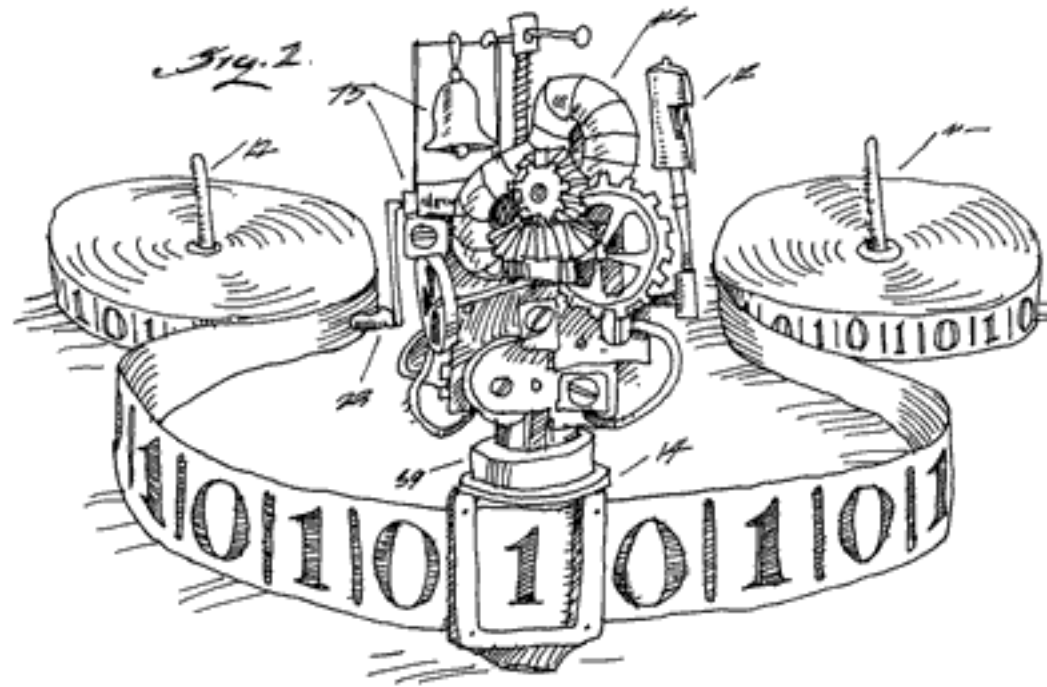
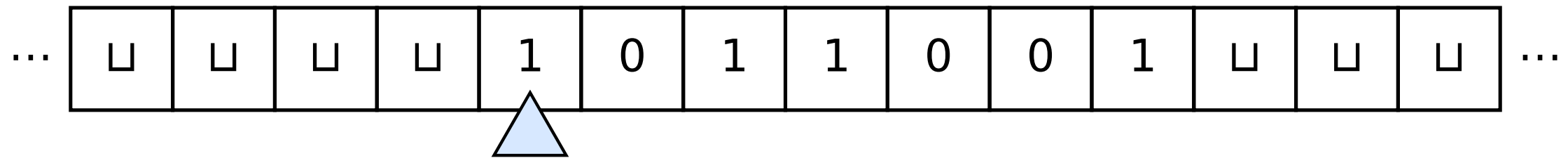


CSC4010: Introduction to Theoretical Computer Science



Lecture 4

Recap



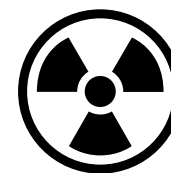
Turing Machine Model:

- An infinite, one-dimensional “tape”. The tape divided into “cells”. Each cell contains a single symbol.
- There is a “**head**” pointing at one cell of the tape.
- The machine can be in one of finitely many “**internal states**”.
- In each step, the machine decides:
 - What to write in current cell,
 - Which cell to move in (left/right),
 - And, the next state.
- The decision is based only on the symbol being read and the current state the machine is in.

Formal Definition of Turing Machine

Mathematically, a Turing Machine is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of “states”).
- $q_{\text{accept}}, q_{\text{reject}}, q_0 \in Q$ and $q_{\text{accept}} \neq q_{\text{reject}}$.
- Σ is a finite set of symbols (“the tape alphabet”).
- $\sqcup$ is a symbol (the “blank symbol”).
- $\{0, 1, \sqcup\} \subseteq \Sigma$ and $\sqcup \notin \{0, 1\}$.
- δ is a function $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\}$. (The “transition function”).



This definition is different from Sipser. (Both definitions are equivalent.)

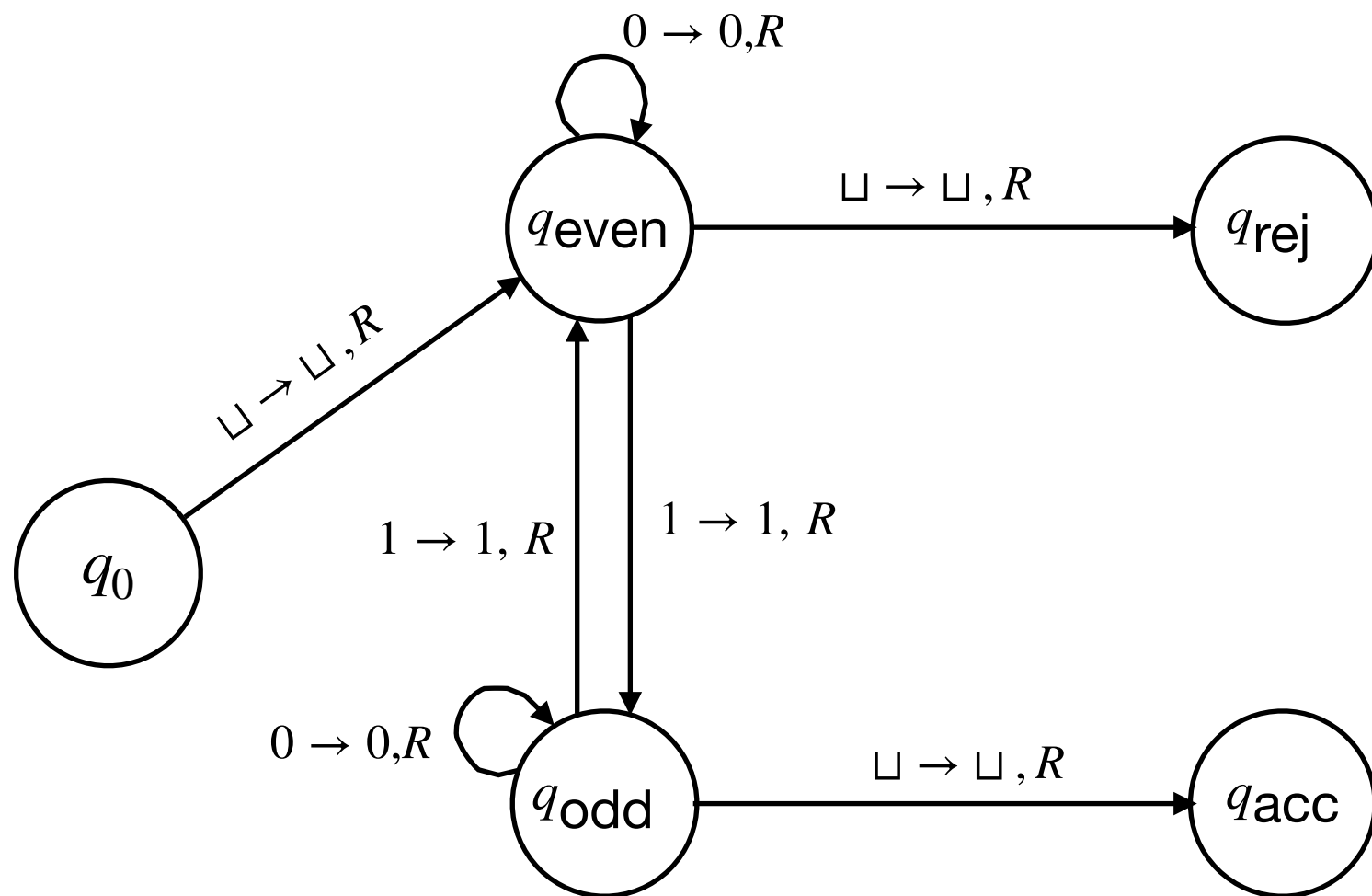
Example: Parity

- **Informal Problem Statement:** “Given $w \in \{0,1\}^*$, determine whether the number of ones in w is even or odd”.
- **Can be formulated as:**

$\text{PARITY} = \{w \in \{0,1\}^* \text{ such that } w \text{ has an odd number of 1's}\}$

- Design a Turing Machine for PARITY.

Example: Parity



- $Q = \{q_0, q_{\text{acc}}, q_{\text{rej}}, q_{\text{odd}}, q_{\text{even}}\}$

- Let's look at δ :

	0	1	$\sqcup$
q_0			
q_{even}			
q_{odd}			
q_{accept}			
q_{reject}			

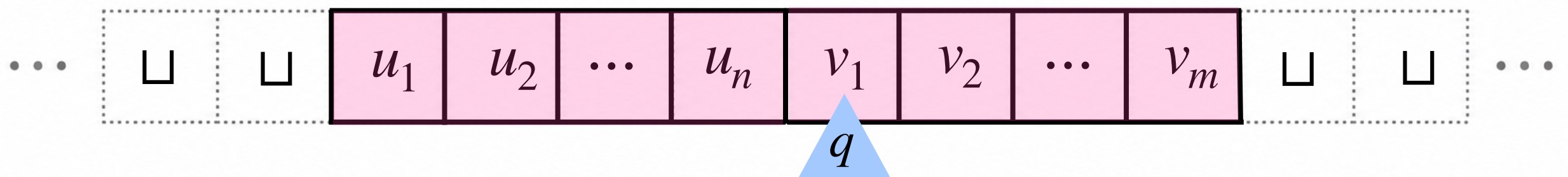
Transition Function

The transition function δ describes the **local evolution** of the computation.

What about the **global evolution**?

Configurations of a Turing Machine

- Let $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ be a Turing Machine.
- A **configuration** of M is a triple (u, q, v) where $u \in \Sigma^*, q \in Q, v \in \Sigma^*$ and $v \neq \epsilon$. Interpretation:
 - The tape currently contains uv .
 - The machine is in state q and the “head” is pointing at the first symbol of v .



Configuration Shorthand

- Instead of (u, q, v) , we often write uqv .
- The configuration uqv of a Turing Machine is a string over the alphabet $\Sigma \cup Q$.
- We can assume without loss of generality that $\Sigma \cap Q = \emptyset$.

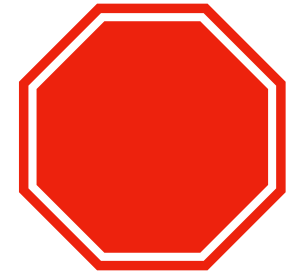
The Initial Configuration

- Let $w \in \{0,1\}^*$ be the input.
- The initial configuration of M on w is $q_0 \sqcup w$.

The “next” configuration

- For any configuration uqv , we define $\text{NEXT}(uqv)$ as follows:
 - Break uqv into individual symbols: $uqv = u_1, u_2, \dots, u_n q v_1, v_2, \dots, v_m$.
 - If $\delta(q, v_1) = (q', b, R)$, then $\text{NEXT}(uqv) = u_1 u_2 \dots u_n b q' v_2, v_3 \dots v_m$.
 - **Edge Case:** If $m = 1$, then $\text{NEXT}(uqv) = u_1 u_2 \dots u_n b q' \sqcup$.
 - If $\delta(q, v_1) = (q', b, L)$, then $\text{NEXT}(uqv) = u_1 u_2 \dots u_{n-1} q' u_n b v_2, v_3 \dots v_m$.
 - **Edge Case:** If $n = 0$, then $\text{NEXT}(uqv) = q' \sqcup b v_2 v_3 \dots v_m$.
 - We write $\text{NEXT}_M(uqv)$ if M is not clear from context.

