Theory of Computation

Chapter 1

Regular Languages

1.1 Languages

Definition 1.1. An **alphabet** is a finite nonempty set of symbols.

Definition 1.2. Let Σ be an alphabet.

- A **string** over Σ is a finite sequence of symbols from Σ. The collection of all strings over Σ is denoted by Σ^*.
- The **length** of a string w, denoted by $|w|$, is the number of symbols it contains.
- The string containing no symbols is called the **empty string**, denoted by ϵ.

Definition 1.3. A subset of Σ^* is called a **language** over Σ.

1.2 Deterministic Finite State Automata

Definition 1.4. A **deterministic finite state automaton** (DFA) 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.
- $\delta : Q \times \Sigma \to Q$ is a function, called the **transition function**.
- $q_0 \in Q$ is called the **start state**.
- $F \subseteq Q$ is called the **accepting states**.

Definition 1.5. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. For each string $w \in \Sigma^*$, we define $\delta_w : Q \to Q$ as follows, where $a \in \Sigma$ and $x \in \Sigma^*$.

- $\delta_\epsilon(p) = p$ for each $p \in Q$.
- $\delta_a(p) = \delta(p, a)$ for each $p \in Q$.
- $\delta_{xa}(p) = \delta(\delta_x(p), a)$ for each $p \in Q$.

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

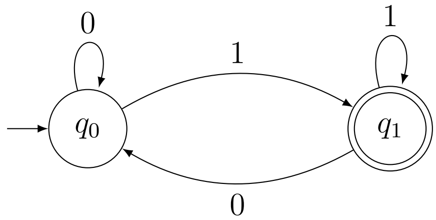
- We say that A accepts a string $w \in \Sigma^*$ if $\delta_w(q_0) \in F$.
- The **language** of A, denoted $L(A)$, is defined as the set of strings that are accepted by A.

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

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

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

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

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

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

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_1L_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_1L_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 $i, j \in \{1, \ldots, n\}$ and $k \in \{0, \ldots, n\}$, 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 $x_0, x_1, \ldots, x_\ell$ 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 $u_0, u_1, \ldots, u_{\ell+1}$ 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 $h \in \{0, \ldots, \ell\}$. 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 $h \in \{1, \ldots, \ell\}$, 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 symbols, called **variables**.
- Σ is a finite set of symbols, called **terminals**, and we have $V \cap \Sigma = \varnothing$.
- R is a finite set of **rules**, where each rule is a pair (A, γ) with $A \in V$ and $\gamma \in (V \cup \Sigma^*)$.
- $S \in V$ is the **start variable**.

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

- For any variable $A \in V$ and for any strings $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$, 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 strings $\alpha, \beta \in (V \cup \Sigma)^*$, we say that α **derives** β under G, denoted by

  $$\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta,$$

  if $\alpha = \beta$ or there exists a string $\gamma \in (V \cup \Sigma)^*$ 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.

- We say that G **accepts** a string $w \in \Sigma^*$ if

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

- The **language** of G, denoted $L(G)$, is the set of strings that are accepted by G.

Definition 2.4. A language L is **context-free** if there is a CFG G with $L(G) = L$.

2.2 Pushdown Automata

Definition 2.5. A **pushdown automaton (PDA)** is a 7-tuple

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

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.
- $\delta \subseteq (Q \times \Sigma \times \Gamma) \times (Q \times \Gamma^*)$ is the transition relation.
- $q_0 \in Q$ is the initial state.
- $Z \in \Gamma$ is the initial symbol.
- $F \subseteq Q$ is the set of accepting states.

Definition 2.6. Let $A = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ be a PDA. For each string $w \in \Sigma^*$, we define $\delta_w \subseteq (Q \times \Gamma^*) \times (Q \times \Gamma^*)$ such that the following properties are satisfied for each $a \in \Sigma$ and $x \in \Sigma^*$.

