Lesson 8: Turing machines

**Theme:** Turing machines as a model of general computation.

We reserve a special symbol ⊥, called the *blank* symbol.

A Turing machine (TM) is a system \( M = (\Sigma, \Gamma, Q, q_0, q_{\text{acc}}, q_{\text{ rej}}, \delta) \), where each component is as follows.

- \( \Sigma \) is a finite alphabet, called the *input* alphabet, where \( \sqcup \notin \Sigma \).
- \( \Gamma \) is a finite alphabet, called the *tape* alphabet, where \( \Sigma \cup \Gamma \) and \( \sqcup \in \Gamma \).
- \( Q \) is a finite set of states.
- \( q_0 \in Q \) is the initial state.
- \( q_{\text{acc}}, q_{\text{ rej}} \in Q \) are two special states called the *accept* and *reject* states, respectively.
- \( \delta : Q - \{q_{\text{acc}}, q_{\text{ rej}}\} \times \Gamma \to Q \times \Gamma \times \{\text{Left}, \text{Right}\} \) is the transition function.

Intuitively, the intuitive meaning of \( \delta(p, a) = (q, b, \alpha) \) is as follows. When the head reads a symbol \( a \), if \( M \) is in state \( p \), it “writes” symbol \( b \) on top of \( a \), enters state \( q \), and the head moves left, if \( \alpha = \text{Left} \), or moves right, if \( \alpha = \text{Right} \).

To describe how a TM computes, we need a few terminologies. A configuration of \( M \) is a string \( C \) from \( (Q \cup \Gamma)^* \) which contains *exactly one symbol* from \( Q \). We call such symbol the state of \( C \). Intuitively, a configuration \( C = a_1 \cdots a_{i-1} \text{pa}_i \cdots a_m \) means the content of the tape \( a_1 \cdots a_m \) and that \( M \) is in state \( p \) with the head in position \( i \).

On input word \( w \in \Sigma^* \), the *initial* configuration of \( M \) on \( w \) is the string \( q_0w \). A configuration is called *accepting*, if it contains \( q_{\text{ acc}} \), and it is called *rejecting*, if it contains \( q_{\text{ rej}} \). A halting configuration is either an accepting or a rejecting configuration.

Let \( C = a_1 \cdots a_{i-1} \text{pa}_i \cdots a_m \) be a configuration, where \( a_1, \ldots, a_m \in \Gamma \) and \( p \in Q \) such that \( p \neq q_{\text{ acc}}, q_{\text{ rej}} \). The transition \( \delta \) yields the subsequent configuration \( C' \), denoted by \( C \vdash C' \), as follows.

- If \( \delta(p, a_i) = (q, b, \text{ Left}) \) and \( i \geq 2 \), then \( C' = a_1 \cdots a_{i-2} q a_{i-1} b a_{i+1} \cdots a_m \).
- If \( \delta(p, a_i) = (q, b, \text{ Right}) \) and \( i \leq m - 1 \), then \( C' = a_1 \cdots a_{i-1} b q a_{i+1} \cdots a_m \).
- If \( \delta(p, a_i) = (q, b, \text{ Right}) \) and \( i = m \), then \( C' = a_1 \cdots a_{m-1} b q \sqcup \).

The *run* of \( M \) on \( w \) is the (possibly infinite) sequence:

\[
C_0 \vdash C_1 \vdash C_2 \vdash \cdots, \tag{1}
\]

where \( C_0 \) is the initial configuration of \( M \) on \( w \).

\( M \) stops when it reaches a configuration \( C = a_1 \cdots a_{i-1} \text{pa}_i \cdots a_m \) where there is no \( C' \) where \( C \vdash C' \). For such case, we say that \( M \) *halts on* \( w \) *in configuration* \( C \) and \( C \) must satisfy one of the two conditions below holds.

- \( C \) is a halting configuration.
- \( i = 1 \) and \( \delta(p, a_i) = (q, b, \text{ Left}) \), i.e., the head still moves left when it is already on the leftmost position of the tape and “falls” off the tape.
If $M$ halts in an accepting configuration, then we say that $M$ accepts $w$. If it halts in a rejecting configuration, then we say that $M$ rejects $w$. Proposition 8.1 below states that we can always assume that when a Turing machine halts, it halts in either an accepting or rejecting configuration.

**Proposition 8.1** For every Turing machine $M$, there is another Turing machine $M'$ such that for every $w \in \Sigma^*$ the following holds.

- $M$ accepts $w$ if and only if $M'$ accepts $w$.
- $M$ rejects $w$ if and only if $M'$ rejects $w$.

For all $w$ neither accepted nor rejected by $M$, $M'$ does not halt on $w$.

In other words, Proposition 8.1 implies that we can assume that on any input, the head of $M'$ never falls off the tape.

