Turing Machine

Turing Machine

- **Mathematical model** of a **symbolic computing** device.
- Proposed by **Alan Turing** in **1935**[a].

[a] **On Computable Numbers, with an application to the Entschidungsproblem**, *Proc, London Mathematical Society*, 1936, pp. 230-265.
Alan Turing the enigma, by Andrew Hodges, Pub. Vintage, ISBN 0099116413.

Turing Machine : Mathematical Definition

A **Turing Machine (TM) M** can be specified by a **7-tuple**[a] of data, $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where

- **Q** is a **finite set** of **states**.
- **Σ** is the **input alphabet** and the special symbol **blank** (⊔) is not in **Σ**.
- **Γ** is the **tape alphabet**, $\Sigma \subset \Gamma$ and **blank** (⊔) is in **Γ**.

[a] **Introduction to the Theory of Computation**, by **Michael Sipser**, Brooks/Cole (Thompson Learning), ISBN 981-240-226-8.

Turing Machine : Mathematical Definition

- $\delta : \mathbf{Q} \times \Gamma \longrightarrow \mathbf{Q} \times \Gamma \times \{\mathbf{L}, \mathbf{R}\}$ is the **state transition function**.
- $\mathbf{q_0} \in \mathbf{Q}$ is the **start state**.
- $\mathbf{q_a} \in \mathbf{Q}$ is the **accept halt state**.
- $\mathbf{q_r} \in \mathbf{Q}$ is the **reject halt state**.

