Part Two

Computability Theory

The Church-Turing Machines

Decidability

Reducibility

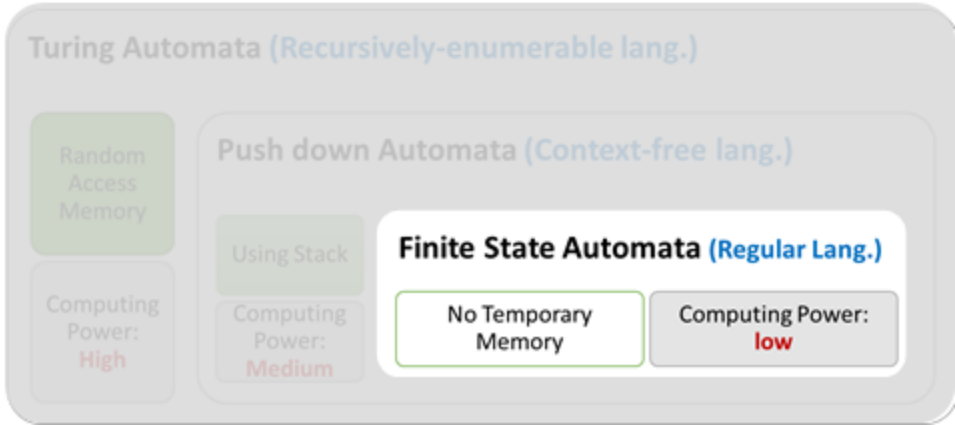
Chapter 3

Introduction

In chapter 1 :

- We learned that
  - we can describe many **languages** (regular languages) using two equivalent methods:
    - finite automata (DFA, NFA)
      - They have a small amount of memory
    - regular expressions (RE)

- We found out
  - There are some **languages** that cannot be described using the methods above
    - Non-regular languages
    - Example: $\{0^n1^n \mid n \geq 0\}$



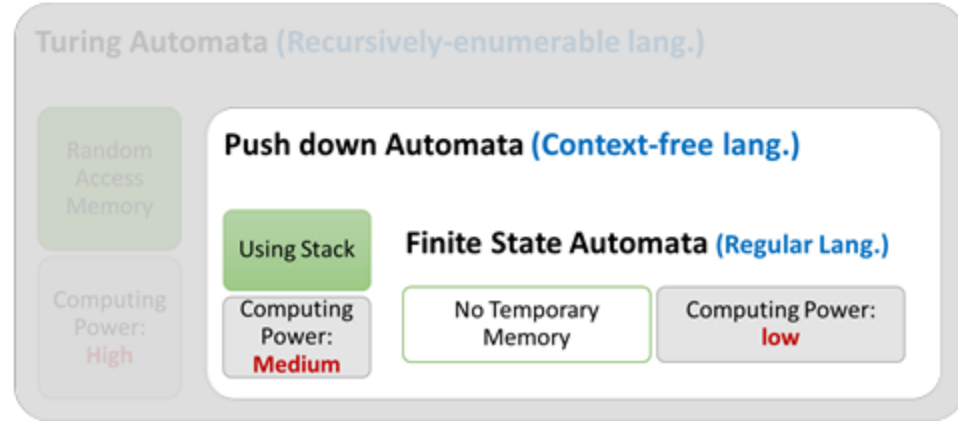
Chapter 3

Introduction

In chapter 2 :

- We learned that
  - There is a **more powerful method** of describing languages:
    - called context-free grammars (CFG)
  - Associated languages with CFG: context-free languages (CFL).
  - Associated automata: Pushdown automata
    - They have an unlimited memory that is usable only in the last in, first out manner of a stack.

- We found out
  - There are some **languages** that cannot be described using the method above
    - Non-context free languages
    - Example: $\{a^n b^n c^n \mid n \geq 0\}$



Chapter 3

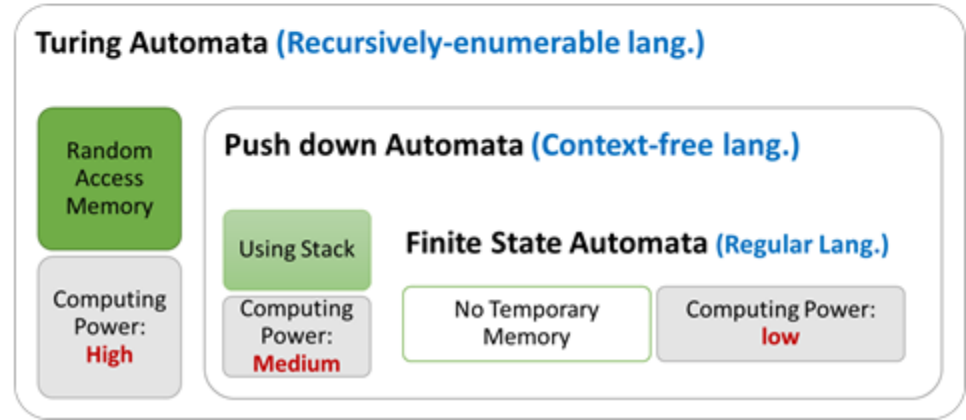
Introduction

In chapter 3 :

- We present a much **more powerful method** of describing languages:
  - called Turing machine
  - first proposed by Alan Turing in 1936

- Similar to a **finite automaton** but with an unlimited and unrestricted memory.

- Some languages that they are beyond the capabilities of FA and PDA models can be described using Turing machine.



Chapter 3 : The Church–Turing Thesis

3.1 Turing Machines

Formal definition of a Turing machine

Examples of Turing machines

3.2 Variants of Turing Machines

Multitape Turing machines

Nondeterministic Turing machines

Enumerators

Equivalence with other models

3.3 The Definition of Algorithm

Hilbert's problem

Terminology for describing Turing machines

Introduction

- In 1936 :
  - Church's Lambda calculus
  - Kleene's(general) recursive functions
  - Turing machines
  - Post's production systems

- All these systems were proved equivalent.
- Church and Turing then proposed equivalent theses. "**Church's Thesis** " , "**Turing's Thesis**,", or the "**Church-Turing Thesis**.

Introduction

Church-Turing Thesis: is a hypothesis about the nature of computable functions.

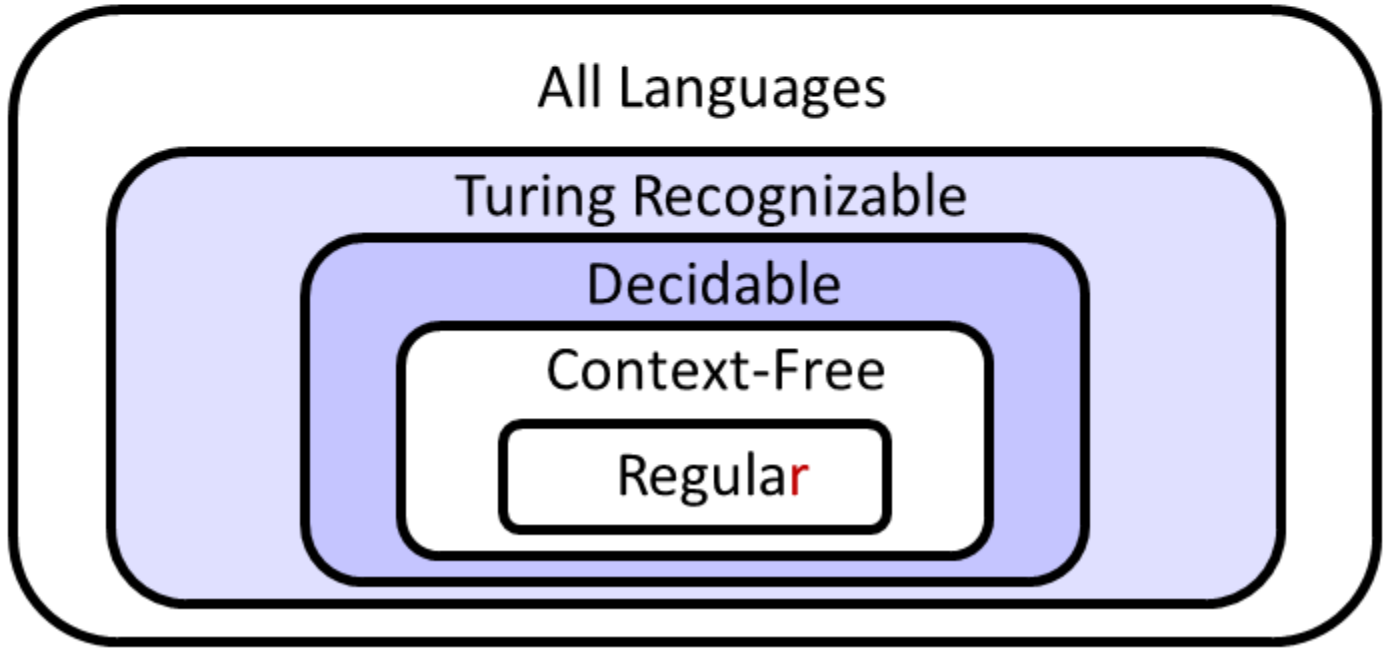
- It states that

> a function on the natural numbers is computable by a human being following an algorithm, ignoring resource limitations, if and only if it is computable by a Turing machine.

A Turing machine

- is a much more accurate model of a **general purpose computer**.
- can do everything that a real computer can do.

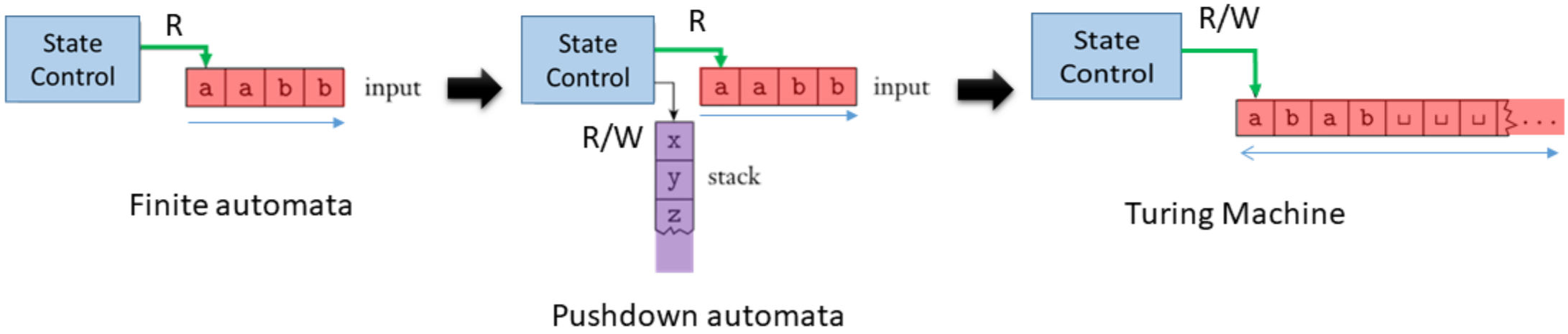
- Even a Turing machine cannot solve certain problems.
  - these problems are beyond the theoretical limits of computation.



TURING MACHINES

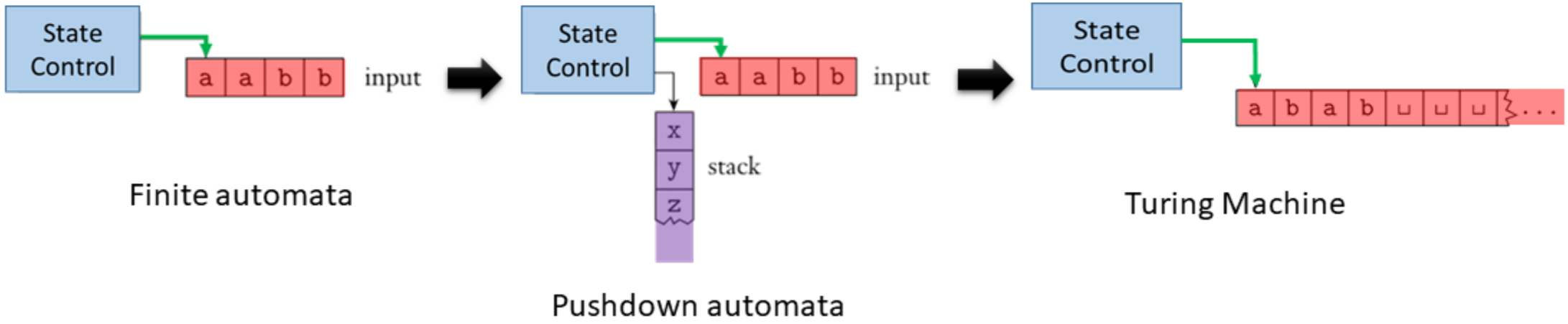
- **A Turing machine model**
  - uses an infinite tape (in one direction) as its **unlimited memory**.
  - has a tape head that can read and write symbols and move around on the tape.

- Initially the **tape** contains only the input string and is blank everywhere else.



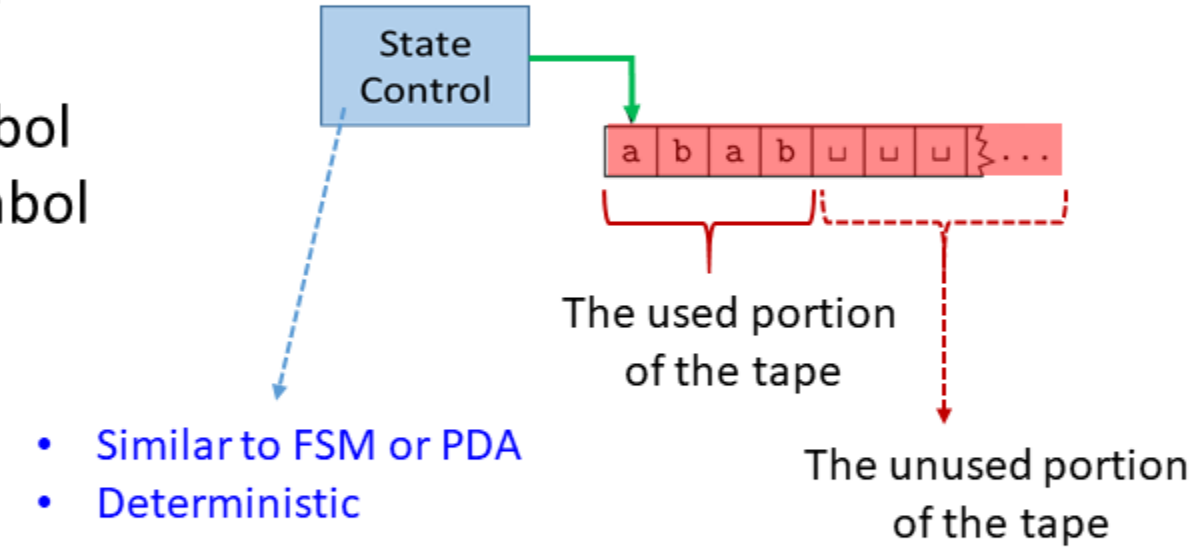
TURING MACHINES

- **Tape Alphabet**
  - Typical: $\sum = \{0,1\}$
  - Common: $\sum = \{0,1,a,b,x,y,\#,\$\}$
  - The "Blank" symbol is special and $\sqcup \notin \Sigma$.



TURING MACHINES

- **The Tape head**
  - shows the current position
  - initially at the leftmost square
  - can move left or right
  - can read the current symbol
  - can write the current symbol

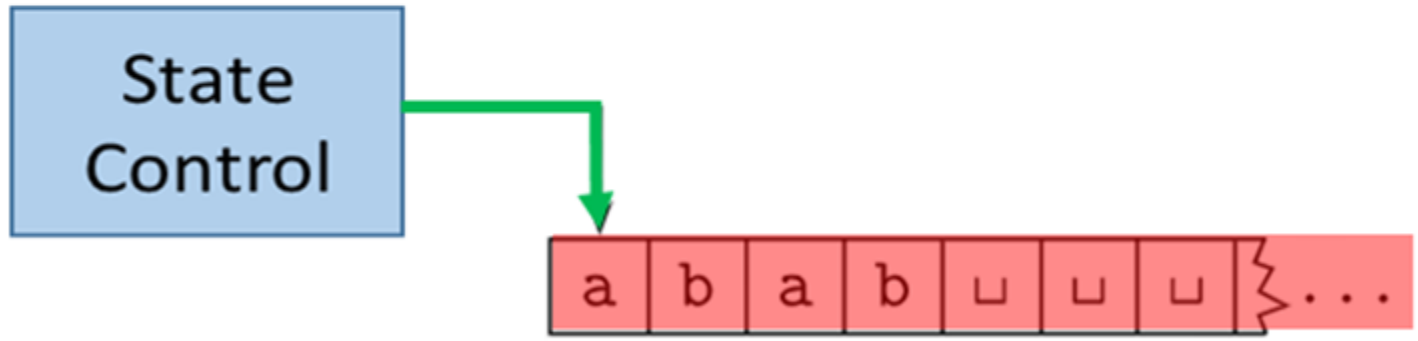


- Similar to FSM or PDA
- Deterministic

TURING MACHINES

The differences between finite automata and Turing machines

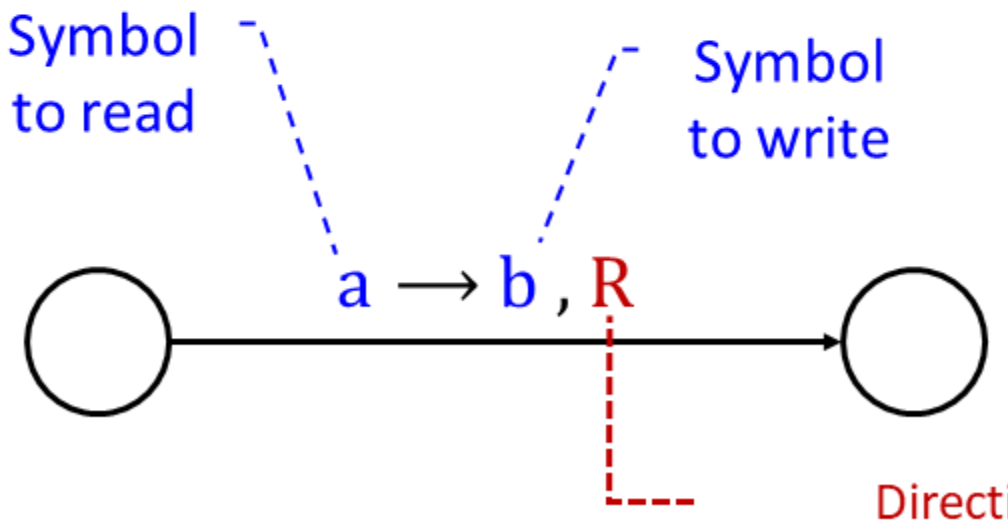
1. A Turing machine can both write on the tape and read from it.
2. The **read–write head** can move both to the left and to the right.
3. The **tape** is infinite.
4. The special states for rejecting and accepting take effect immediately.



