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

1 Finite Automata

1.1 Deterministic Finite Automaton(DFA)

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.

Definition 2 (String accepted by DFA)**.** A DFA has one only one final state for a given string. The string is said to be accepted by the DFA if the final state is also an accepting state of the DFA.

1.2 Regular Language

Definition 3 (Language of Automaton)**.** $L(M)$ of an automaton M is the set of all strings x that are accepted by the automaton.

Definition 4 (Extended transition function of DFA)**.** 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 5 (Regular language)**.** A language L is a regular language if $\exists$ some finite automaton M such that $L(M) = L$.

1.3 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.

Theorem 3 (Contatenation)**.** If L_1, L_2 are regular languages, so is

$$L_1 L_2 = \{xy|x \in L_1, y \in L_2\}$$

Theorem 4 (Kleene Closure)**.** If L is a regular language, so is

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Note. $L^0 = \{\epsilon\}, L^1 = L, L^2 = LL$ and so on.

1.4 Regular Expression

Definition 6 (Regular Expression)**.** A regular expression over a finite alphabet Σ and operators $*, +, \cdot$ is defined as

1. $\varnothing, \epsilon$ and $a \in \Sigma$ are regular expressions.
2. If R is a regular expression so is R^*.
3. If R_1, R_2 are regular expressions so are $R_1 + R_2$ and $R_1 R_2$.

Example 1. Let $\Sigma = \{0,1\}$. $\epsilon, 1, 0^*, 1+0, 11, (1+01)^*$ are all examples of regular expressions.

Definition 7 (Precedence of Operators)**.** $*, \cdot, +$ is the order of decreasing precedence.

Definition 8 (Language represented by a regular expression)**.**

1. $\varnothing \to \varnothing$
2. $\epsilon \to \{\epsilon\}$
3. $a \to \{a\}$
4. $R \to L \implies R^* \to L^*$
5. $R_1 \to L_1$ and $R_2 \to L_2 \implies R_1 + R_2 \to L_1 \cup L_2$
6. $R_1 \to L_1$ and $R_2 \to L_2 \implies R_1 R_2 \to L_1 L_2$

Theorem 5. Every regular expression represents a unique regular language.

Note. Every regular language represents a regular expression. However, this expression is not unique.

Definition 9 (Equality)**.** Two regular expressions are said to be equal if they represent the same language.

Theorem 6. For $\Sigma = \{0,1\}$, $(0^*1^*)^* = (0+1)^*$. Both represent Σ^*.

1.5 Non-deterministic Finite Automaton(NFA)

Definition 10 (Non-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. $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition 'function'.
4. $q_0 \in Q$ is the start state.
5. $F \subseteq Q$ is the set of accepting states.

Note. $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Note. $\mathcal{P}(Q)$ is the set of all subsets of Q.

Note. The transition function of an NFA is not a mathematical function. For a given state $q \in Q$ and symbol $a \in \Sigma$, $\delta(q, a)$ may not exist or may have multiple values in Q. This is what mathematicians call a relation.

Note. The definition of regular languages refers to finite automaton which includes NFA.

Definition 11 (String accepted by NFA)**.** An NFA can end up in mutiple final states or no state at all with some input string. If **any one** of these final states is an accepting state of the NFA, we say the NFA accepts the string.