Theory of Computation

Chapter 1

Regular Languages

1.1 Strings and Languages

Definition 1.1. An **alphabet** is a finite set of symbols. A **string** over alphabet Σ is a finite sequence

$$w = a_1 a_2 \cdots a_n$$

with $a_1, \ldots, a_n \in \Sigma$, where n is called the **length** of the string w. The **empty string** is the unique string of length zero, which is denoted by ϵ.

Definition 1.2. The **concatenation** of strings

$$u = a_1 a_2 \cdots a_n \quad \text{and} \quad v = b_1 b_2 \cdots b_m$$

is defined as the string

$$uv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m.$$

Definition 1.3. Let Σ^* denote the set of all strings over Σ. A subset of Σ^* is called a **language** over Σ.

Definition 1.4. The **concatenation** of languages L_1 and L_2 is

$$L_1 L_2 = \{w_1 w_2 : w_1 \in L_1 \text{ and } w_2 \in L_2\}.$$

For any language L, we define $L^0 = \{\epsilon\}$ and $L^{n+1} = L^n L$ for all integers $n \geq 0$. Also, we define $L^* = \bigcup_{n \geq 0} L^n$.

1.2 Deterministic Finite State Automata

Definition 1.5. A **deterministic finite state automaton (DFA)** is a 5-tuple

$$M = (Q, \Sigma, \delta, s, F),$$

where each component is as follows.

- Q is a finite set of **states**.
- Σ is an alphabet.
- $\delta : Q \times \Sigma \to Q$ is a **transition function**.
- $s \in Q$ is the **start state**.
- $F \subseteq Q$ is the set of **final states**.

Definition 1.6. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

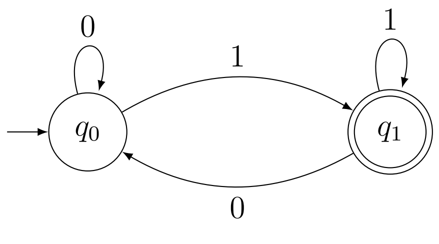
- We say that A accepts a string $w \in \Sigma^*$ if $\delta_w(q_0) \in F$.
- The **language** of A, denoted $L(A)$, is defined as the set of strings that are accepted by A.

Definition 1.7. A language L is **regular** if there exists a DFA A such that $L(A) = L$.

Example. Let $A = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ be a DFA, where the transition function δ is as follows.

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

Instead of using the formal definition, one can also draw a state diagram of A as follows.



It can be shown that a string $w \in \{0, 1\}^*$ is accepted by A if and only if w ends with 1. Thus, the language $L = \{w \in \{0, 1\}^* : w \text{ ends with } 1\}$ is regular.

Theorem 1.8. If L is a regular language over Σ, then $\Sigma^* \setminus L$ is also regular.

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA with $L = L(A)$. Let $A' = (Q, \Sigma, \delta, q_0, Q \setminus F)$. Then for each $w \in \Sigma^*$, we have

$$w \in L(A') \quad \Leftrightarrow \quad \delta_w(q_0) \in Q \setminus F \quad \Leftrightarrow \quad w \notin L(A).$$

Thus, $L(A') = \Sigma^* \setminus L(A)$, implying that $\Sigma^* \setminus L$ is regular. $\square$

Theorem 1.9. If L_1 and L_2 are regular languages over Σ, then $L_1 \cup L_2$ is also regular.

Proof. Let
$$A_1 = (Q_1, \Sigma, \delta^{(1)}, q_1, F_1) \quad \text{and} \quad A_2 = (Q_2, \Sigma, \delta^{(2)}, q_2, F_2)$$
be DFAs with $L_1 = L(A_1)$ and $L_2 = L(A_2)$. We construct the DFA

$$A = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F)$$

as follows.

- $\delta((p,q),a) = (\delta^{(1)}(p,a), \delta^{(2)}(q,a))$ for each $p \in Q_1$, $q \in Q_2$ and $a \in \Sigma$.
- $F = \{(p,q) : p \in F_1 \text{ or } q \in F_2\}$.