Rules of Operation

At each step of the computation:

1. Read the current symbol
2. Update (write) the same cell.
3. Move exactly one cell either left or right.
   - At the left end of the tape, there is no move left and the head will stay at left end.



$0 \longrightarrow 0 , R$

No update and move to right

State Control

- is similar to FSM.
- It has
  - A **unique** Initial state
  1. Exactly two final states
     1. The "accept" state
     2. The "reject" state

Computation can

- Halt and accept
  - Whenever the machine enters the accept state, computation immediately Halts.
- Halt and reject
  - Whenever the machine enters the reject state, computation immediately Halts.
- Loop
  - The machine fails to Halt.

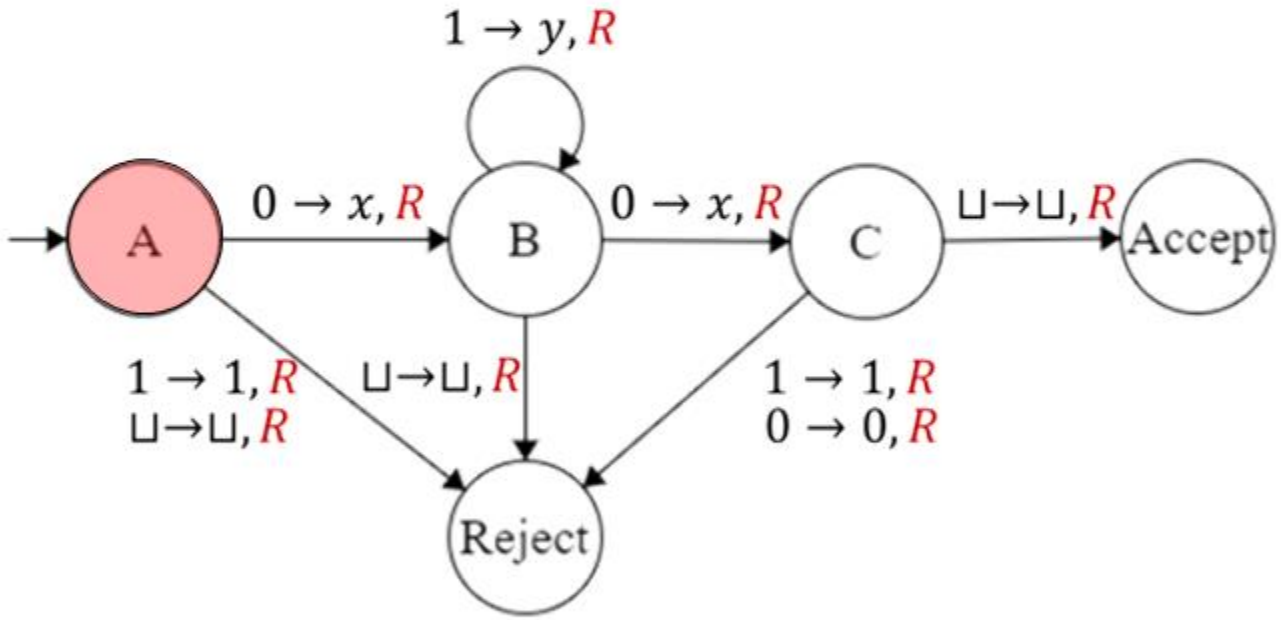
State Control

- is similar to FSM.
- It has
  - A **unique** Initial state
  - Exactly **two final states**
    1. The “accept” state
    2. The “reject” state

Example 1

Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.

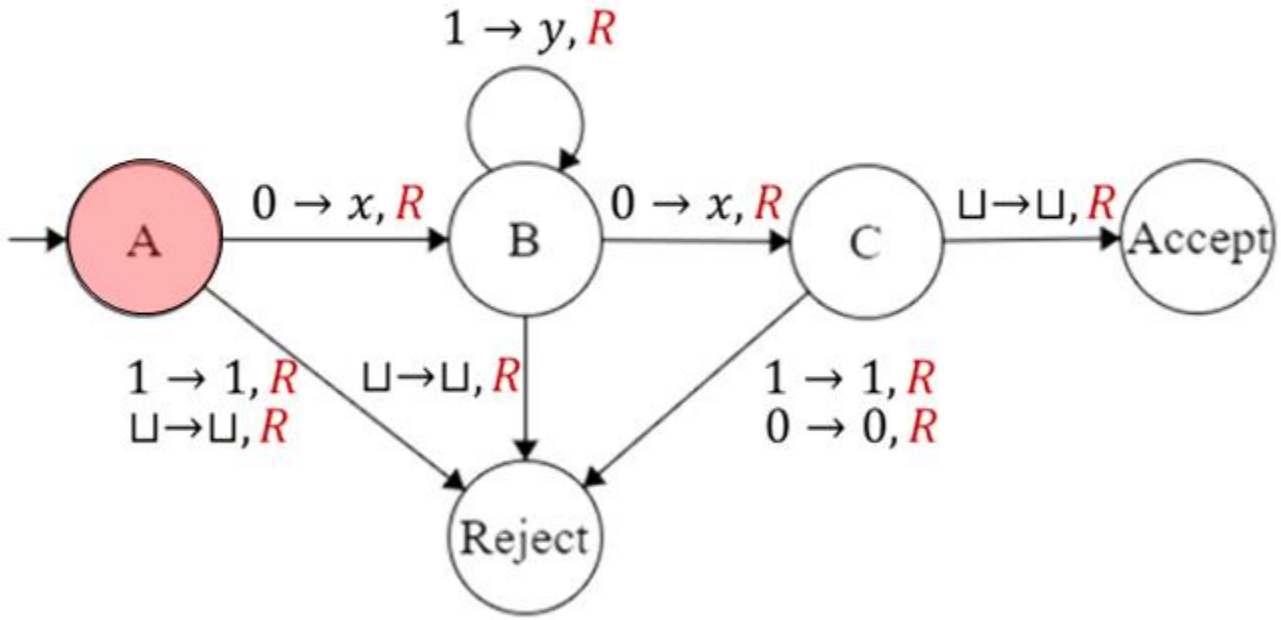
if w = 01110



If an edge is missing, assume it leads to **Reject**.

Example 1

Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.

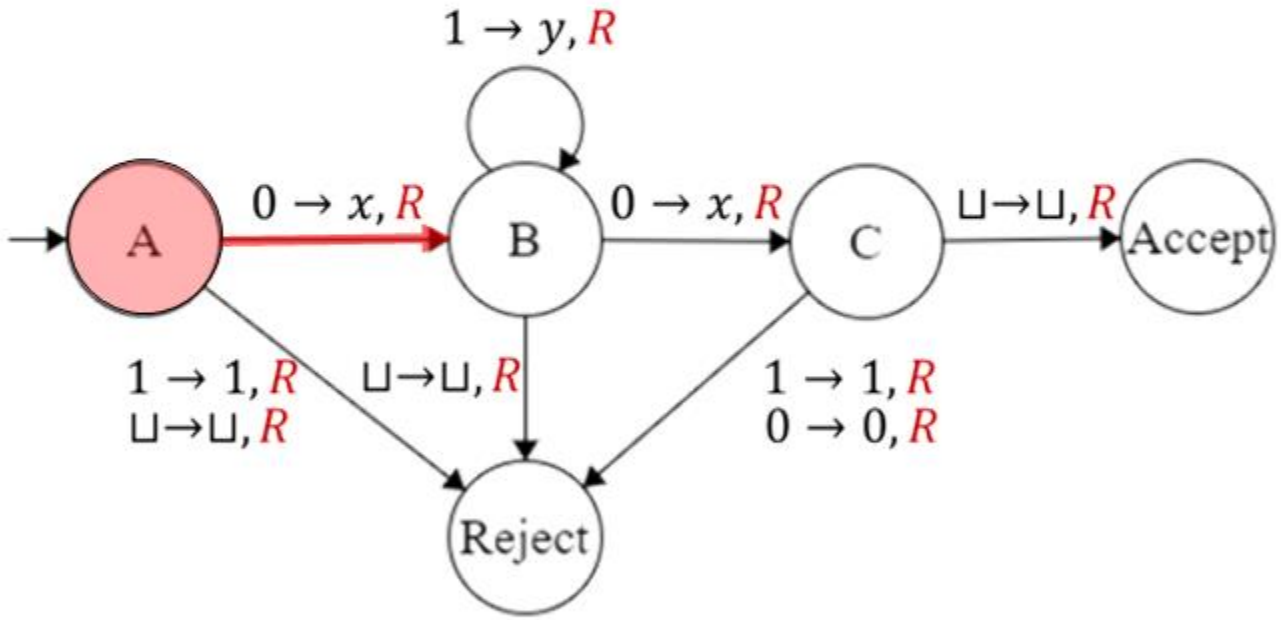


W

0	1	1	1	0	⊔	⊔	A

Example 1

Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.



0 ⟶ x , R

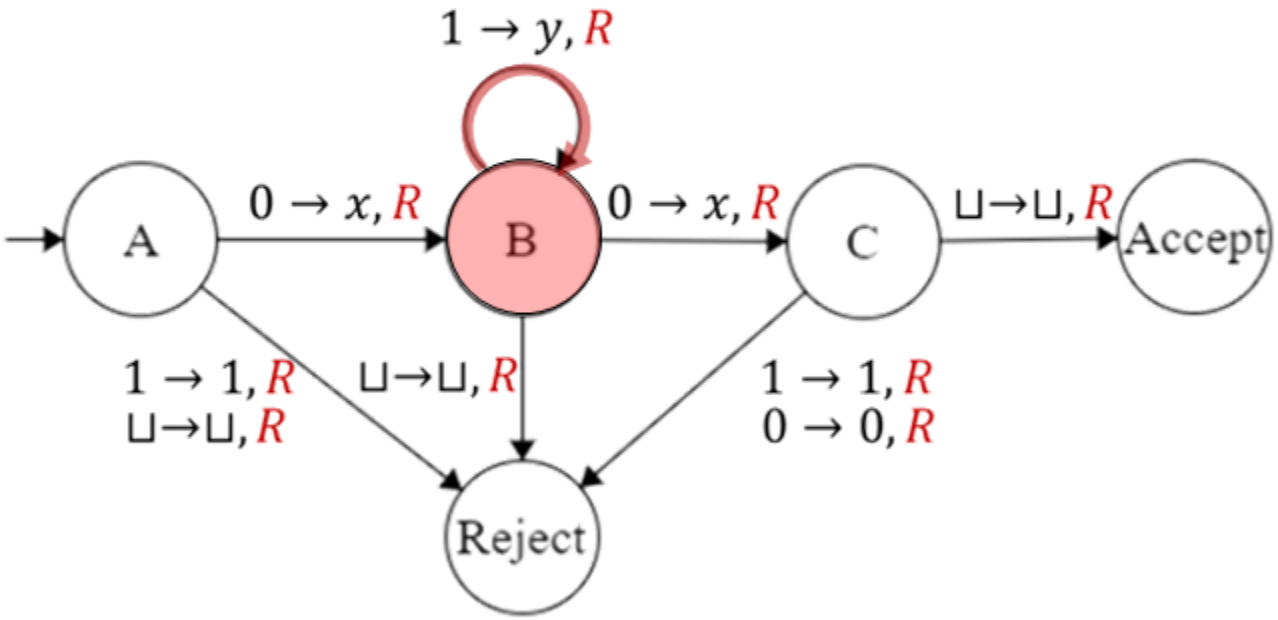
	w						
	0	1	1	1	0	⊔	⊔
	x	**1**	1	1	0	⊔	⊔

A

B

Example 1

Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.

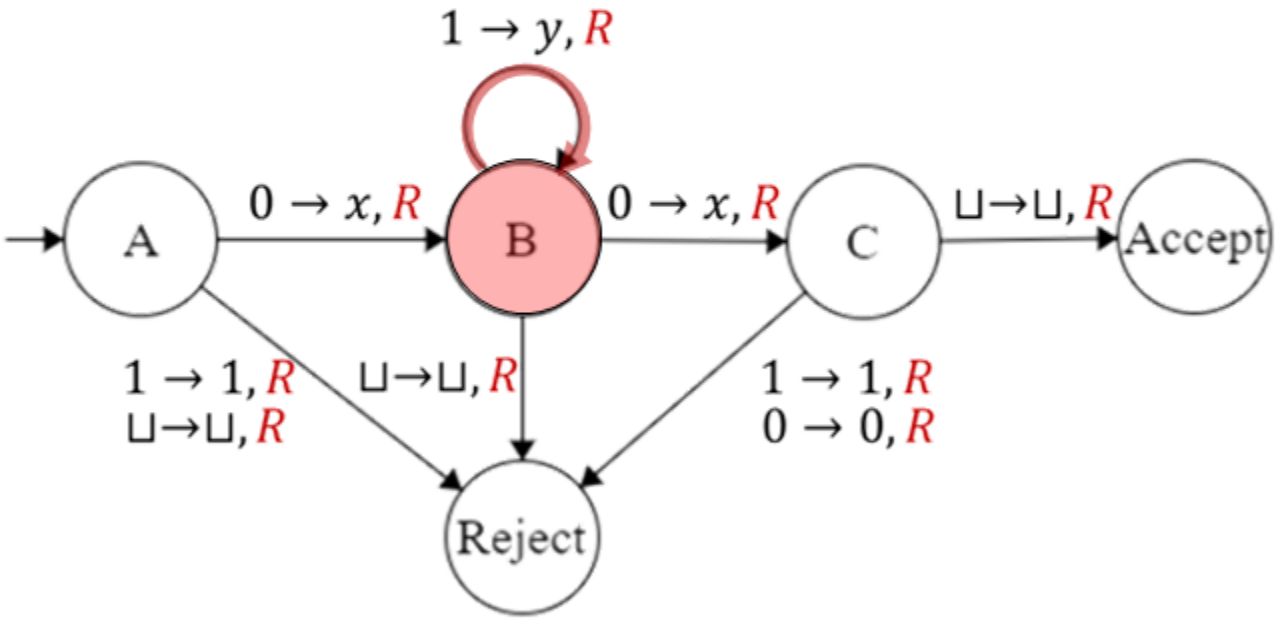


0 ⟶ x , R
1 ⟶ y , R

	w						
	0	1	1	1	0	⊔	⊔
A	x	**1**	1	1	0	⊔	⊔
B	x	y	**1**	1	0	⊔	⊔
B							

Example 1

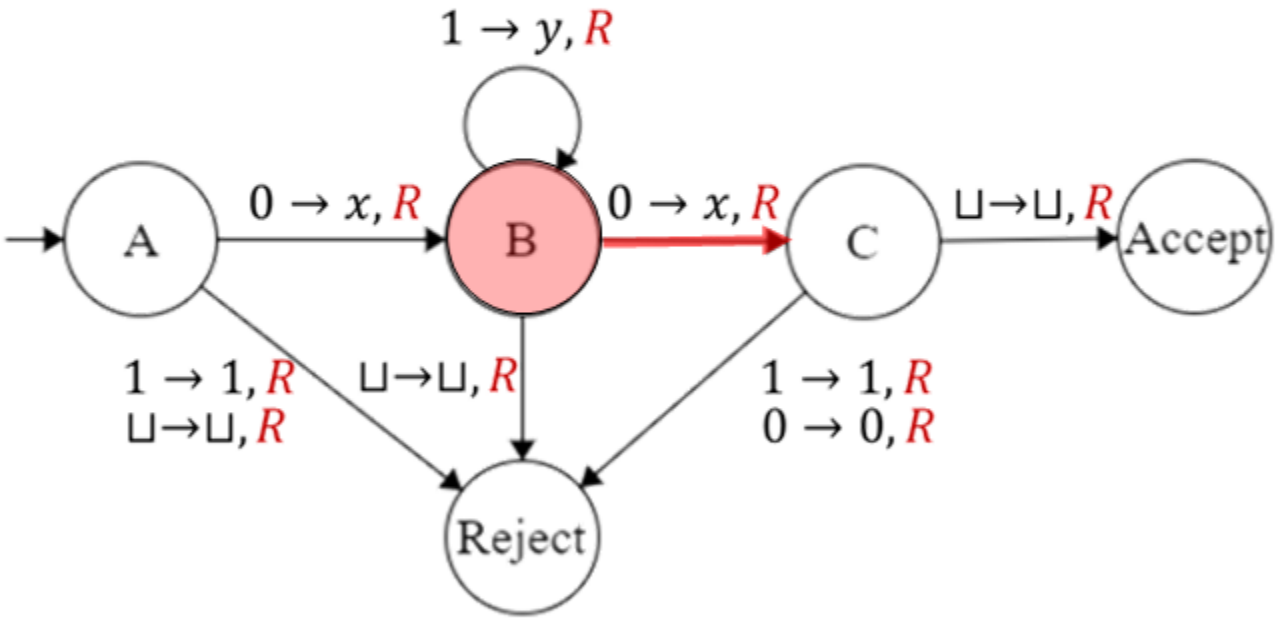
Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.



