Formal Definition of Turing Machine

A **(standard) Turing machine (TM)** is a 5-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0)$ where

- Q is a finite set of **states;**
- Σ is the **input alphabet;**
- Γ is the **tape alphabet;**
- the partial function

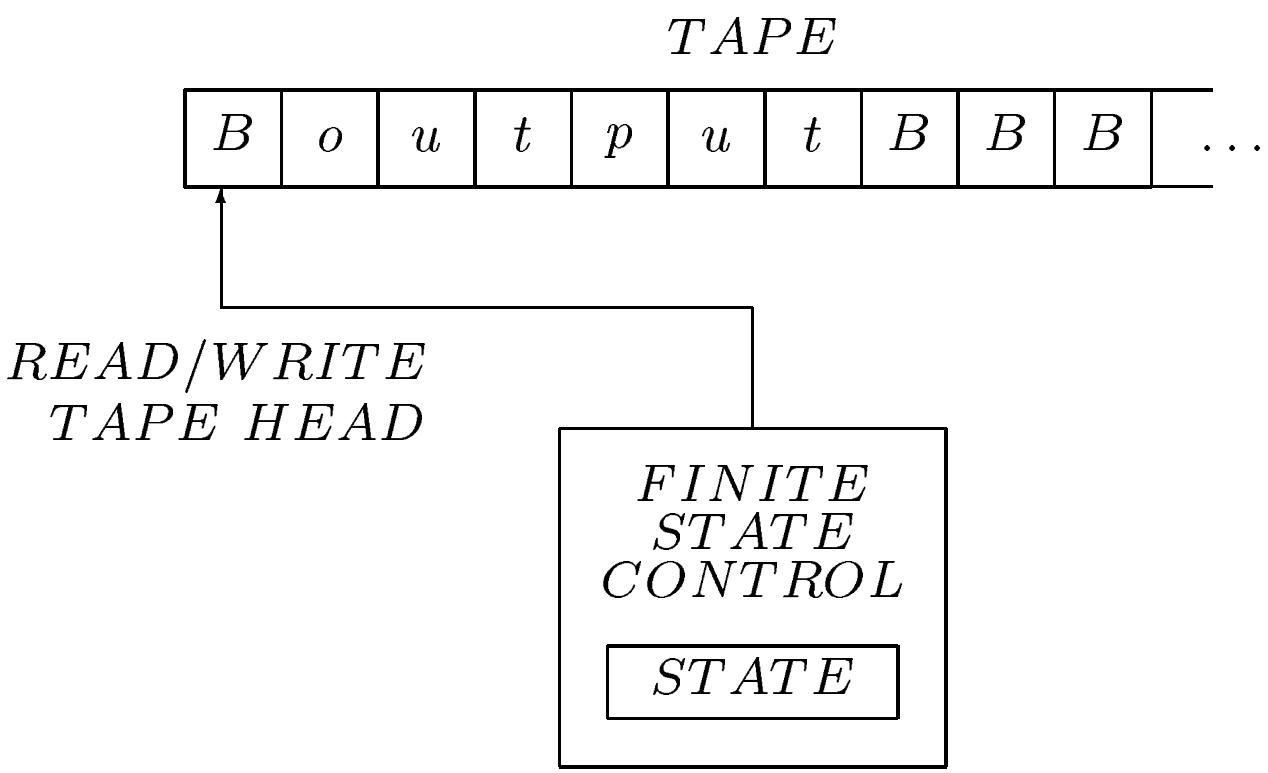
$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

  is the **transition function;** and
- $q_0 \in Q$ is the **start state.**

There is an element $B \in \Gamma$ called **blank.** We require $\Sigma \subset \Gamma - \{B\}$.

The Model

A typical mental model for a Turing machine looks like this:



A Turing machine **terminates abnormally** whenever a computation tries to move left from the leftmost tape square.

Example

Let M_1 be the Turing machine given by:

$$\begin{aligned} M_1 &= (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0) \\ Q_1 &= \{q_0, q_1, q_2\} \\ \Sigma_1 &= \{a\} \\ \Gamma_1 &= \{a, B\} \end{aligned}$$

The transition function

$$\delta : Q_1 \times \Gamma_1 \rightarrow Q_1 \times \Gamma_1 \times \{L, R\}$$

is given by this table:

q	$\delta_1(q, a)$	$\delta_1(q, B)$
q_0	undefined	(q_1, B, R)
q_1	(q_1, a, R)	(q_2, B, L)
q_2	(q_2, B, L)	undefined

Try the model on input aaa.

Configurations

A **configuration** of M is an element of

$$\Gamma^* Q (\{B\} \cup \Gamma^* (\Gamma - \{B\})).$$

For example, TM M_1 has the configuration

$$Baaq_1a,$$

which means

- The contents of the tape are $Baaa$ followed by an infinite sequence of B's;
- The TM is in state q_1; and
- The tape head is pointing to the fourth tape square (the one to the right of the state).