Turing Machine : Physical View

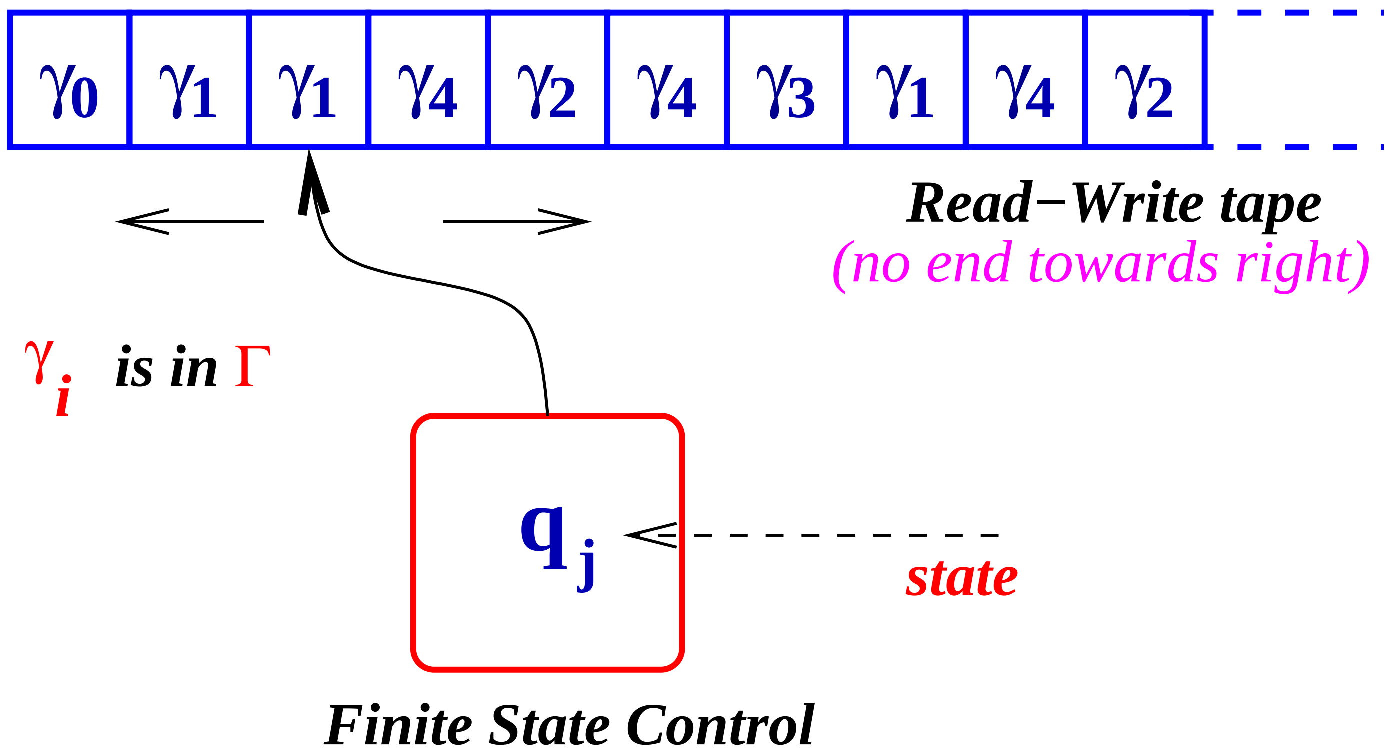

Figure 1: **Turing Machine**

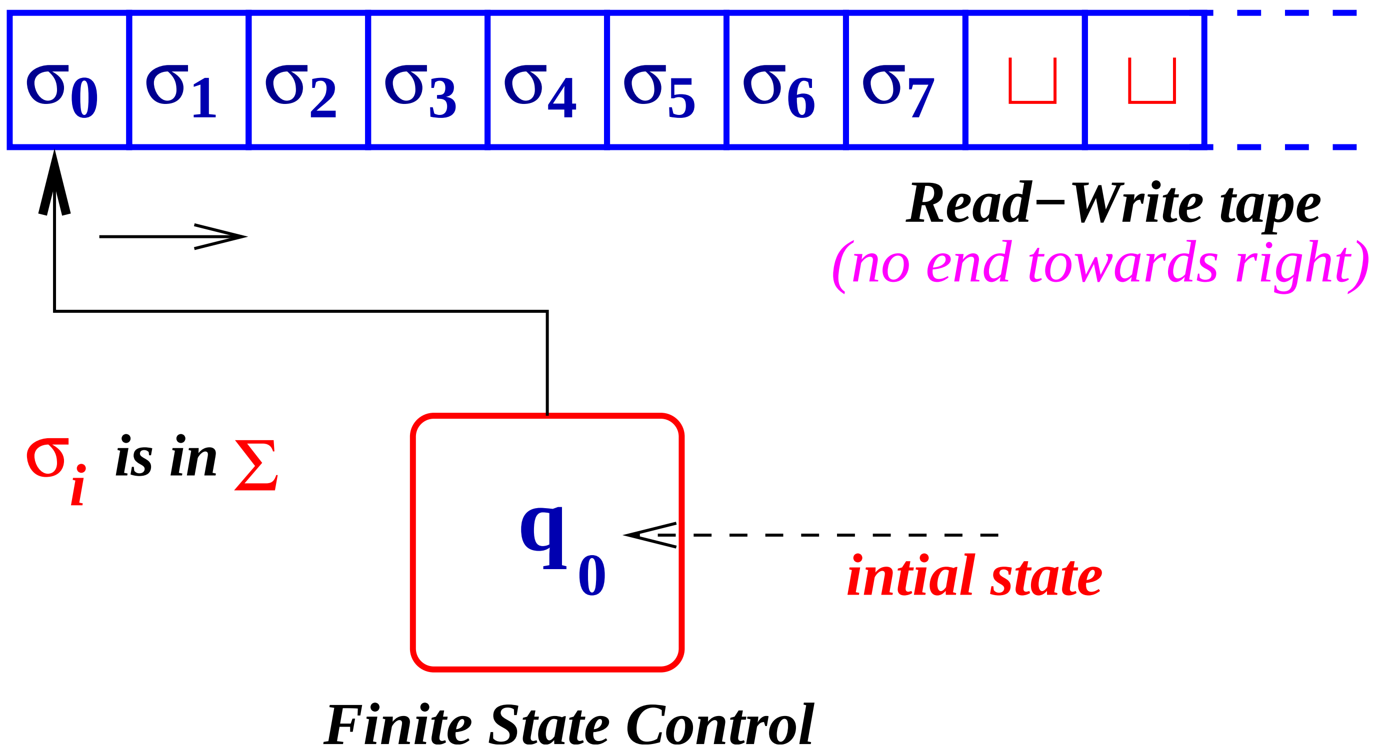

Figure 2: **TM : Initial Configuration**

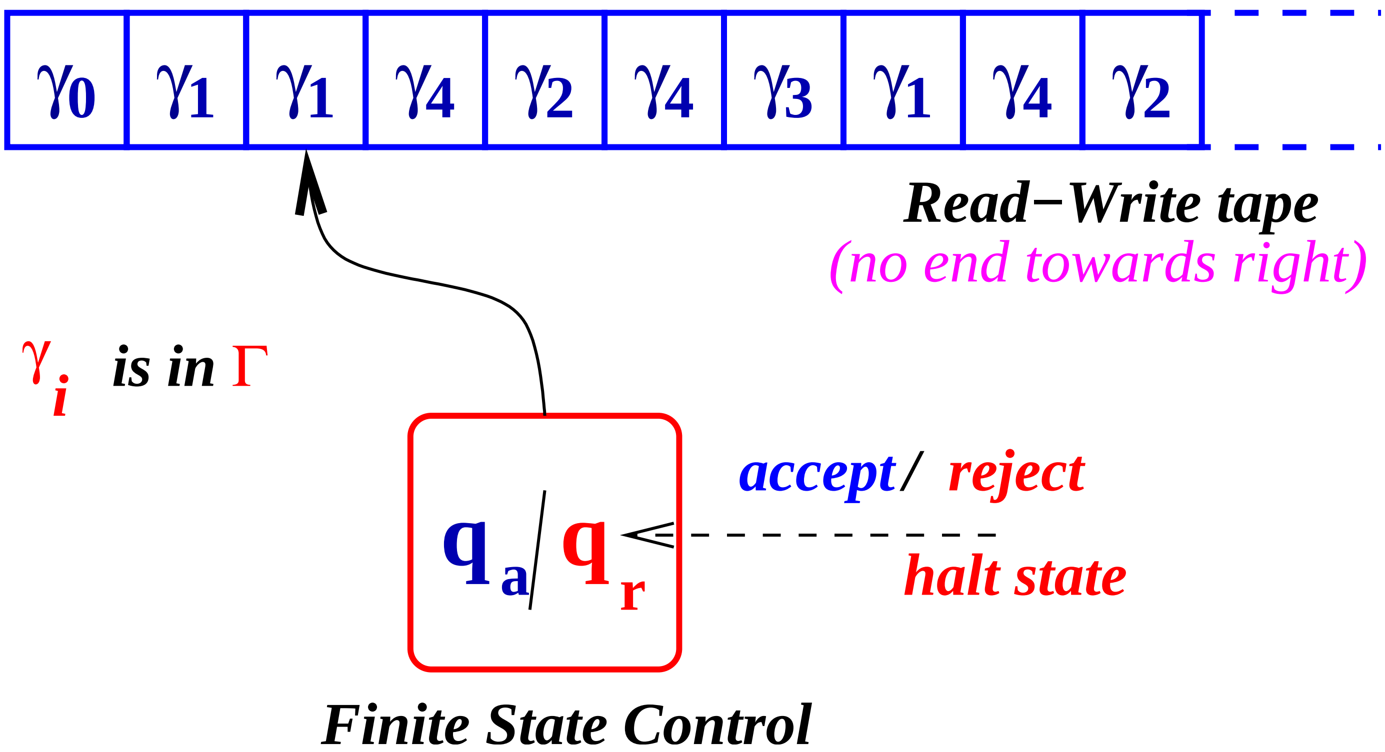

Figure 3: **TM : Halt Configurations**

Turing Machine : One Step Computation

Curr. St.	In. Sym.	Trans. Fun.	Nxt, St.	Sym. Wr.	Mv.
$\mathbf{q}$	γ_i	$((\mathbf{q}, \gamma_i), (\mathbf{p}, \gamma_j, \mathbf{L}))$	$\mathbf{p}$	γ_j	$\mathbf{L}$
$\mathbf{q}$	γ_i	$((\mathbf{q}, \gamma_i), (\mathbf{p}, \gamma_j, \mathbf{R}))$	$\mathbf{p}$	γ_j	$\mathbf{R}$

