1. Finite Automata and Regular Language

Deterministic Finite Automata (DFA): A 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set called the *states*;
- Σ is a finite set called the *alphabet*;
- $\delta : Q \times \Sigma \to Q$ is the *transition function*;
- $q_0 \in Q$ is the *start state*;
- $F \subseteq Q$ is the set of *acceptance states*.

Computation by DFA: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let $w = w_1 w_2 \cdots w_n$ be a string with $w_i \in \Sigma$ for all $i \in [n]$. Then M **accepts** w if there exists a sequence of states $r_0, r_1, \cdots, r_n$ in Q such that

- $r_0 = q_0$;
- $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, 1, \cdots, n-1$;
- $r_n \in F$.

For a set A, we say that M **recognize** A if $A = \{l \mid M \text{ accepts } l\}$

Regular language: A language is called **regular** iff some finite automata (here we say DFA) recognizes it..

Regular operators: Let A and B be *languages* (the subset of Σ^*). We define the following three regular operators.

- **Union**: $A \cup B = \{x \mid x \in A \vee x \in B\}$;
- **Concatenation**: $A \circ B = \{xy \mid x \in A \wedge y \in B\}$;
- **Kleene star**: $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \wedge x_i \in A\}$.

Nondeterministic Finite Automata (NFA): A 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set called the *states*;
- Σ is a finite set called the *alphabet*;
- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the *transition function*, where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$;
  - $\mathcal{P}(Q)$ can be $\varnothing$, thus we can ignore those transitions.
- $q_0 \in Q$ is the *start state*;
- $F \subseteq Q$ is the set of *acceptance states*.

Computation by NFA: Let $N = (Q, \Sigma, \delta, q_0, F)$ be a NFA and let w be a string. Then N **accepts** w if we can write w as $y_1 y_2 \cdots y_m$, where $y_i \in \Sigma_\epsilon$ for all $i \in [m]$ and a sequence of states $r_0, r_1, \cdots, r_m$ exists in Q such that

- $r_0 = q_0$;
- $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, 1, \cdots, m-1$;
- $r_m \in F$.

For a set A, we say that N **recognize** A if $A = \{l \mid N \text{ accepts } l\}$

Theorem. Every NFA has an equivalent DFA, i.e., they recognize the same language.

Tips. DFA to NFA simple. Let Q_{DFA} be $\mathcal{P}(Q_{NFA})$.

Proof. (NFA to DFA) (true, but impractical)

- *Step 1*. For any state $q \in Q$, compute its *silently reachable class* $E(q)$.

$$\begin{array}{|l|}\hline \textbf{initially set } E(q) = \{q\}; \\ \quad \textbf{repeat} \\ \qquad E'(q) = E(q) \\ \qquad \forall x \in E(q), \textbf{if } \exists y \in \delta(x,\epsilon) \wedge y \notin E(q), E(q) = E(q) \cup \{y\} \\ \quad \textbf{until } E(q) = E'(q) \\ \textbf{return } E(q). \\ \hline \end{array}$$

- *Step 2*. Build the equivalent DFA. $N = (Q, \Sigma, \delta, q_0, F) \Longrightarrow M = (Q', \Sigma, \delta', q_0', F')$.
  - $Q' = \mathcal{P}(Q)$;
  - $\delta'(R, a) = \cup\{E(q) \mid q \in Q \wedge (\exists r \in R)(q \in \delta(r, a))\}$;
  - $q_0' = E(q_0)$;
  - $F' = \{R \subseteq Q' \mid R \cap F \neq \varnothing\}$

Corollary. A language is **regular** iff some NFA recognizes it.

Theorem. The class of regular languages is closed under $\{\cup, \circ, ^*\}$

- **Union**

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$;
2. q_0 is the new start state;
3. $F = F_1 \cup F_2$;
4. For any $q \in Q$ and any $a \in \Sigma_\epsilon$

$$\delta(q,a) = \begin{cases} \{q_1, q_2\} & q = q_0 \wedge a = \epsilon \\ \emptyset & q = q_0 \wedge a \neq \epsilon \\ \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

- **Concatenation**

1. $Q = Q_1 \cup Q_2$;
2. the start state is q_1;
3. the set of accept states is F_2;
4. For any $q \in Q$ and any $a \in \Sigma_\epsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 - F_1 \\ \delta_1(q,a) & q \in F_1 \wedge a \neq \epsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \wedge a = \epsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

- **Kleene Star**

1. $Q = \{q_0\} \cup Q_1$;
2. the new start state is q_0;
3. $F = \{q_0\} \cup F_1$;
4. For any $q \in Q$ and any $a \in \Sigma_\epsilon$

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 - F_1 \\ \delta_1(q, a) & q \in F_1 \wedge a \neq \epsilon \\ \delta_1(q, a) \cup \{q_1\} & q \in F_1 \wedge a = \epsilon \\ \{q_1\} & q = q_0 \wedge a = \epsilon \\ \emptyset & q = q_0 \wedge a \neq \epsilon \end{cases}$$

Lemma. The class of regular languages is closed under complementation and intersection.