	w							
0 ⟶ x , R	0	1	1	1	0	⊔	⊔	A
1 ⟶ y , R	x	1	1	1	0	⊔	⊔	B
1 ⟶ y , R	x	y	**1**	1	0	⊔	⊔	B
	x	y	y	**1**	0	⊔	⊔	B

Example 1

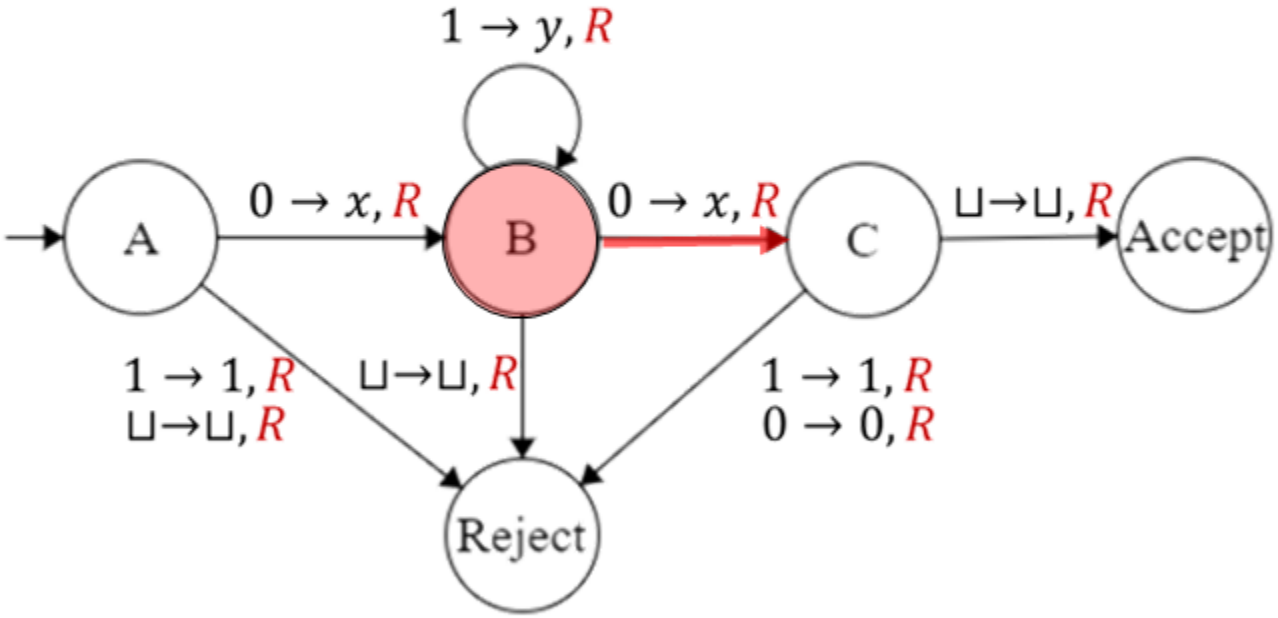
Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.



	w							
0 ⟶ x , R	0	1	1	1	0	⊔	⊔	A
1 ⟶ y , R	x	1	1	1	0	⊔	⊔	B
1 ⟶ y , R	x	y	1	1	0	⊔	⊔	B
1 ⟶ y , R	x	y	y	**1**	0	⊔	⊔	B
	x	y	y	y	**0**	⊔	⊔	B

Example 1

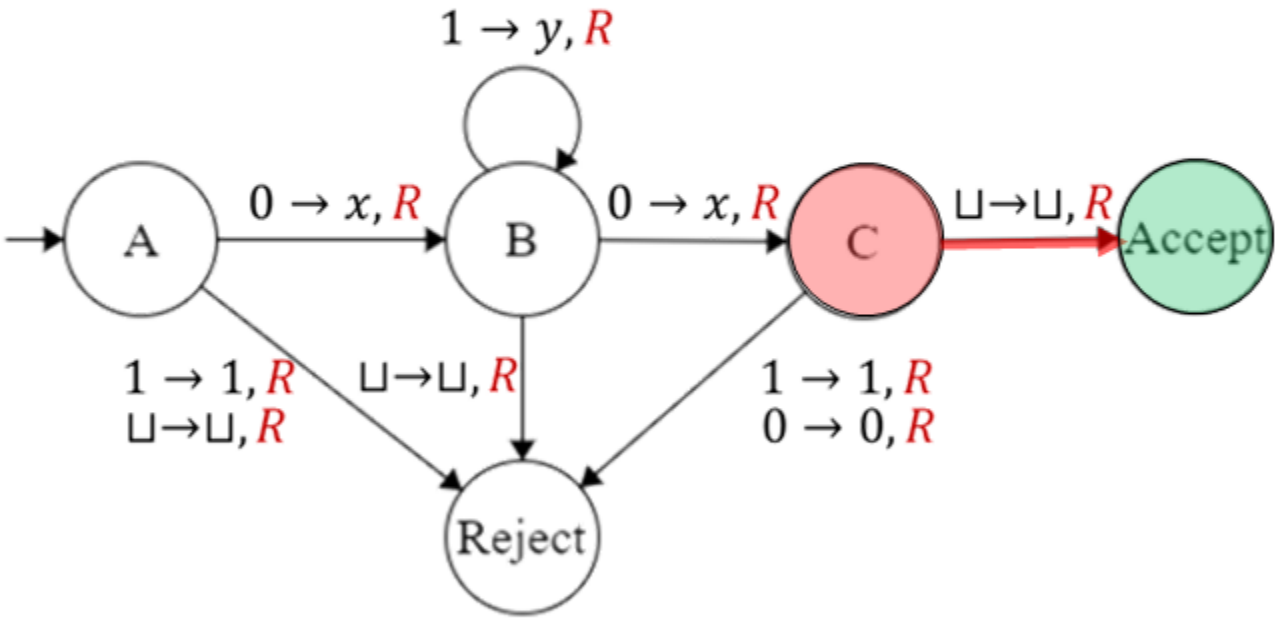
Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.



	w							
0 ⟶ x , R	0	1	1	1	0	⊔	⊔	A
1 ⟶ y , R	x	1	1	1	0	⊔	⊔	B
1 ⟶ y , R	x	y	1	1	0	⊔	⊔	B
1 ⟶ y , R	x	y	y	1	0	⊔	⊔	B
0 ⟶ x , R	x	y	y	y	**0**	⊔	⊔	B
	x	y	y	y	x	⊔	⊔	C

Example 1

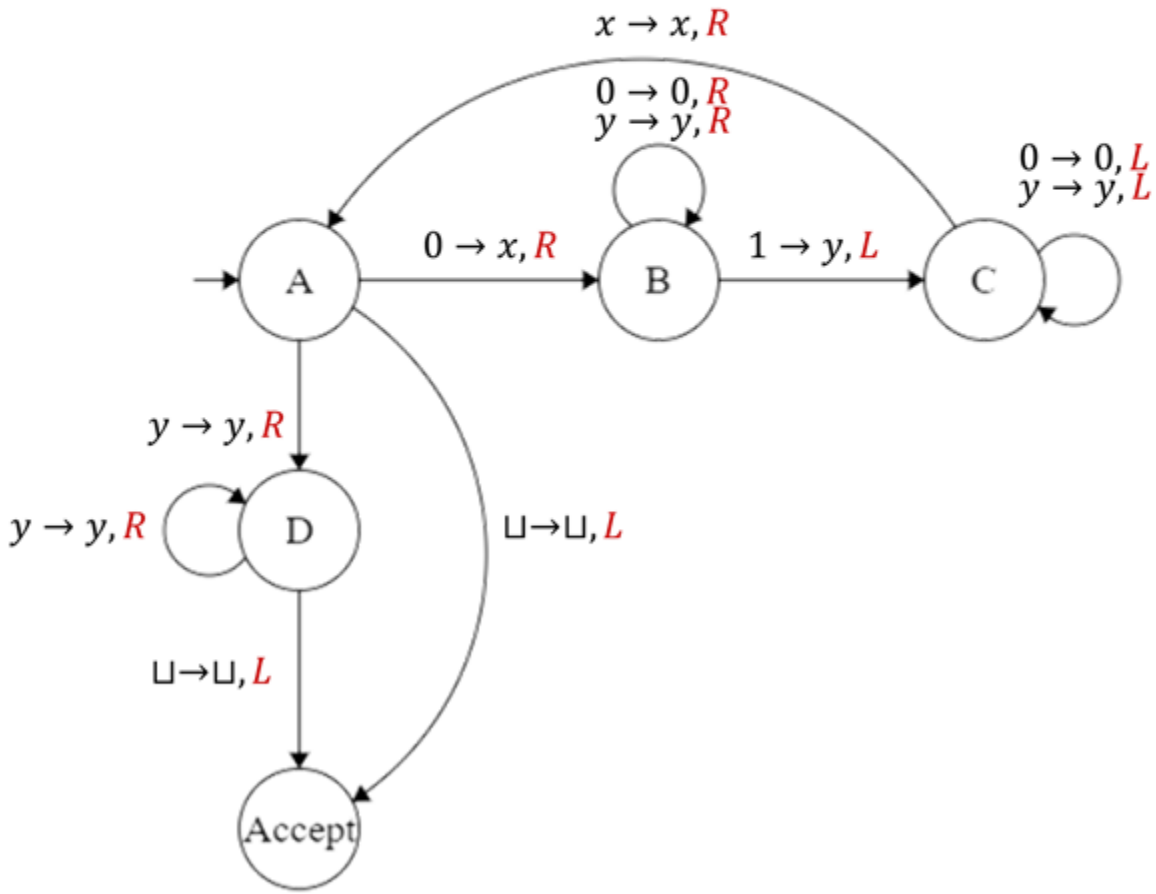
Show that the following Turing machine M1 can be used for testing membership in the language $A = \{w = 01^*0 | w \in \{0,1\}^*\}$.



				w				
0 ⟶ x , R	0	1	1	1	0	⊔	⊔	A
1 ⟶ y , R	x	1	1	1	0	⊔	⊔	B
1 ⟶ y , R	x	y	1	1	0	⊔	⊔	B
1 ⟶ y , R	x	y	y	1	0	⊔	⊔	B
0 ⟶ x , R	x	y	y	y	**0**	⊔	⊔	B
⊔ ⟶ ⊔, R	x	y	y	y	x	⊔	⊔	C
Accept	x	y	y	y	x	⊔	⊔	

Example 2

The following Turing machine M2 has been designed for testing membership in the language $B = \{w = 0^n 1^n | w \in \{0,1\}^* , n > 0\}$.



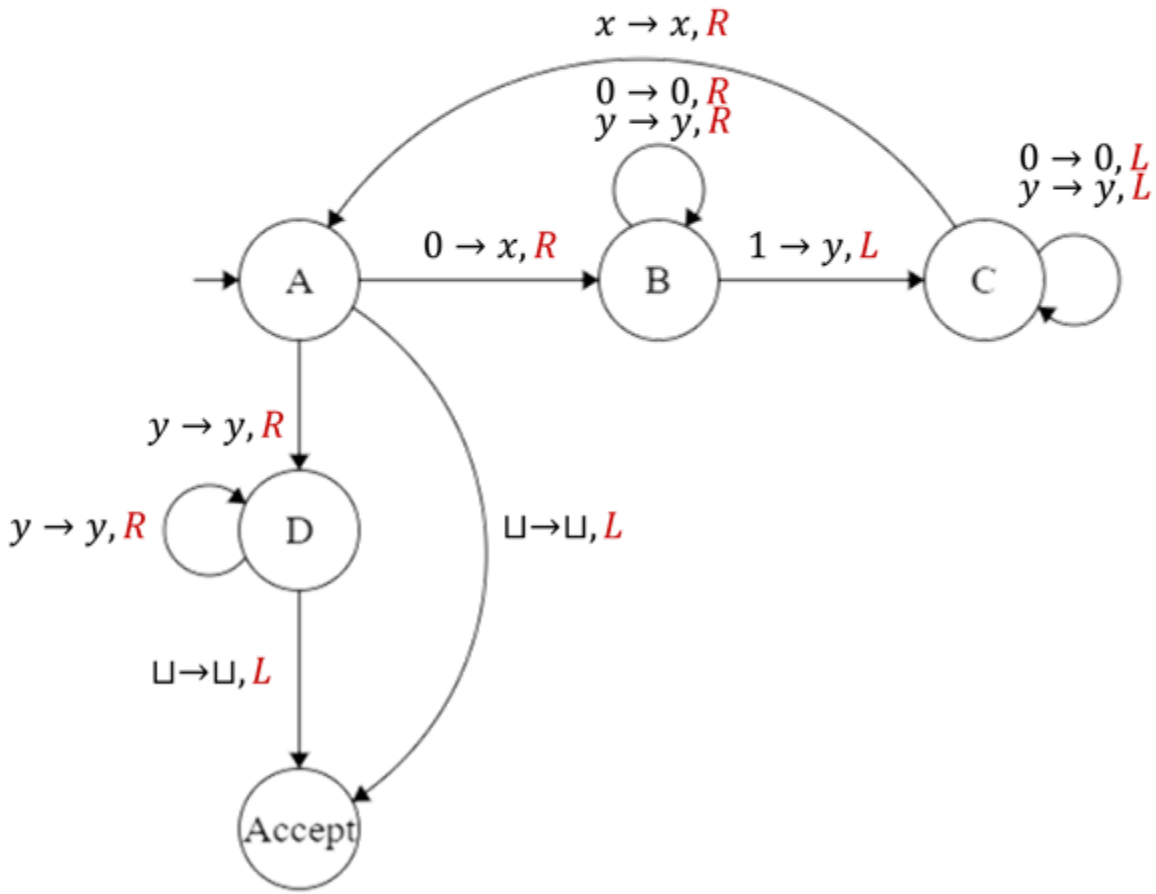
Algorithm:

- Change 0 to x
- Move right to first 1
  - If none: Reject
- Change 1 into y
- Move left to leftmost 0
- Repeat until no more 0s
- Make sure no more 1 remain.

- Does it work?
- Does it contain bugs?

Example 2

The following Turing machine M2 has been designed for testing membership in the language $B = \{w = 0^n 1^n | w \in \{0,1\}^*, n > 0\}$.



Computation history

0 0 0 0 1 1 1 1

X 0 0 0 1 1 1 1

X 0 0 0 Y 1 1 1

0 X 0 0 Y 1 1 1

⇓*

X X X X Y Y Y Y

X X X X Y Y Y Y

Example 3

- Construct a Turing machine M1 for testing membership in the language

$$B = \{w\#w | w \in \{0,1\}^*\}.$$

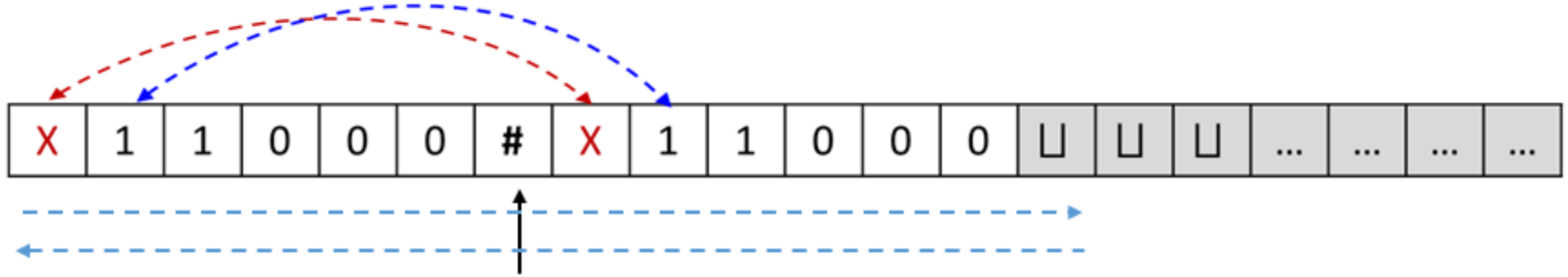
- The TM M1 should accept if its input is a member of B and reject otherwise.

