CS331 : THEORY OF COMPUTATION

Lecture 4

July 29, 2003

EXAMPLES

Example 1: Let us define a Language L_1

$$\Sigma = \{0\}$$

$$L_1 = \{0^n | n = 6k, k \geq 1\}$$

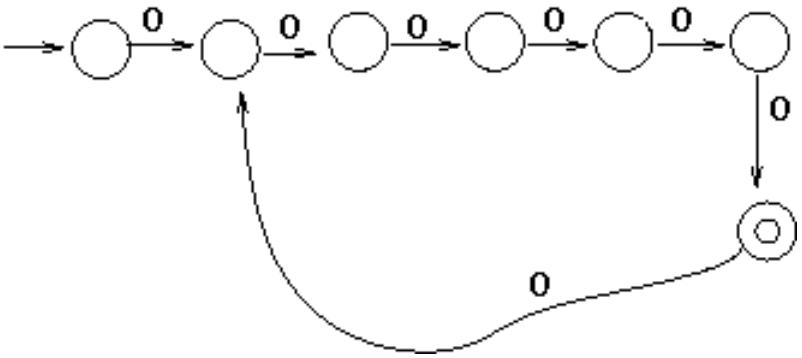

Figure 1: Transition Diagram for DFA recognizing language L1

In this model while reading the input string the information required to check the validity of string is finite. Precisely we have to remember 6 states. Intermixing of any two of them may lead to incorrect recognition. In this language there are finite number of states and thus finite amount of information stored in them.

Example 2: Let us define a Language L_2

$$\Sigma = \{0, 1\}$$

$$L_2 = \{0^n 1^n | n \geq 1\}$$

To check acceptance of strings of this language, we have to remember number of 0's which can be arbitrarily large. So we cannot express this language with any Finite Automaton.

Example 3: Let us define a language L_3 such as

$$\Sigma = \{0\}$$

$$L_3 = \{0^{n^2} | n \geq 1\}$$

Here though the situation is not like L_2, but the difference between the length of strings to be accepted keeps on increasing. So, the information to be remembered becomes arbitrarily large. Due to that, we cannot recognize this language using finite amount of information. We are sure that this language cannot be recognized by DFA.

1 NON-DETERMINISTIC FINITE AUTOMATION (NFA)

As we know that in **DFA**

$$Q : finite\ set\ of\ states$$

$$\Sigma : alphabet$$

$$s_0 \in Q : starting\ state$$

$$F \subseteq Q : set\ of\ accepting\ states$$

Transition Function in DFA is defined as

$$\delta : Q \times \Sigma \rightarrow Q$$

1.1 Difference between DFA and NFA

In **DFA** the transition function gives exactly one state, when applied from some state on an input symbol. In the nondeterministic automaton there can be several possible next states , and the automaton 'guesses' (always correctly) which next state (of the set of possible next states) will lead to acceptance of the input string. **DFA** is a particular case of **NFA**.
So, transition function in NFA is

$$\delta : Q \times \Sigma \rightarrow 2^Q$$

Here the Transition Function will give set of states

$$\Sigma : alphabet$$

$$Q : finite\ set\ of\ states$$

$$\delta : Q \times \Sigma \rightarrow 2^Q$$

Here **NFA** has four states : A,B,C and D

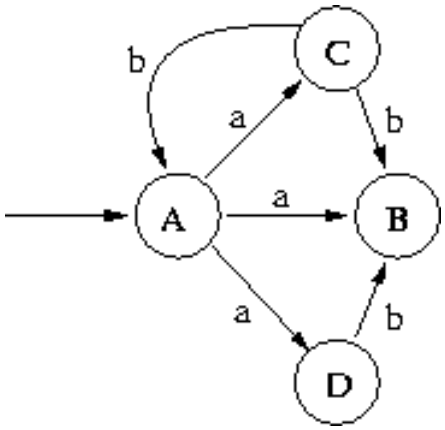

Figure 2: Transition Diagram of an NFA : N_1

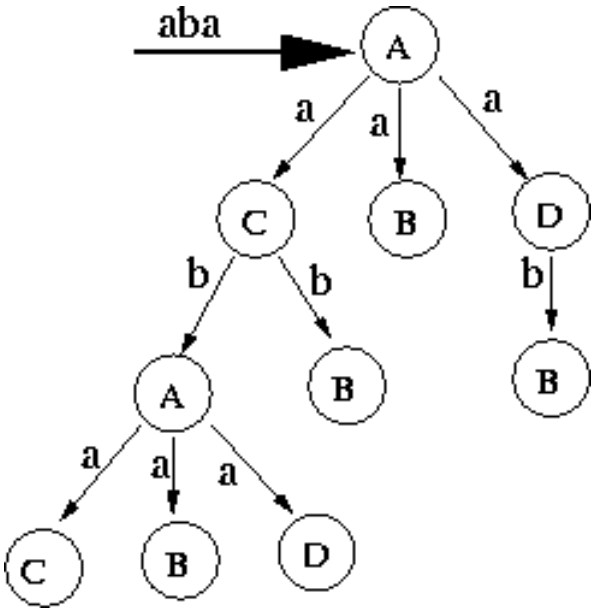

Figure 3: Possible transitions in N_1 on seeing the input "aba"

1.2 Acceptance condition of NFA

In **DFA** the acceptance condition is

$$\hat{\delta}(s_0, w) \in F$$

It means that on applying string w to the automata with s_0 as the starting state, we should end up in a final(accepting) state.

But in **NFA** the acceptance condition is defined as:

$$\hat{\delta}(s_0, w) \bigcap F \neq \phi$$

An input sequence is accepted by a **NFA** if there exists a sequence of transitions, corresponding to the input sequence , that leads from the initial state to some final state.

$$\delta : Q \times \Sigma \rightarrow 2^Q$$

$$\hat{\delta} : Q \times \Sigma^* \rightarrow 2^Q$$

$$Here\ \Sigma^*\ is\ set\ of\ strings$$

$$\hat{\delta}(q, \varepsilon) = \{q\}; \forall q \in Q$$

$$\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$$

$$= \bigcup_{p \in \hat{\delta}(q,x)} \delta(p.a)$$

Applying to our example in fig3

$$\hat{\delta}(A, \varepsilon) = \{A\}$$

$$\hat{\delta}(A, a\varepsilon) = \delta(A, a) = \{B, C, D\}$$

$$\hat{\delta}(A, ab) = \bigcup_{p \in \{B,C,D\}} \delta(p, b) = \delta(B, b) \cup \delta(C, b) \cup \delta(D, b)$$

$$\implies \phi \cup \{A, B\} \cup \{B\} = \{A, B\}$$

$$\hat{\delta}(A, aba) = \bigcup_{\hat{\delta}(A,ab)} \delta(p, a) = \bigcup_{p \in \{A,B\}} \delta(p, a)$$

$$= \delta(A, a) \cup \delta(B, a)$$
$$= \{B, C, D\} \cup \phi$$
$$= \{B, C, D\}$$