Intro to Complexity Theory

Kunwar Shaanjeet Singh Grover

Contents

Chapter 1

Introduction

1.0.1 Automata, Computability, and Compelxity

What are the fundamental capabilities and limitations of computers?

- Complexity Theory: What makes some problems harder than others?
- Computability Theory
- Automata theory: Deals with the definations and properties of mathematical models of computation

1.0.2 Mathematical notions and terminology

Strings and Languages We define an **alphabet** to be any nonempty finite set. The members of the alphabet are the **symbols** of the alphabet. Generally $\sum$ and Γ are used to designate alphabets.

A **string over an alphabet** is a finite sequence of symbols from the alphabet written next to one another and not seperated by commas.

Chapter 2

Regular Languages

2.1 Finite Automata

Defination 1. *A **finite automation** is a 5-tuple $(Q, \sum, \delta, q_0, F)$ where*

1. *Q is a finite set called the **states***
2. *$\sum$ is a finite set called the **alphabet***
3. *$\delta : Q \times \sum \rightarrow Q$ is the **transition function***
4. *$q_0 \in Q$ is the **start state**, and*
5. *$F \subseteq Q$ is the **set of accept states***

If A is the set of all strings that machine M accepts, we say that A is the **language of machine** M and write $L(M) = A$. We say that M **recognizes** A or that M **accepts** A.

A machine may accept several strings, but it always recognizes only one lnaguage.

Formal defination of Computation

Let $M = (Q, \sum, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \ldots w_n$ be a string where each w_i is a member of the alphabet $\sum$. Then M **accepts** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

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

Condition 1 says that the machine starts in the start state, Condition 2 says that the machine goes from the state to state according to the transition function. Condtion 3 sayas that machine accepts its input if it ends up in accept state. We say that M **recognizes language** A if $A = \{w \mid Macceptsw\}$

Defination 2. *A language is called a* ***regular language*** *if some finite automaton recognizes it.*

2.1.1 The Regular Operations

Defination 3. *Let A and B be languages. We define the regular operations* ***union, concatenation,*** *and* ***star*** *as follows:*

- ***Union*** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- ***Concatenation:*** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
- $A^* = \{x_1x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

2.2 NonDeterminism

When the machine was in a given state and read the next input symbol, we knew what the next state would be, it is determined. We call this **deterministic** computation. In a **nondeterministic** machine, several choices may exist for the next state at any point. Nondeterminism is a generalization of determinism, so every deterministic finite automaton is automatically a nondeterministic finite automaton.

2.2.1 Formal defination of a nondeterministic finite automaton

Defination 4. *A* ***nondeterministic finite automaton*** *is a 5-tuple* $(Q, \sum, \delta, q_0, F)$, *where*

1. *Q is a finite set of states,*
2. *$\sum$ is a finite alphabet,*
3. *$\delta : Q \times \sum_\epsilon \to P(Q)$ is the transition function,*
4. *$q_0 \in Q$ is the start state, and*
5. *$F \subseteq Q$ is the set of accept states.*

The formal defination of computation for an NFA is similar to that or a DFA. Let $N = \{Q, \sum, \delta, q_0, F\}$ be an NFA and w a string over the alphabet $\sum$. Then we say that N **accepts** w if we can write w as $w = y_1y_2 \ldots y_m$, where each y_i is a member of $\sum_\epsilon$ and a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions

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

Condition 1 says that the machine starts out in the start state. Condition 2 says that state r_{i+1} is one of the allowable next states when N is in the state r_i and reading y_{i+1}. Observe that $\delta(r_i, y_{i+1})$ is the *set* of allowable next states and so we say that r_{i+1} is a member of that set. Finally, condition 3 says that the machine accepts its input if the last state is an accept state.

2.2.2 Equivalence of NFAs and DFAs

Theorem 1. *Every nondeterministic finite automaton has an equivalent deterministic finite automaton*

Proof idea: We keep track of of the current states and create a transition function based on that. If there are k states o the NFA, then it has 2^k subsets of states. To keep track of these states, the DFA will have 2^k states.

Corollary 2. *A language is regular if and only if some nondeterministic finite automaton recognizes it.*

2.2.3 Closure under the regular operations

Theorem 3. *The class of regular languages is closed under the union operation*

Proof: Let:

$$N_1 = (Q_1, \sum, \delta_1, q_1, F_1)\text{recognize } A_1\text{, and}$$
$$N_2 = (Q_2, \sum, \delta_2, q_2, F_2)\text{recognize } A_2.$$

Construct $N = (Q, \sum, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$
   The states of N are all the states of N_1 and N_2, with the addition of a new start state q_0.
2. The state q_0 is the start state of N.
3. The set of accept states $F = F_1 \cup F_2$.
4. Define δ so that for any $q \in Q$ and any $a \in \sum_\epsilon$,

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

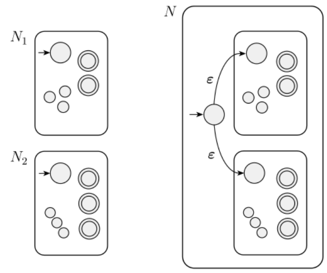

Figure 2.1: Construction of an NFA N to recognize $A_1 \cup A_2$

Theorem 4. *The class of regular languages is closed under the concatenation operation*

The formal proof is similar to the previous proof. The follwing image describes the contruction:

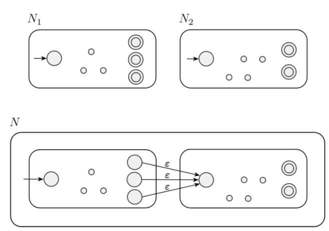

Figure 2.2: Construction of N to recognize $A_1 \circ A_2$

Theorem 5. *The class of regular languages is closed under the star operation.*

The formal proof is similar to union proof. The following image describes the construction:

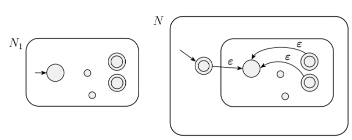

Figure 2.3: Construction of N to recognize A^*