Example 3

$$B = \{w\#w | w \in \{0,1\}^*\}.$$

- The input is too long for you to remember it all,
- But you are allowed to move back and forth over the input and make marks on it.

- **Strategy:**
  - zig-zag to the corresponding places on the two sides of the # and determine whether they match.
  - Place marks on the tape to keep track of which places correspond.



TURING MACHINES : Example

- In summary, M1's algorithm is as follows. $B = \{w\#w | w \in \{0,1\}^*\}$.

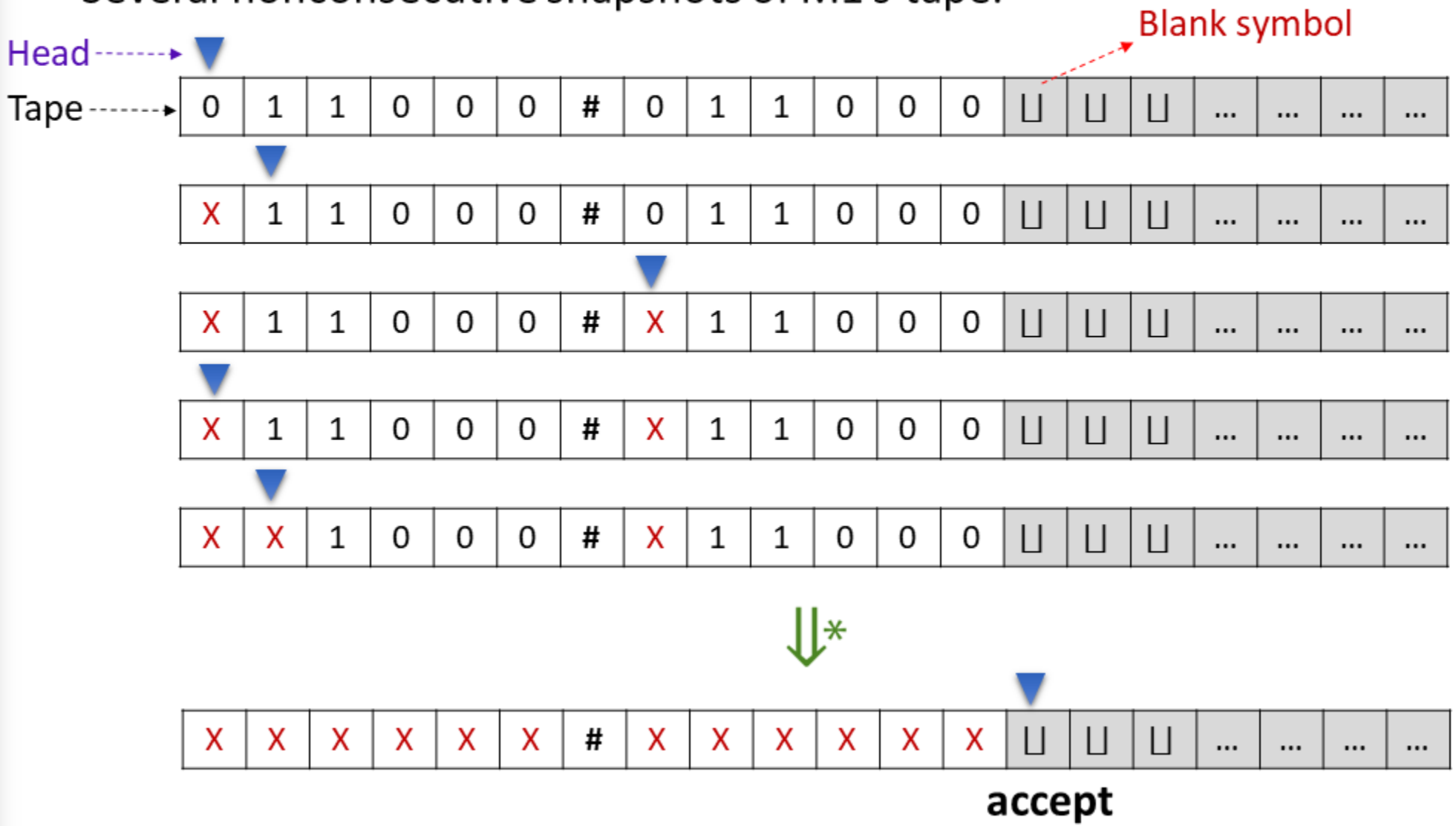
M_1 = "On input string w:

1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, *reject*. Cross off symbols as they are checked to keep track of which symbols correspond.
2. When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right of the #. If any symbols remain, *reject*; otherwise, *accept*."

0	1	1	0	0	0	#	0	1	1	0	0	0	⊔	⊔	⊔	...	...	...	...

TURING MACHINES : Example

- Several nonconsecutive snapshots of M1's tape.



FORMAL DEFINITION OF A TURING MACHINE

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

FORMAL DEFINITION OF A TURING MACHINE

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, 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*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,

FORMAL DEFINITION OF A TURING MACHINE

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, 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*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta\colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

$$\delta(q, a) = (r, b, \text{L})$$

COMPUTATION IN TURING MACHINE

A Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ computes as follows:

- **Initial Configuration:** Initially, M receives its input $w = w_1 w_2 \dots w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is blank.

- **Start :** The head starts on the **leftmost square** of the tape.
  - the first blank appearing on the tape marks the end of the input.

- **Computation process:** Once M has started,
  - the **computation proceeds according to the rules** described by the **transition function**.

COMPUTATION IN TURING MACHINE (Cont.)

- If M ever tries to move its head to the left off the left-hand end of the tape,
  - the head stays in the same place for that move, even though the transition function indicates L.

- Halt: The computation continues **until**
  - it enters either the accept or reject states, at which point it halts.
  - If neither occurs, M goes on forever. (**LOOP**)

CONFIGURATIONOF THE TURING MACHINE

- As a Turing machine computes, changes occur in
  - the current state,
  - the current tape contents, and
  - the current head location.

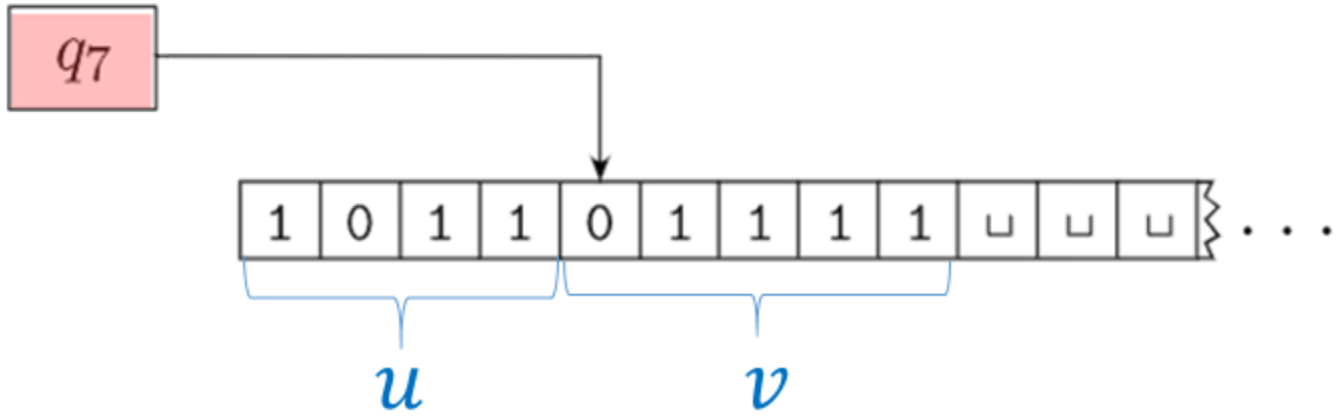
- A **setting of these three items** is called a configuration of the Turing machine.

CONFIGURATIONOF THE TURING MACHINE

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

- Example: $1011\boldsymbol{q_7}\mathbf{0}1111$

u v



CONFIGURATIONOF THE TURING MACHINE

- Suppose that we have
  - Tape symbols a, b,and c in Γ,
  - Strings u and v in Γ^*
  - states q_i and q_j.
  - two configurations $C_1 = uaq_ibv$ and $C_2 = uq_jacv$

- Say that
- if (for a leftward move)

uaq_ibv yields uq_jacv $\delta(q_i, b) = (q_j, c, L)$.

- Say that
- if (for a rightward move)

uaq_ibv yields $uacq_jv$ $\delta(q_i, b) = (q_j, c, R)$.

$C_1 \rightarrow C_2$

CONFIGURATIONOF THE TURING MACHINE

- **Special cases**: when the head is at one of the ends of the configuration.
  - **the left-hand end:**
    - if the transition is left-moving : $q_i b\boldsymbol{v}$ yields $q_j c\boldsymbol{v}$
      - prevent the machine from going off the left-hand end of the tape
    - if the transition is right-moving : $q_i bv$ yields $cq_j v$.

CONFIGURATIONOF THE TURING MACHINE

- **Special cases**: when the head is at one of the ends of the configuration.
  - **the right-hand end:**
    - if the transition is left-moving :
      - we can handle this case as before
    - if the transition is right-moving : uaq_i is equivalent to $uaq_i\sqcup$.
      - we can handle this case as before with the head no longer at the right-hand end

CONFIGURATIONOF THE TURING MACHINE

Different configurations of TM M on string w

- Start configuration $(q_0 w)$:
  - indicates that the machine is in the **start state** q_0 with its head at the leftmost position on the tape configuration.
- Accepting configuration:
  - The state of the configuration is q_{accept}
- Rejecting configuration:
  - state of the configuration is q_{reject}
- Halting configurations:
  - Accepting and rejecting configurations

CONFIGURATIONOF THE TURING MACHINE

A Turing machine M accepts input 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.

$$C_1, C_2, \ldots, C_k$$

Start configuration

Accept configuration

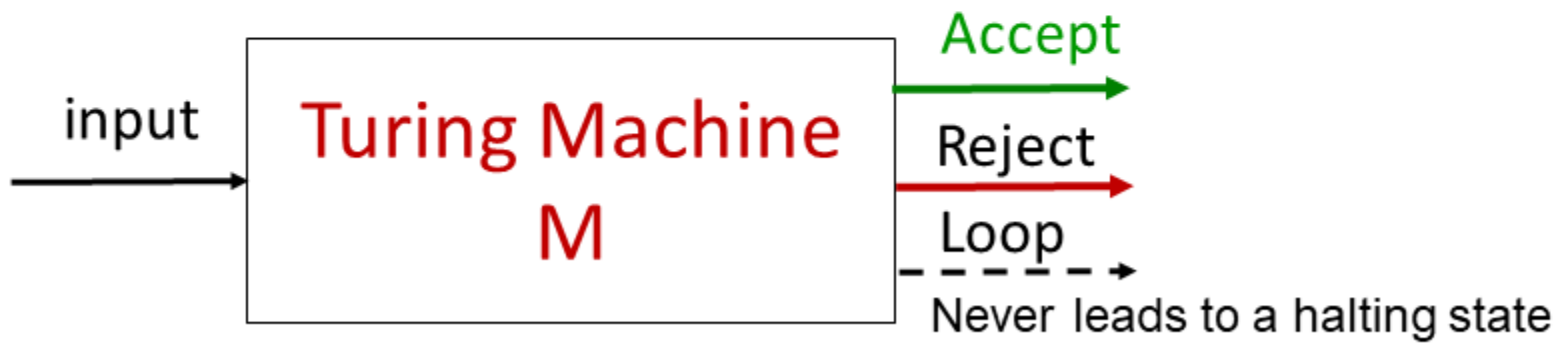
Definition: Turing-recognizable

- The collection of strings that TM M accepts is the language of M, or the language recognized by M, denoted $L(M)$.

Definition: Turing-recognizable

- The collection of strings that TM M accepts is the language of M, or the language recognized by M, denoted $L(M)$.

- Call a language Turing-recognizable if some Turing machine recognizes it.
  - Also called : recursively enumerable language



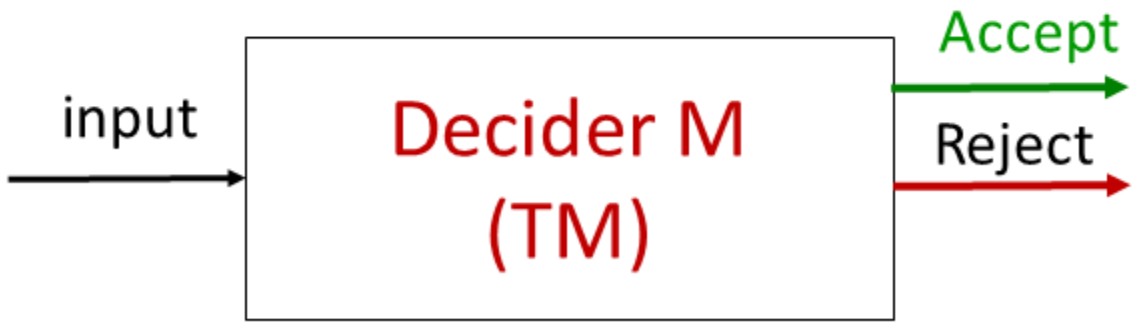
Definition: Decider

- we **prefer** Turing machines that halt on all inputs;
  - such machines never loop.

Definition: Decider

- 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.

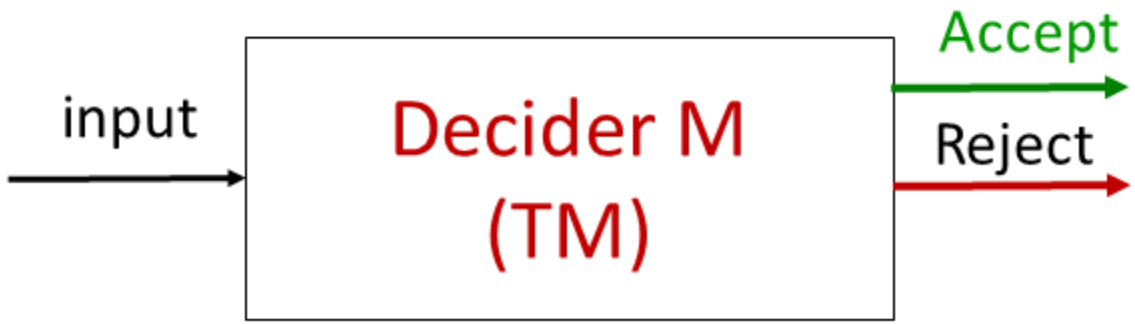


Definition: Decider

- 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.

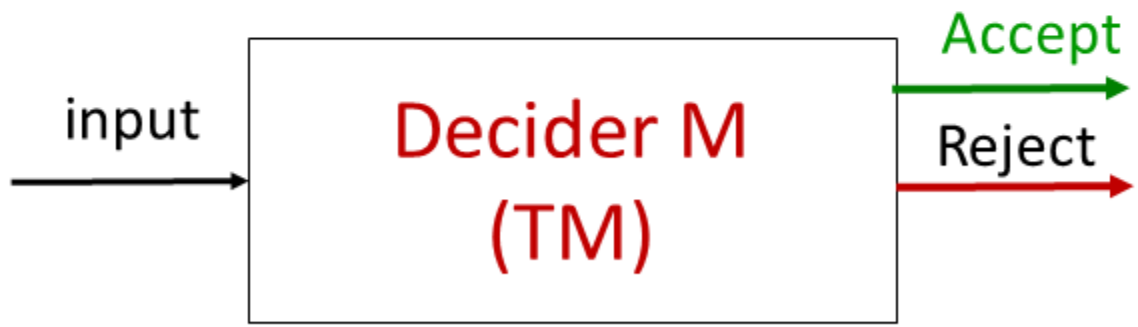
- A decider that recognizes some language also is said
  - to decide that language.



Definition: Turing-decidable

- Call a language Turing-decidable or simply decidable
  - if some Turing machine decides it.
  - Also called : recursive language

- Every decidable language is Turing-recognizable.



Example 4

- The following Turing machine (TM) $M2$ decides $A = \{0^{2^n} | n \geq 0\}$,
  - the language consisting of all strings of 0s whose length is a power of 2.

Example 4

- The following Turing machine (TM) $M2$ decides $A = \{0^{2^n} | n \geq 0\}$,
  - the language consisting of all strings of 0s whose length is a power of 2.