- $\delta_\epsilon = \{(p, \alpha, p, \alpha) : p \in Q \text{ and } \alpha \in \Gamma^*\}$.
- $\delta_a = \{(p, X\alpha, q, \gamma\alpha) : (p, a, X, q, \gamma) \in \delta \text{ and } \alpha \in \Gamma^*\}$.
- $\delta_{xa} = \{(p, \alpha, q, \beta) : (p, \alpha, r, \gamma) \in \delta_x \text{ and } (r, \gamma, q, \beta) \in \delta_a \text{ for some } (r, \gamma) \in Q \times \Gamma^*\}$.

Definition 2.7. Let $A = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ be a PDA.

- We say that A **accepts** a string $w \in \Sigma^*$ if $(q_0, Z, q, \gamma) \in \delta_w$ for some $q \in F$ and $\gamma \in \Gamma^*$.
- The **language** of A, denoted $L(A)$, is the set of strings that are accepted by A.

Chapter 3

Decidability

3.1 Turing Machines

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

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F, F'),$$

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 $\Sigma \subseteq \Gamma$.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ is the **transition function**.
- $q_0 \in Q$ is the **initial state**.
- $B \in \Gamma \setminus \Sigma$ is a special symbol, called the **blank symbol**.
- $F, F' \subseteq Q$ are the sets of **accepting states** and **rejecting states**, respectively, with $F \cap F' = \varnothing$.

Definition 3.2. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F, F')$ be a TM. A **configuration** of M is a pair (q, α), where $q \in Q$, α is a string over $\Gamma \cup \{\blacktriangleright\}$, and the symbol $\blacktriangleright$ appears exactly once before some symbol in α. For each configuration (p, α) of M, with

$$\alpha = X_1 \cdots X_{i-2} X_{i-1} \blacktriangleright X_i X_{i+1} \cdots X_k$$

and $\delta(p, X_i) = (q, Y, d)$, we define its **subsequent configuration** (q, β) as follows.

- If $d = -1$ and $i \geq 2$, then
$$\beta = X_1 \cdots X_{i-2} \blacktriangleright X_{i-1} Y X_{i+1} \cdots X_k.$$
- If $d = -1$ and $i = 1$, then
$$\beta = \blacktriangleright B Y X_2 \cdots X_k.$$
- If $d = +1$ and $i \leq k - 1$, then
$$\beta = X_1 \cdots X_{i-1} Y \blacktriangleright X_{i+1} \cdots X_k.$$
- If $d = +1$ and $i = k$, then
$$\beta = X_1 \cdots X_{k-1} Y \blacktriangleright B.$$

3.2 Programs

Definition 3.3. An **ordered alphabet** is a finite ordered set of symbols. If

$$\Gamma = (C_0, C_1, \ldots, C_m)$$

is an ordered alphabet, then the **successor** of each symbol in Γ is defined as follows.

- The successor of C_i is C_{i+1} for each $i \in \{0, \ldots, m-1\}$.
- The successor of C_m is C_0.

Definition 3.4. We define the language of $\boldsymbol{\mathcal{P}''}$ **programs** over the alphabet $\{\lambda, R, (,)\}$ as follows.

- λ and R are a $\mathcal{P}''$ programs.
- If P_1 and P_2 are $\mathcal{P}''$ programs, then P_1P_2 is a $\mathcal{P}''$ program.
- If P is a $\mathcal{P}''$ program, then (P) is a $\mathcal{P}''$ program.

A $\boldsymbol{\mathcal{P}''}$ **machine** is a $\mathcal{P}''$ program equipped with an alphabet Σ of input symbols and an ordered alphabet

$$\Gamma = (B, C_1, \ldots, C_m)$$

of tape symbols, where $\Sigma \subseteq \Gamma \setminus \{B\}$.

Definition 3.5. Let P be a $\mathcal{P}''$ program over input alphabet Σ and tape alphabet $\Gamma = (B, C_1, \ldots, C_m)$. A **configuration** of P is a string α over $\Gamma \cup \{\blacktriangleright\}$, where the symbol $\blacktriangleright$ appears exactly once before some symbol in α.