**Some important terminologies.**

- We say that $M$ recognizes a language $L$, if:
  1. for every word $w \in L$, $M$ accepts $w$;
  2. for every word $w \notin L$, $M$ does not accept $w$.

  Note that $M$ does not accept $w$ can have two meanings: either $M$ rejects $w$, or $M$ does not halt on $w$.

- We say that $M$ decides a language $L$, if for every word $w$,
  1. if $w \in L$, $M$ accepts $w$,
  2. if $w \notin L$, $M$ rejects $w$.

  Note that this implies $M$ halts on every word $w \in \Sigma^*$.

- A language $L$ is recognizable/recursively enumerable (r.e.), if there is a TM $M$ that recognizes $L$.

- A language $L$ is decidable/recursive, if there is a TM $M$ that decides $L$.

  Otherwise, it is called undecidable.

**Appendix**

**A Turing machines with Stay option**

In some textbooks, Turing machines are defined such that the head can stay put, instead of moving Left or Right. Formally, a transition can be of the form:

$$(q, a) \rightarrow (p, b, \alpha), \quad \text{where } \alpha \in \{\text{Left, Right, Stay}\}$$

If $\alpha = \text{Stay}$, then the head stays where it is. Such Stay option is obviously equivalent to making two moves: Right, and followed by Left, thus, does not add any power of computation.
B Putting a marker on the leftmost cell of the tape

To prevent the head falls off the tape, we reserve a special symbol $\triangleleft$ that can be used to mark the leftmost cell of the tape of Turing machines. We describe a TM, denoted by $M_{sr}$, that on input $w \in \Sigma^*$, it will always halt in the accepting configuration $q_{acc}w$.

The following is $M_{sr} = \langle \Sigma, \Gamma, Q, q_0, q_{acc}, q_{rej}, \delta \rangle$ for the case of $\Sigma = \{0, 1\}$.

- $\Sigma = \{0, 1\}$.
- $\Gamma = \{\triangleleft, 0, 1, \sqcup\}$.
- $Q = \{q_0, p, r, s, q_{acc}, q_{rej}\}$.
- $\delta$ consists of the following:

\[
\begin{align*}
(q_0, 1) & \rightarrow (p, \triangleleft, \text{Right}) & (p, 1) & \rightarrow (p, 1, \text{Right}) \\
(q_0, 0) & \rightarrow (r, \triangleleft, \text{Right}) & (p, 0) & \rightarrow (r, 0, \text{Right}) \\
(q_0, \sqcup) & \rightarrow (q_{acc}, \triangleleft, \text{Right}) & (p, \sqcup) & \rightarrow (s, \sqcup, \text{Left}) \\
(q_0, \triangleleft) & \rightarrow (q_{rej}, \triangleleft, \text{Right}) & (p, \triangleleft) & \rightarrow (q_{rej}, \triangleleft, \text{Right}) \\
(r, 1) & \rightarrow (p, 1, \text{Right}) & (s, 0) & \rightarrow (s, 0, \text{Left}) \\
(r, 0) & \rightarrow (r, 0, \text{Right}) & (s, 1) & \rightarrow (s, 1, \text{Left}) \\
(r, \sqcup) & \rightarrow (s, \sqcup, \text{Left}) & (s, \triangleleft) & \rightarrow (q_{acc}, \triangleleft, \text{Right}) \\
(r, \triangleleft) & \rightarrow (q_{rej}, \triangleleft, \text{Right}) & (s, \sqcup) & \rightarrow (q_{rej}, \triangleleft, \text{Right})
\end{align*}
\]

The construction above can be easily generalized for arbitrary $\Sigma$.

This $M_{sr}$ can now be run as a precursor of an arbitrary Turing machine whose head never moves left whenever it reads the marker $\triangleleft$. Thus, we can always assume that the head never falls off the tape.

C Encoding an arbitrary alphabet into the binary alphabet $\{0, 1\}$

Turing machines are usually defined with arbitrary input and tape alphabets. It is not difficult to show that any alphabet can be “encoded” with binary alphabet.

Suppose $\Gamma = \{a_1, \ldots, a_n, \sqcup\}$. Each symbol $a_i$ can then be encoded with a 0-1 string of length $\lceil \log_2 n \rceil$. For example, if $\Gamma = \{a_1, \ldots, a_5, \sqcup\}$, we can encode $a_1$ with 000, $a_2$ with 001, $a_3$ with 010, $a_4$ with 011, and $a_5$ with 100. We denote by $\langle a_i \rangle$ the encoding of the symbol $a_i$. For a word $w \in \Gamma^*$, $\langle w \rangle$ denotes the encoding of $w$ by replacing each symbol $a_i$ in $w$ with $\langle a_i \rangle$. For example, if $w = a_1a_2a_3a_4$, $\langle w \rangle = \langle a_1 \rangle \langle a_3 \rangle \langle a_2 \rangle \langle a_1 \rangle = 000100001000$.