M_2 = "On input string w:

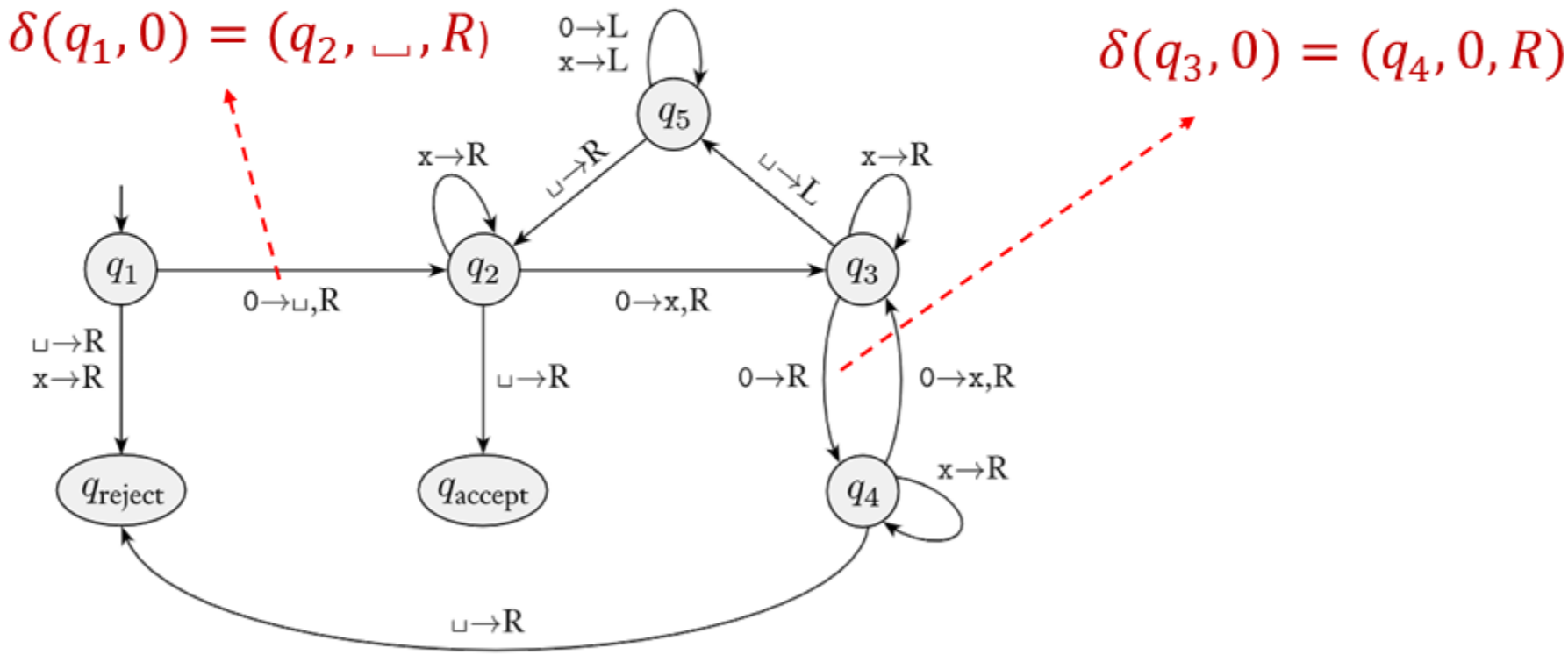
1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, *accept*.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, *reject*.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1."

0	0	0	0	0	0	0	0	⊔	⊔	⊔	...
⊔	X	0	X	0	X	0	X	⊔	⊔	⊔	...

(cuts the number of 0s in half.)