Halting Configurations

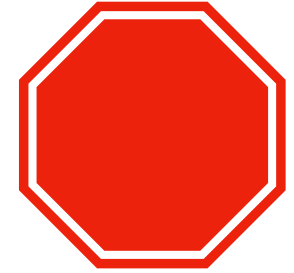


- An **accepting** configuration is of the form $uq_{\text{accept}}v$.
- An **rejecting** configuration is of the form $uq_{\text{reject}}v$.
- A **halting configuration** is either an **accepting** or a **rejecting** configuration.

Computational History

- Let $w \in \{0,1\}^*$ be an input string.
- Let C_0 be the initial configuration of M on w i.e. $C_0 = q_0 \sqcup w$.
- Inductively, for each $i \in \mathbb{N}$, let $\text{NEXT}(C_i) = C_{i+1}$.
- The **computational history** of M on w is the sequence $C_1, C_2, \dots, C_T$, where C_T is the first halting configuration in the sequence.
- If there is no such C_T , then the computational history is infinite: $C_1, C_2, C_3, \dots$.

Halting and Looping

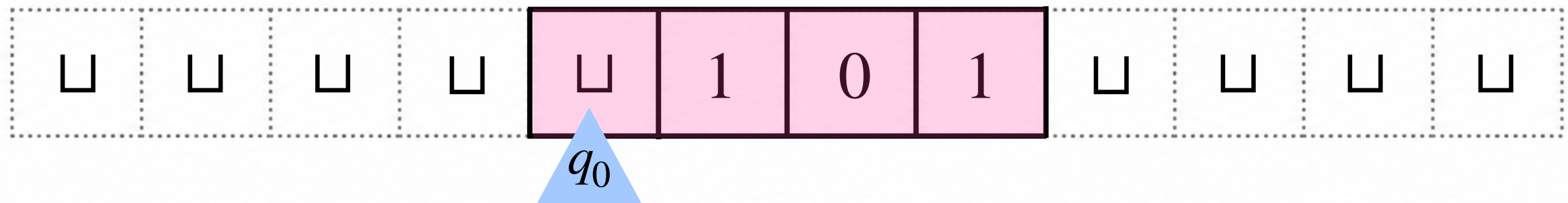
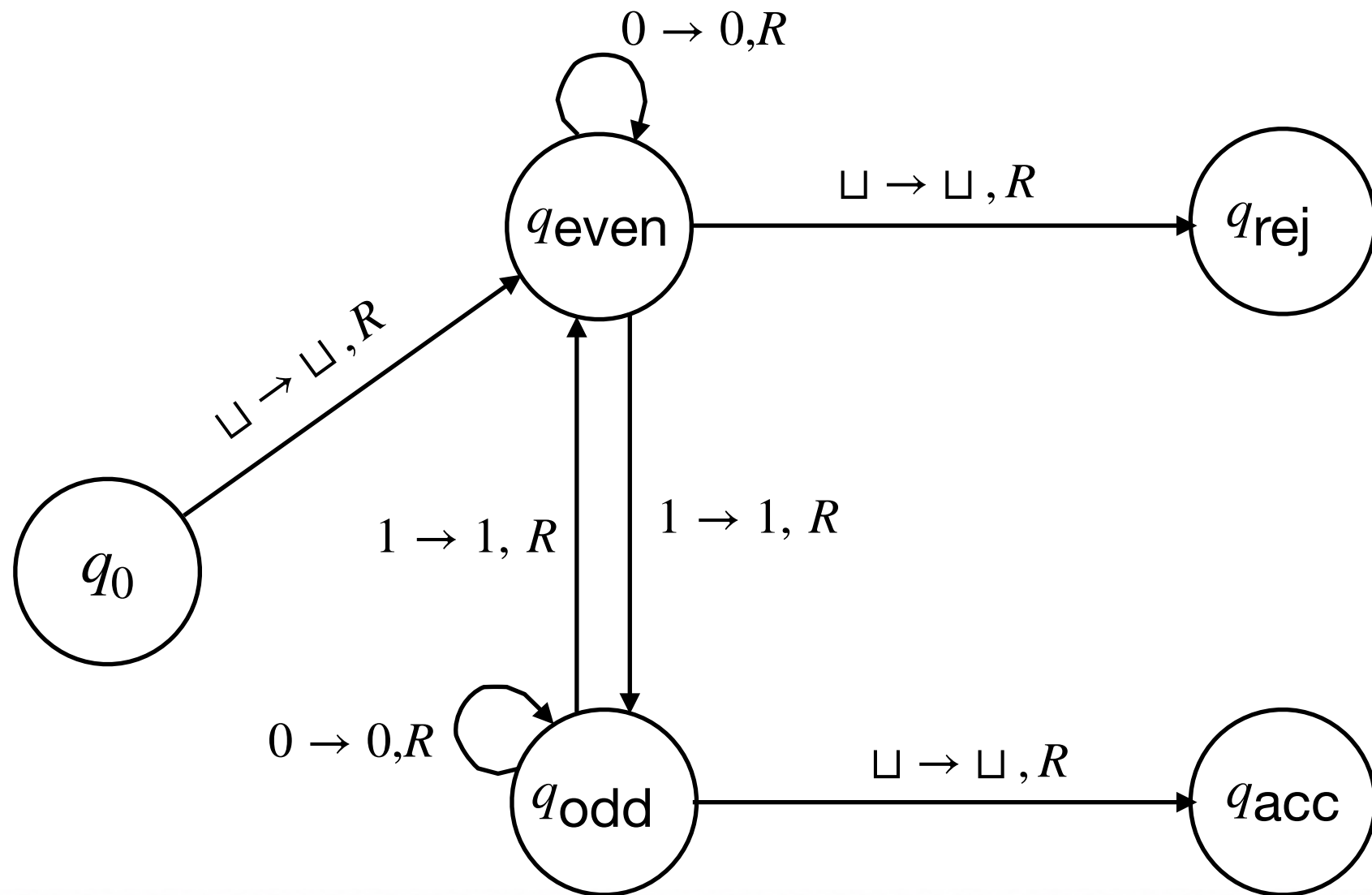


- If the computational history of M on w is finite, we say M **halts** on w .
- Otherwise, M **loops** on w .

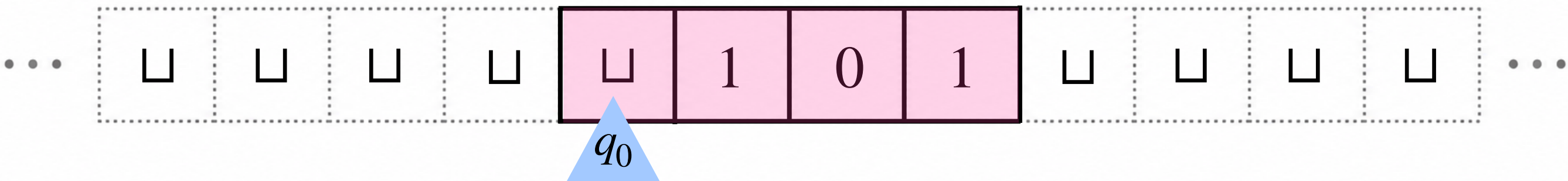
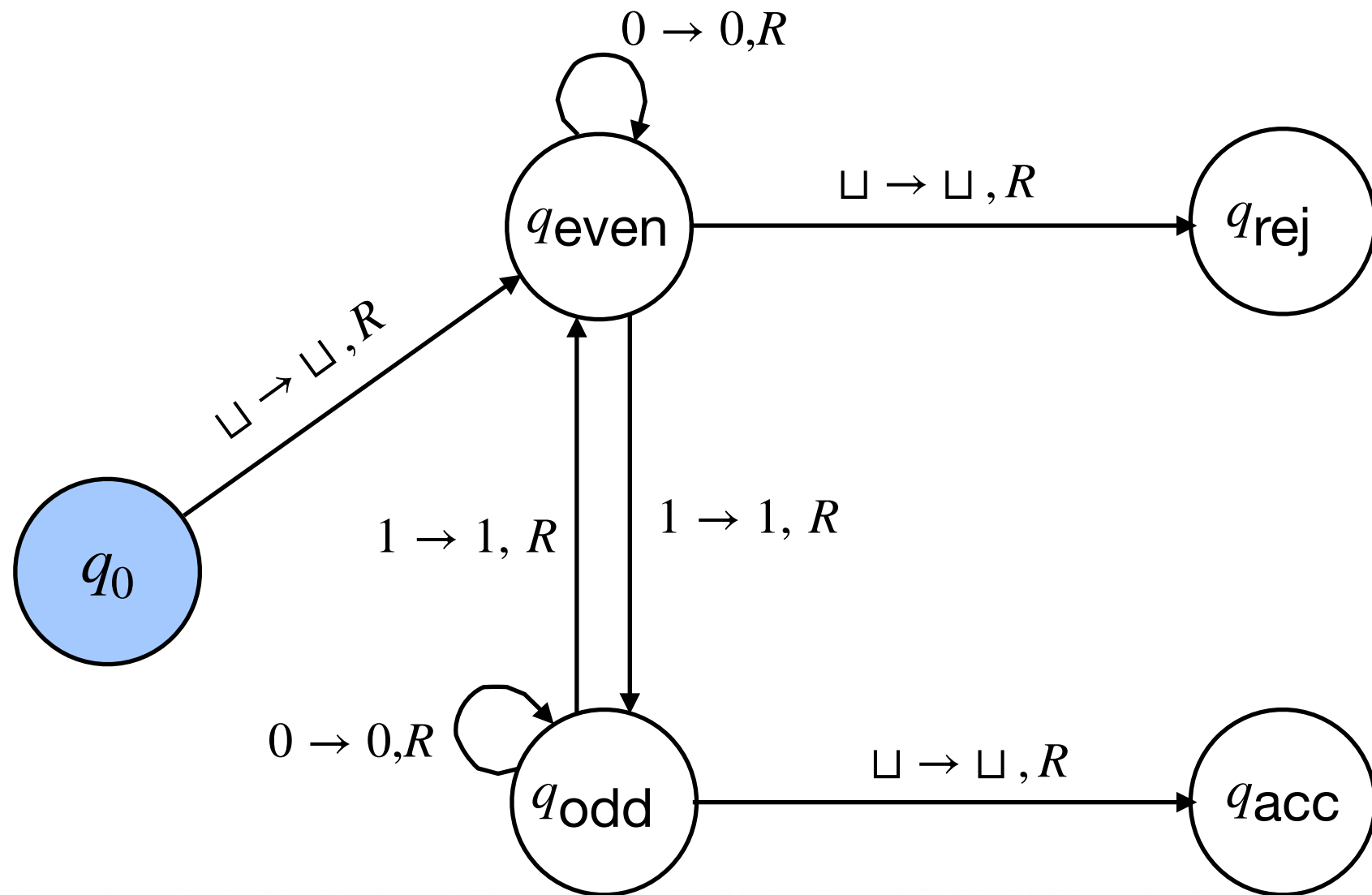
Accepting and Rejecting

- Suppose M halts on w .
- The computational history is finite, $C_0, C_1, \dots, C_T$.
- If C_T is an accepting configuration, we say M accepts w .
- If C_T is a rejecting configuration, we say M rejects w .