It can be shown that for each string $w \in \Sigma^*$, we have

$$\begin{aligned} w \in L(A) \quad &\Leftrightarrow \quad \delta_w((q_1,q_2)) \in F \\ &\Leftrightarrow \quad \delta_w^{(1)}(q_1) \in F_1 \text{ or } \delta_w^{(2)}(q_2) \in F_2 \\ &\Leftrightarrow \quad w \in L(A_1) \text{ or } w \in L(A_2). \end{aligned}$$

Thus, $L(A) = L(A_1) \cup L(A_2)$, implying that $L_1 \cup L_2$ is regular. $\square$

Corollary 1.10. If L_1 and L_2 are regular languages over Σ, then $L_1 \cap L_2$ is also regular.

Proof. Straightforward since by De Morgan's law we have

$$L_1 \cap L_2 = \Sigma^* \setminus ((\Sigma^* \setminus L_1) \cup (\Sigma^* \setminus L_2)). \qquad \square$$

1.3 Nondeterministic Finite State Automata

Definition 1.11. A **nondeterministic finite state automaton** (NFA) is a tuple

$$A = (Q, \Sigma, \delta, q_0, F),$$

where each component is as follows.

- Q is a finite set of **states**.
- Σ is a finite set of input symbols.
- $\delta \subseteq Q \times \Sigma \times Q$ is a relation, called the **transition relation**.
- $q_0 \in Q$ is called the **start state**.
- $F \subseteq Q$ is called the **accepting states**.

Definition 1.12. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA. For each string $w \in \Sigma^*$, we define $\delta_w \subseteq Q \times Q$ as follows, where $a \in \Sigma$ and $x \in \Sigma^*$.

- $\delta_\epsilon = \{(p, q) : p = q\}$.
- $\delta_a = \{(p, q) : (p, a, q) \in \delta\}$.
- $\delta_{xa} = \{(p, q) : (p, r) \in \delta_x \text{ and } (r, q) \in \delta_a \text{ for some } r \in Q\}$.

Definition 1.13. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA.

- We say that A accepts a string $w \in \Sigma^*$ if there exists $q \in F$ such that $(q_0, q) \in \delta_w$.
- The **language** of A, denoted $L(A)$, is defined as the set of strings that are accepted by A.

Theorem 1.14. For every NFA A, there is a DFA A' with $L(A') = L(A)$.

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$. We construct $A' = (\mathcal{P}(Q), \Sigma, \Delta, \{q_0\}, \Phi)$ as follows.

- $\Delta : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ is the function with

$$\Delta_a(P) = \bigcup_{p \in P} \{q \in Q : (p, q) \in \delta_a\}$$

  for any $P \subseteq Q$ and $a \in \Sigma$.

- $\Phi = \{P \subseteq Q : P \cap F \neq \varnothing\}$.

Now we prove that for any $w \in \Sigma^*$, for any $q \in Q$ and for any $P \subseteq Q$, we have $q \in \Delta_w(P)$ if and only if $(p, q) \in \delta_w$ for some $p \in P$. For the induction basis, let $w = \epsilon$, and we have

$$q \in \Delta_\epsilon(P) \quad \Leftrightarrow \quad q \in P \quad \Leftrightarrow \quad (p, q) \in \delta_\epsilon \text{ for some } p \in P.$$

For the induction step, let $w = xa$, where x is any string and a is any symbol. Note that by the construction of Δ, we have $q \in \Delta_a(P)$ if and only if $(p, q) \in \delta_a$ for some $p \in P$. Thus, we can conclude that

$$
\begin{aligned}
q \in \Delta_{xa}(P) \quad &\Leftrightarrow \quad q \in \Delta_a(\Delta_x(P)) \\
&\Leftrightarrow \quad (r, q) \in \delta_a \text{ for some } r \in \Delta_x(P) \\
& \quad \text{and } (p, r) \in \delta_x \text{ for some } p \in P \\
&\Leftrightarrow \quad (p, q) \in \delta_{xa} \text{ for some } p \in P.
\end{aligned}
$$