We have the following proposition that shows that we can always assume that the Turing machines under consideration always work on tape alphabet $\Gamma = \{\triangleleft, 0, 1, \sqcup\}$, where $\triangleleft$ is the marker that marks the leftmost cell of the tape.

**Proposition 8.2** Let $M = \langle \Sigma, \Gamma, Q, q_0, q_{acc}, q_{rej}, \delta \rangle$ be a TM, where $\Gamma = \{a_1, \ldots, a_n, \sqcup\}$. Let $K = \lceil \log_2 n \rceil$. Let $\langle a_i \rangle$ be an encoding of symbol $a_i$ with 0-1 string of length $K$. There is a TM $M' = \langle \{0, 1\}, \{\triangleleft, 0, 1, \sqcup\}, Q', q_0', q_{acc}, q_{rej}, \delta' \rangle$ such that for every word $w \in \Sigma^*$, the following holds.

$M$ accepts $w$ if and only if $M'$ accepts $\langle w \rangle$.

Intuitively, $M'$ simulates $M$ by reading the tapes by blocks of $\lceil \log_2 n \rceil$ cells. It then remembers the block that it reads in its states, and “simulates” the transitions of $M$ accordingly.

Formally, $M = \langle \{0, 1\}, \{0, 1, \sqcup\}, Q, q_0, q_{acc}, q_{rej}, \delta \rangle$ is defined as follows. Let $\{0, 1\} \subseteq K$, i.e., the set of all 0-1 strings of length less than or equal to $K = \lceil \log_2 n \rceil$. 

3/4
Lesson 8: Turing machines

\[ Q' = (Q \times \{0,1\}^{<K}) \cup (Q \times \{L_1, \ldots, L_K, R_1, \ldots, R_K\}) \]
\[ \cup (Q \times \{L, R\} \times \{W\} \times \{0,1\}^{<K}). \]

\[ q_0' = (q_0, \epsilon). \]

\[ \delta' \] is defined as follows.

- For every \( u \in \{0,1\}^{<K-1} \), for every \( p \in Q - \{q_{acc}, q_{rej}\} \), \( \delta' \) consists of the following transitions.

\[ ((p, u), 0) \rightarrow ((p, u0), 0, \text{Right}) \]
\[ ((p, u), 1) \rightarrow ((p, u1), 1, \text{Right}) \]

- For every \( (q, a) \rightarrow (p, b, \text{Left}) \in \delta \), for every \( d \in \{0,1,\} \), \( \delta' \) consists of the following transitions.

\[ ((q, a), d) \rightarrow ((p, L, W, b), d, \text{Left}) \]

- For every \( (q, a) \rightarrow (p, b, \text{Right}) \in \delta \), for every \( d \in \{0,1,\} \), \( \delta' \) consists of the following transitions.

\[ ((q, a), d) \rightarrow ((p, R, W, b), d, \text{Left}) \]

- For every \( p \in Q \), for every \( c \in \{0,1\} \), for every \( v \in \{0,1\}^{<K-1} \) and \( v \neq \epsilon \), for every \( d \in \{0,1,\} \), for every \( \beta \in \{L, R\} \), \( \delta' \) consists of the following transitions.

\[ ((p, \beta, W, vc), d) \rightarrow ((p, \beta, W, v), c, \text{Left}) \]

- For every \( p \in Q \), for every \( d \in \{<,0,1,\} \), \( \delta' \) consists of the following transitions.

\[ ((p, L, W, \epsilon), d) \rightarrow ((p, L_1), d, \text{Right}) \]

- For every \( p \in Q \), for every \( d \in \{<,0,1,\} \), \( \delta' \) consists of the following transitions.

\[ ((p, R, W, \epsilon), d) \rightarrow ((p, R_k), d, \text{Right}) \]

- For every \( p \in Q \), for every \( d \in \{0,1,\} \), \( \delta' \) consists of the following transitions.

\[ ((p, L, W, \epsilon), d) \rightarrow ((p, L_k), d, \text{Right}) \]

- For every \( p \in Q \), for every \( i \in \{2, \ldots, k\} \), for every \( d \in \{0,1,\} \), \( \delta' \) consists of the following transitions.

\[ ((p, R_i), d) \rightarrow ((p, R_{i-1}), d, \text{Right}) \]
\[ ((p, L_i), d) \rightarrow ((p, L_{i-1}), d, \text{Left}) \]

- For every \( p \in Q \), for every \( d \in \{0,1,\} \), \( \delta' \) consists of the following transitions.

\[ ((p, R_1), d) \rightarrow ((p, \epsilon), d, \text{Right}) \]
\[ ((p, L_1), d) \rightarrow ((p, \epsilon), d, \text{Left}) \]

All the other transitions not specified above are assumed to enter \( q_{rej} \).