

1 Formal definition of a Turing machine

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

- Q is the set of states
- Σ is the input alphabet *not* containing special blank symbol \sqcup
- Γ is the tape alphabet satisfying $\Sigma \subset \Gamma$ and $\sqcup \in \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
- $q_0 \in Q$ is the start state
- $q_{accept} \in Q$ is the accept state
- $q_{reject} \in Q$ is the reject state, $q_{reject} \neq q_{accept}$

2 Turing machine computation

Tape content is unbounded but always finite, and the first (leftmost) blank symbol marks the end of tape content.

A configuration C_1 yields the configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step. A **configuration** consists of:

- The current state
- The tape content
- The head location

The **start configuration** on an input $w \in \Sigma^*$ consists of start state q_0 , w as the tape content, and the head location is the first (leftmost) position of the tape.

A configuration is **accepting** if its state is q_{accept} . A configuration is **rejecting** if its state is q_{reject} .

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

A Turing machine M **accepts** an input w if there is a sequence of configurations C_1, C_2, \dots, C_k such that:

1. C_1 is the start configuration of M on input w
2. C_i yields C_{i+1} for $1 \leq i \leq k - 1$
3. C_k is an accepting configuration

The **language** of M , denoted $L(M)$, is the set of strings accepted by M .

3 Turing-recognisable languages

A language L is **Turing-recognisable** if there is a Turing machine M that recognises it, i.e. L is the language of M .

Turing-recognisable means the same thing as **semi-decidable** and **recursively enumerable**.

If M recognises L , it may or may not halt on words not in L .

4 Turing-decidable languages

A language L is **Turing-decidable** if there is a Turing machine M that accepts every $w \in L$ and rejects every $w \notin L$.

Turing-decidable means the same thing as **recursive**.

If M decides L , it always halts.

5 Multitape Turing machines

A **Multitape Turing machine** is like a single tape Turing machine with several tapes, each with its own head. The only difference in the formal definition is the transition function, which is now:

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

where k is the number of tapes.

Theorem. Every multitape Turing machine has an equivalent single tape Turing machine.

6 Non-deterministic Turing machines

A **non-deterministic Turing machine** has a transition function:

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

Theorem. Every non-deterministic Turing machine has an equivalent deterministic Turing machine.

Proof. Consider the tree of all possible computations of the non-deterministic Turing machine. Start from the root (the start configuration) and do a breadth-first search. Accept only if an accepting configuration is found.

- DFS would not work
- Can use a multitape Turing machine to implement the BFS

7 Church-Turing thesis

Intuitive notion of an algorithm is equivalent to the mathematical concept of an algorithm defined by Turing machines (or any other formal model of computation, such as λ -calculus, Post machines, recursive functions)

8 Universal Turing machine

Every Turing machine M can be encoded as a word over a finite alphabet. Use $\langle M \rangle$ to denote the **encoding** of a Turing machine M .

Theorem. There is a Turing machine U that takes a two-part input – the encoding of a Turing machine M ($\langle M \rangle$) and a word w , and simulate M on w . U is called a **universal Turing machine**.

9 Halting problem

Halting problem: Given an encoding of a Turing machine M and a word w , does M terminate on w ?

Proposition. The Halting problem is Turing-recognisable.

Proof. Run a universal Turing machine on the pair $(\langle M \rangle, w)$. Accept if the computation eventually terminates.

Proposition. The Halting problem is not Turing-decidable.

Proof. Assume, for contradiction, there is a Turing machine H that decides the Halting problem.

$$H(\langle M \rangle, w) = \begin{cases} \text{accept} & \text{if } M \text{ terminates on } w \\ \text{reject} & \text{if } M \text{ does not terminate on } w \end{cases}$$

Use H as a black box to create an instance of the Halting problem, on which, H fails.

Consider a Turing machine D that takes the description of a single Turing machine M as an input and does the following:

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } H(\langle M \rangle, \langle M \rangle) \text{ rejects} \\ \text{loop} & \text{if } H(\langle M \rangle, \langle M \rangle) \text{ accepts} \end{cases}$$

What happens when D runs on its own encoding, $\langle D \rangle$?

1. D terminates on $\langle D \rangle$. By the construction of D , $H(\langle D \rangle, \langle D \rangle)$ rejects, giving a wrong answer
2. D does not terminate on $\langle D \rangle$. By the construction of D , $H(\langle D \rangle, \langle D \rangle)$ accepts, giving a wrong answer.

10 Turing-recognisable vs. Turing-decidable

Theorem. A language L is Turing-decidable if and only if both L and its complement, \bar{L} , are Turing-recognisable.

Proof. Suppose M_1 recognises L , and M_2 recognises \bar{L} .

On an input w , run M_1 and M_2 in parallel – i.e. simulate alternating steps of M_1 and M_2 on a multitape Turing machine.

Either M_1 or M_2 must eventually accept – accept if M_1 accepts and reject if M_2 accepts.

11 Co-Halting problem

Given an encoding of a Turing machine M , and word w , is it the case that M doesn't terminate on w , i.e. is not Turing-recognisable.

12 Step-counter predicate

Step (M, w, k) if and only if the machine M terminates on w in no more than k steps, is Turing-decidable.