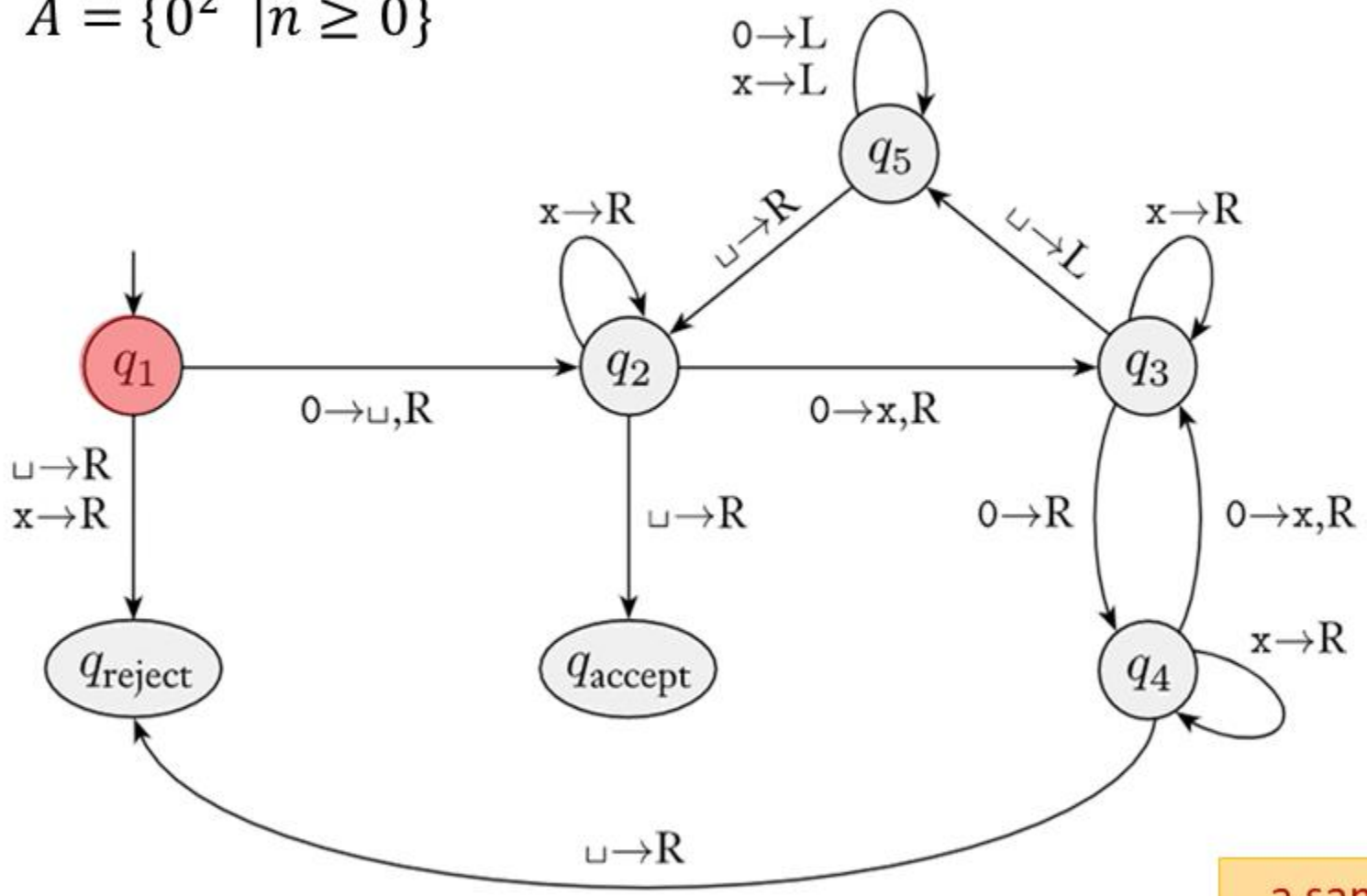
Example 4

- $A = \{0^{2^n} | n \geq 0\}$,



Example 4

- $A = \{0^{2^n} | n \geq 0\}$

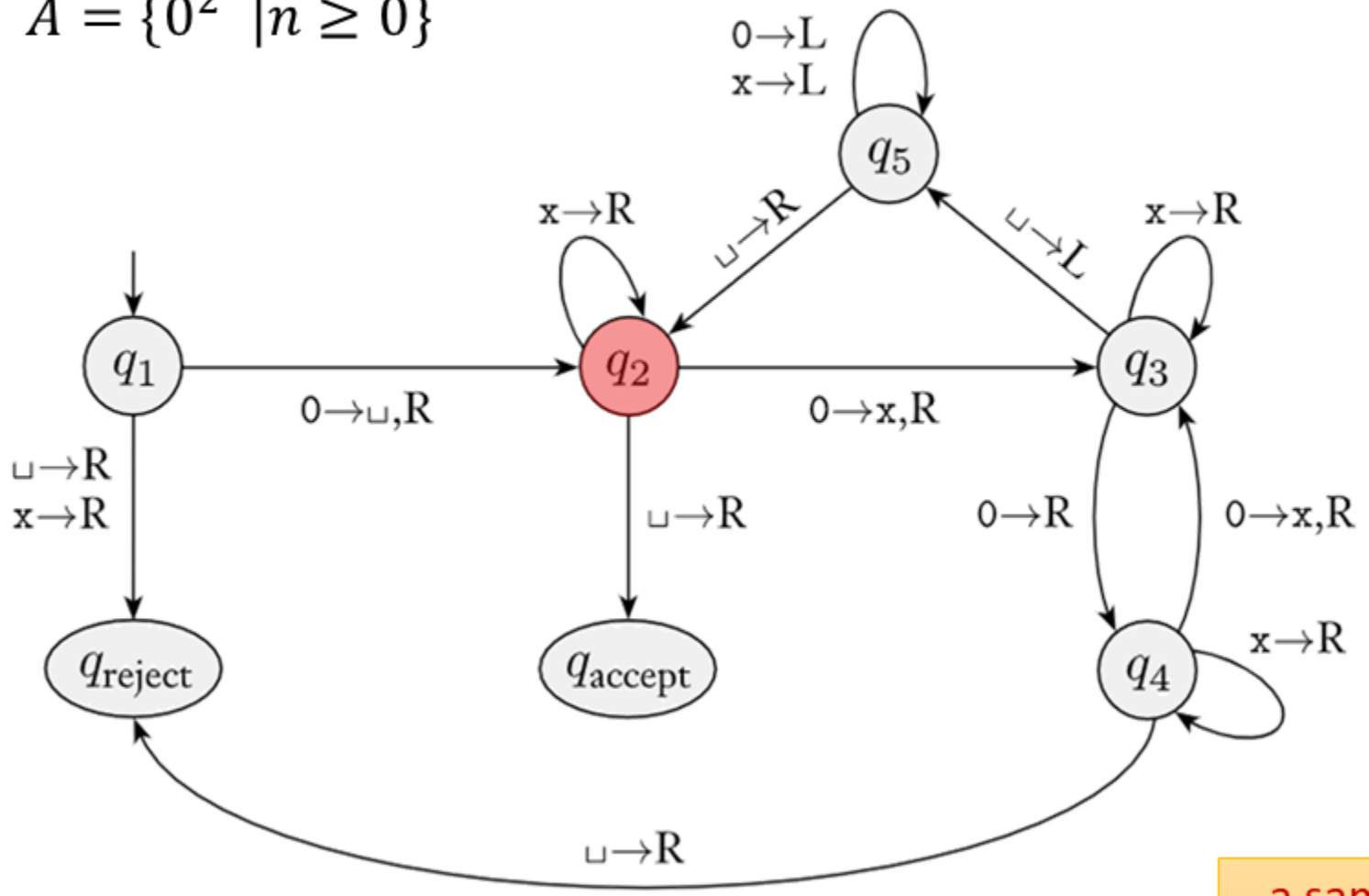


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

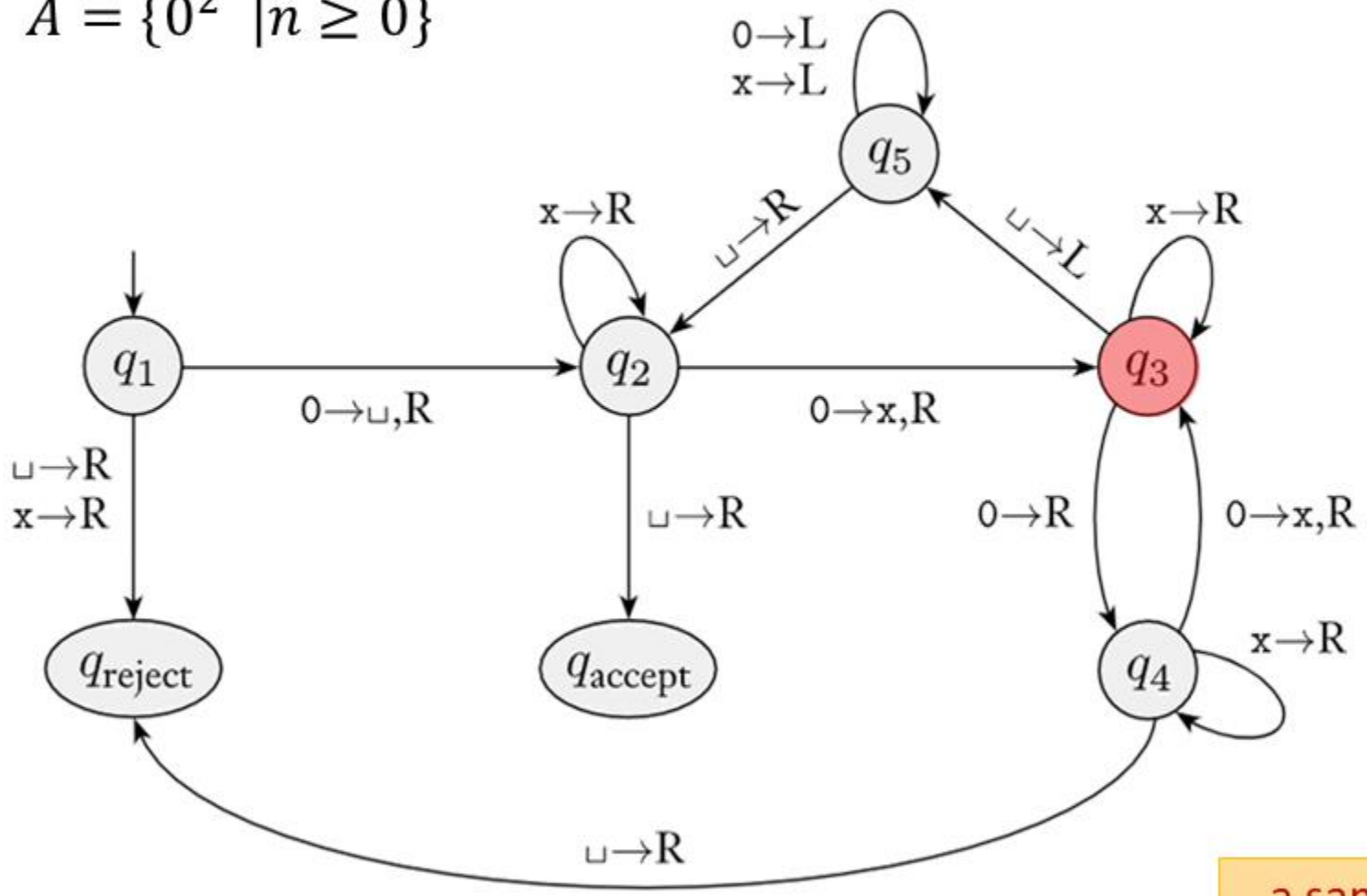


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

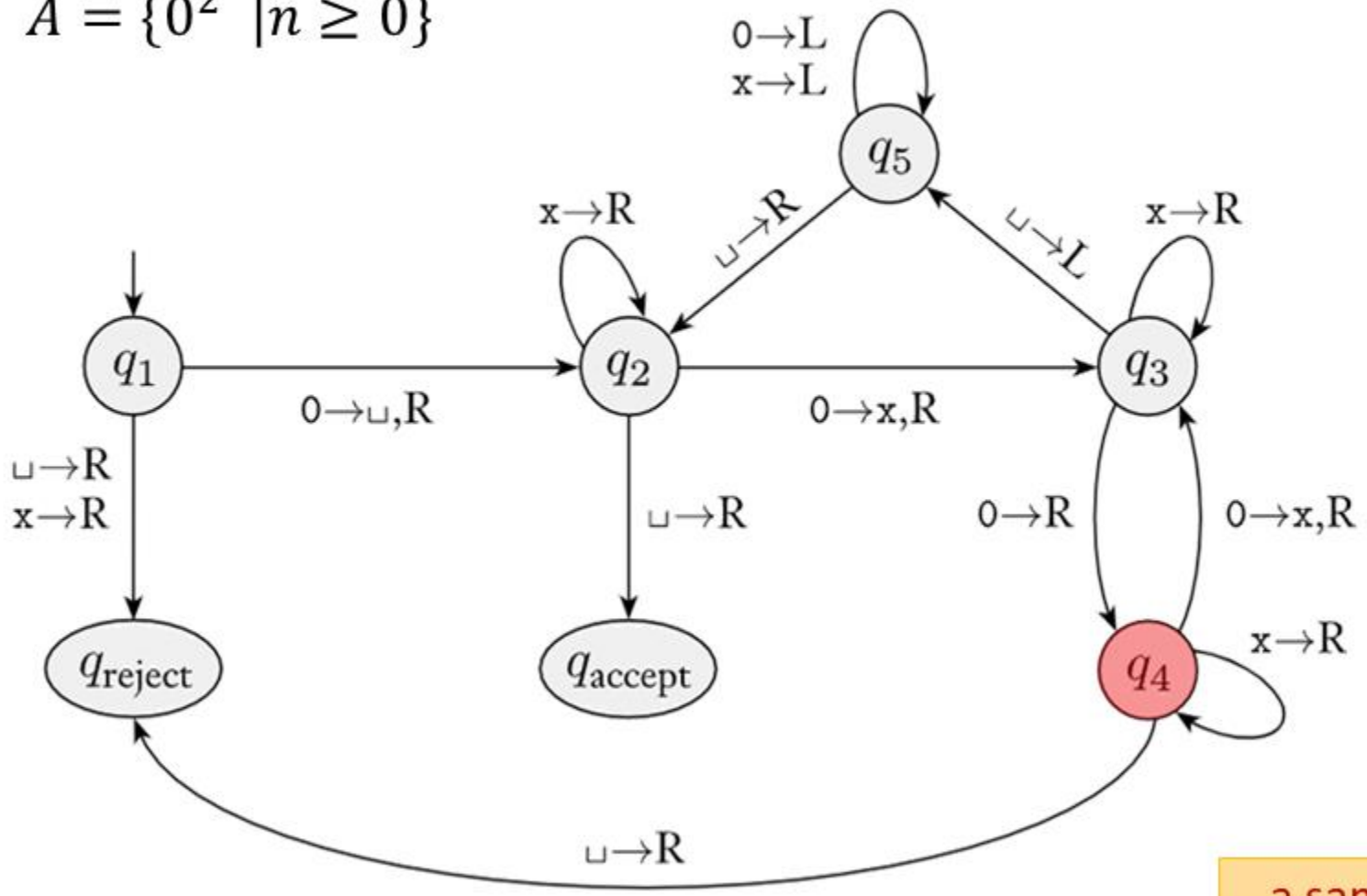


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

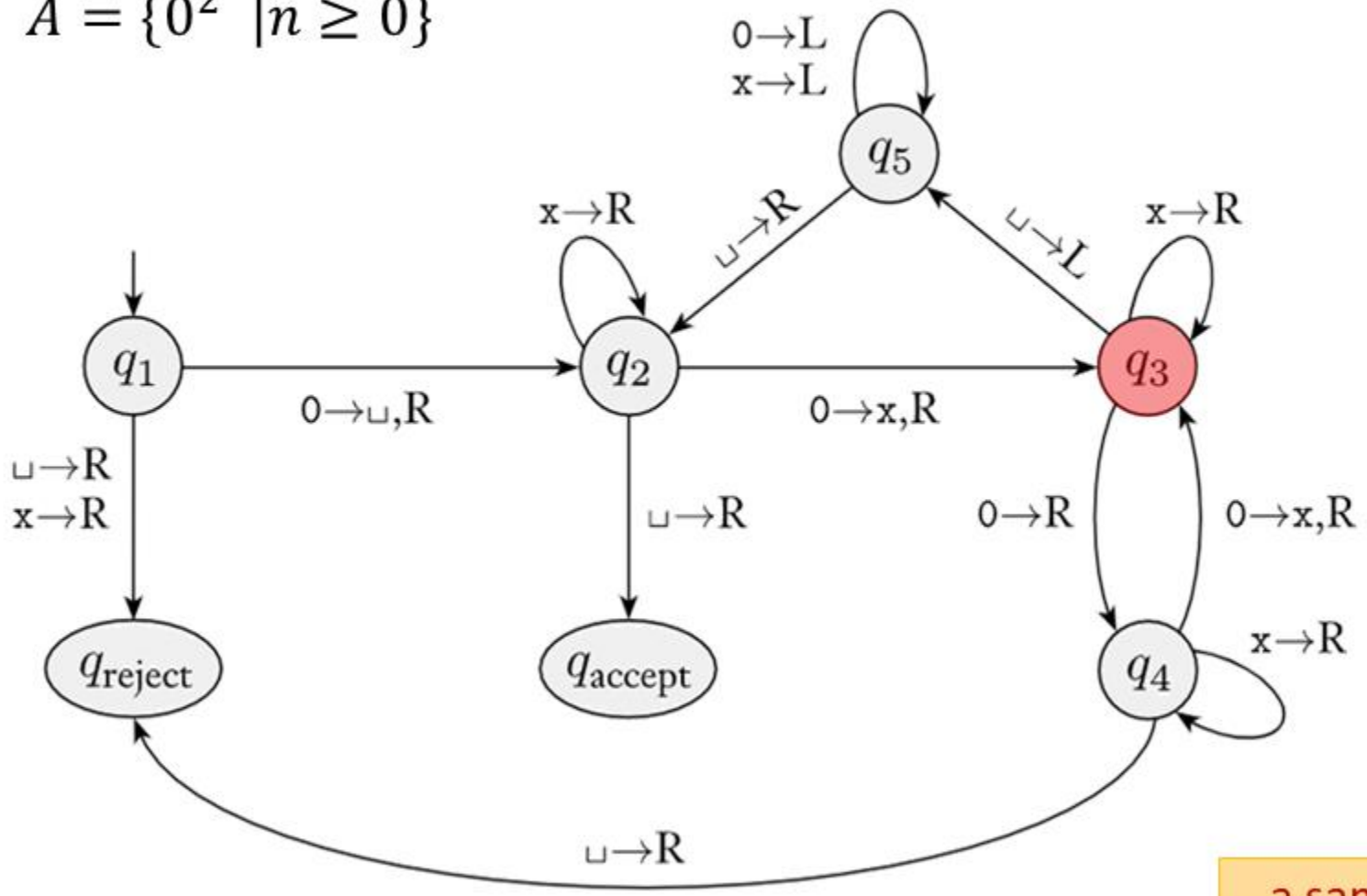


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

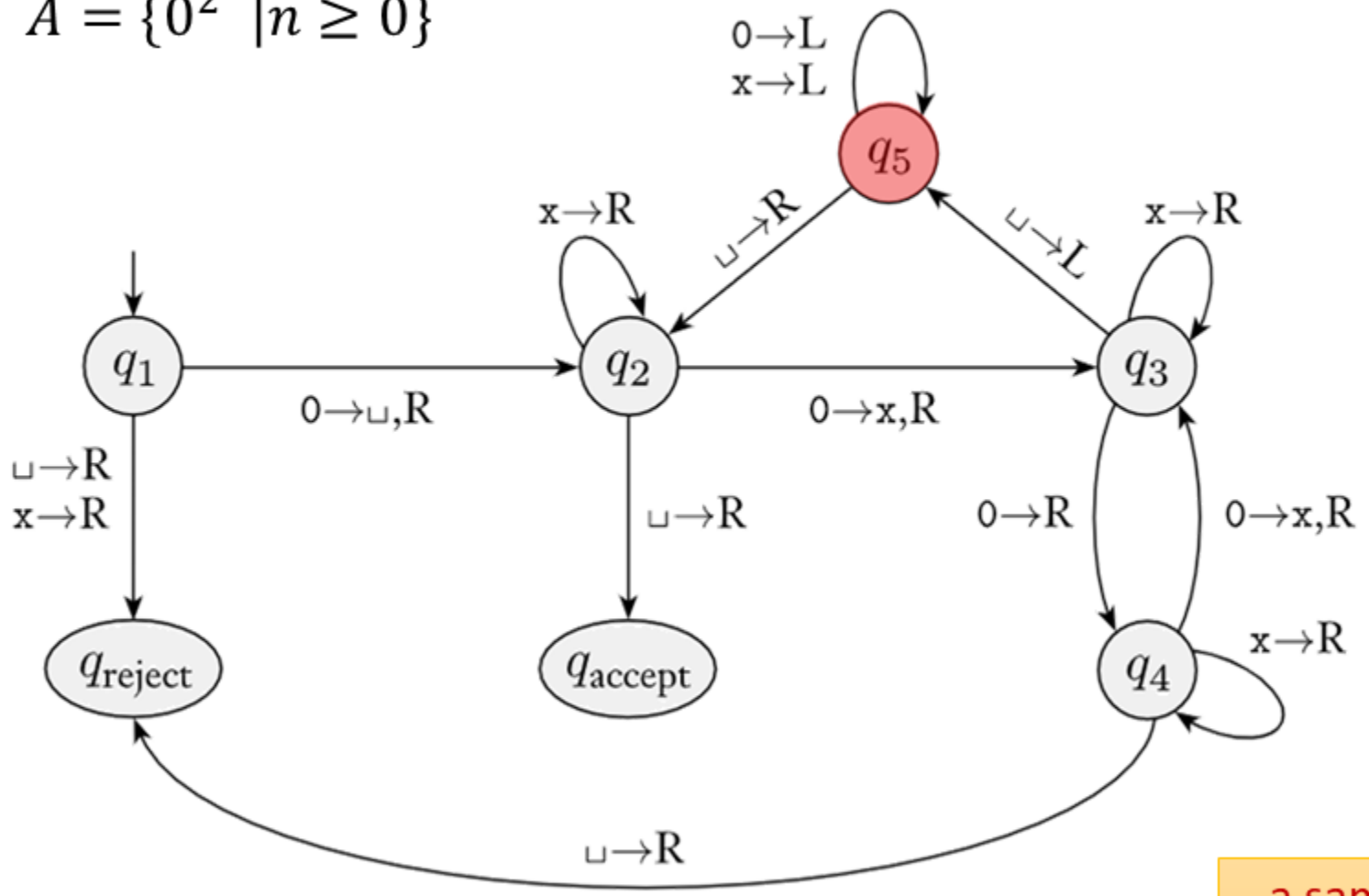


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

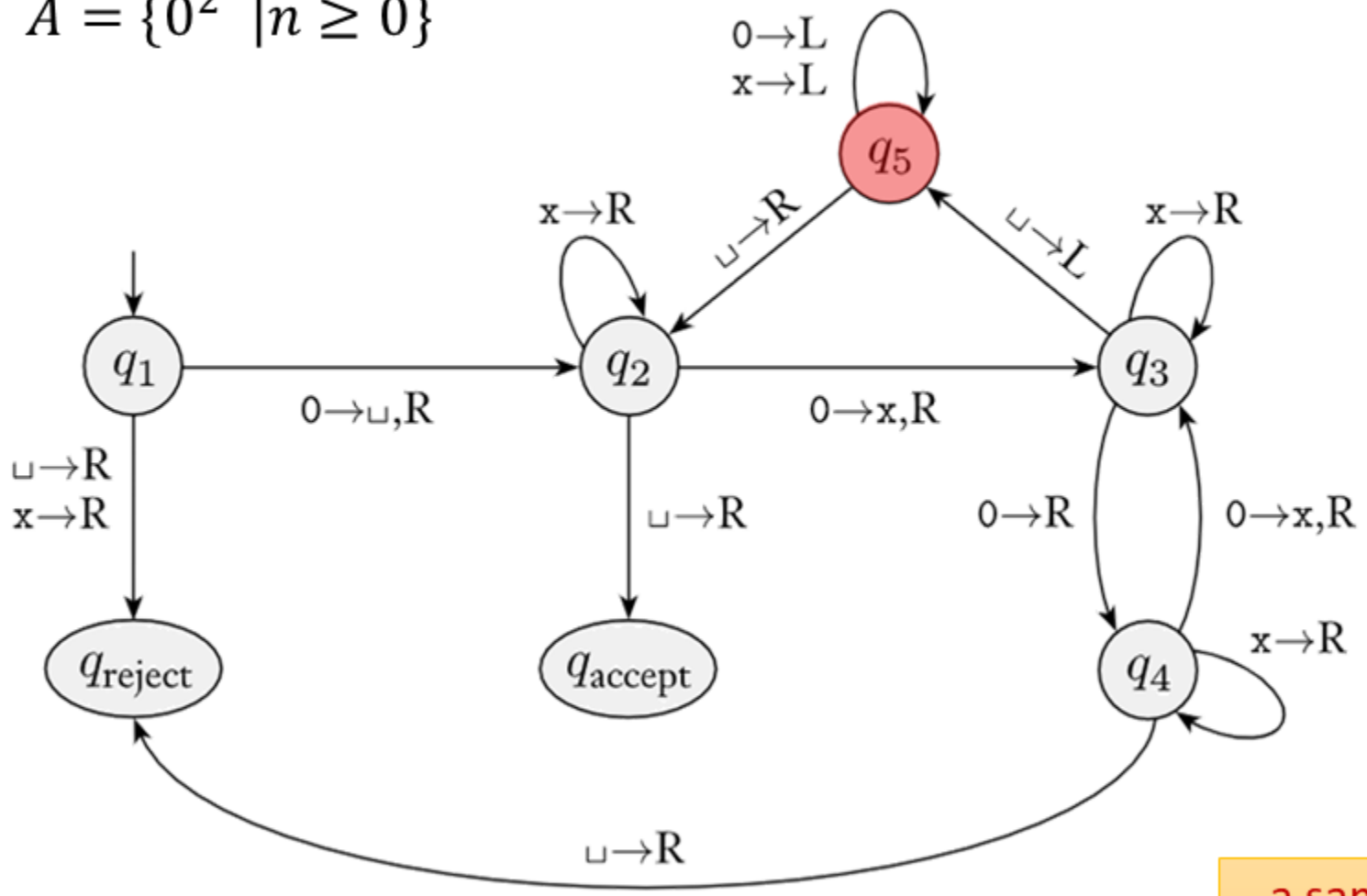


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

