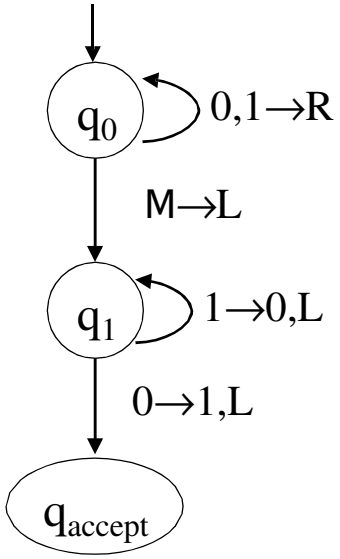
REJECT

Another Turing Machine Example - 2

- A TM to add 1 to a binary number (with a 0 in front)
- M = "On input w
  1. Go to the right end of the input string
  2. Move left as long as a 1 is seen, changing it to a 0.
  3. Change the 0 to a 1, and halt.
- For example, to add 1 to w = 0110011
  - Change all the ending 1's to 0's $\Rightarrow$ 0110000
  - Change the next 0 to a 1 $\Rightarrow$ 0110100
- Now let's design a TM for this problem.

Another Turing Machine Example - 2

A TM to add 1 to a binary number (with a 0 in front)



Language of a Turing Machine

Accepting (Recognizing) String:

- A Turing machine M accepts an input string w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where
  1. C_1 is the start configuration of M on input w,
  2. Each C_i yields C_{i+1}, and
  3. C_k is an accepting configuration.

Language of a Turing Machine M (or Language Recognized by M):

- The language of a Turing machine M, $L(M)$ is the set of strings w that are recognized by M.
- More formally,

$$L(M) = \{ w \mid q_0 w \Rightarrow^* C \text{ where } q_0 w \text{ is the starting configuration of } M \text{ on input } w \text{ and } C \text{ is an accepting configuration of } M. \}$$

Language of a Turing Machine

Turing-Recognizable:

- **A language *L* is Turing-recognizable if some Turing machine recognizes it.**
  - **It is also called a recursively enumerable language.**

- When we start a Turing machine on an input, three outcomes are possible.
  - The machine may ***accept***, ***reject***, or ***loop***.
  - By ***loop*** we mean that **the machine simply does not halt**.
  - **Looping** may entail any simple or complex behavior that never leads to a halting state.

Language of a Turing Machine

- A ***Turing machine M can fail to accept*** an input by ***entering q_{reject} state*** or ***looping***.
  - We prefer Turing machines that halt on all inputs; such machines never loop.
  - These machines are called deciders because they always make a decision to accept or reject.
  - **A decider that recognizes some language also is said to decide that language.**

Turing-Decidable (Decidable):

- **A language L is Turing-decidable (or decidable) if some Turing machine (which is a decider) decides it.**
  - **It is also called a recursive language.**

- **Every decidable language is Turing-recognizable.**
  - **But there are languages that are Turing-recognizable but not decidable.**

Turing Machines: Story So Far!

- **Turing Machines are the most general model of computation.**
- **Computations of a TM are described by a sequence of configurations.**
  - Accepting Configuration: contains state q_{accept}
  - Rejecting Configuration: contains state q_{reject}
  - Starting Configuration for input w: $q_0\ w$ where q_0 is the start state
- **Turing-recognizable languages**
  - TM halts in an accepting configuration if w is in the language.
  - TM may halt in a rejecting configuration or go on indefinitely if w is not in the language.
- **Turing-decidable languages**
  - TM halts in an accepting configuration if w is in the language.
  - TM halts in a rejecting configuration if w is not in the language.