Finally we prove that $L(A') = L(A)$, which is given by

$$
\begin{aligned}
w \in L(A') \quad &\Leftrightarrow \quad \Delta_w(\{q_0\}) \in \Phi \\
&\Leftrightarrow \quad \Delta_w(\{q_0\}) \cap F \neq \varnothing \\
&\Leftrightarrow \quad q \in \Delta_w(\{q_0\}) \text{ for some } q \in F \\
&\Leftrightarrow \quad (p, q) \in \delta_w \text{ for some } q \in F \text{ and } p \in \{q_0\} \\
&\Leftrightarrow \quad (q_0, q) \in \delta_w \text{ for some } q \in F \\
&\Leftrightarrow \quad w \in L(A).
\end{aligned}
$$

□

Theorem 1.15. If L_1 and L_2 are regular languages over Σ, then $L_1 L_2$ is also regular.

Proof. Let $A_1 = (Q_1, \Sigma, \delta^{(1)}, q_1, F_1)$ and $A_2 = (Q_2, \Sigma, \delta^{(2)}, q_2, F_2)$ be NFAs such that $L_1 = L(A_1)$ and $L_2 = L(A_2)$. We construct an NFA

$$A = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F)$$

as follows.

- $\delta = \delta^{(1)} \cup \delta^{(2)} \cup \{(p, a, q) \in Q \times \Sigma \times Q : p \in F_1 \text{ and } (q_2, a, q) \in \delta^{(2)}\}$.
- If $q_2 \in F_2$, let $F = F_1 \cup F_2$. Otherwise, let $F = F_2$.

It can be shown that $L(A) = L(A_1)L(A_2)$, and thus $L_1 L_2$ is regular. □

Theorem 1.16. If L is a regular language over Σ, then L^* is also regular.

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA with $L = L(A)$. We construct an NFA

$$A' = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', F \cup \{q_0'\})$$

with

$$\delta' = \delta \cup \{(p, a, q) \in Q \times \Sigma \times Q : p \in F \cup \{q_0'\} \text{ and } (q_0, a, q) \in \delta\}.$$

It can be shown that $L(A') = (L(A))^*$, and thus L^* is regular. □

1.4 Regular Expressions

Definition 1.17. Let Σ be an alphabet. A **regular expression** over Σ is a string in the minimal language over $\Sigma \cup \{\varnothing, \epsilon, ^*, +, (,)\}$ that satisfies the following conditions.

1. $\varnothing$ is a regular expression.
2. ϵ is a regular expression.
3. If $a \in \Sigma$, then a is a regular expression.
4. If e_1 and e_2 are regular expressions, then so is $(e_1 e_2)$.
5. If e_1 and e_2 are regular expressions, then so is $(e_1 + e_2)$.
6. If e is a regular expression, then so is $(e)^*$.

Definition 1.18. A regular expression e over an alphabet Σ defines a language $L(e)$ as follows.

1. $L(\varnothing) = \varnothing$.
2. $L(\epsilon) = \{\epsilon\}$.
3. $L(a) = \{a\}$ for each $a \in \Sigma$.
4. $L((e_1 e_2)) = L(e_1)L(e_2)$ for each regular expressions e_1 and e_2.
5. $L((e_1 + e_2)) = L(e_1) \cup L(e_2)$ for each regular expressions e_1 and e_2.
6. $L((e)^*) = L(e^*)$ for each regular expression e.

Remark. From now on, we may omit parentheses if there is no ambiguity.

Lemma 1.19. If e is a regular expression over an alphabet Σ, then $L(e)$ is regular.

Proof. It can be easily shown that $\varnothing$ and $\{\epsilon\}$ are regular. Moreover, $\{a\}$ is regular for each $a \in \Sigma$. Thus, by Theorem 1.9, Theorem 1.15 and Theorem 1.16, we can conclude that for all regular expressions e, $L(e)$ is regular. $\square$

Lemma 1.20. If L is a regular language over an alphabet Σ, then there is a regular expression e over Σ such that $L(e) = L$.

