Turing Machines

Introduction and Motivation

In this lecture we introduce ***Turing Machines*** and discuss some of their properties.

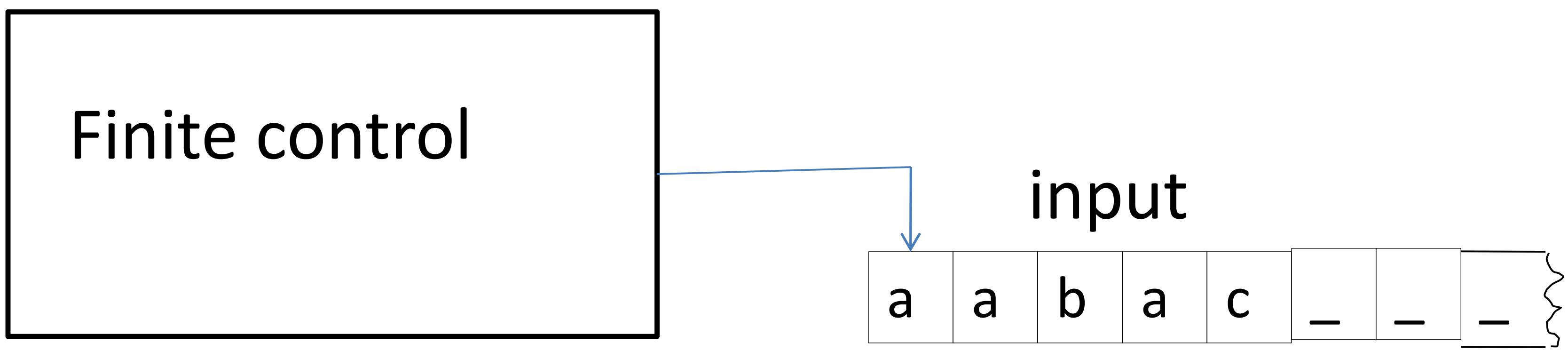
Turing Machines

A **Turing Machine** is a finite state machine augmented with an infinite tape.

The tape head can go in both directions. It can read and write from/to any cell of the semi-infinite tape.

Once the TM reaches an **accept (reject resp.)** state it **accepts (rejects resp.) immediately**.

Schematic of a Turing Machine



The tape head can go in both directions. It can read and write from/to any cell of the semi-infinite tape. The _ symbol marks the input's end.

TM – A Formal Definition

A ***Turing Machine*** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$**,** where:

1. Q is a finite set called the ***states***.
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** , _ .
3. Γ is the ***tape alphabet,*** $\_ \in \Gamma$ and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the ***transition function***.
5. $q_0 \in Q$ is the ***start state***.

TM – A Formal Definition

A ***Turing Machine*** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:

6. $q_{accept} \in Q$ is the ***accept state***, and
7. $q_{reject} \in Q$ is the ***reject state***, where $q_{reject} \neq q_{accept}$

The Transition Function - Domain

Let M be a Turing machine defined by $(Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$. At any given time M is in some state, $q \in Q$, and its head is on some tape square containing some tape symbol $\gamma \in \Gamma$. The transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L,R\}$, depends on the machine state q and on the tape symbol γ.

The Transition Function - Range

The range of the transition function are triples of the type (q', γ', d), where q' is M's next state, γ' is the symbol written on the tape cell over which the head was at the beginning of the transition (namely γ is replaced with γ') and $d \in \{L, R\}$ is the direction towards which the tape head has made a step.

Turing machine – A Computation

Computation of M always starts at state q_0, and the input is on the leftmost n cells where n is the input's ***length***. The tape's head is over the tape's cell 0 – the leftmost cell.

Computation of M ends either when it reaches $q_{accept} \in Q$ - this is an ***Accepting Computation****.* Or when it reaches $q_{reject} \in Q$ - this is a ***Rejecting Computation****.*

Configurations

A configuration of a Turing machine M is a concise description M's state and tape contents. It is written as $C=uqv$ and its meaning is:

1. The state of M is q.
2. The content of M's tape is uv , where u resides on the leftmost part of the tape.
3. The head of M resides over the leftmost (first) symbol of v.
4. The tape cells past the end of v hold blanks.

Configurations

Configuration C_1 of M ***yields*** Configuration C_2, if

M can legally go from C_1 to C_2 in a single step.

For example:

Assume that $a,b,c \in \Gamma$, $u,v \in \Gamma^*$, and $q_i, q_j \in Q$.