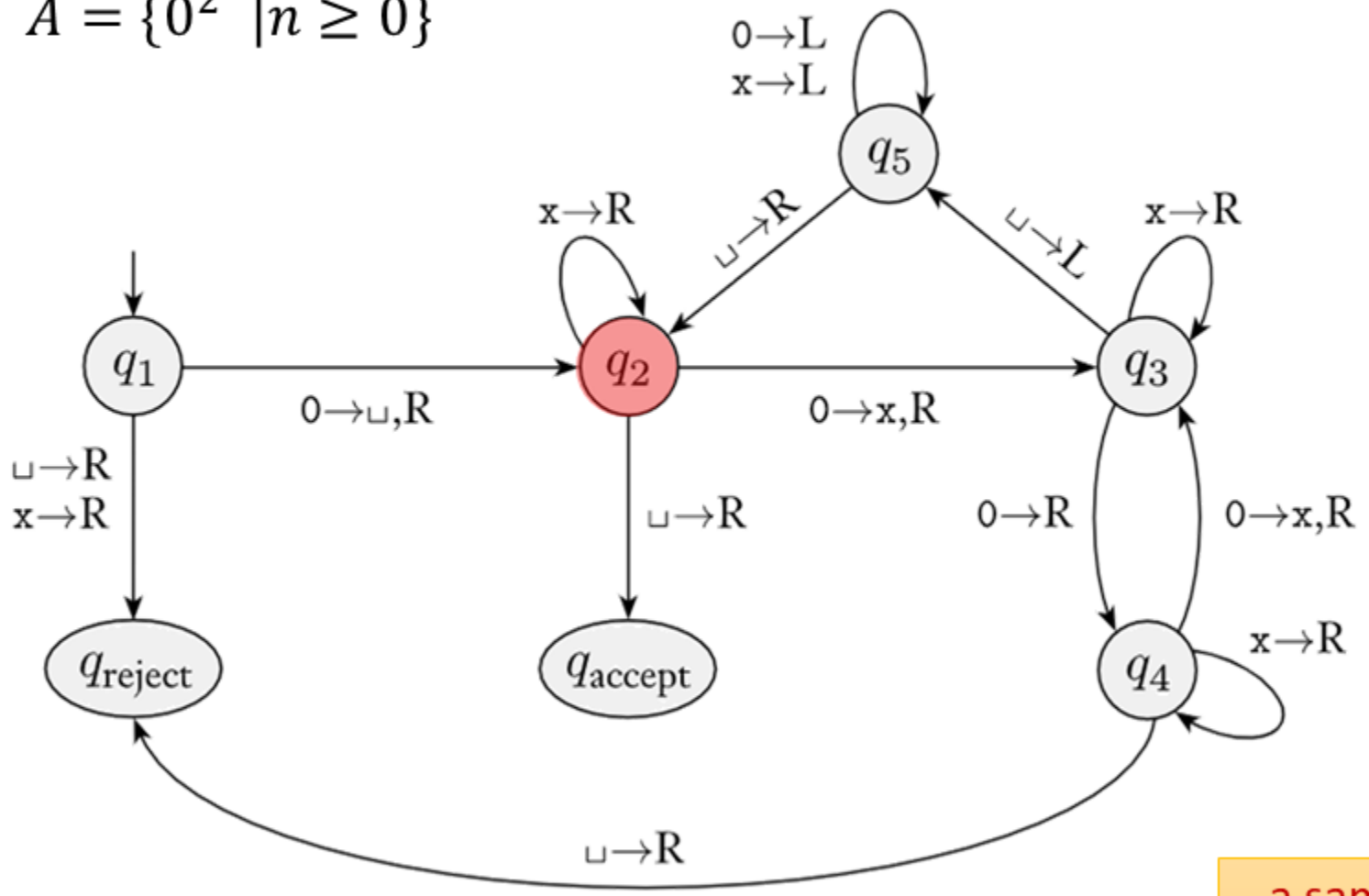


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

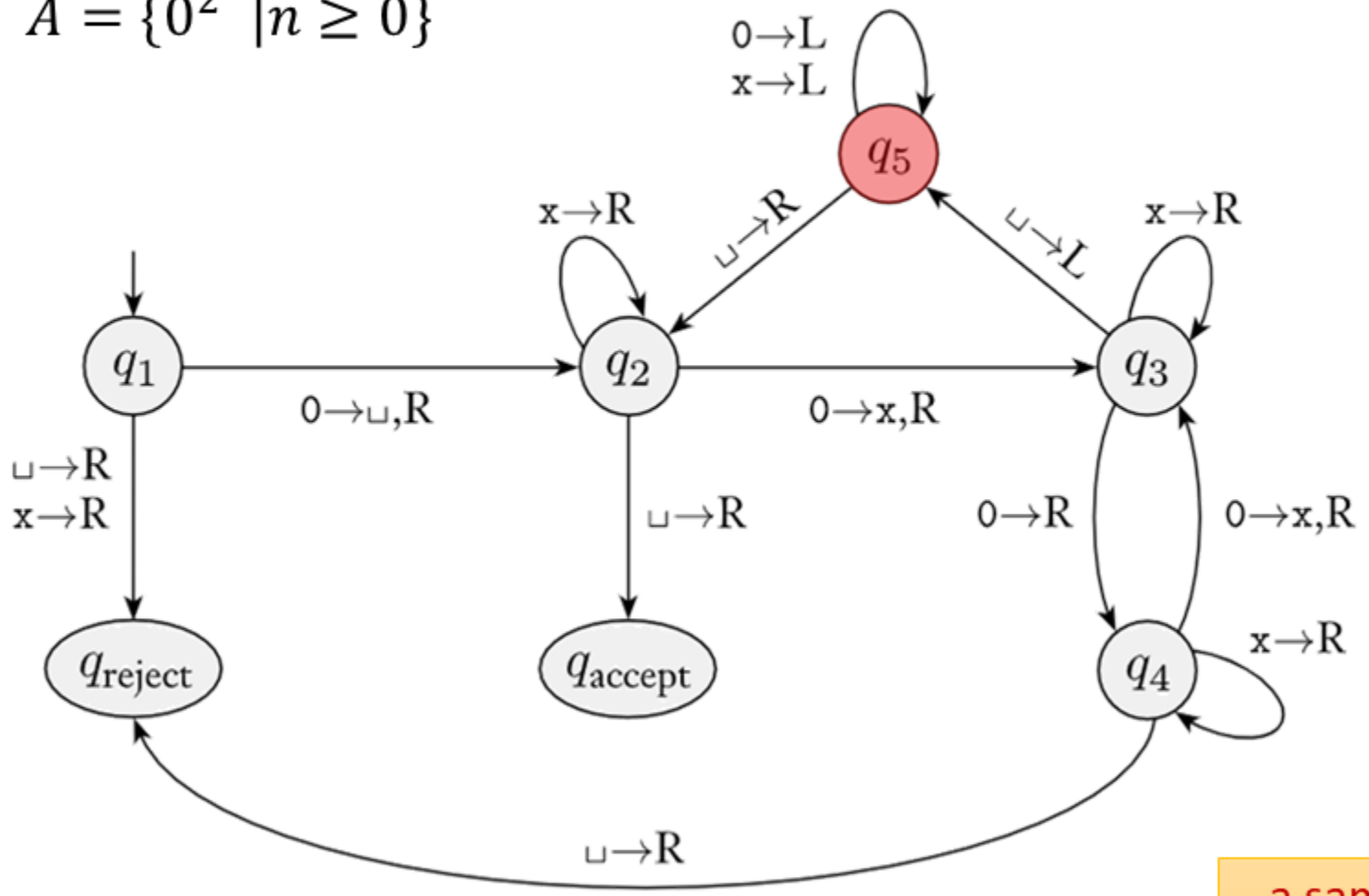


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

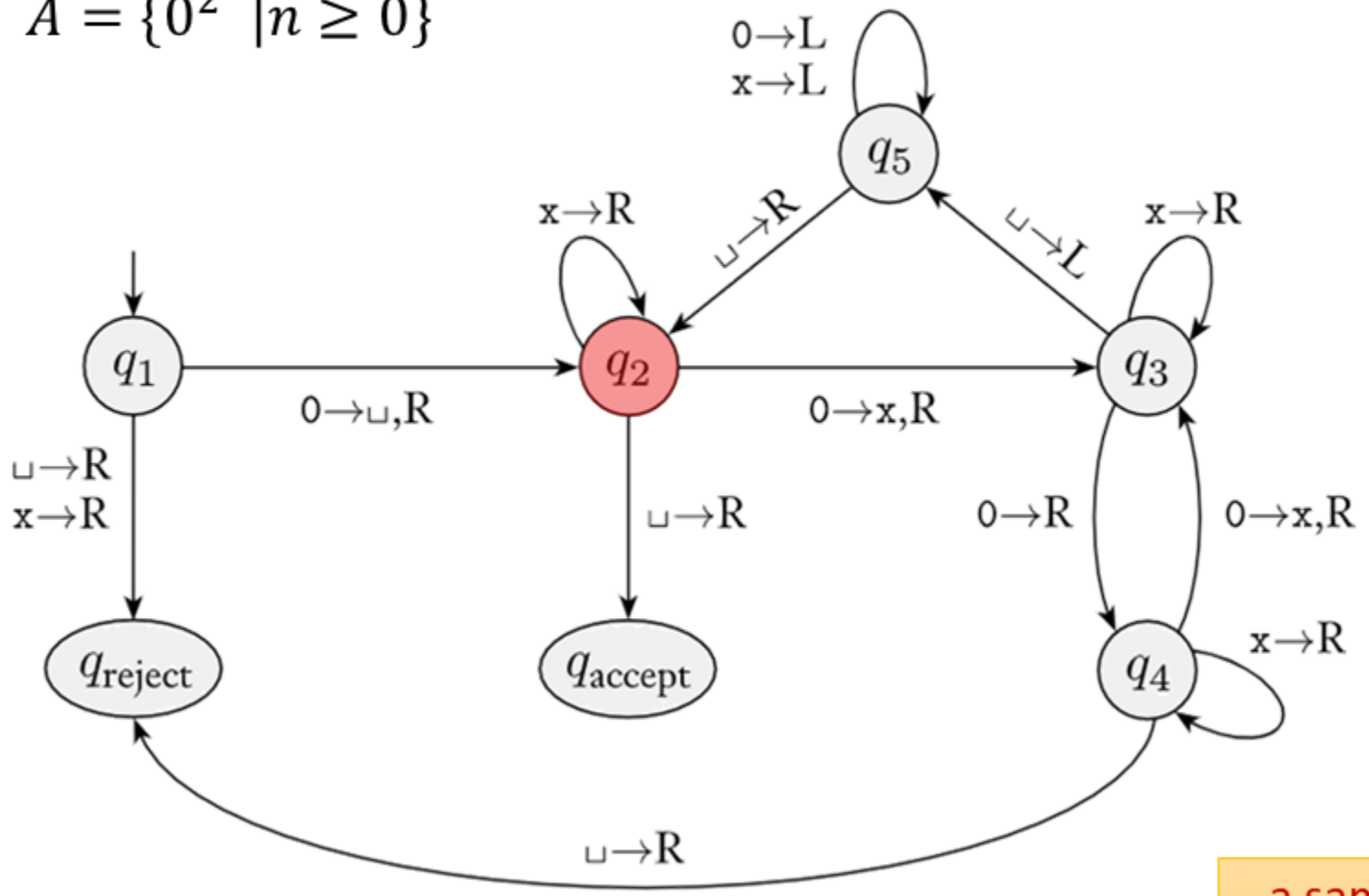


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

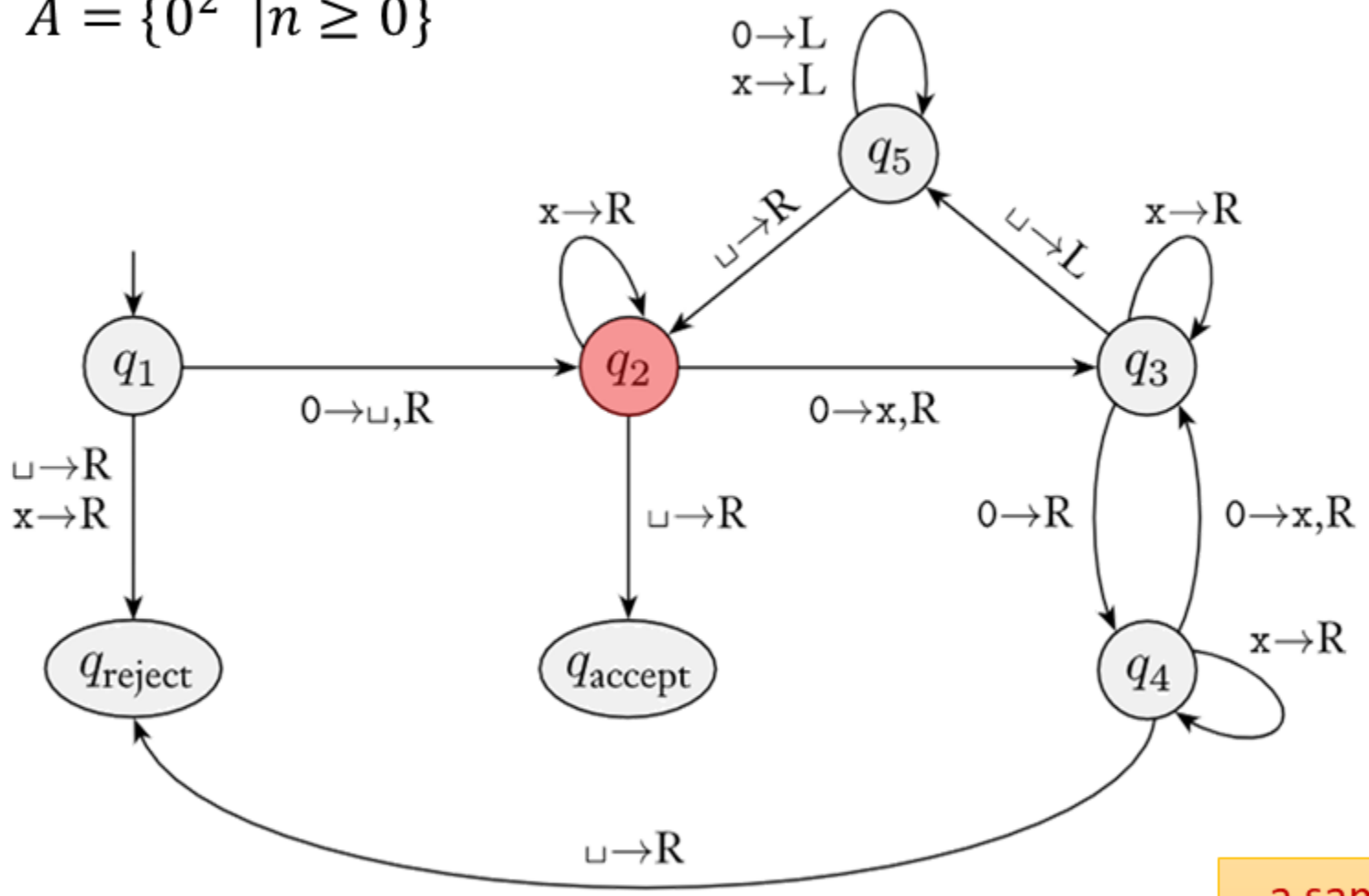


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$

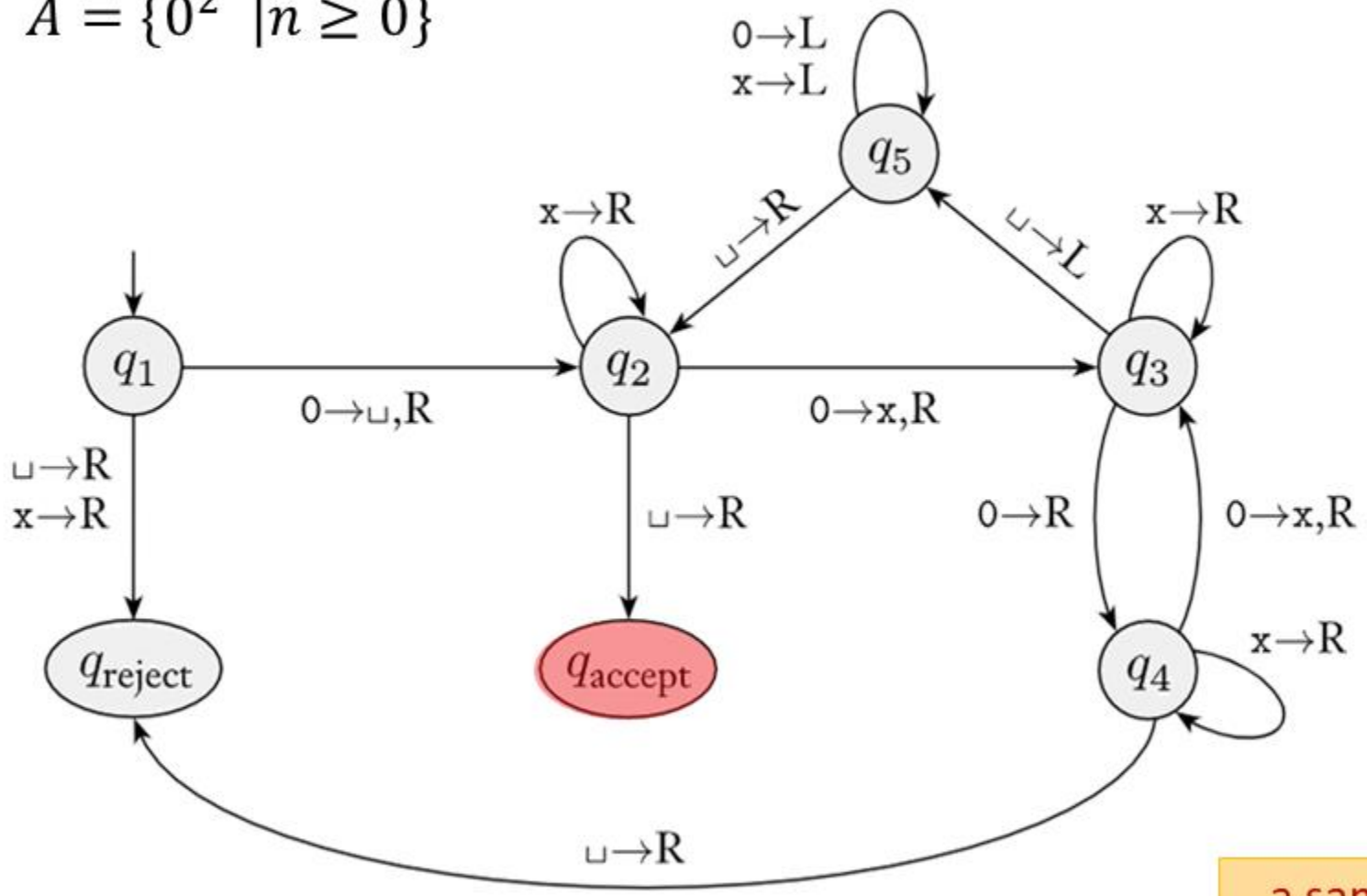


a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 4

- $A = \{0^{2^n} | n \geq 0\}$



a sample run of this machine on input 0000

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{accept}$

Example 5

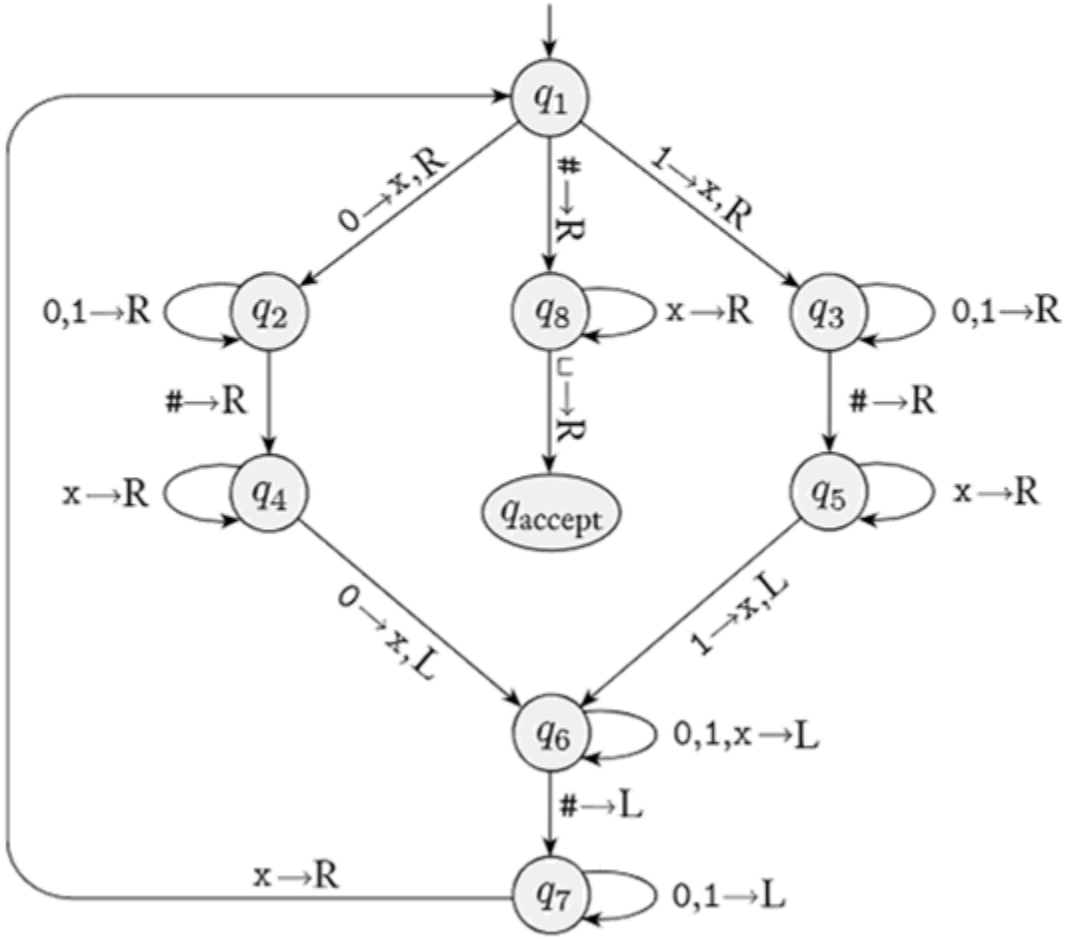
Formal definition of Turing machine (TM) M_1 that decides