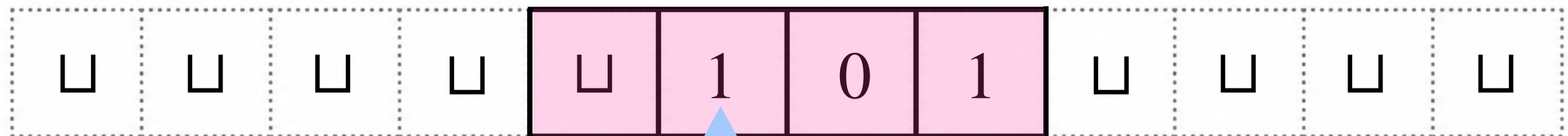
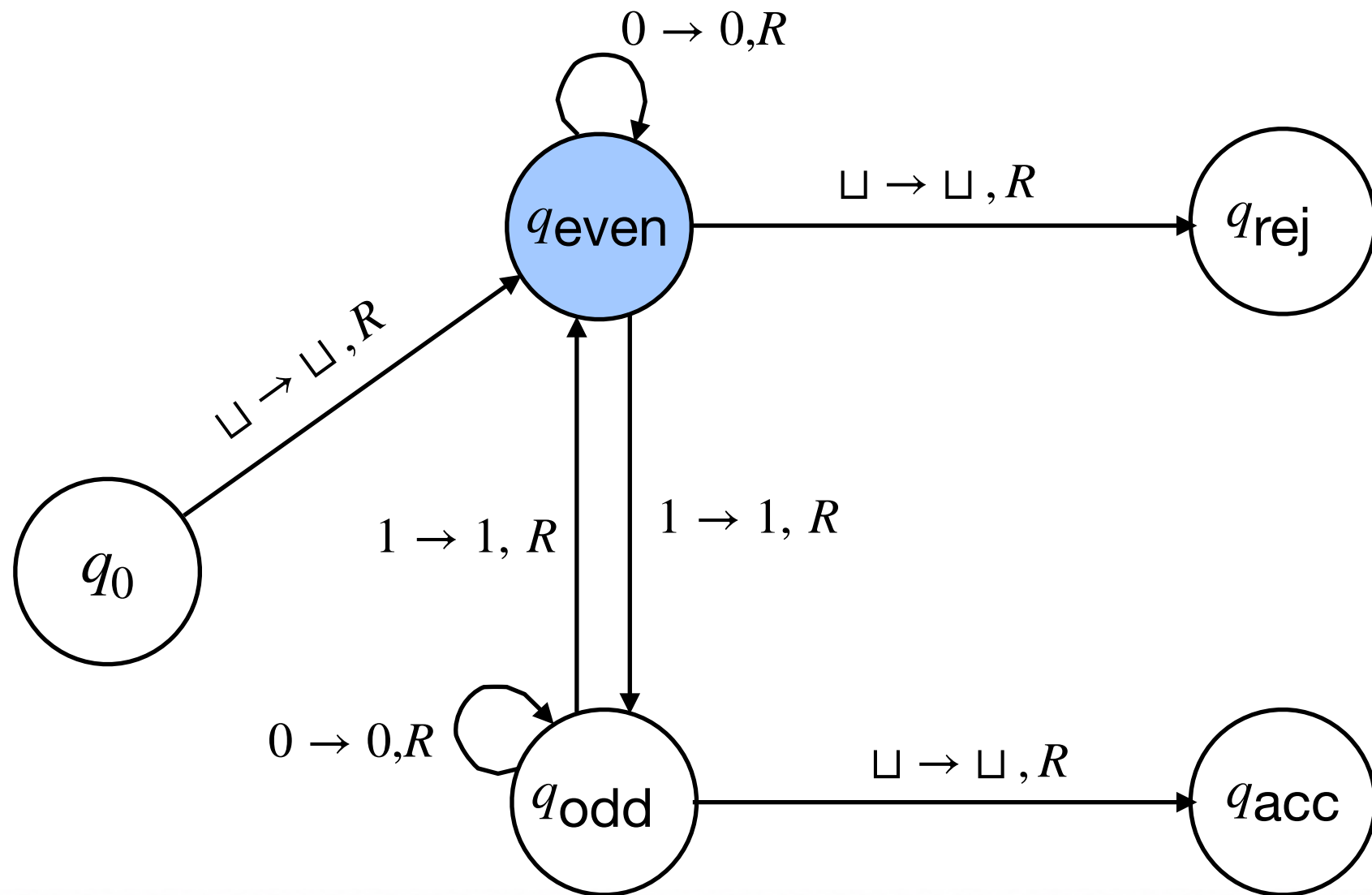
Example: Parity



Example: Parity

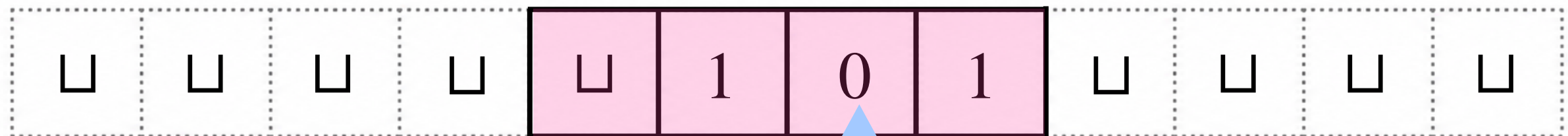
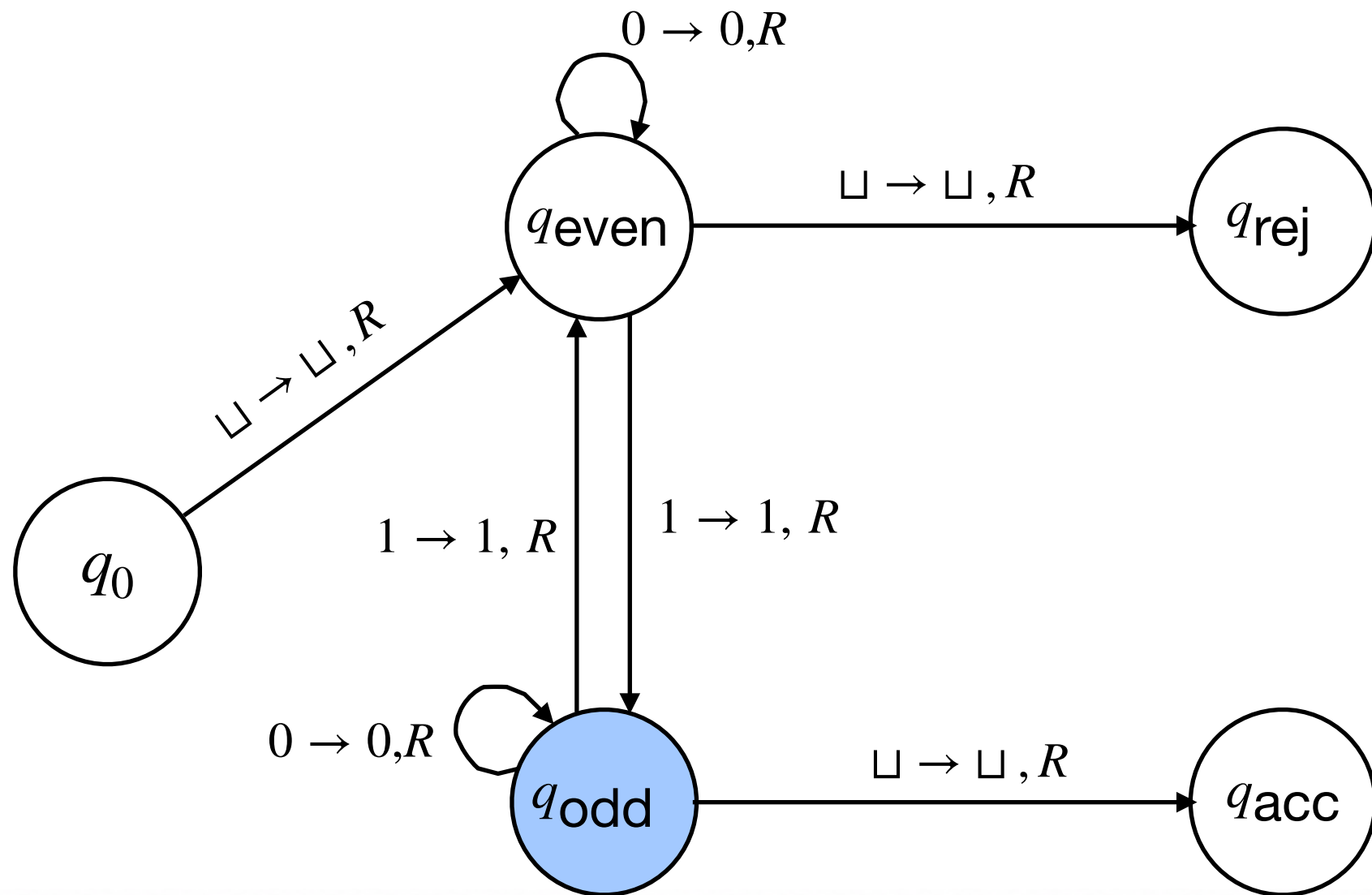


Example: Parity

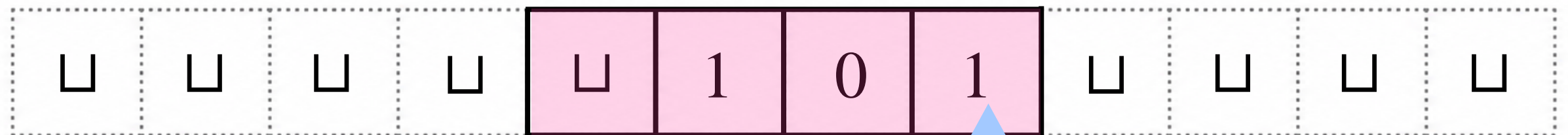
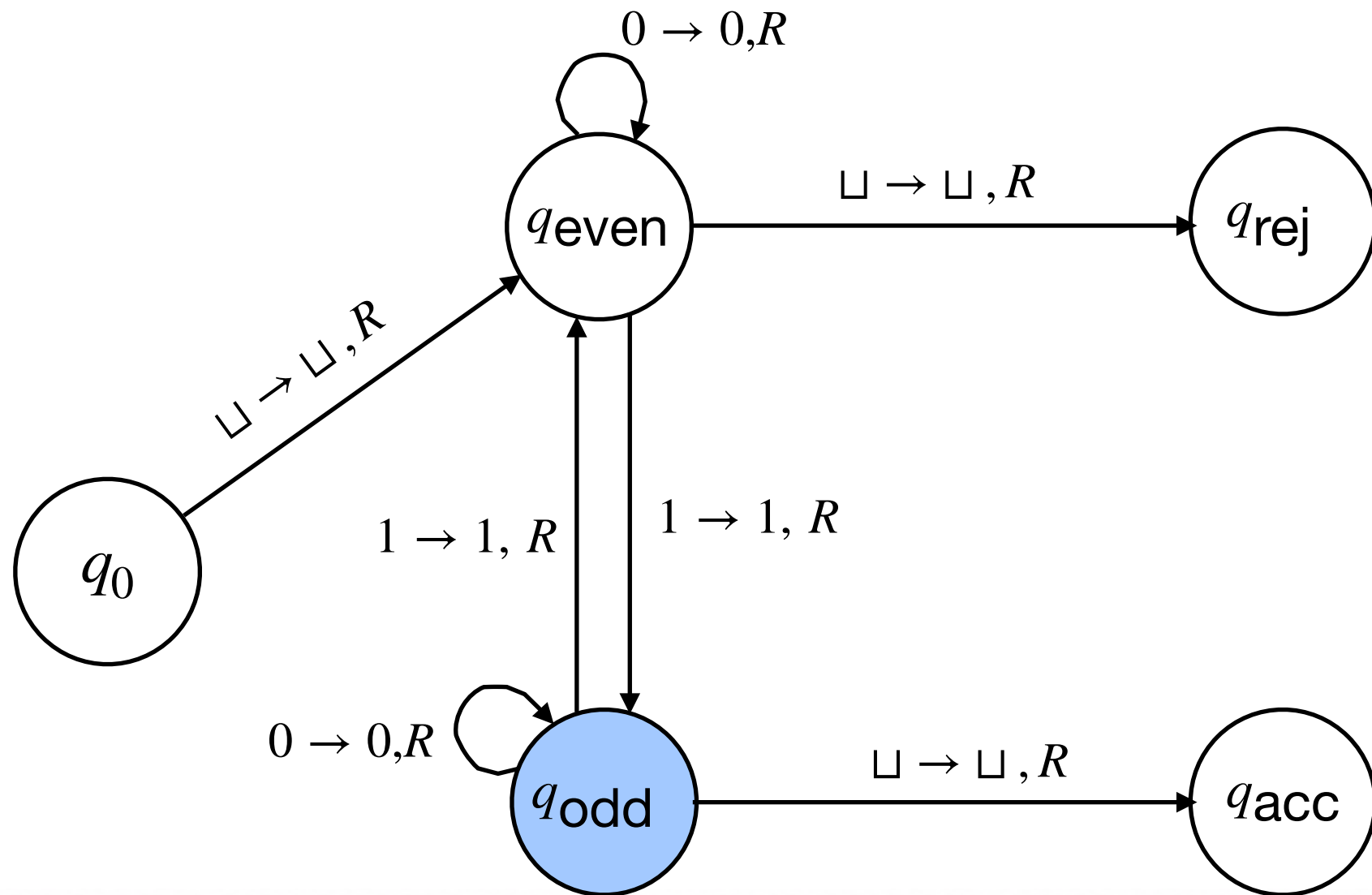