Proof. Since L is regular, there exists a DFA $A = (Q, \Sigma, \delta, q_0, F)$ with $L(A) = L$. Suppose that $Q = \{p_1, p_2, \ldots, p_n\}$ with $p_1 = q_0$. For any $i, j \in \{1, \ldots, n\}$ and for any $k \in \{0, \ldots, n\}$, let $L_{ij}^{(k)}$ denote the language of strings w such that

- $\delta_w(p_i) = p_j$, and
- for each string x with $\epsilon \sqsubset x \sqsubset w$, we have $\delta_x(p_i) = p_\ell$ for some $\ell \in \{1, \ldots, k\}$.

We are going to prove that for all $i, j \in \{1, \dots, n\}$ and $k \in \{0, \dots, n\}$, there exists a regular expression $e_{ij}^{(k)}$ such that

$$L\big(e_{ij}^{(k)}\big) = L_{ij}^{(k)}.$$

The proof is by induction on k. For the induction basis, let $k = 0$. Let $\Pi_{ij} \subseteq \Sigma$ denote the set of symbols a with $\delta_a(p_i) = p_j$. If $i \neq j$, we have

$$L_{ij}^{(0)} = \bigcup_{a \in \Pi_{ij}} \{a\},$$

and thus we can construct $e_{ij}^{(0)}$ by

$$e_{ij}^{(0)} = \sum_{a \in \Pi_{ij}} a.$$

(If $\Pi_{ij} = \varnothing$, then the summation is defined as $\varnothing$.) If $i = j$, we have

$$L_{ii}^{(0)} = \{\epsilon\} \cup \bigcup_{a \in \Pi_{ii}} \{a\},$$

and thus we can construct $e_{ii}^{(0)}$ by

$$e_{ii}^{(0)} = \epsilon + \sum_{a \in \Pi_{ii}} a.$$

Now for the induction step, let $k \geq 1$. Suppose that $w \in L_{ij}^{(k)}$. If there is no string x with $\epsilon \sqsubset x \sqsubset w$ such that $\delta_x(p_i) = p_k$, then we have

$$w \in L_{ij}^{(k-1)}.$$

Otherwise, let $x_0, x_1, \dots, x_\ell$ be all strings with $\epsilon \sqsubset x_0 \sqsubset x_1 \sqsubset \cdots \sqsubset x_\ell \sqsubset w$ such that

$$\delta_{x_0}(p_i) = \delta_{x_1}(p_i) = \cdots = \delta_{x_\ell}(p_i) = p_k.$$

Let $u_0, u_1, \dots, u_{\ell+1}$ be strings such that

$$w = u_0 u_1 \cdots u_{\ell+1},$$

and $x_h = u_0 u_1 \cdots u_h$ for each $h \in \{0, \dots, \ell\}$. Since $u_0 \in L_{ik}^{(k-1)}$, $u_{\ell+1} \in L_{kj}^{(k-1)}$, and $u_h \in L_{kk}^{(k-1)}$ for each $h \in \{1, \dots, \ell\}$, it follows that

$$w \in L_{ik}^{(k-1)} \left(L_{kk}^{(k-1)}\right)^* L_{kj}^{(k-1)}.$$

As a result, we have

$$L_{ij}^{(k)} \subseteq L_{ij}^{(k-1)} \cup L_{ik}^{(k-1)} \left(L_{kk}^{(k-1)}\right)^* L_{kj}^{(k-1)},$$

implying

$$L_{ij}^{(k)} = L_{ij}^{(k-1)} \cup L_{ik}^{(k-1)} \left(L_{kk}^{(k-1)}\right)^* L_{kj}^{(k-1)}.$$

Therefore, we can construct $e_{ij}^{(k)}$ by

$$e_{ij}^{(k)} = e_{ij}^{(k-1)} + e_{ik}^{(k-1)} \left(e_{kk}^{(k-1)} \right)^* e_{kj}^{(k-1)}.$$

Now we can construct the regular expression e with $L(e) = L$ as follows. Let Φ be the set of integers $j \in \{1, \ldots, n\}$ such that $p_j \in F$. Since

$$L = \bigcup_{j \in \Phi} L_{1j}^{(n)},$$

we can construct e by

$$e = \sum_{j \in \Phi} e_{1j}^{(n)},$$

which completes the proof. □

Theorem 1.21. Let Σ be an alphabet. A language L over Σ is regular if and only if there is a regular expression e over Σ such that $L(e) = L$.