$$B = \{w\#w | \mathrm{w} \in \{0,1\}^*\}.$$

Example 5

Formal definition of Turing machine (TM) M_1 that decides

$$B = \{w\#w | \mathrm{w} \in \{0,1\}^*\}.$$

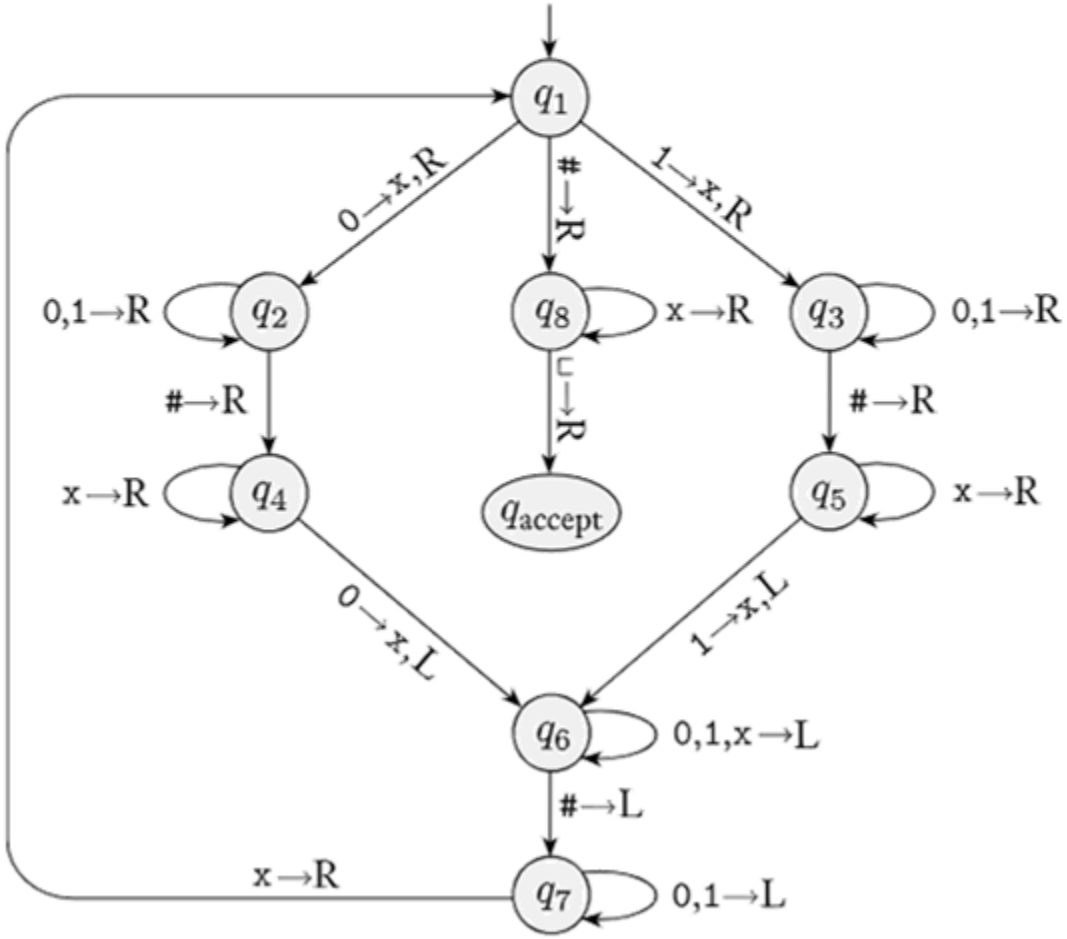


Example 5

Formal definition of Turing machine (TM) M_1 that decides

$$B = \{w\#w | w \in \{0,1\}^*\}.$$

- $\boldsymbol{M_1} = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$
- $\boldsymbol{Q} = \{q_1, \dots, q_8, q_{accept}, q_{reject}\}$
- $\boldsymbol{\Sigma} = \{0,1,\#\}, and\ \ \Gamma = \{0,1,\#,x, \sqcup\}.$
- The **transition function** can be described with a state diagram



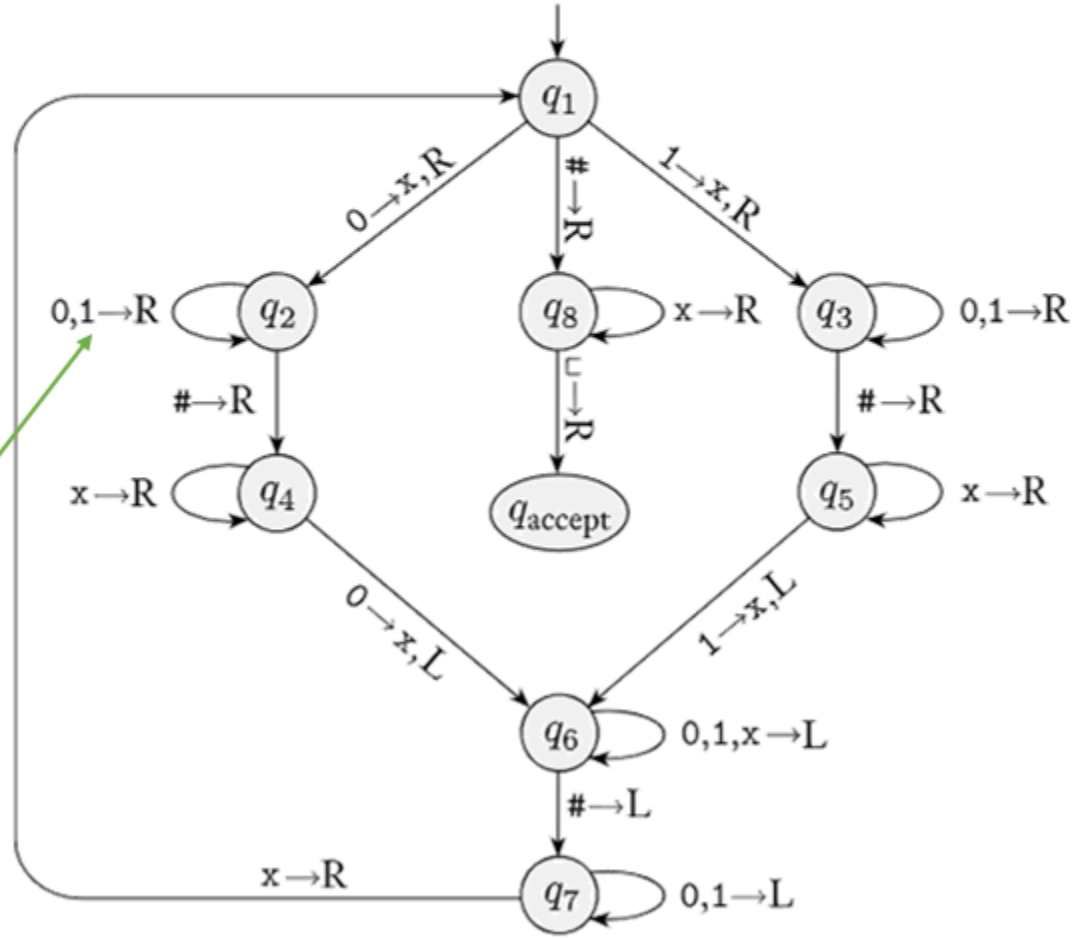
Example 5

Formal definition of Turing machine (TM) M_1 that decides

$$B = \{w\#w | w \in \{0,1\}^*\}.$$

- $\boldsymbol{M_1} = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$
- $\boldsymbol{Q} = \{q_1, \dots, q_8, q_{accept}, q_{reject}\}$
- $\boldsymbol{\Sigma} = \{0,1,\#\}, and\ \ \Gamma = \{0,1,\#,x,\sqcup\}.$
- The **transition function** can be described with a state diagram

The machine stays in q2 and moves to the right when it reads a 0 or a 1 in state q2.



- To simplify the figure, we didn't show the reject state.

Example 6

The TM M_3 decides the language $C = \{a^i b^j c^k | i \times j = k \; and i, j, k \geq 1\}$.

- This machine is doing some elementary arithmetic.

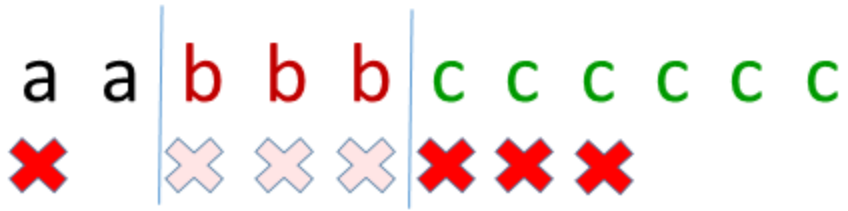
Example 6

The TM M_3 decides the language $C = \{a^i b^j c^k | i \times j = k \text{ and } i, j, k \geq 1\}$.

- This machine is doing some elementary arithmetic.

M_3 = "On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+b^+c^+$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and the c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*.
4. Restore the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's have been crossed off, determine whether all c's also have been crossed off. If yes, *accept*; otherwise, *reject*."



First iteration of Stage 3

Example 8

- Here, a TM M_4 is solving what is called the element distinctness problem.
- It is given a list of strings over {0,1} separated by #s and its job is to accept if all the strings are different. The language is

$$E = \{\#x_1\#x_2\#\cdots\#x_l \mid each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.

Placing the 1st mark

#101#11#1#1011

#̇101#̇11#1#1011

#̇101#11#̇1#1011

#̇101#11#1#̇1011

#101#̇11#̇1#1011

#101#̇11#1#̇1011

#101#11#̇1#̇1011

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.

Placing the 2nd mark

#101#11#1#1011

#̇101#̇11#1#1011

#̇101#11#̇1#1011

#̇101#11#1#̇1011

#101#̇11#̇1#1011

#101#̇11#1#̇1011

#101#11#̇1#̇1011

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.

#101#11#1#1011

#̇101#̇11#1#1011

#̇101#11#̇1#1011

#̇101#11#1#̇1011

#101#̇11#̇1#1011

#101#̇11#1#̇1011

#101#11#̇1#̇1011

Comparing two strings

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.

#101#11#1#1011

#̇101#̇11#1#1011

#̇101#11#̇1#1011

#̇101#11#1#̇1011

#101#̇11#̇1#1011

#101#̇11#1#̇1011

#101#11#̇1#̇1011

Updating the marked #s

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."

#101#11#1#1011

#̇101#̇11#1#1011

#̇101#11#̇1#1011

#̇101#11#1#̇1011

#101#̇11#̇1#1011

#101#̇11#1#̇1011

#101#11#̇1#̇1011

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."

#101#11#1#1011

#̇101#̇11#1#1011

#̇101#11#̇1#1011

#̇101#11#1#̇1011

#101#̇11#̇1#1011

#101#̇11#1#̇1011

#101#11#̇1#̇1011

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."

#101#11#1#1011

#̇101#̇11#1#1011

#̇101#11#̇1#1011

#̇101#11#1#̇1011

#101#̇11#̇1#1011

#101#̇11#1#̇1011

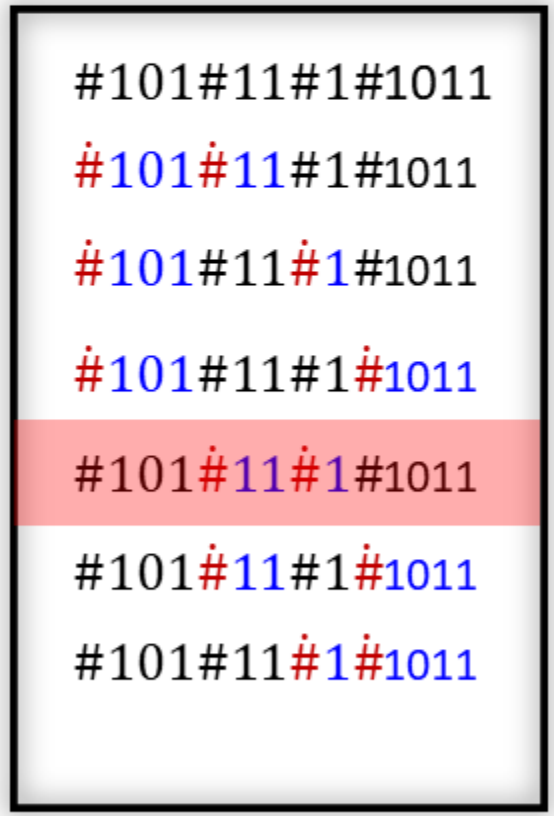
#101#11#̇1#̇1011

Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."

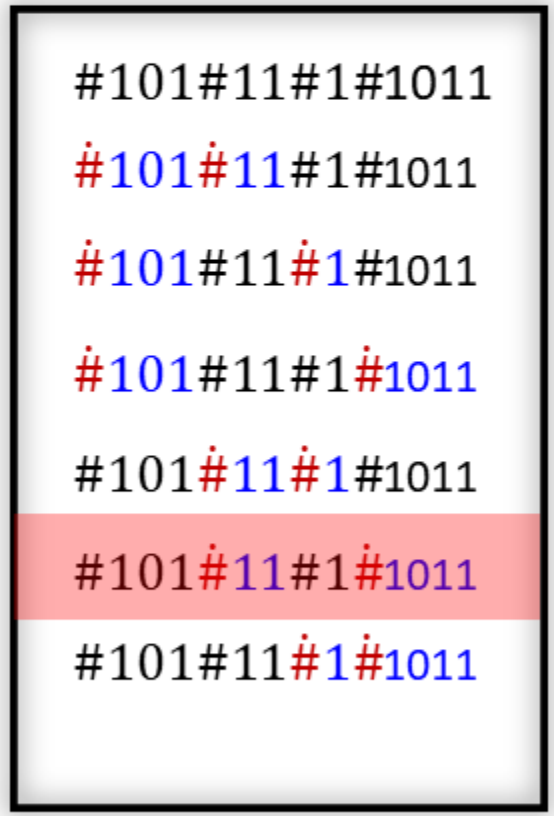


Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."

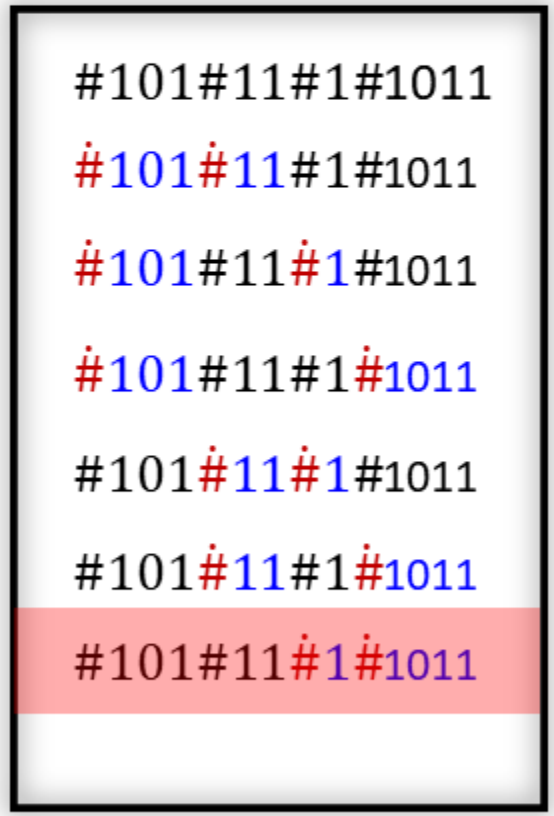


Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."



Example 8

$$E = \{\#x_1\#x_2\#\cdots\#x_l | each\ x_i \in \{0,1\}^* and\ x_i \neq x_j\ for\ each\ i \neq j\}$$

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to stage 3."