Theory of Computation

Based on lectures by Dr. Arpit Sharma
Notes taken by Rwik Dutta

These notes are not endorsed by the lecturers, and I have modified them (often significantly) after lectures. They are nowhere near accurate representations of what was actually lectured, and in particular, all errors are almost surely mine.[1]

Contents

1 Finite Automata **2**
1.1 Language of Finite Automata 2
1.2 Set Operations on Regular Languages 2

[1]This is how Dexter Chua describes his lecture notes from Cambridge. I could not have described mine in any better way.

1 Finite Automata

Definition 1 (Deterministic Finite Automaton)**.** A collection $(Q, \Sigma, \delta, q_0, F)$ such that

1. Q is a finite set of *states.*
2. Σ is a finite *alphabet.*
3. $\delta : Q \times \Sigma \to Q$ is the *transition function.*
4. $q_0 \in Q$ is the start state.
5. $F \subseteq Q$ is the set of *accepting states.*

We will use automata to solve set membership problems, i.e., given a finite alphabet Σ and a finite language $L \subset \Sigma^*$, we need to find whether a given string $x \in L$.

Note. Before a finite automaton has received any input, it is in its initial state, which is an accepting state precisely if the null string is accepted.

Note. At each step, a finite automaton is in one of a finite number of states (it is a finite automaton because its set of states is finite). Its response depends only on the current state and the current symbol.

1.1 Language of Finite Automata

Definition 2 (Language of Automaton)**.** $L(M)$ of an automaton M is the set of all strings x that are accepted by the automaton.

Definition 3 (Extended transition function)**.** Let $M(Q, \Sigma, \delta, q_0, F)$ be a finite automaton. The extended transition function is given by

$$\begin{aligned} \delta^* : Q \to \Sigma^* \\ \delta^*(q, \epsilon) = q \\ \delta^*(q, xa) = \delta(\delta^*(q, x), a) \end{aligned}$$

for all $q \in Q, x \in \Sigma^*, a \in \Sigma$. $\epsilon \in \Sigma^*$ represents the empty string.

Hence, $x \in L(M)$ if $\delta^*(q_0, x) \in F$.

Definition 4 (Regular language)**.** A language L is a regular language if $\exists$ some finite automaton M such that $L(M) = L$.

1.2 Set Operations on Regular Languages

Regular languages are closed under certain operations.

Theorem 1. If $M(Q, \Sigma, \delta, q_0, F)$ accepts L, L^C is accepted by the finite automaton $M'(Q, \Sigma, \delta, q_0, F')$ where

$$F' = Q \backslash F = F^C$$

Lemma 1.1. L is a regular language $\implies$ L^C is a regular language.

Theorem 2. Let $M_1(Q_1, \Sigma, \delta_1, q_1, F_1)$ accept L_1 and $M_2(Q_2, \Sigma, \delta_2, q_2, F_2)$ accept L_2. Let $M(Q, \Sigma, \delta, q_0, F)$ be a finite automaton with

$$Q = Q_1 \times Q_2$$

$$q_0 = (q_1, q_2)$$

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

We have $L(M) =$

1. $L_1 \cup L_2$ if $F = \{(p, q) | p \in F_1 \text{ or } q \in F_2\}$
2. $L_1 \cap L_2$ if $F = \{(p, q) | p \in F_1 \text{ and } q \in F_2\}$
3. $L_1 \backslash L_2$ if $F = \{(p, q) | p \in F_1 \text{ and } q \notin F_2\}$

Lemma 2.1. L_1, L_2 are regular languages with the same alphabet $\implies$ $L_1 \cup L_2, L_1 \cap L_2$ and $L_1 \backslash L_2$ are regular languages.