Proof. Straightforward by Lemma 1.19 and Lemma 1.20. □

Chapter 2

Context-Free Languages

2.1 Context-Free Grammars

Definition 2.1. A **context-free grammar (CFG)** is a 4-tuple

$$G = (V, \Sigma, R, S),$$

where each component is as follows.

- V is a finite set of **variables**.
- Σ is a finite set of **terminals** with $V \cap \Sigma = \varnothing$.
- R is a finite set of **rules**, where each rule is a pair (A, γ) with $A \in V$ and $\gamma \in (V \cup \Sigma)^*$.
- S is a special variable in V, called the **start variable**.

Definition 2.2. Let $G = (V, \Sigma, R, S)$ be a CFG.

- For any $A \in V$ and for any $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$, we say that $\alpha A \beta$ **yields** $\alpha\gamma\beta$ under G, denoted by
$$\alpha A \beta \underset{G}{\Rightarrow} \alpha\gamma\beta,$$
if there is a rule $(A, \gamma) \in R$.
- For any $\alpha, \beta \in (V \cup \Sigma)^*$, we say that α **derives** β under G, denoted by
$$\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta,$$
if $\alpha = \beta$, or there exists $\gamma \in (V \cup \Sigma)^*$ such that $\alpha \overset{*}{\underset{G}{\Rightarrow}} \gamma$ and $\gamma \underset{G}{\Rightarrow} \beta$.

Definition 2.3. Let $G = (V, \Sigma, R, S)$ be a CFG. For any string $w \in \Sigma^*$, if

$$S \overset{*}{\underset{G}{\Rightarrow}} w,$$

then we say that G **accepts** w. The set of string accepted by G is called the **language** of G, denoted by $L(G)$. A language L is **context-free** if $L = L(G)$ for some CFG G.

2.2 Pushdown Automata

Definition 2.4. A **pushdown automaton (PDA)** is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, s, \perp, F),$$

where each component is as follows.

- Q is a finite set of **states**.
- Σ is a finite alphabet, called the **input alphabet**.
- Γ is a finite alphabet, called the **stack alphabet**.
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*$ is the **transition relation**.
- $s \in Q$ is the **initial state**.
- $\perp \in \Gamma$ is the **initial stack symbol**.
- $F \subseteq Q$ is the set of **final states**.

Definition 2.5. A **configuration** of a PDA $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ is a triple

$$(q, w, \gamma),$$

where $q \in Q$ is the current state, $w \in \Sigma^*$ is the unprocessed input, and $\gamma \in \Gamma^*$ is the current stack. The **initial configuration** of M on input string w is $(s, w, \perp)$.

We define the single-step relation $\vdash_M$ such that for any $p, q \in Q$, $a \in \Sigma$, $h \in \Gamma$, $w \in \Sigma^*$ and $\beta, \gamma \in \Gamma^*$,

$$(p, aw, h\gamma) \vdash_M (q, w, \beta\gamma)$$

holds if and only if $(p, a, h, q, \beta) \in \delta$. The reflexive transitive closure of $\vdash_M$ is denoted by $\vdash_M^*$.

Definition 2.6. Let $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ be a PDA. A string $w \in \Sigma^*$ is **accepted** by M if

$$(s, w, \perp) \vdash_M^* (q, \epsilon, \gamma)$$

for some $q \in F$ and $\gamma \in \Gamma^*$. The **language** $L(M)$ accepted by M is defined as the collection of strings that are accepted by M.

Theorem 2.7. If L is context-free, then there is a PDA M that accepts L.

Chapter 3

Decidability

3.1 Turing Machines

Definition 3.1. A **Turing machine** is an 8-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_{\mathrm{acc}}, q_{\mathrm{rej}}),$$

where each component is as follows.