For an input $w \in \Sigma^*$, the **initial** or **start configuration** is

$$q_0 Bw.$$

Yields Relation

The **yields (in one step) relation** $\underset{M}{\vdash}$ is a binary relation on well-formed configurations.

Right Moves.

$$\delta(q_i, \sigma) \;=\; (q_j, \tau, R)$$

If $v = \lambda$, then

$$uq_i\sigma v \;\underset{M}{\vdash}\; u\tau q_j B;$$

otherwise,

$$uq_i\sigma v \;\underset{M}{\vdash}\; u\tau q_j v.$$

Left Moves.

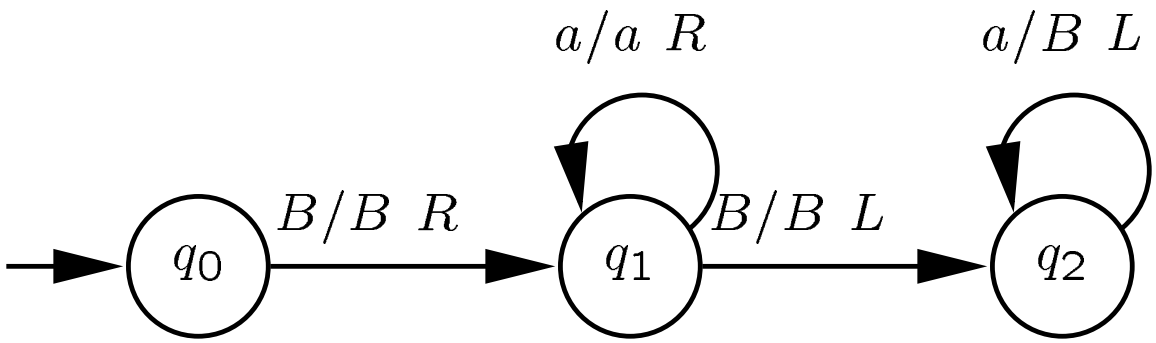
$$\delta(q_i, \sigma) \;=\; (q_j, \tau, L)$$

If $u \neq \lambda$, then $u = x\gamma$ and

$$uq_i\sigma v \;\underset{M}{\vdash}\; xq_j\gamma\tau v.$$

Example

The previous example M_1 has state diagram



For input $w = aaa$, the initial configuration is

$$q_0Baaa.$$

The complete computation is

$$\begin{aligned} q_0Baaa &\vdash Bq_1aaa \vdash Baq_1aa \\ &\vdash Baaq_1a \vdash Baaaq_1B \\ &\vdash Baaq_2a \vdash Baq_2a \\ &\vdash Bq_2a \vdash q_2B. \end{aligned}$$

As there is no other configuration that follows q_2B, the Turing machine **halts.**

Yields in t Steps

As usual, we have the notion of **yields in t steps:**

$$wq_ix \overset{t}{\underset{M}{\vdash}} yq_jz.$$

Yields in Zero or More Steps

Taking the union of all these relations, we get **yields (in zero or more steps):**

$$wq_ix \overset{*}{\underset{M}{\vdash}} yq_jz$$

holds if and only if there exists a $t \geq 0$ such that

$$wq_ix \overset{t}{\underset{M}{\vdash}} yq_jz.$$

Exercise

Design a Turing machine M_2 that **copies its input.** In particular, the effect of M_2 on input u should be

$$q_0 Bu \overset{*}{\underset{M}{\vdash}} q_j BuBu,$$

where $q_j \neq q_0$ is some designated finishing state.

Assume that $\Sigma = \{0, 1\}$.

Acceptance by Halting

A string $w \in \Sigma^*$ is **accepted by halting** by the Turing machine M if the computation by M on input w eventually halts (not terminates abnormally).

Acceptance by Final State

A **Turing machine with final states** is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, a standard Turing machine augmented with a set $F \subset Q$ of final states.

A string $w \in \Sigma^*$ is **accepted by final state** by the Turing machine M if the computation by M on input w eventually halts in a final state.

The **language** $L(M)$ **accepted** by the Turing machine M is the set of all strings accepted by M (by final state).

The class of languages accepted by some TM is the class of **recursively enumerable** languages.

Example

Design a TM M_3 with final states that accepts the language

$$L_3 = \{ww \mid w \in \{0,1\}^*\}.$$

Exercise

Design a Turing machine M_4 that accepts the following language:

$$L_4 = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$$

?

Use acceptance by final state **or** by halting, as you like.

Equivalence of Definitions of Acceptance

Theorem 9.3.2. A language L is accepted by halting if and only if L is accepted by final state.

Proof:

First suppose that $M = (Q, \Sigma, \Gamma, \delta, q_0)$ accepts L by halting. Then $M' = (Q, \Sigma, \Gamma, \delta, q_0, Q)$ accepts L by final state.