- **Complement**: $\bar{A} = \Sigma^* - A$;
- **Intersection**: $A \cap B = \{x \mid x \in A \wedge x \in B\}$.

Proof.

- **Complement**: suppose $N = (Q, \Sigma, \delta, q, F)$ is a DFA, then $\bar{N} = (Q, \Sigma, \delta, q, Q - F)$ will recognize $\bar{A}$;
- **Intersection**: $A \cap B = \overline{\bar{A} \cup \bar{B}}$, since the class of regular languages is closed under complement and union, then it should be closed under intersection.

Regular Expression: Given alphabet Σ, we say that R is a regular expression if R is:

- a for some $a \in \Sigma$;
- ε (empty character);
- $\varnothing$ (empty set);
- $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions;
- $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, sometimes written as $R_1 R_2$;
- (R_1^*), where R_1 is a regular expression.

Language Defined by Regular Expressions

Regular expression R	**Language** $L(R)$
a	$\{a\}$
ϵ	$\{\epsilon\}$
$\varnothing$	$\varnothing$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
(R_1^*)	$L(R_1)^*$

Theorem. A language is regular iff some regular expression describes it.

- ($\Longleftarrow$) by induction, simple;

- ($\Longrightarrow$) Some idea as Floyd-warshall Algorithm. Let i denote q_i. Let $R(i,j,k)$ be the regular expression from i to j and we use the intermediate state not greater than k.

$$R(i,j,k) = R(i,j,k-1) \cup R(i,k,k-1)R(k,k,k-1)^* R(k,j,k-1)$$

  Then,

$$L(M) = \cup\{R(1,j,n) \mid j \in F\}$$

Examples.

- $L_1 = \{l \in \{0,1\}^* \mid l \text{ has an equal number of 0s and 1s}\}$ is regular (NFA is easy to construct);
- $L_2 = \{l \in \{0,1\}^* \mid l \text{ has an equal number of occurence of 01 and 10 as substrings}\}$ is not regular.
- **Explanation**: In L_1, we must remember the previous string, so we need counting and the state must be infinite; but in L_2, the difference between occurence of 01 and 10 as substrings can only be $0, \pm 1$, thus, we can use finite state to remember it.
- By the way, we can use the pumping lemma to prove that L_2 is not regular (First prove $\{0^n 1^n \mid n \in \mathbb{N}\}$ is not regular, then let $L_2 \cap 0^* 1^* = \{0^n 1^n \mid n \in \mathbb{N}\}$ and use contradiction to prove it.)

Lemma. (*The pumping lemma for regular languages*) If A is a regular language, then there is a number p (i.e., **the pumping length**) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions.

1. $|y| > 0$;
2. $|xy| \le p$;
3. for each $i \ge 0$, we have $xy^i z \in A$.

Any string xyz in A can be pumped along y.

Proof. Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and $p = |Q| + 1$. According to the pigeonhole principle, the recognizing route must have a cycle, say y. Then the lemma is proved.

Problems from Formal Language Theory

- **Accpetance Problem**: does a given string belong to a given language?
- **Emptiness Problem**: is a given language empty?
- **Equality**: are two given language equal?

Problems for DFA

- **Accpetance Problem**: Given a DFA (NFA) A and a string w, does A accept w?
- **Emptiness Problem**: Given a DFA (NFA) A, is the language $L(A)$ empty?
- **Equality**: Given two DFA (NFA) A and B, is $L(A)$ equal to $L(B)$?

Proof.

- Take DFA as an example. Simple.
- Take DFA as an example, use DFS/BFS.
- Use the lemma $L(A) = L(B) \iff (L(A) - L(B)) \cup (L(B) - L(A)) = \varnothing$. We can use A and B to construct the DFA that can express $(L(A) - L(B)) \cup (L(B) - L(A))$. Then we can use the algorithm of emptiness problem to solve it.