If the **scanned square** is the **leftmost**, there will be no **left movement**.

Turing Machine : An Example

Consider the following TM, **M**,

$$
\begin{aligned}
Q &= \{q_0, q_1, q_a, q_r\}, \\
\Sigma &= \{1\}, \\
\Gamma &= \{1, \sqcup\}, \\
\delta &= \left\{ \begin{array}{l}
((q_0, 1), (q_1, 1, R)), \\
((q_1, 1), (q_0, 1, R)), \\
((q_0, \sqcup), (q_a, \sqcup, L)), \\
((q_1, \sqcup), (q_r, \sqcup, L))
\end{array} \right\}
\end{aligned}
$$

Turing Machine Computation : An Example

Start

Tape: $\uparrow_{q_0}$ 1 1 1 1 **Trans:** $((q_0, 1), (q_1, 1, R))$

Tape: 1 $\uparrow_{q_1}$ 1 1 1 **Trans:** $((q_1, 1), (q_0, 1, R))$

Tape: 1 1 $\uparrow_{q_0}$ 1 1 **Trans:** $((q_0, 1), (q_1, 1, R))$

Tape: 1 1 1 $\uparrow_{q_1}$ 1 **Trans:** $((q_1, 1), (q_0, 1, R))$

Tape: 1 1 1 1 $\uparrow_{q_0}$ $\sqcup$ **Trans:** $((q_0, \sqcup), (q_a, \sqcup, L))$

Tape: 1 1 1 $\uparrow_{q_a}$ 1

Halt

Turing Machine Configuration

A turing machine **configuration** is a snap-shot of its computation.

Tape: 1 1 1 $\uparrow_{q_1}$ 1

A *configuration* **C** is an element of $\mathbf{\Gamma^* \times Q \times \Gamma^*}$. Let $\mathbf{C = (x, q, y)}$.

- The **head** is going to **read** the **leftmost symbol** of $\mathbf{y} \neq \varepsilon$.
- If $\mathbf{x} = \varepsilon$, then the **head** is scanning the **leftmost square** of the tape.

Computation as a Binary Relation

- An **one step computation** may be viewed as a **binary relation ($\Rightarrow$)** over the collection of **configurations** $\mathcal{C}_M$ of the TM **M**.
- Let $\mathbf{C_1}, \mathbf{C_2} \in \mathcal{C}_\mathbf{M}$, then we may have the following possibilities.

Computation as a Binary Relation

- $C_1 = (x\gamma, q, \gamma_1 y)$, $\mathbf{C_2} = (\mathbf{x}, \mathbf{p}, \gamma\gamma_2\mathbf{y})$, and $\mathbf{delta}(\mathbf{q}, \gamma_1) = (\mathbf{p}, \gamma_2, \mathbf{L})$.
- $C_1 = (x, q, \gamma_1\gamma y)$, $\mathbf{C_2} = (\mathbf{x}\gamma_2, \mathbf{p}, \gamma\mathbf{y})$, and $\delta(\mathbf{q}, \gamma_1) = (\mathbf{p}, \gamma_2, \mathbf{R})$.
- $C_1 = (x, q, \gamma_1)$, $\mathbf{C_2} = (\mathbf{x}\gamma_2, \mathbf{p}, \sqcup)$, and $\delta(\mathbf{q}, \gamma_1) = (\mathbf{p}, \gamma_2, \mathbf{R})$.
- $C_1 = (\varepsilon, q, \gamma_1 y)$, $\mathbf{C_2} = (\varepsilon, \mathbf{p}, \gamma_2\mathbf{y})$, and $\delta(\mathbf{q}, \gamma_1) = (\mathbf{p}, \gamma_2, \mathbf{L})$, and
- $C_1 = (x\gamma, q, \gamma_1)$, $\mathbf{C_2} = (\mathbf{x}, \mathbf{p}, \gamma)$, and $\delta(\mathbf{q}, \gamma_1) = (\mathbf{p}, \sqcup, \mathbf{L})$.