q_{even}

Example: Parity

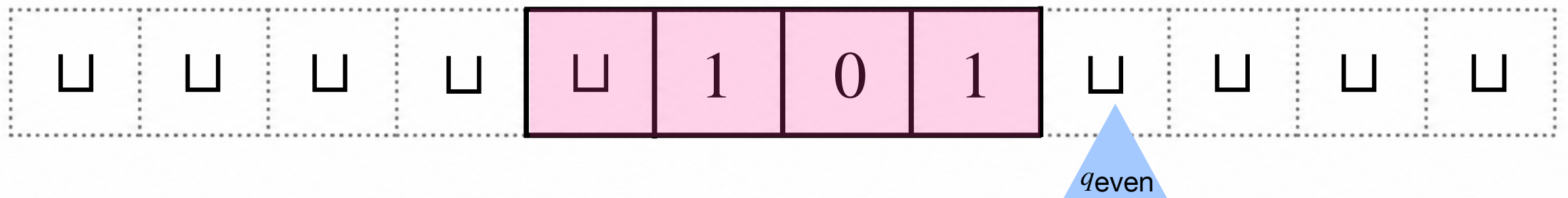
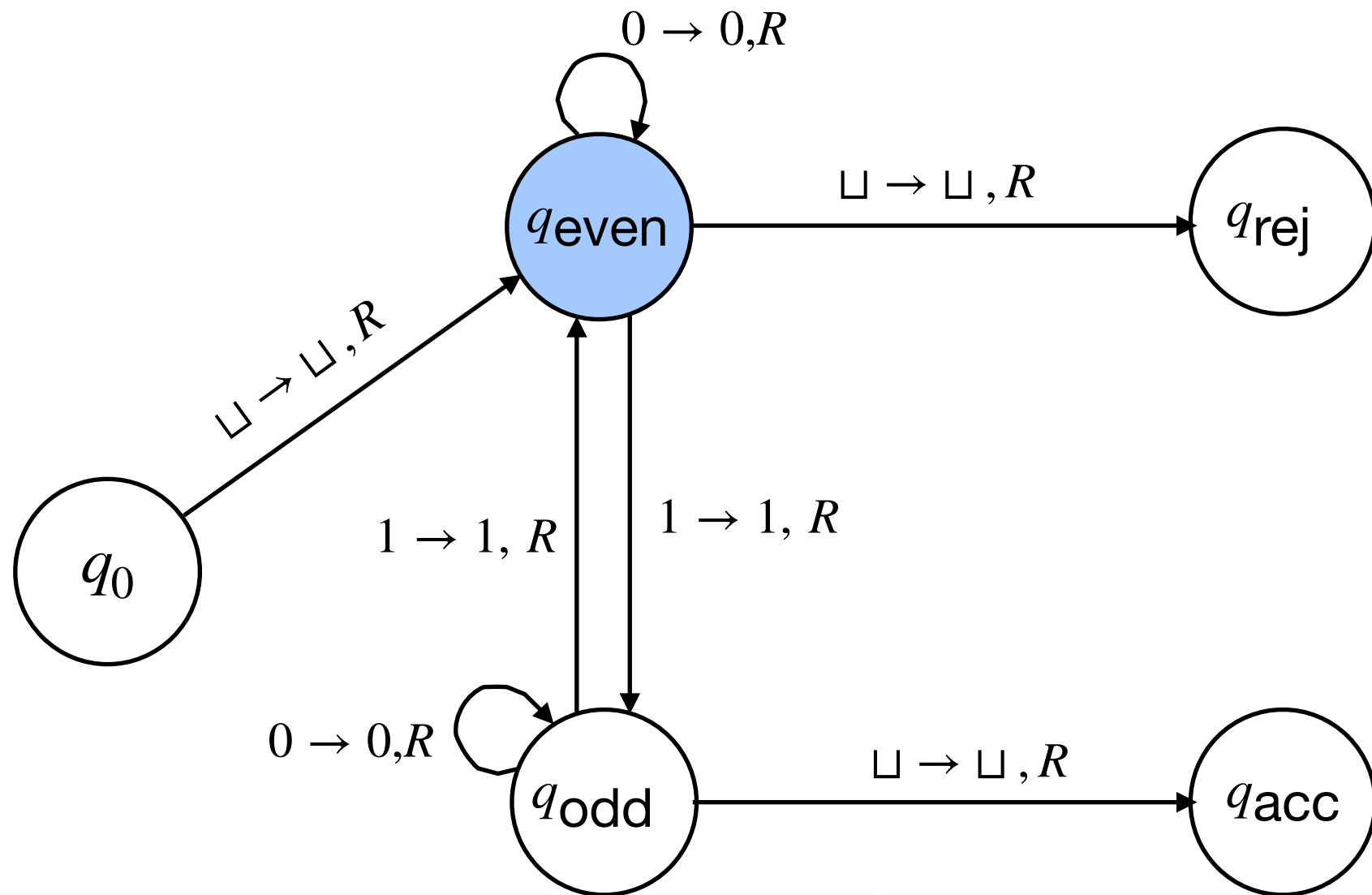


Example: Parity

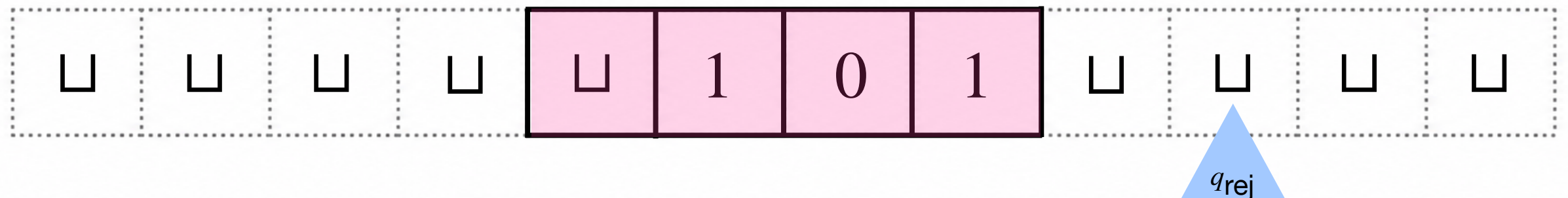
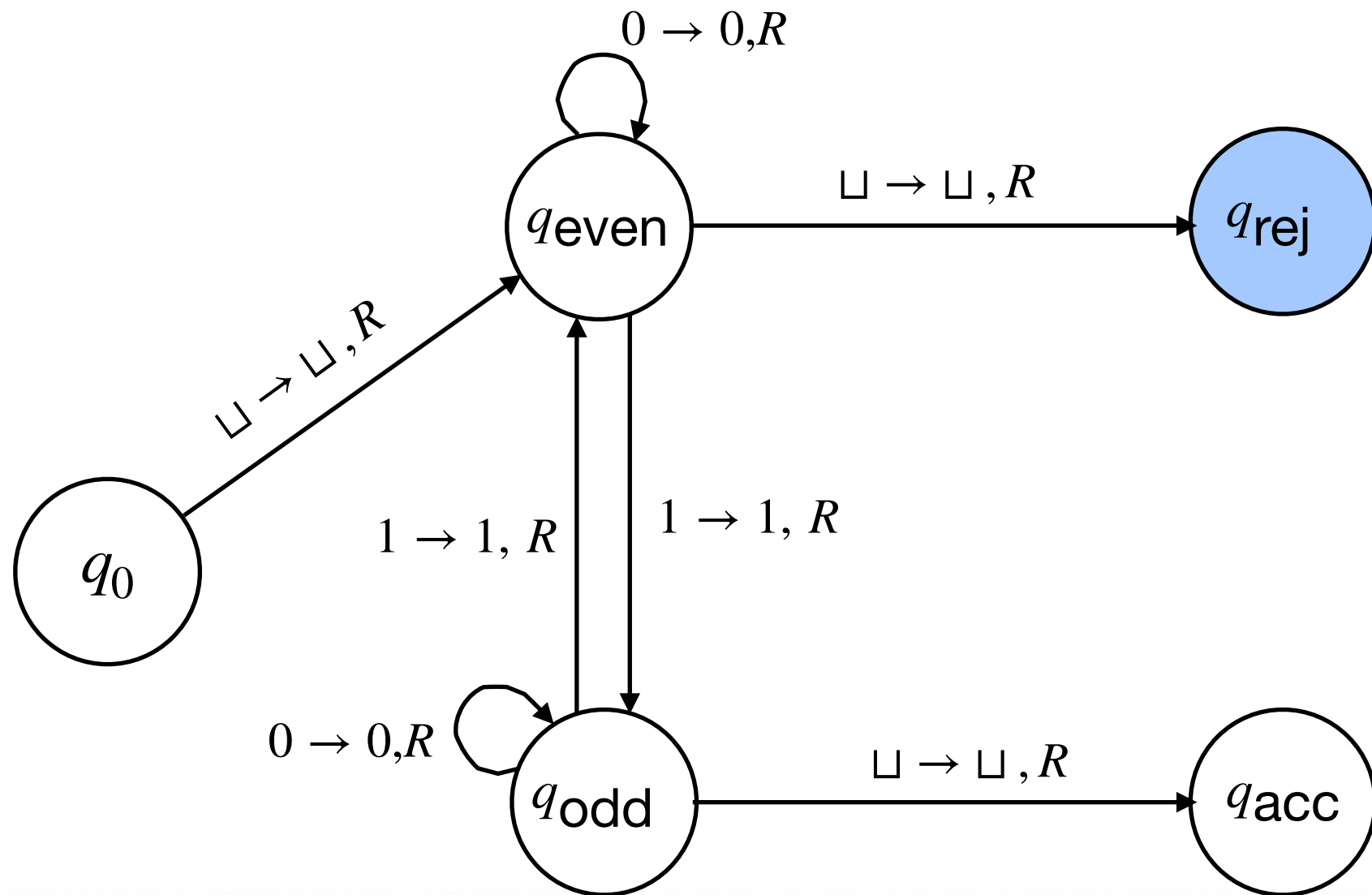


q_{odd}

Example: Parity



Example: Parity



**Which problems can be solved through
computation?**

Deciding a Language

- Let M be a Turing Machine and let $L \subseteq \{0,1\}^*$.
- We say that M **decides** L if
 - M accepts every $x \in L$.
 - M rejects every $x \in \{0,1\}^* \setminus L$.
- ◎ This is mathematical model of what it means to “**solve a problem**”.

Examples of Decidable Languages

Example: Strings containing 01

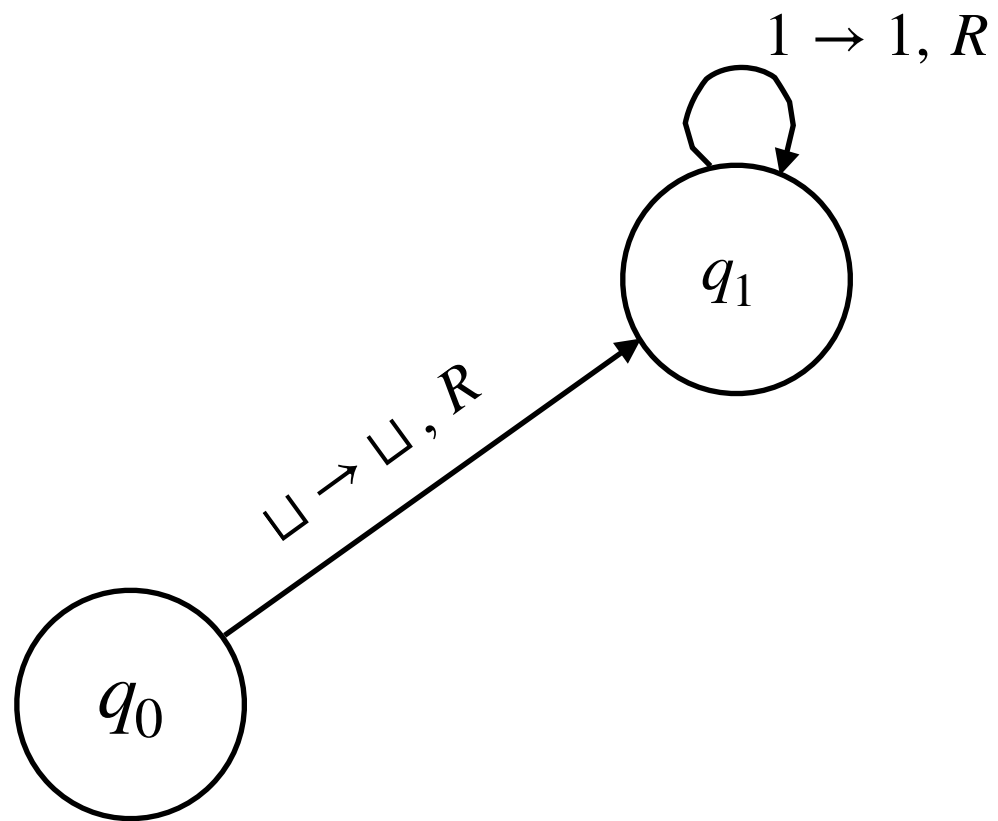
- **Informal Problem Statement:** “Given $x \in \{0,1\}^*$, determine whether it contains 01 as a substring”.