- Q is the finite set of **states**.
- Σ is the finite set of **input symbols**.
- Γ is the finite set of **tape symbols** with $\Sigma \subseteq \Gamma$.
- $\delta : (Q \setminus \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}) \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\}$ is the **transition function**.
- $q_0 \in Q$ is the **initial state**.
- $\sqcup \in \Gamma \setminus \Sigma$ is a special symbol, called the **blank symbol**.
- q_{acc} and q_{rej} are distinct states in Q, called the **accepting state** and the **rejecting state**, respectively.

Definition 3.2. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_{\mathrm{acc}}, q_{\mathrm{rej}})$ be a Turing machine.

- A **configuration** of M is a triple in $Q \times \{1, 2, \dots\} \times \Gamma^*$.
- We define a binary relation $\vdash_M$ over $Q \times \{1, 2, \dots\} \times \Gamma^*$ such that for any $p, q \in Q$, $i, j \in \{1, 2, \dots\}$ and $u, v \in \Gamma^*$,

  $$(p, i, u) \underset{M}{\vdash} (q, j, v)$$

  if and only if

  $$\begin{aligned} u^{(1)} \cdots u^{(i-1)} u^{(i+1)} \cdots u^{(n)} \sqcup \sqcup \cdots &= v^{(1)} \cdots v^{(i-1)} v^{(i+1)} \cdots v^{(m)} \sqcup \sqcup \cdots \\ \delta(p, u^{(i)}) &= (q, v^{(i)}, j - i). \end{aligned}$$

  Let $\vdash_M^{(n)}$ denote the nth power of $\vdash_M$, and let $\vdash_M^* = \bigcup_{n \in \mathbb{N}} \vdash_M^{(n)}$.

Remark. $\vdash_M$ is a partial function.

Definition 3.3. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_{\text{acc}}, q_{\text{rej}})$ be a Turing machine. Let $u \in \Sigma^*$.

- We say that M **accepts** u if $(q_0, 1, w) \vdash_M^* (q_{\text{acc}}, j, v)$ for some $j \in \{1, 2, \dots\}$ and $v \in \Gamma^*$.
- We say that M **rejects** u if $(q_0, 1, w) \vdash_M^* (q_{\text{rej}}, j, v)$ for some $j \in \{1, 2, \dots\}$ and $v \in \Gamma^*$.
- We say that M **halts** on input u if M either accepts or rejects u.

If M halts on u, then we have the following definitions.

- The **running time** of M on input u is the integer t such that

$$(q_0, 1, u) \overset{(t)}{\underset{M}{\vdash}} (q_{\text{acc}}, j, v) \qquad \text{or} \qquad (q_0, 1, u) \overset{(t)}{\underset{M}{\vdash}} (q_{\text{rej}}, j, v),$$

  where $j \in \{1, 2, \dots\}$ and $v \in \Sigma^*$.

- The **accessed space** of M on input u is the maximum integer s such that

$$(q_0, 1, u) \overset{*}{\underset{M}{\vdash}} (q, s, v),$$

  where $q \in Q$ and $v \in \Sigma^*$.

Definition 3.4. Let L be a language over Σ. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_{\text{acc}}, q_{\text{rej}})$ be a Turing machine.

- We say that M **recognizes** L if for each $w \in \Sigma^*$,

$$w \in L \quad \Leftrightarrow \quad M \text{ accepts } w.$$

  A language is **recursively enumerable** if it is recognized by some Turing machine. The collection of recursively enumerable languages is denoted by **RE**.

- We say that M **decides** L if for each $w \in \Sigma^*$,

$$\begin{aligned} w \in L \quad &\Rightarrow \quad M \text{ accepts } w \\ w \notin L \quad &\Rightarrow \quad M \text{ rejects } w. \end{aligned}$$

  A language is **recursive** if it is decided by some Turing machine. The collection of recursive languages is denoted by **R**.

Remark. If M decides L, then M recognizes L. Thus, $\mathbf{R} \subseteq \mathbf{RE}$.