Computation as a Binary Relation

A **configuration** $\mathbf{C_s}$ produces a **configuration** $\mathbf{C_d}$ after finite number of steps (may be zero) if

$$\mathbf{C_s} \Rightarrow^* \mathbf{C_d},$$

where $\Rightarrow^*$ is the **reflexive-transitive closure** of $\Rightarrow$.

In other words, either $\mathbf{C_s} = \mathbf{C_d}$ or there are $n > 1$ **configurations**, $\mathbf{C_1}, \cdots, \mathbf{C_n}$, so that $\mathbf{C_1} = \mathbf{C_s}$ and $\mathbf{C_n} = \mathbf{C_s}$ and $\mathbf{C_i} \Rightarrow \mathbf{C_{i+1}}$, for $1 \leq i < n$.

Start and Halting Configurations

- **Start Configuration:** $(\varepsilon, \mathbf{q_0}, \mathbf{x} \in \mathbf{\Sigma}^*)$.
- **Halting Configuration:** $(\mathbf{x}, \mathbf{q_a}, \mathbf{y})$ - **accept halt**.
- **Halting Configuration:** $(\mathbf{x}, \mathbf{q_r}, \mathbf{y})$ - **reject halt**.

Language Accepted by a Turing Machine

Let **M** be a TM. The language **accepted** by **M**,

$$\mathbf{L}(\mathbf{M}) = \{\mathbf{x} \in \mathbf{\Sigma}^* : (\varepsilon, \mathbf{q_0}, \mathbf{x}) \Rightarrow^* (\mathbf{y}, \mathbf{q_a}, \mathbf{z})\}.$$

We may write $(\mathbf{x}, \mathbf{p}, \mathbf{y})$ as **xpy** provided $Q \cap \Gamma = \emptyset$.

Turing Acceptable and Decidable Languages

Let $\mathbf{L}$ be a language over the alphabet $\mathbf{\Sigma}$. The language $\mathbf{L}$ is called **Turing acceptable** or **Turing recognisable** if there is a Turing machine $\mathbf{M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)}$ so that $\mathbf{L(M) = L}$.

The language $\mathbf{L}$ is called **Turing decidable** if there is a Turing machine $\mathbf{M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)}$ so that $\mathbf{L(M) = L}$ and also $\mathbf{L(\overline{M}) = \overline{L}}$, where $\mathbf{\overline{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_r, q_a)}$ is same as $\mathbf{M}$ except the **accept** and the **reject halt states** are interchanged.

Turing Acceptable and Decidable Languages

- Every **Turing decidable** language is **Turing acceptable/recognisable**.
- But a **Turing recognisable** language may not be **Turing decidable** i.e. there is a Turing machine that on **any input** from the **language** enters the **accept state** and halts. But it may not (always) enter the **reject state** on some input outside the language.
- There are languages that are not even **Turing recognisable**.