Proof Continued

Now suppose that $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts L by final state. Define a new, standard Turing machine $M' = (Q \cup \{q_R\}, \Sigma, \Gamma, \delta', q_0)$ as follows:

- Whenever $\delta(q, x)$ is defined, define $\delta'(q, x) = \delta(q, x)$.
- If $q \in Q - F$ and $\delta(q, x)$ is undefined, define $\delta'(q, x) = (q_R, x, R)$.
- For all $x \in \Gamma$, define $\delta'(q_R, x) = (q_R, x, R)$.

Think of q_R as a reject state. Whenever M would halt in a non-final state, M' goes to state q_R and moves right forever. Hence M accepts by final state if and only if M' halts.

Variations

- A **multitrack Turing machine** has k tracks on one tape. The tape head can read or write a k-tuple $(\sigma_1, \sigma_2, \ldots, \sigma_k) \in \Gamma^k$.

- A **multitape Turing machine** has k tapes, each with a read/write head that moves independently.

- A **Turing machine with a two-way tape** has a tape that extends infinitely in two directions.

Example

Design a Turing machine with two tapes to accept the language of palindromes:

$$L_5 = \{w \in \{0,1\}^* \mid w = w^R\}.$$

You may use moves L, R, and S (stationary) for each of the two heads.

Equivalence of Variations

Theorem 9.4.1. A language L is accepted by a multitrack Turing machine if and only if L is accepted by a standard Turing machine.

Theorem 9.6.1. A language L is accepted by a multitape Turing machine if and only if L is accepted by a standard Turing machine.

Theorem 9.5.1. A language L is accepted by a Turing machine with a two-way tape if and only if L is accepted by a standard Turing machine.

Nondeterministic Turing Machine

A **Nondeterministic Turing machine (NTM)** is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where everything is defined as for a TM with final states except the transition function maps as follows:

$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\}).$$

A string $w \in \Sigma^*$ is **accepted** by M if

$$q_0 B w \overset{*}{\underset{M}{\vdash}} \alpha q_i \beta,$$

where $\alpha q_i \beta$ is a halting configuration and $q_i \in F$.

In particular, if there is some computation of M on input w that halts in a final state, then M accepts w.

An Example Where Nondeterminism Helps

Design a Turing machine to accept this language discussed earlier:

$$L_3 = \{ww \mid w \in \{0,1\}^*\}.$$

The Power of Nondeterminism

Theorem. If L is accepted by a nondeterministic Turing machine, then L is accepted by a standard Turing machine.

Proof: Start with a NTM M and input w. The computation of M on w defines a computation tree where each node is a configuration. The initial configuration q_0Bw is at the root.

Design a standard (deterministic) Turing machine M' that traverses the configuration tree in a breadth-first manner and accepts if it ever finds a halting configuration of M containing a final state.

Church's Thesis

Any language that can be accepted by any model of computation can be accepted by a standard Turing machine.

Anything that can be computed can be computed by a standard Turing machine.

Enumerating a Language

Let M be a Turing machine with a special state q_{enum}. M **enumerates** a string $w \in \Sigma^*$ if

$$q_0 B \overset{*}{\underset{M}{\vdash}} q_{enum} B w B u.$$

Think of the configuration $q_{enum} B w B u$ as specifying "print w."

M **enumerates** the language L if

$$L = \{w \in \Sigma^* \mid M \text{ enumerates } w\}.$$

Example

Design a Turing machine to enumerate the language

$$\begin{aligned} L_6 &= \left\{0^{2^i} \mid i \geq 0\right\} \\ &= \{0, 00, 0000, 00000000, \ldots\}. \end{aligned}$$

Recursive Enumeration

Theorem. L is recursively enumerable if and only if L is enumerated by some Turing machine.

Proof: First suppose that L is enumerated by some TM $M = (Q, \Sigma, \Gamma, \delta, q_0)$. Then construct a TM M' that works as follows:

- Start in configuration $q_0' Bw$. Replace the second blank with a \$, arriving at configuration $Bw\$q_0 B$.
- Run M, interrupting whenever M enters q_{enum}. During the interrupt, check the enumerated string against w. If there is a match, then M' accepts w by halting.

Argue that M' accepts L.

Now suppose that TM M accepts L.
Construct a TM M' to enumerate L.
Conceptually M' works as follows:

- M' is able to list the elements of Σ^* in length lexicographic order. For $\Sigma = \{0, 1\}$, the order is $\lambda, 0, 1, 00, 01, 10, 11, \ldots$.
- Imagine a table reporting results of computations of M:

		λ	0	1	00	01	...
S	1	no	no	no	no	no	...
t	2	no	no	no	no	no	...
e	3	no	yes	no	yes	no	...
p	4	no	yes	yes	yes	no	...
s	5	no	yes	yes	yes	no	...
	⋮	⋮	⋮	⋮	⋮	⋮	

M' need only visit every table entry, by simulating M in a dovetailed fashion, to find every "yes." At each "yes," M' enumerates the corresponding string.