- **Can be formulated as:**

$$L_{01} = \{w \in \{0,1\}^* \text{ such that } w \text{ contains substring } 01\} .$$

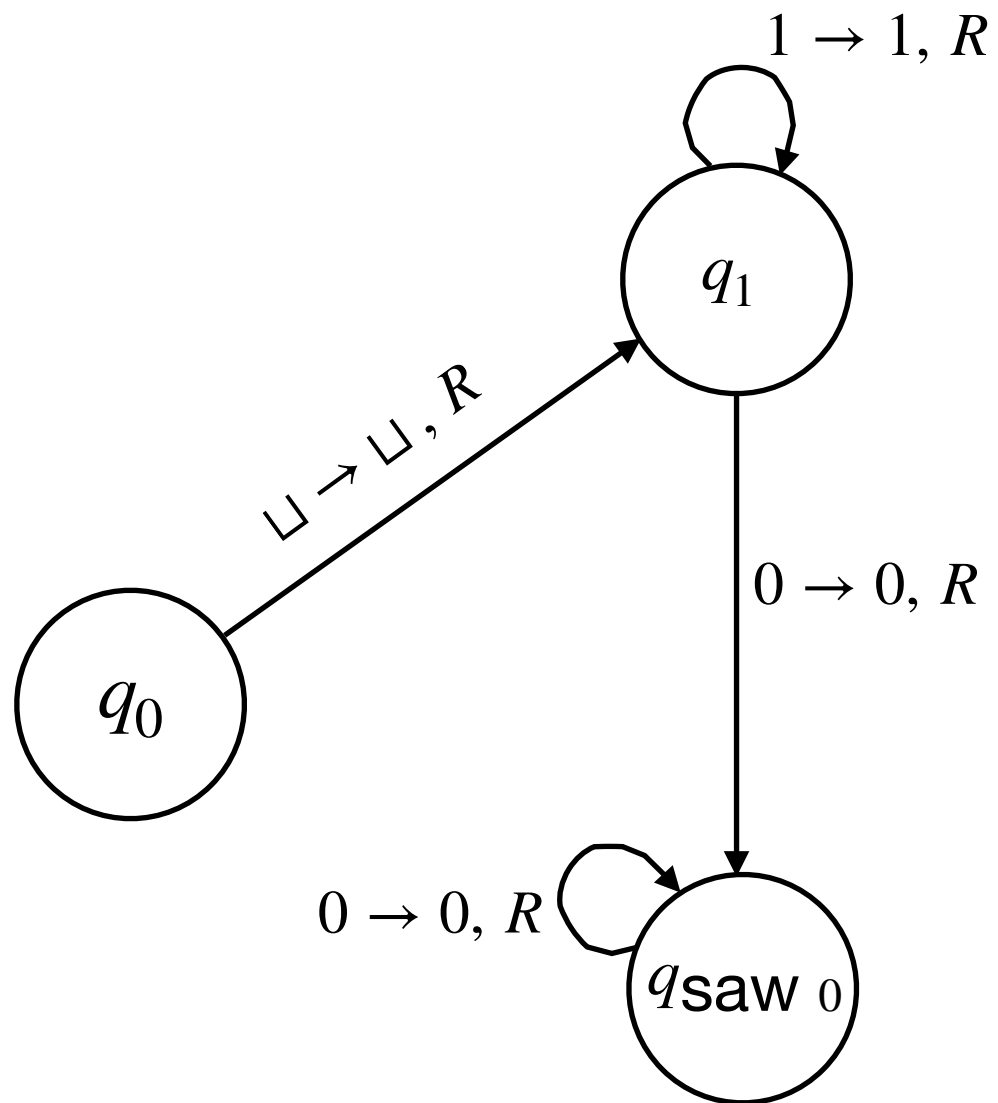
- Design a Turing Machine for L_{01} .

Example: L_{01}



- q_0 = "Start State"
- q_1 = "Haven't seen a 0 yet"

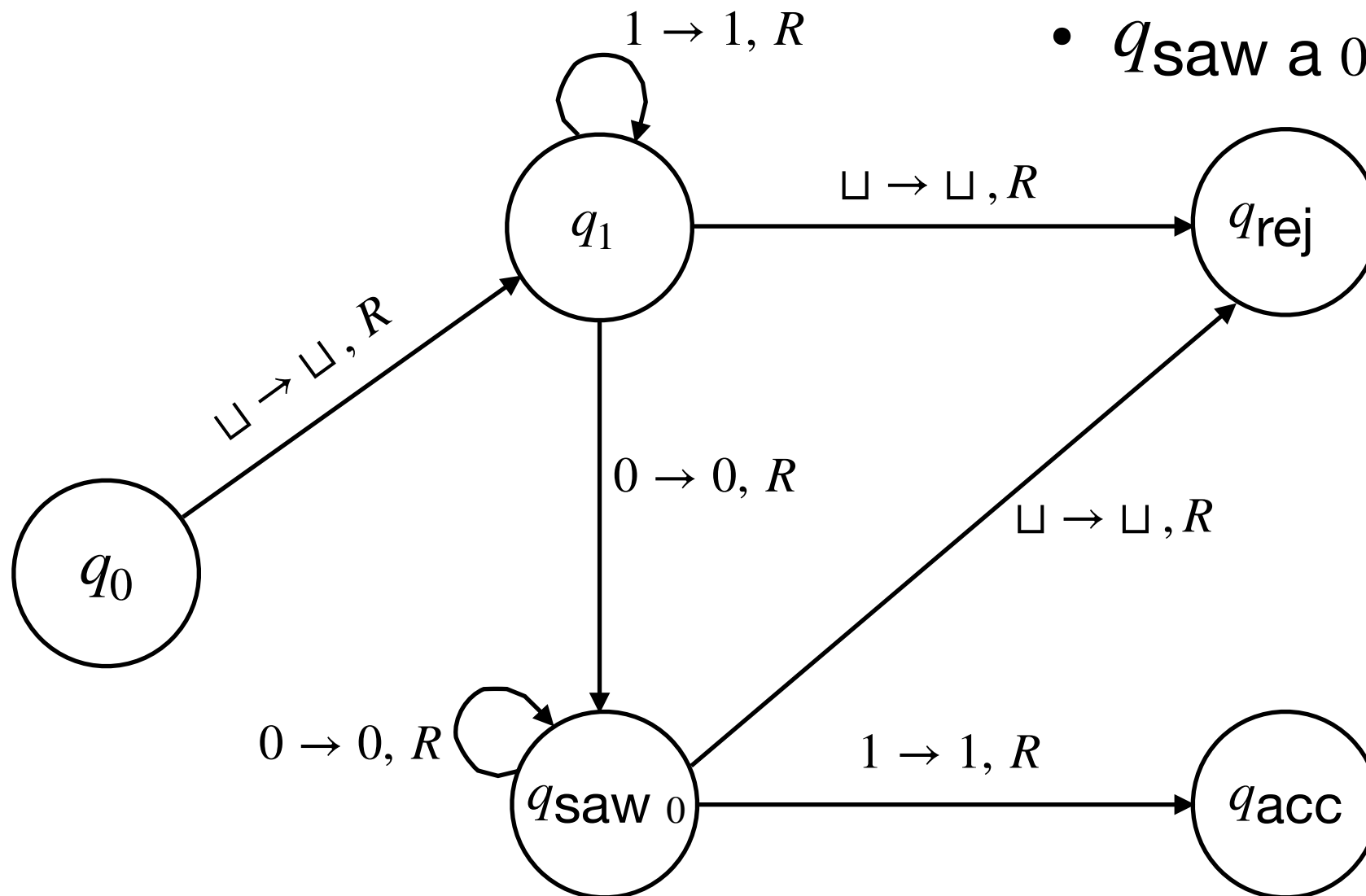
Example: L_{01}



- q_0 = "Start State"
- q_1 = "Haven't seen a 0 yet"
- $q_{\text{saw } 0}$ = "saw a 0"

Example: L_{01}

- q_0 = "Start State"
- q_1 = "Haven't seen a 0 yet"
- $q_{\text{saw } 0}$ = "saw a 0"



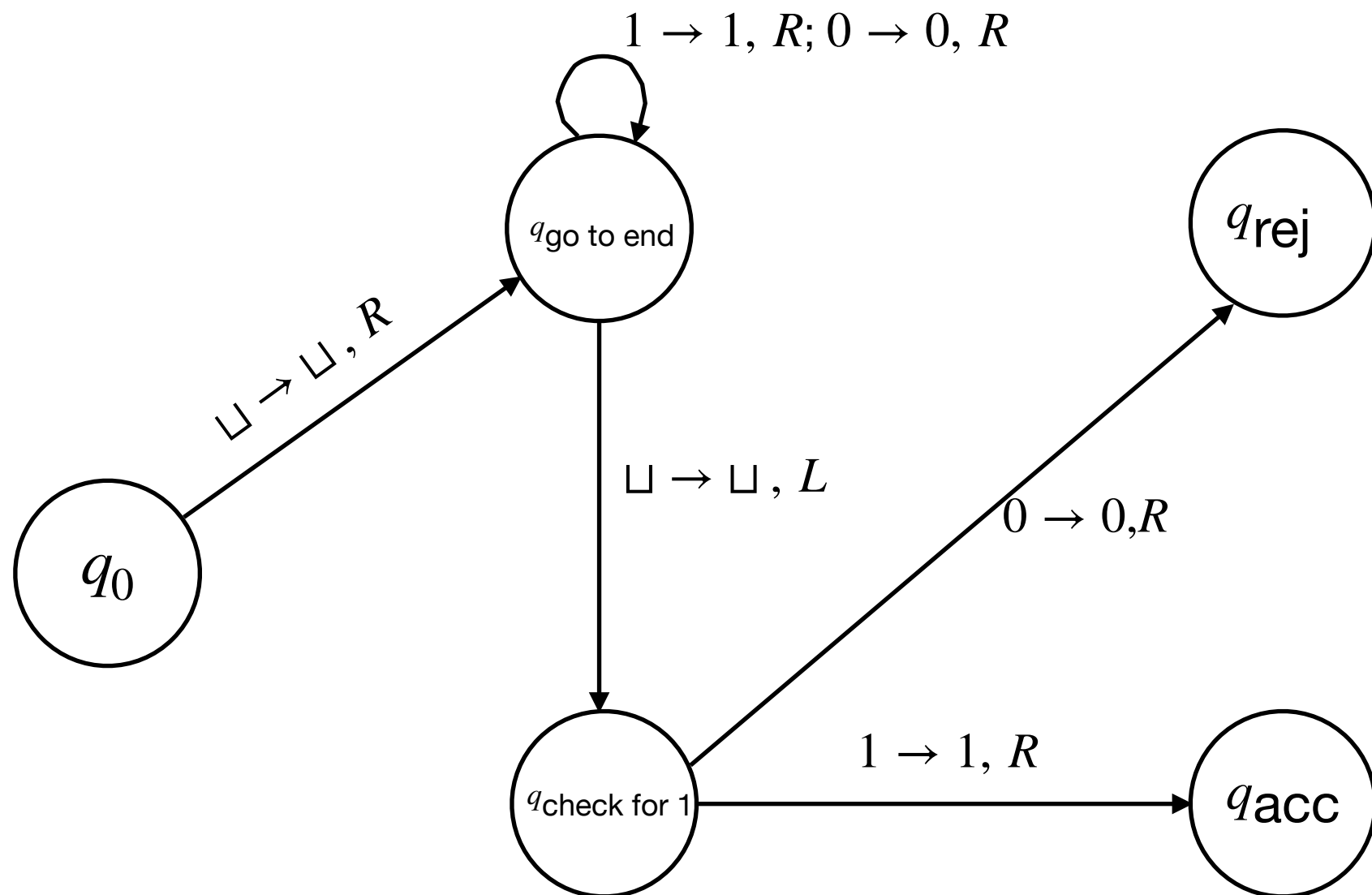
Example: Strings Ending in 1

- **Informal Problem Statement:** “Given $x \in \{0,1\}^*$, determine whether it ends in 1”.
- **Can be formulated as:**

$$L_1 = \{w \in \{0,1\}^* \text{ such that } w \text{ ends in } 1\} .$$

- Design a Turing Machine for L_1 .

