Definitions:

- A **Deterministic Turing machine** is a 7-tuple:
  \[ M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}) \]
  where
  - \( Q \) is a finite set of states
  - \( \Sigma \) is a finite input alphabet
  - \( \Gamma \) is a finite tape alphabet, \( \Sigma \subseteq \Gamma \)
  - \( \delta \) is a transition function, \( \delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\} \)
  - \( q_0 \) is the start state
  - \( q_{\text{accept}} \) is the accepting state
  - \( q_{\text{reject}} \) is the rejecting state

  The machine \( M \) is assumed to stop when it enters either \( q_{\text{accept}} \) or \( q_{\text{reject}} \).

- An **Alphabet** is a finite set of symbols.

- A **Language** is a set of strings over an alphabet.

- Turing machines may be studied as “language deciders” or as “function evaluators”. As a “language decider”, a Turing machine is given an input string \( w \) and halts in either \( q_{\text{accept}} \) or \( q_{\text{reject}} \) indicating \( w \in L \) or \( w \notin L \) respectively.

- In the context of language deciders, we use the notation \( L(M) \) to indicate the language accepted by Turing machine \( M \).

- Turing machines may be studied as “function evaluators”. In this case, a Turing machine is defined as:
  \[ M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{halt}}) \]
  where \( Q, \Sigma, \Gamma, \delta, q_0 \) are as described above, and \( q_{\text{halt}} \) is the halting state. The machine \( M \) is assumed to stop when it enters \( q_{\text{halt}} \).

  The contents of the tape when the machine is started in \( q_0 \) is considered the function input, \( x \). The contents of the tape when the machine stops in \( q_{\text{halt}} \) is considered the function value, \( f(x) \).

- A function \( f \) is **computable** if and only if there exists a Turing machine which computes \( f(x) \) on input \( x \).

- A **Non-deterministic Turing machine** is a 7-tuple:
  \[ N = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}) \]
  where
  - \( Q \) is a finite set of states
  - \( \Sigma \) is a finite input alphabet
  - \( \Gamma \) is a finite tape alphabet, \( \Sigma \subseteq \Gamma \)
  - \( \delta \) is a transition function, \( \delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}} \). I.e., on each computation step, \( N \) has a set of possible moves which are performed non-deterministically.
  - \( q_0 \) is the start state
- $q_{\text{accept}}$ is the accepting state
- $q_{\text{reject}}$ is the rejecting state

The machine $N$ is assumed to stop when it enters either $q_{\text{accept}}$ or $q_{\text{reject}}$.

- An instantaneous description (a.k.a. configuration) is a string $\alpha q \beta$ where:
  - The contents of the tape is the string $\alpha \beta$
  - The machine is in state $q$
  - The read/write head is positioned over the first symbol in $\beta$.

- An accepting configuration is a configuration which contains $q_{\text{accept}}$.

- An rejecting configuration is a configuration which contains $q_{\text{reject}}$.

- A computation history is a sequence of instantaneous descriptions.

$$I_0 \vdash I_1 \vdash I_2 \ldots \vdash I_t$$

such that:

- $I_0$ is the start configuration $q_0 w$
- $I_t$ is a halting configuration, i.e., contains $q_{\text{accept}}$ or $q_{\text{reject}}$
- Each configuration $I_j$ leads to the next configuration $I_{j+1}$ according to the transition function $\delta$.

- A Turing machine accepts an input string $w$ if and only if there exists a computation history leading to an accepting configuration.

- A polynomial time bounded Turing machine is a Turing machine which, given input $w$ of length $n$ will halt within $p(n)$ steps, where $p(n)$ is a polynomial in $n$.

- A function $f$ is a polynomial time computable function if and only if there exists a polynomial time bounded Turing machine which computes $f$.

- The class $\mathcal{P}$ is the set of languages (or problems) which can be decided (solved) by a deterministic polynomial time bounded Turing machine.

- The class $\mathcal{NP}$ is the set of languages (or problems) which can be decided (solved) by a non-deterministic polynomial time bounded Turing machine.

- Some authors define the class $\mathcal{NP}$ as the set of problems which have deterministic polynomial time verifiers.\(^1\)

- It is unknown whether $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$.

- A language $L$ is polynomial time mapping reducible to $L_0$ if and only if there exists a polynomial time computable function $f$ such that for every input string $w$ over the alphabet of $L$, $f(w)$ is an output string over the alphabet of $L_0$ and $w \in L$ if and only if $f(w) \in L_0$.

- A language $L_0$ is $\mathcal{NP}$-complete if and only if two conditions hold:
  1. $L_0$ in $\mathcal{NP}$
  2. every language $L$ in $\mathcal{NP}$ is polynomial time mapping reducible to $L_0$.

- If only the second condition above holds, then we say $L_0$ is $\mathcal{NP}$-hard.

---

\(^1\) A verifier is a deterministic Turing machine which can check whether an alleged solution is correct.