We say that uaq_ibv yields uq_jacv, if $\delta(q_i,b)=(q_j,c,L)$,

for a leftward movement of the head.

We say that uaq_ibv yields $uacq_jv$, if $\delta(q_i,b)=(q_j,c,R)$,

for a rightward movement of the head.

Configurations – Special Cases

Configuration $q_i bv$ yields $q_j cv$ if the head is at the beginning of the tape and the transition is **left-moving**, because the head cannot go off the left-hand end of the tape.

Configuration uaq_i is equivalent to $uaq_i\_$ because the empty part of the tape is always filled out with blanks.

Computations

The ***start*** Configuration of M on input w is q_0w, which indicates that M is at its initial state, q_0 , it's head is on the first cell of its tape and the tape's content is the input w.

Any configuration in which of M reaches state q_{accept}, is an ***accepting configuration***.

Any configuration in which M reaches state q_{reject} ,is a ***rejecting configuration***.

Computations

Accepting and rejecting configurations are ***halting configurations***.

A TM M ***accepts*** word w if there exists a computation (a sequence of configurations) of M, $C_1, C_2, \ldots, C_k$ satisfying:

1. $C_1 = q_0 w$ is the starting state of M on input w.
2. For each $i, 1 \le i < k$, C_i yields C_{i+1}, and
3. C_k is an accepting configuration.

Computation Outcomes

A Computation of a Turing machine M may result in three different ***outcomes***:

1. M may ***accept*** **–** By halting in q_{accept} .
2. M may ***reject*** **–** By halting in q_{reject} .
3. M may ***loop*** **–** By not halting **for ever**.

Note: When M is running, it is not clear whether it is **looping** . Meaning M **may stop eventually but nobody can tell.**

Turing Recognizers

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

A language is ***Turing Recognizable*** (or ***recursively enumerable***) if there exists a Turing machine that recognizes it.

Turing Deciders

Since it is hard to tell whether a running machine is ***looping***, we prefer machines that halt on ***all inputs***. These machines are called ***deciders***.

A decider that recognizes a language L is said to ***decide*** L.

A language is ***Turing decidable*** (or ***recursive***) if there exists a Turing machine that decides it.

An Example

Consider the language $L = \{0^{2^n} | n \geq 0\}$ containing strings of 0-s whose length is an integral power of 2.

The language L is neither regular nor CFL.

In the next slide we present a high level description of TM M_2 to decide L. The description format follows the textbook.

An Example

$M_2 =$ "On input string w:

1. Sweep the tape left to right, crossing every second 0.
2. If in stage 1 the tape has a single 0, *accept*.
3. If in stage 1 the tape has an odd number of 0-s **greater than 1**, *reject*.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1. "

Explanation

M_2 works as follows:

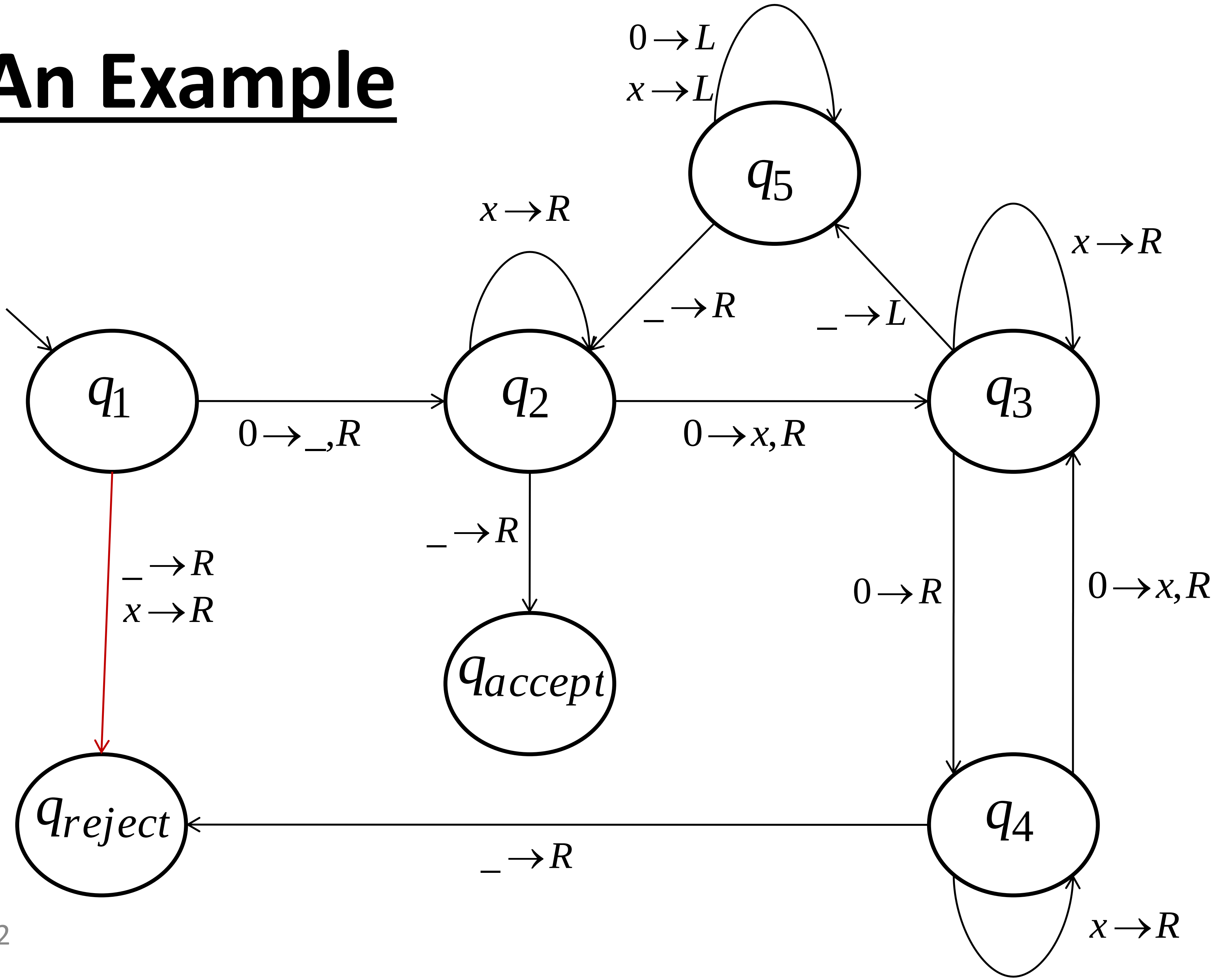
Each iteration of stage 1 cuts the number of 0-s in half. As it sweeps across its tape on stage 1 it “calculates” whether the number of 0-s it sees is odd or even. If the number of 0-s is odd and greater than 1, the input length cannot be a power of 2, so it rejects. If the number of 0-s is 1, the input length is a power of 2 and it accepts.

An Example

In the following slide the transition function of M_2 is presented.

Note: $\Sigma = \{0\}$, $\Gamma = \{0, x, \_\}$.

An Example


$0 \rightarrow L$
$x \rightarrow L$
q_5
$x \rightarrow R$
$x \rightarrow R$
$\_ \rightarrow R$
$\_ \rightarrow L$
q_1
q_2
q_3
$0 \rightarrow \_, R$
$0 \rightarrow x, R$
$\_ \rightarrow R$
$\_ \rightarrow R$
$x \rightarrow R$
$0 \rightarrow R$
$0 \rightarrow x, R$
q_{accept}
q_{reject}
q_4
$\_ \rightarrow R$
$x \rightarrow R$

Example2

Consider the language $L = \{ w\#w \mid w \in \{0,1\}^* \}$.

A simple method to check whether a string w is in L is: Read the first character of w, store it, and mark it off. Then scan w until the character # is found, if the first character past # is equal to the stored character, cross it and go back to the last crossed character, On the tape's beginning.

Example2

Repeat this procedure until all the string w is scanned. If an unexpected character is found, **reject**. Otherwise, **accept.**

In the next slide we present a high level description of TM M_1 to decide L. The description format follows the text book.

Example2

$M_1 =$ "On input string w:

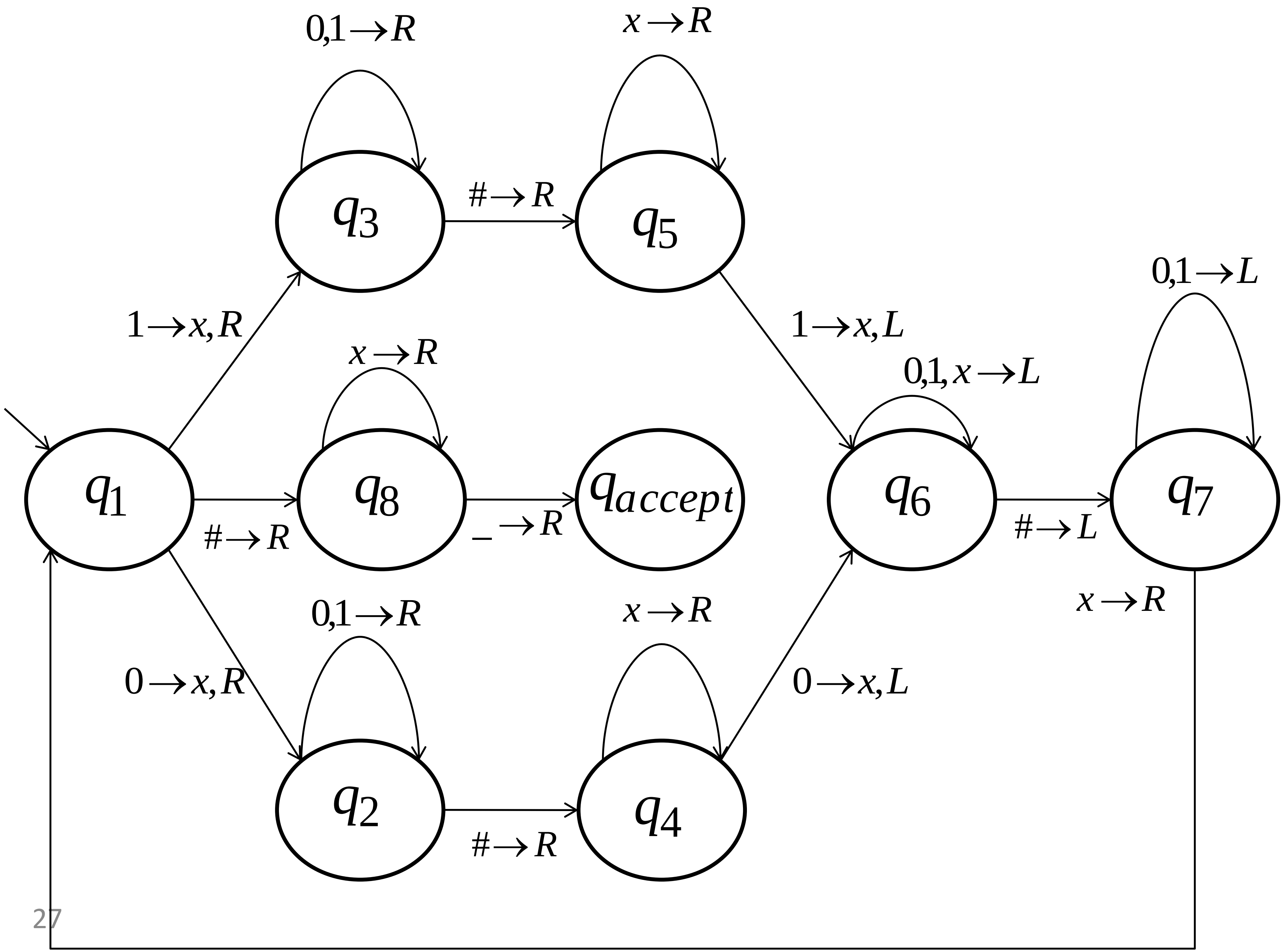
1. Store the leftmost symbol on the tape and cross it out by writing x.
2. Go right past #, if # not found, **reject**.
3. compare the leftmost non x symbol to the stored symbol. If not equal, **reject**.
4. Cross out the compared symbol. Return the head to the left-hand end of the tape.
5. Go to stage 1. "

Example2

In the following slide the transition function of M_2 is presented.

Note:

1. $\Sigma = \{0,1,\#\}$, $\Gamma = \{0,1,\#,x,\_\}$.
2. In this description, state q_{reject} and all its incoming transitions are omitted. Wherever there is a missing transition, it goes to q_{reject}.


q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_{accept}
$1 \to x, R$
$0 \to x, R$
$\# \to R$
$0,1 \to R$
$0,1 \to R$
$\# \to R$
$\# \to R$
$x \to R$
$x \to R$
$x \to R$
$\_ \to R$
$1 \to x, L$
$0 \to x, L$
$0,1,x \to L$
$\# \to L$
$0,1 \to L$
$x \to R$

Example2

Note: states q_2 and q_4 "store" the bit 0, while states q_3 and q_5 "store" the bit 1.
In other words: These two segments are identical, but when they **merge** each segment uses the value it stored.