Example: L_1

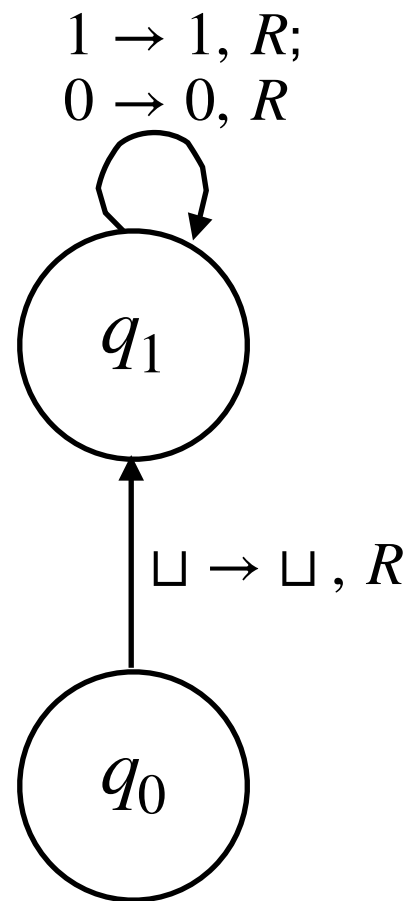


More Examples

Example: TM for COPY

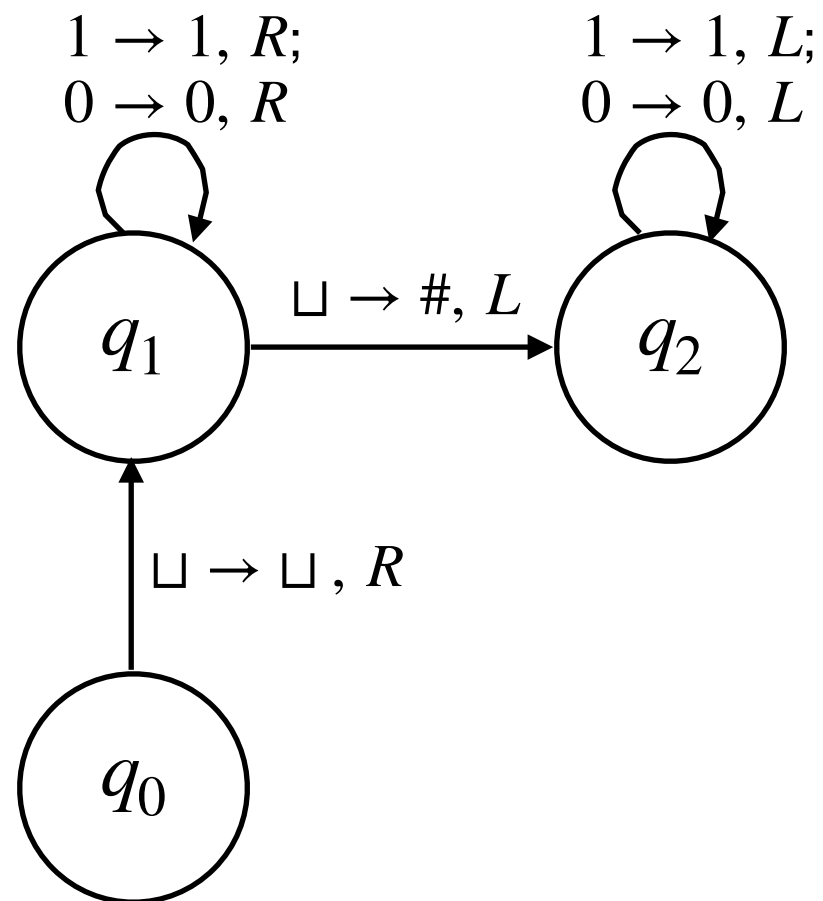
- Takes as input a string $x \in \{0,1\}^*$ and replaces it with $x\#x$ on the tape and accepts.

Example: TM for COPY



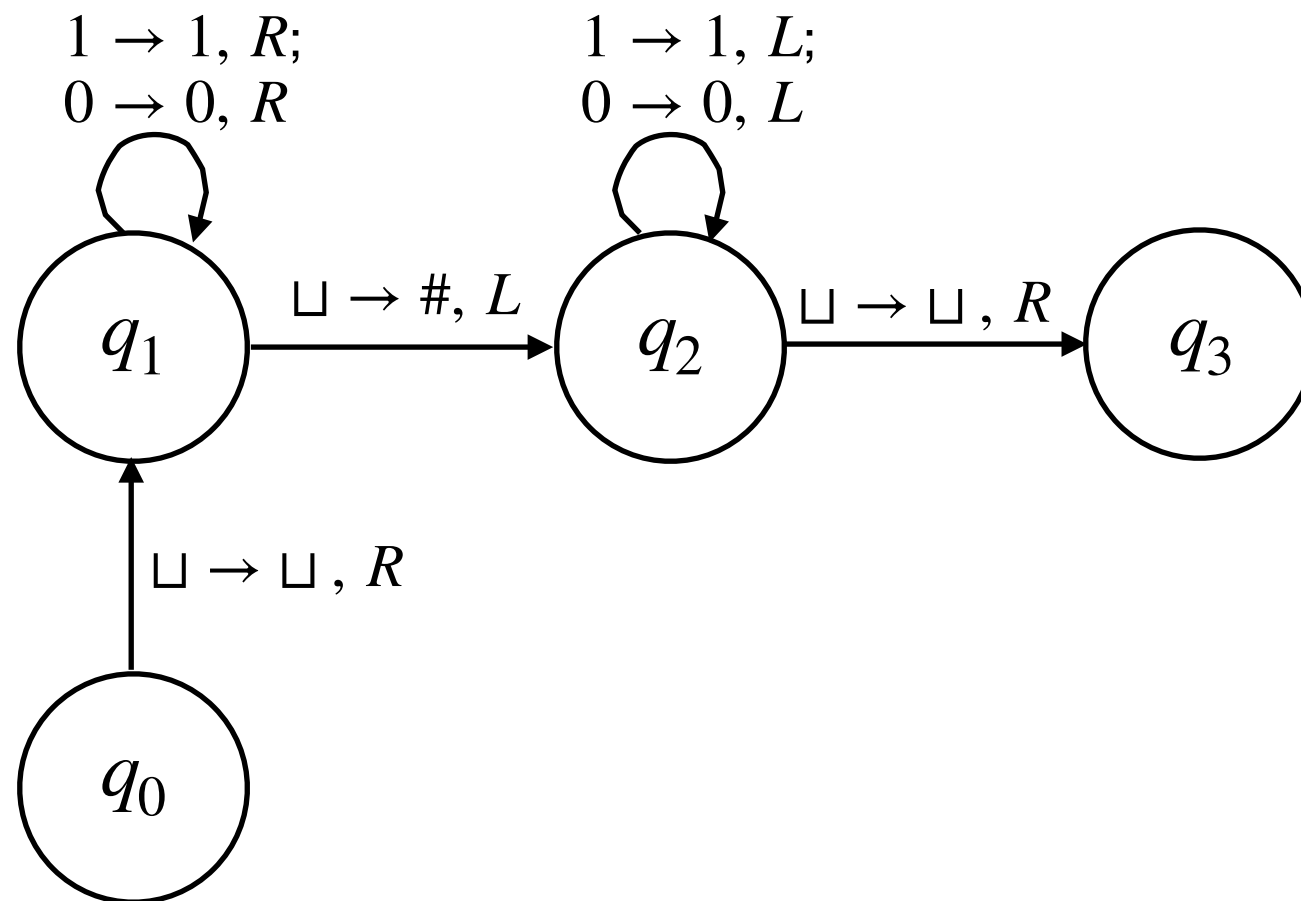
- q_0 = "Start State"
- q_1 = "Append # to end of w"

Example: TM for COPY



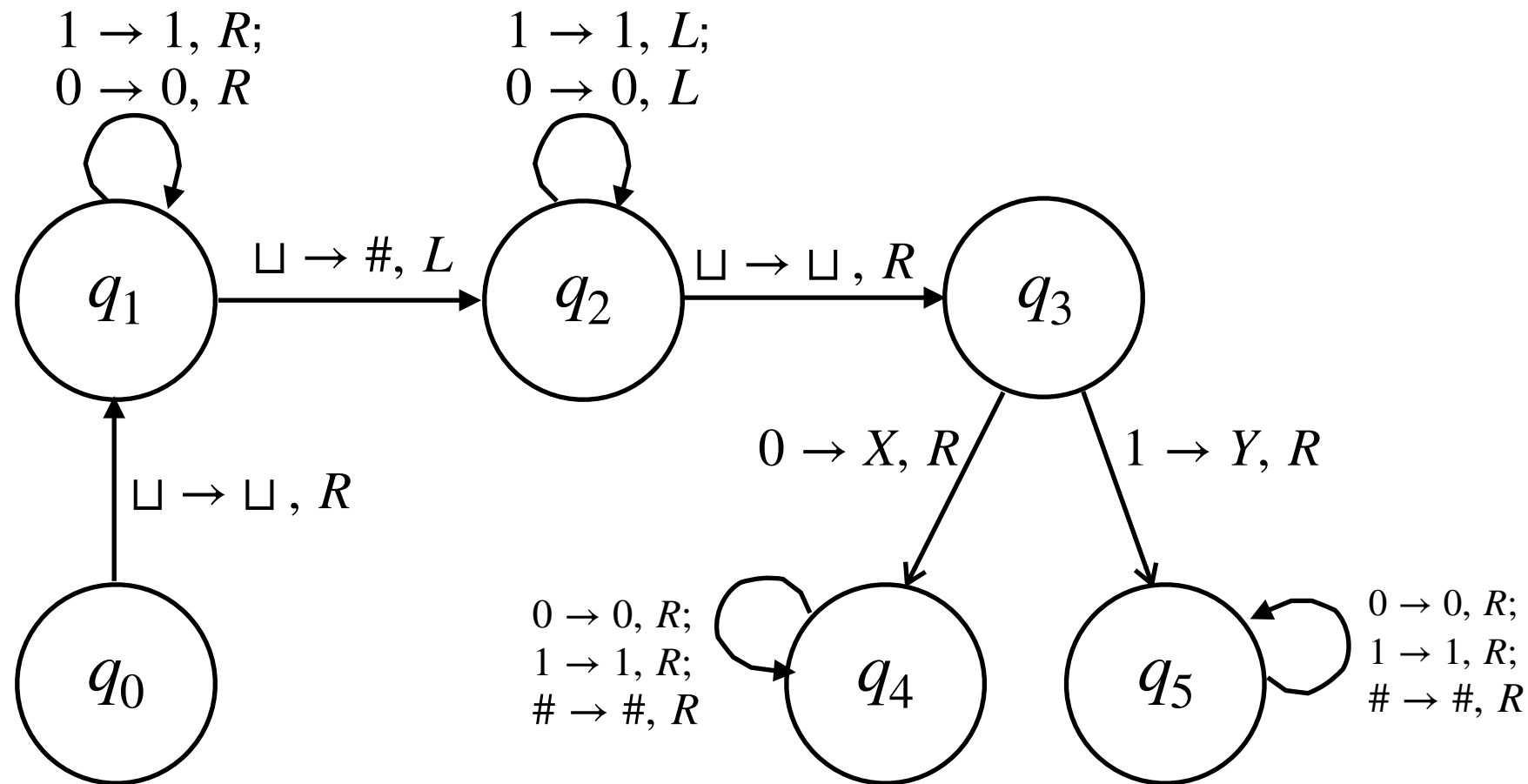
- q_0 = "Start State"
- q_1 = "Append # to end of w"
- q_2 = "Goto beginning of string"