3.2 Variants of Turing Machines

Definition 3.5. A ***k*-tape Turing machine** is

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_{\mathrm{acc}}, q_{\mathrm{rej}}),$$

where

$$\delta : (Q \setminus \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}) \times \Gamma^k \to Q \times \Gamma^k \times \{-1, 0, +1\}^k$$

is the **transition function** and other components are the same as those in the definition of Turing machine.

- A **configuration** of M is a triple in $Q \times \{1, 2, \dots\}^k \times (\Gamma^*)^k$, and we define the binary relation $\vdash_M$ over the configurations of M such that for any $p, q \in Q$, $i_1, \dots, i_k, j_1, \dots, j_k \in \{1, 2, \dots\}$ and $u_1, \dots, u_k, v_1, \dots, v_k \in \Gamma^*$,

  $$(p, (i_1, \dots, i_k), (u_1, \dots, u_k)) \quad \underset{M}{\vdash} \quad (q, (j_1, \dots, j_k), (v_1, \dots, v_k))$$

  if and only if

  $$u_\kappa^{(1)} \cdots u_\kappa^{(i_\kappa - 1)} u_\kappa^{(i_\kappa + 1)} \cdots u_\kappa^{(n_\kappa)} \sqcup \sqcup \cdots \quad = \quad v_\kappa^{(1)} \cdots v_\kappa^{(i_\kappa - 1)} v_\kappa^{(i_\kappa + 1)} \cdots v_\kappa^{(m_\kappa)} \sqcup \sqcup \cdots$$

  for all $\kappa \in \{1, \dots, k\}$ and

  $$\delta(p, (u_1^{(i_1)}, \dots, u_k^{(i_k)})) \quad = \quad (q, (v_1^{(i_1)}, \dots, v_k^{(i_k)}), (j_1 - i_1, \dots, j_k - i_k)).$$

- We say that M **accepts** (resp., **rejects**) $w \in \Sigma^*$ if

  $$(q_0, (1, 1, \dots, 1), (w, \epsilon, \dots, \epsilon)) \quad \overset{*}{\underset{M}{\vdash}} \quad (q_{\mathrm{acc}}, (j_1, j_2, \dots, j_k), (v_1, v_2, \dots, v_k))$$
  $$\Big(\text{resp., } (q_0, (1, 1, \dots, 1), (w, \epsilon, \dots, \epsilon)) \quad \overset{*}{\underset{M}{\vdash}} \quad (q_{\mathrm{rej}}, (j_1, j_2, \dots, j_k), (v_1, v_2, \dots, v_k))\Big)$$

  for some $j_1, j_2, \dots, j_k \in \{1, 2, \dots\}$ and $v_1, v_2, \dots, v_k \in \Gamma^*$. If M either accepts or rejects w, then we say that M **halts** on w.

- If M halts on $w \in \Sigma^*$, then the **running time** of M on input w is the integer t with

  $$(q_0, (1, 1, \dots, 1), (w, \epsilon, \dots, \epsilon)) \quad \overset{(t)}{\underset{M}{\vdash}} \quad (q_{\mathrm{acc}}, (j_1, j_2, \dots, j_k), (v_1, v_2, \dots, v_k))$$

  or

  $$(q_0, (1, 1, \dots, 1), (w, \epsilon, \dots, \epsilon)) \quad \overset{(t)}{\underset{M}{\vdash}} \quad (q_{\mathrm{rej}}, (j_1, j_2, \dots, j_k), (v_1, v_2, \dots, v_k)),$$

  and the **accessed space** of M on input w is the maximum sum of integers $s_1, \dots, s_k$ with

  $$(q_0, (1, 1, \dots, 1), (w, \epsilon, \dots, \epsilon)) \overset{*}{\underset{M}{\vdash}} (q, (j_1, j_2, \dots, j_k), (v_1, v_2, \dots, v_k)),$$

  where $q \in Q$, $j_1, j_2, \dots, j_k \in \{1, 2, \dots\}$ and $v_1, v_2, \dots, v_k \in \Gamma^*$.