Turing Decidable Language : An Example

Consider the language

$$\mathbf{L} = \{\mathbf{x} \in \{\mathbf{a}, \mathbf{b}\} : \mathbf{x} = \mathbf{x}^{\mathbf{reverse}} \textbf{ and } |x| \textbf{ is even}\}$$

Following is the Turing machine that decides **L**.

$$\begin{aligned} Q &= \{q_0, \cdots, q_6, q_7\} \\ \Sigma &= \{a, b\} \\ \Gamma &= \{a, b, \#, \sqcup\} \\ q_a &= q_6 \\ q_r &= q_7 \end{aligned}$$

$$\delta' = \left\{ \begin{array}{ll} ((q_0, a), (q_1, \#, R)), & ((q_0, b), (q_2, \#, R)), \\ & ((q_0, \sqcup), (q_6, \sqcup, R)), \\ ((q_1, a), (q_1, a, R)), & ((q_1, b), (q_1, b, R)), \\ & ((q_1, \sqcup), (q_3, \sqcup, L)), \\ ((q_2, a), (q_2, a, R)), & ((q_2, b), (q_2, b, R)), \\ & ((q_2, \sqcup), (q_4, \sqcup, L)), \\ ((q_3, a), (q_5, \sqcup, L)), & ((q_3, b), (q_7, b, R)), \\ & ((q_3, \#), (q_7, \sqcup, R)), \end{array} \right\}$$

$$\delta = \delta' \cup \left\{ \begin{array}{ll} ((q_4, b), (q_5, \sqcup, L)), & ((q_4, a), (q_7, a, R)), \\ & ((q_4, \#), (q_7, \sqcup, R)), \\ ((q_5, a), (q_5, a, L)), & ((q_5, b), (q_5, b, l)), \\ & ((q_5, \#), (q_0, \sqcup, R)), \end{array} \right\}$$

Computation : An Example

$$
\begin{aligned}
(\mathbf{q_0}abba) &\Rightarrow (\#\mathbf{q_1}bba) \\
&\Rightarrow (\#b\mathbf{q_1}ba) \\
&\Rightarrow (\#bb\mathbf{q_1}a) \\
&\Rightarrow (\#bba\mathbf{q_1}\sqcup) \\
&\Rightarrow (\#bb\mathbf{q_3}a) \\
&\Rightarrow (\#b\mathbf{q5}b) \\
&\Rightarrow (\#\mathbf{q5}bb) \\
&\Rightarrow (\mathbf{q5}\#bb)
\end{aligned}
$$

Computation : An Example

$\Rightarrow \quad (\sqcup \mathbf{q_0} bb)$

$\Rightarrow \quad (\sqcup \# \mathbf{q_2} b)$

$\Rightarrow \quad (\sqcup \# b \mathbf{q_2} \sqcup)$

$\Rightarrow \quad (\sqcup \# \mathbf{q_4} b)$

$\Rightarrow \quad (\sqcup \mathbf{q_5} \#)$

$\Rightarrow \quad (\sqcup \sqcup \mathbf{q_0} \sqcup)$

$\Rightarrow \quad (\sqcup \sqcup \sqcup \mathbf{q_6} \sqcup)$ **- Accept Halt**

Graph of a Turing Machine

A Turing machine can be drawn as a **labelled directed graph**.

- Each element of $\mathbf{Q}$ is a vertex of the graph.
- Each transition is a **labelled directed edge**. If $\delta(\mathbf{q}, \gamma_\mathbf{1}) = (\mathbf{p}, \gamma, \mathbf{L})$ be a transition, then there is an edge from the vertex of state $\mathbf{q}$ to the vertex of state $\mathbf{p}$ with a label $\gamma_\mathbf{1} \rightarrow \gamma_\mathbf{2}, \mathbf{L}$. If $\gamma_\mathbf{1} = \gamma_\mathbf{2}$, then we may use a short-hand $\gamma_\mathbf{1} \rightarrow \mathbf{L}$.
- The **start** and the **halt** states are **marked**.