Example: TM for COPY



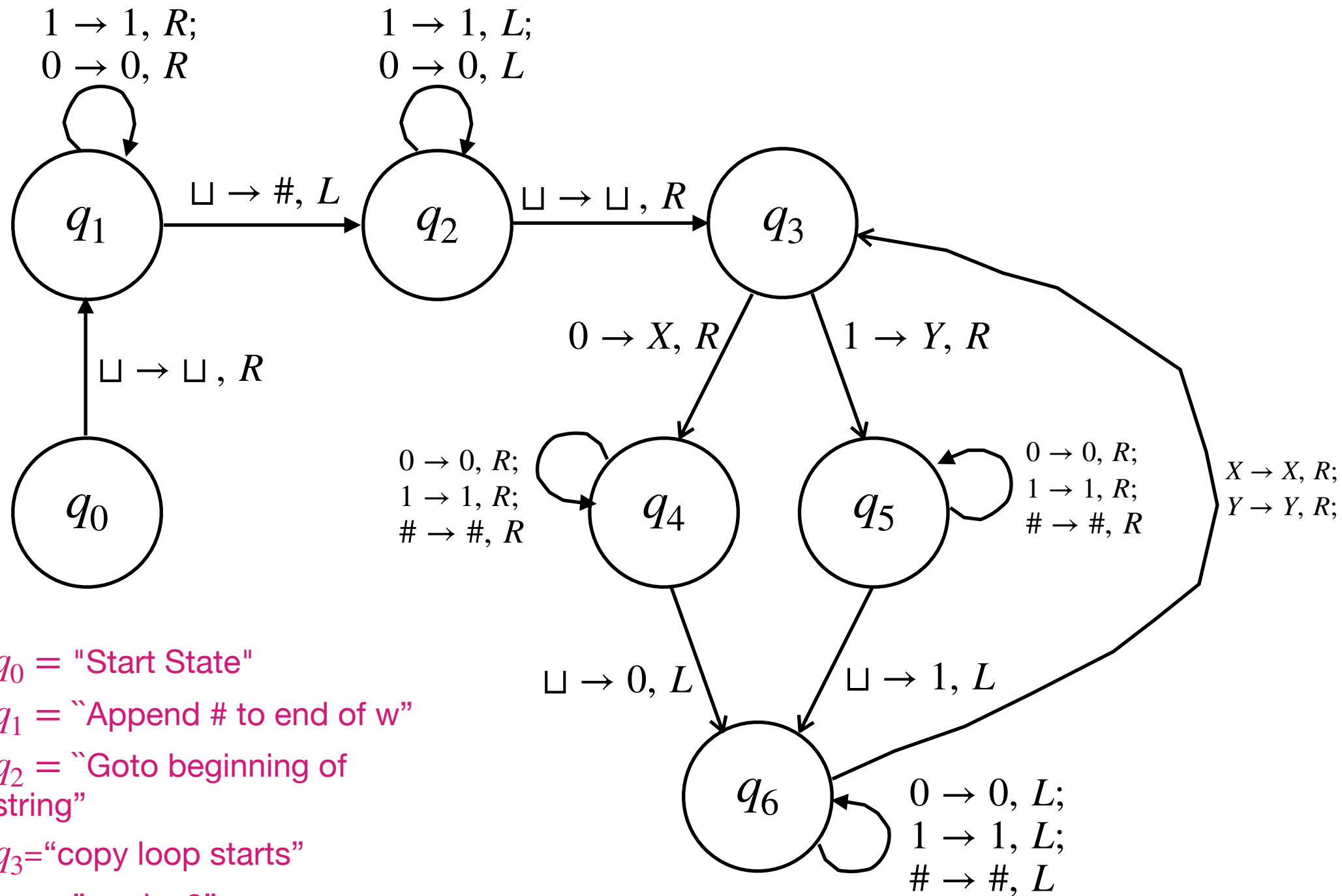
- q_0 = "Start State"
- q_1 = "Append # to end of w"
- q_2 = "Goto beginning of string"
- q_3 = "copy loop starts"

Example: TM for COPY



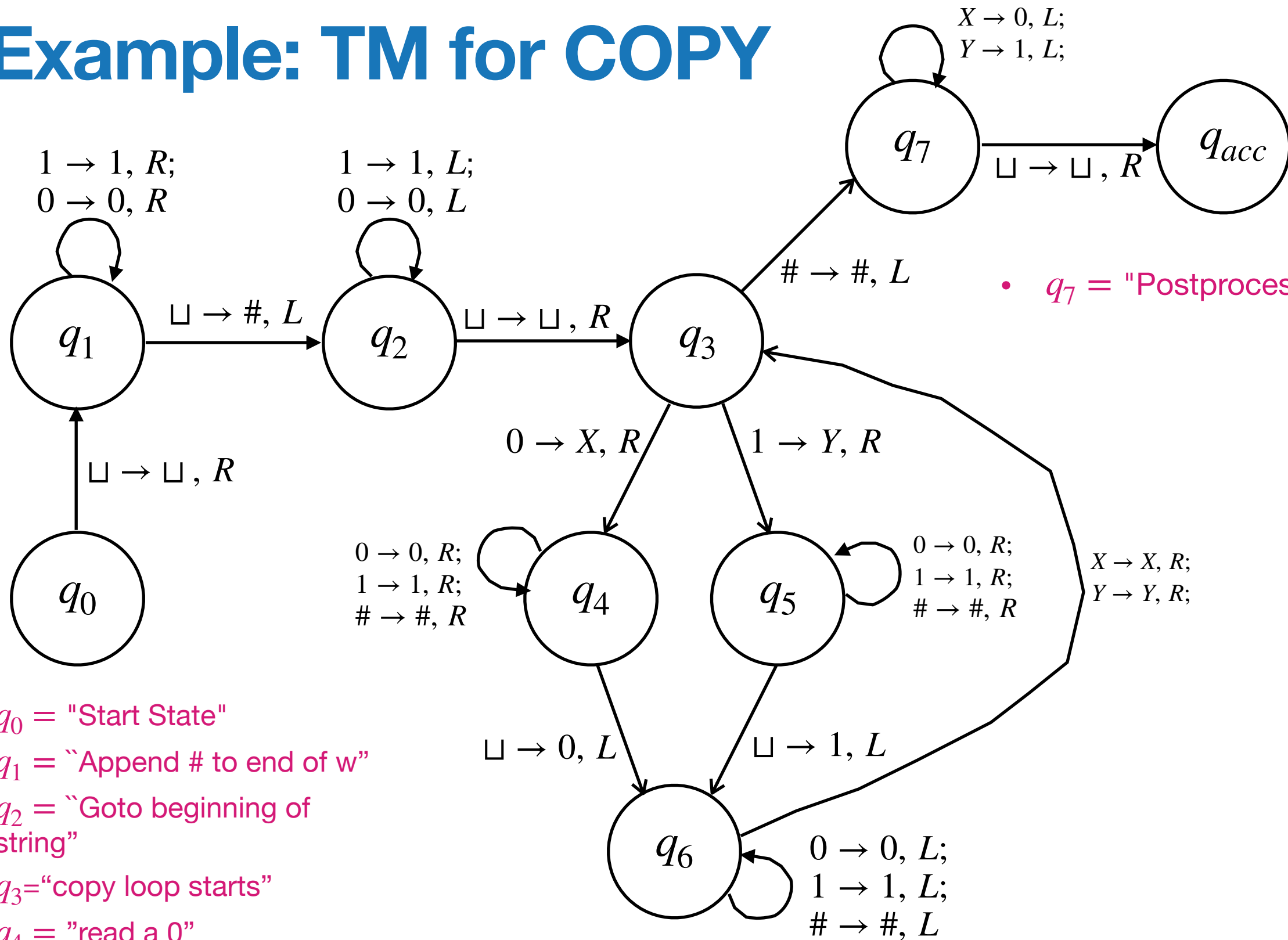
- q_0 = "Start State"
- q_1 = "Append # to end of w"
- q_2 = "Goto beginning of string"
- q_3 = "copy loop starts"
- q_4 = "read a 0"
- q_5 = "read a 1"

Example: TM for COPY



- q_0 = "Start State"
- q_1 = "Append # to end of w"
- q_2 = "Goto beginning of string"
- q_3 = "copy loop starts"
- q_4 = "read a 0"
- q_5 = "read a 1"
- q_6 = "copy a symbol and go back to first uncopied symbol"

Example: TM for COPY



• q_7 = "Postprocessing"

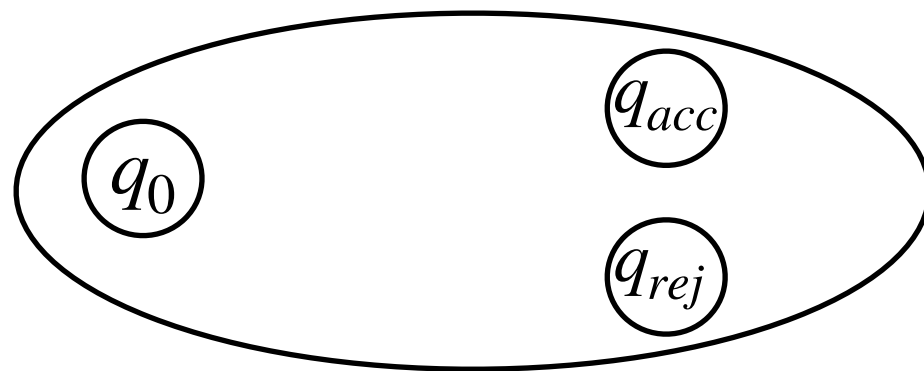
- q_0 = "Start State"
- q_1 = "Append # to end of w"
- q_2 = "Goto beginning of string"
- q_3 = "copy loop starts"
- q_4 = "read a 0"
- q_5 = "read a 1"
- q_6 = "copy a symbol and go back to first uncopied symbol"

Complement of a Decidable Language

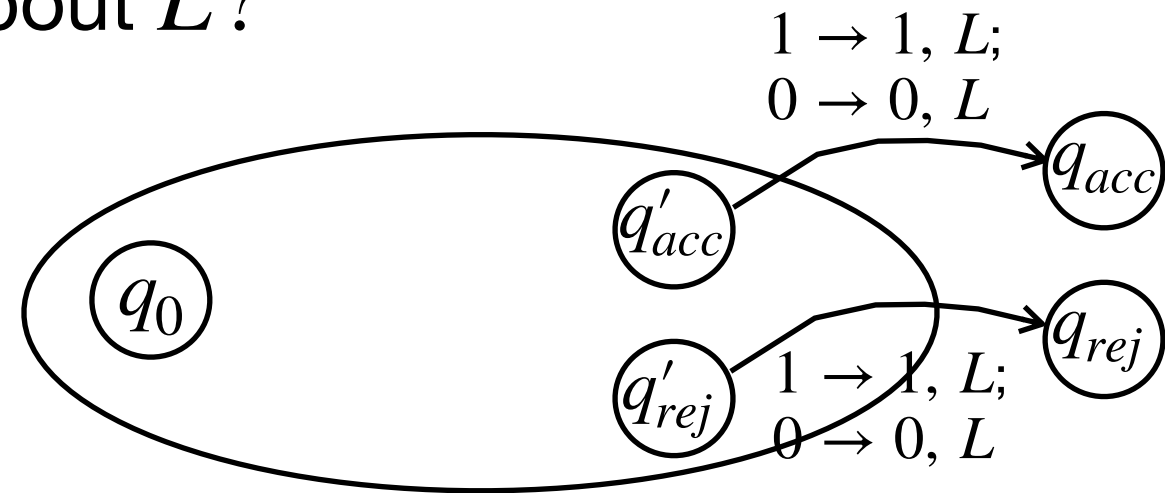
- The complement of a language L , denoted $\bar{L} = \{x \in \{0,1\}^* \setminus L\}$.
- If L is decidable, then what about $\bar{L}$?