Theorem 3.6. Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_{\text{acc}}, q_{\text{rej}})$ be a k-tape Turing machine. We show how to convert M to an equivalent 1-tape Turing machine $M' = (Q', \Sigma, \Gamma', \delta', q_0, \sqcup, q_{\text{acc}}, q_{\text{rej}})$, where we use

$$\Gamma' = \Gamma \times \{\Box, \dot{\Box}\} \cup \{\#\}$$

as the tape symbols. On input $w \in \Sigma^*$, the machine M' performs the following steps:

1. Format the tape by turning its content from
$$w_1 \cdots w_n$$
into
$$\#\dot{w}_1 \cdots w_n \sqcup \# \dot{\sqcup} \# \dot{\sqcup} \# \cdots \# \dot{\sqcup} \#$$
such that the tape of M' can simulate the k tapes of M.

2. Continue performing steps 3 – 4 until it halts.

3. Scan the tape from the first $\#$ to the last $\#$ to determine the k symbols under the dots.

4. Update the tape according to the transition function δ of M. If there is any $\dot{\#}$, then replace it with a $\dot{\sqcup}$ and then insert a $\#$ after it.

It can be shown that if M runs on w in time t, then M' runs on w in time $O(t^2)$. $\Box$

Definition 3.7. A **nondeterministic Turing machine** is

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_{\text{acc}}, q_{\text{rej}}),$$

where

$$\delta \subseteq ((Q \setminus \{q_{\text{acc}}, q_{\text{reg}}\}) \times \Gamma) \times (Q \times \Gamma \times \{-1, 0, +1\})$$

is the **transition relation** and other components are the same as those in the definition of Turing machine.

- A **configuration** of M is a triple in $Q \times \{1, 2, \dots\} \times \Gamma^*$, and we define the binary relation $\vdash_M$ over the configurations of M such that for any $p, q \in Q$, $i, j \in \{1, 2, \dots\}$ and $u, v \in \Gamma^*$,
$$(p, i, u) \underset{M}{\vdash} (q, j, v)$$
if and only if
$$\begin{aligned} u^{(1)} \cdots u^{(i-1)} u^{(i+1)} \cdots u^{(n)} \sqcup \sqcup \cdots &= v^{(1)} \cdots v^{(i-1)} v^{(i+1)} \cdots v^{(m)} \sqcup \sqcup \cdots \\ ((p, u^{(i)}), (q, v^{(i)}, j - i)) &\in \delta. \end{aligned}$$

- Let $u \in \Sigma^*$. We say that M **diverges** on input u (i.e., M does not **halt** on u) if for any integer $t \geq 1$ there exist $q \in Q$, $j \in \{1, 2, \dots\}$ and $v \in \Gamma^*$ such that
$$(q_0, 1, u) \overset{(t)}{\underset{M}{\vdash}} (q, j, v).$$

We say that M **accepts** u if

$$(q_0, 1, u) \quad \overset{*}{\underset{M}{\vdash}} \quad (q_{\text{acc}}, j, v)$$

for some $j \in \{1, 2, \dots\}$ and $v \in \Gamma^*$. We say that M **rejects** u if M neither accepts u nor diverges on u.

- If M halts on $u \in \Sigma^*$, then the **running time** of M on input u is the maximum integer t with

$$(q_0, 1, u) \quad \overset{(t)}{\underset{M}{\vdash}} \quad (q_{\text{acc}}, j, v) \qquad \text{or} \qquad (q_0, 1, u) \quad \overset{(t)}{\underset{M}{\vdash}} \quad (q_{\text{rej}}, j, v),$$

and the **accessed space** of M on input u is the maximum integer s with

$$(q_0, 1, u) \quad \overset{*}{\underset{M}{\vdash}} \quad (q, s, v)$$

where $q \in Q$, $j \in \{1, 2, \dots\}$ and $v \in \Gamma^*$.

Chapter 4

Time Complexity

4.1 P

Definition 4.1. We define

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathbf{TIME}(n^k).$$

4.2 NP

Definition 4.2. We define

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(n^k).$$