Graph of a Turing Mahine : An Example

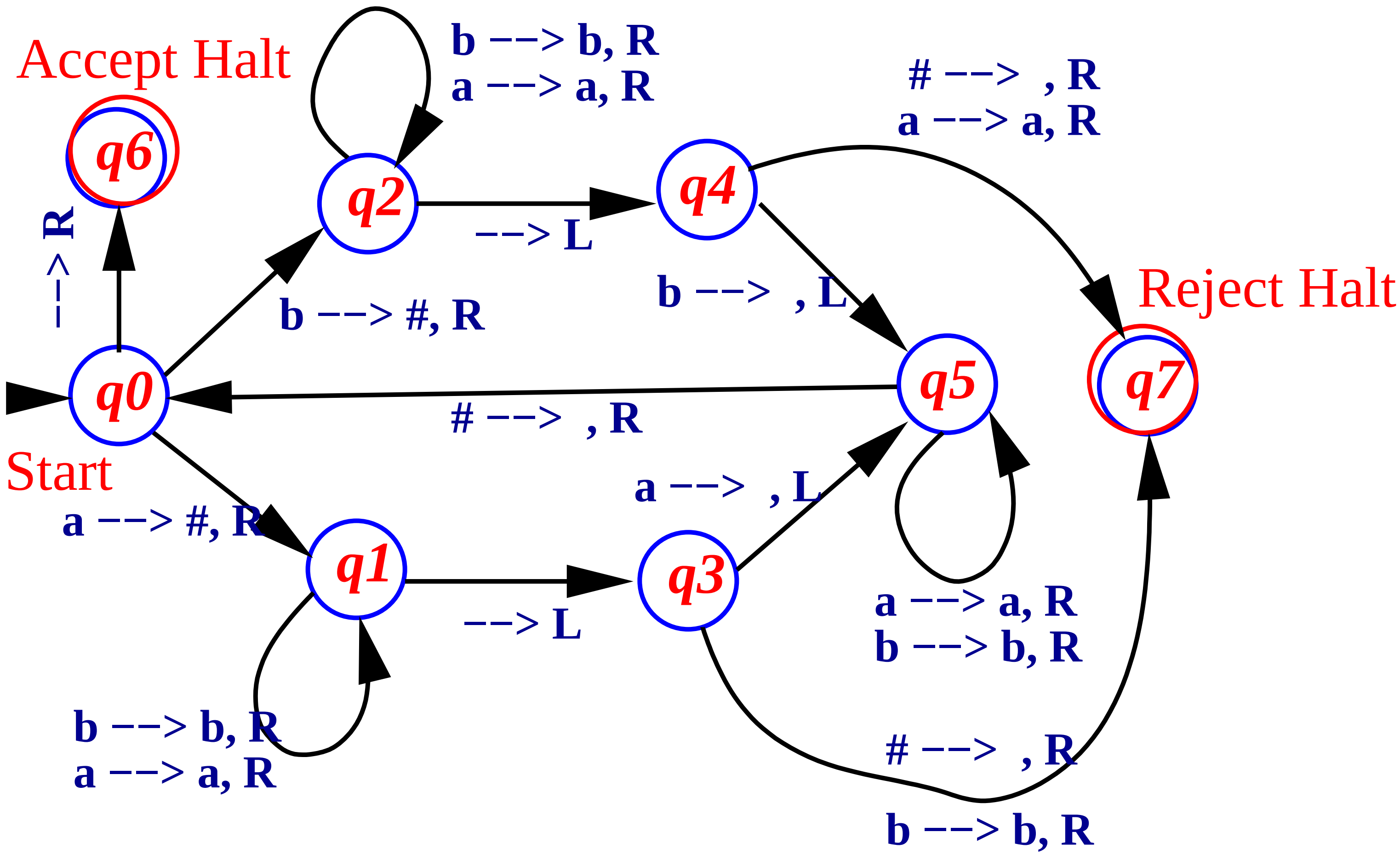

Figure 4: **State Transition Graph**

High Level Description

Input: A string over $\{\mathbf{a}, \mathbf{b}\}$.

Algorithm:

1. Read the current symbol . If it is a **blank**, **accept** the string. Otherwise remember whether it is an '**a**' or a '**b**' and change it to '#'.
2. Move the head to the rightmost nonblank symbol; if it is not same as the symbol read in **step 1**, **reject** the string. Otherwise change it to **blank**.
3. Move the head towards left until '#' is encountered. Change it to a blank, move one step right and goto **step 1**.