Complement of a Decidable Language

- The complement of a language L , denoted $\bar{L} = \{x \in \{0,1\}^* \setminus L\}$.
- If L is decidable, then what about $\bar{L}$?



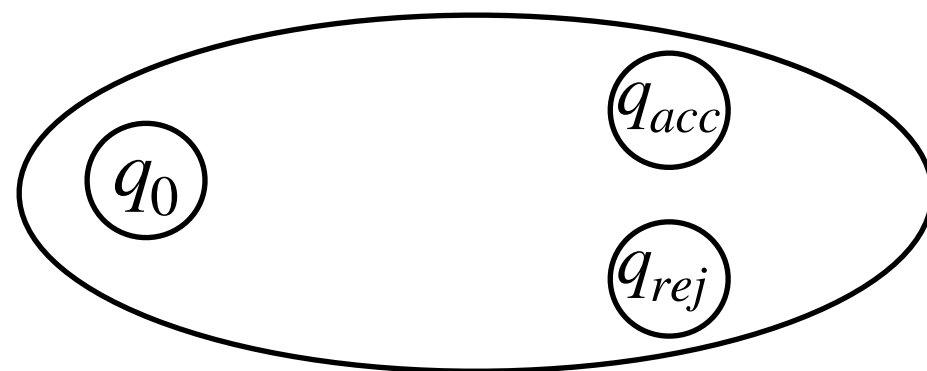
Turing Machine for L



Turing Machine for $\bar{L}$

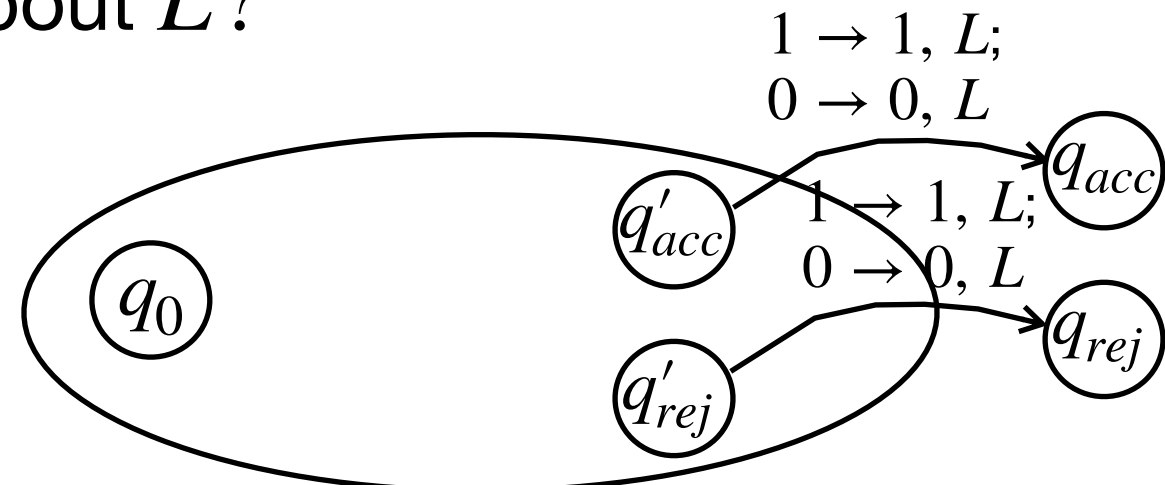
Complement of a Decidable Language

- The complement of a language L , denoted $\bar{L} = \{x \in \{0,1\}^* \setminus L\}$.
- If L is decidable, then what about $\bar{L}$?



Turing Machine for L

$$M_L = \{Q_L, q_0, q_{acc}, q_{rej}, \Sigma, \delta, \sqcup\}$$



Turing Machine for $\bar{L}$

$$M_{\bar{L}} = \{Q_{\bar{L}}, q_0, q_{acc}, q_{rej}, \Sigma, \delta_{\bar{L}}, \sqcup\}$$

$$Q_{\bar{L}} = Q_L \cup \{q'_{acc}, q'_{rej}\}$$

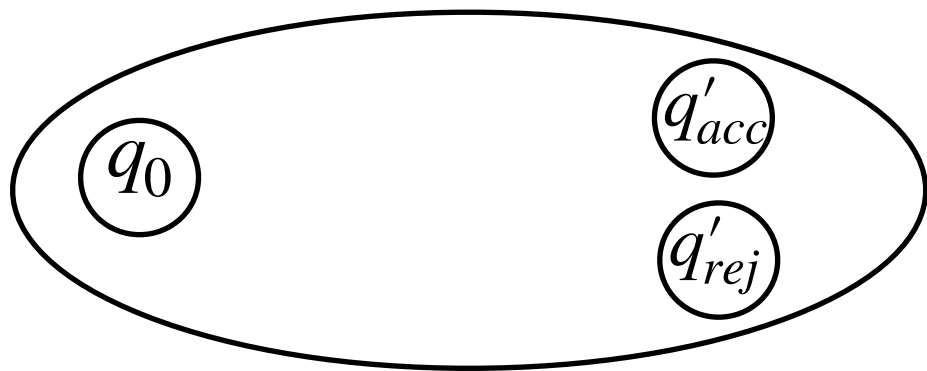
$$\delta_{\bar{L}} : Q_{\bar{L}} \times \Sigma_{\bar{L}} \rightarrow Q_{\bar{L}} \times \Sigma_{\bar{L}} \times \{L, R\}$$

Reverse of a Decidable Language

- Given a decidable language L , is the language $L^R = \{x^R \mid x \in L\}$ decidable?

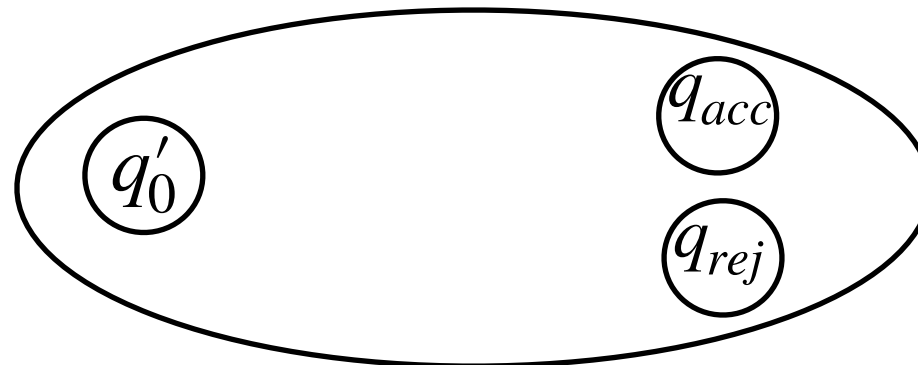
Reverse of a Decidable Language

- Given a decidable language L , is the language $L^R = \{x^R \mid x \in L\}$ decidable?



Reverser Turing Machine:

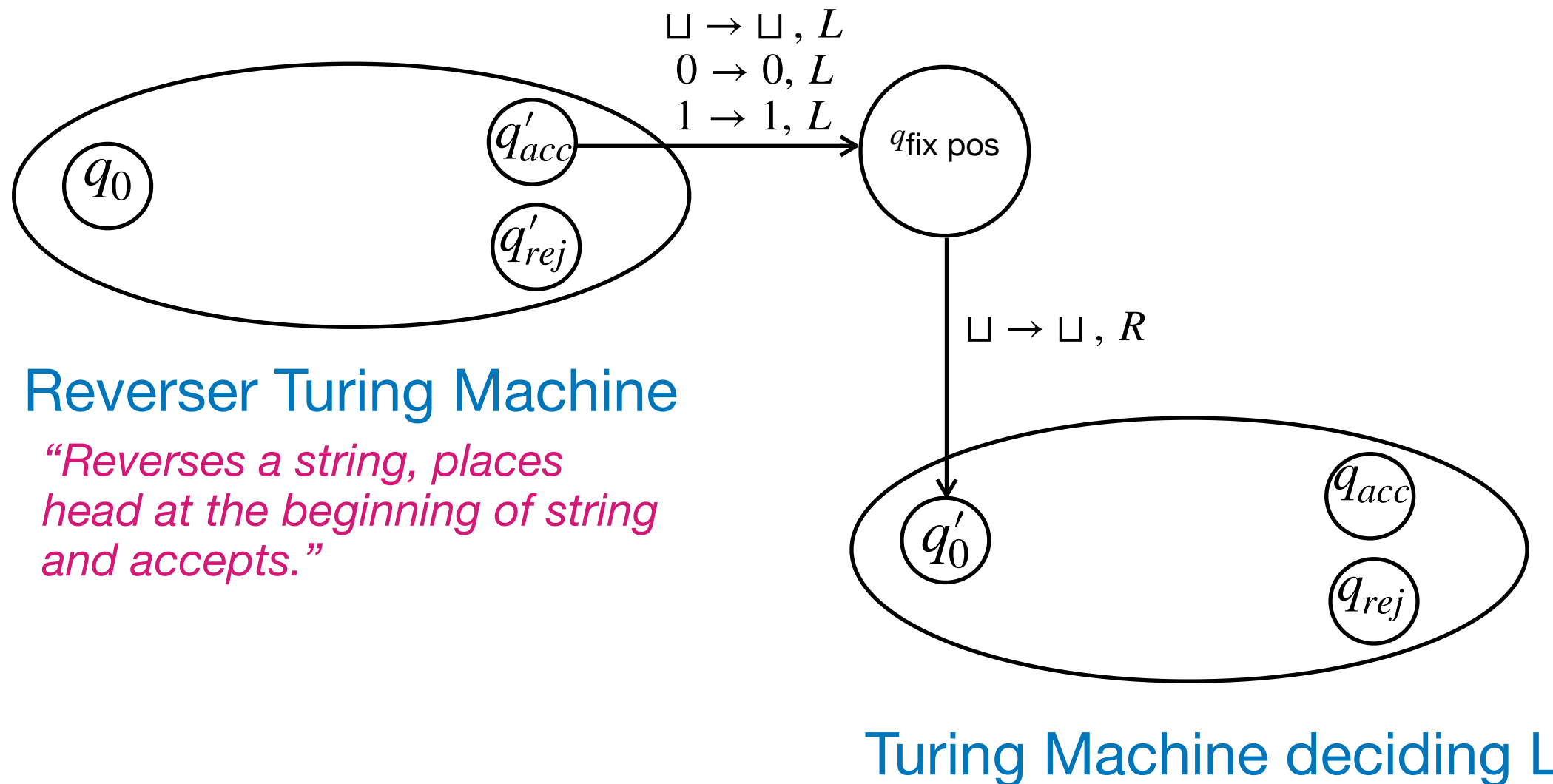
“Reverses a string, places head at the beginning of string and accepts.”



Turing Machine deciding L

Reverse of a Decidable Language

- Given a decidable language L , is the language $L^R = \{x^R \mid x \in L\}$ decidable?



A Note on Representation of TMs

- If we formally define a Turing Machine for each task, it can begin to get fairly complicated.
- As we said earlier, Turing machines are a way to mathematically reason about algorithms. Our goal this week and next is to get comfortable with low-level details of Turing machines to convince ourselves that they capture all algorithms.
- Once we are friends with Turing Machines, we will use them in subsequent classes to discuss and reason about computation, but our focus will be on algorithms.
- More examples of this next week.

Properties of Decidable Languages

- **Claim:** Decidable languages are closed under intersection and union.

Summary

- **Turing Machine** is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$.
- Computational history of a Turing machine on an input $w \in \{0,1\}^*$ is a series of configurations $C_1, C_2, \dots, C_T$, where $C_{i+1} = \text{NEXT}(C_i)$.
- Examples of decidable languages and their closure properties.