MC-304 (Theory of computation)

Anish Sachdeva
DTU/2K16/MC/13

Assignment - II

Q1) What is Chomsky classification of languages? Explain with example?

Ans) Type-3 Grammars (Regular Grammars)

Type-3 grammars generates regular languages. They must have a single non-terminal on the left hand side and a right hand side consisting of a single or terminal followed by a single non-terminal. The productions must be in the form $X \to a$, $X \to aY$ where $X, Y \in N$ (Variables / Non-terminals) and $a \in \Sigma$ (Language / terminals). The rule $S \to \varepsilon$ is allowed if S does not appear on the right side of any rule.

Example

$X \to \varepsilon$

$X \to a | aY$

$Y \to b$

All languages created/generated by type-3 grammar can be expressed as regular expressions and finite automata.

Type-2 (Context Free grammars)

Context free grammars are of the form $A \to \gamma$ where $A \in V$ (Variables / Non-terminals) and ~~$\gamma \in$ Terminals (Σ)~~ $\gamma \in (\Sigma \cup V)^*$ String of terminals and non-terminals. It is a superset of Regular Grammar (Type-3) languages.

Some of Type-2 language may be represented as a regular expression, but not all type-2 grammar can be represented as finite automata.

Some examples of context free grammar are:-

$S \rightarrow aA|bA|\varepsilon$

$A \rightarrow Aa|a$

This is an example of a grammar that accepts the language of all strings that ends with an 'a' and 'ε' $L = \{\varepsilon\} \cup \{A / A \text{ ends with } 'a'\}$

Type-I Grammar (Context-sensitive Languages)

Type-I grammars generate context-sensitive languages. The production must be in the form.

$\alpha A \beta \rightarrow \alpha \gamma \beta$ where

$A \in$ (Non-terminal)

$\alpha, \beta, \gamma \in (\Sigma \cup V)^*$ (collection of non-terminals and terminals)

The strings α and β may be empty, but γ must not be non-empty. The rule language generated by these grammars are recognized by a linear bounded-automaton.

$AB \rightarrow AbBc$

$A \rightarrow bcA$

$B \rightarrow b$

Type-0 Grammar (Recursively-Enumerable Languages)

These languages generate recursively enumerable languages. The production has no restriction. They are also called turing enumerable or turing languages/grammar. They are any phrase structure grammar including all formal grammar.

They generate languages that are recognized by Turing machines.

The production can be in the form $\alpha \to \beta$ where α is a string of terminals and non-terminals with at least one non-terminal and α cannot be null. β is a string of terminals and non-terminals.

Example:-

$S \to ACaB$

$Bc \to acB$

$CB \to DB$

$aD \to Db$

Q2) Check whether the strings: 001100, 00101O, 01010 are in the language generated by the grammar with the production rules

$S \to 0S1 \mid 0A \mid 0 \mid B \mid 1 \Rightarrow S \to 0S1 \mid A \mid B$

$A \to 0A \mid 0$

$B \to 1B \mid 1$

$A = \{0^n, n \geq 1\}$

$B = \{1^n, n \geq 1\}$

$S = \{0^m 0^n 1^m, m \geq 0, n \geq 1\} \cup \{0^m 1^n 1^m, m \geq 0, n \geq 1\}$

i) 001100 : Can't be generated as alternating 0's and 1's similarly (ii) and (iii) can also not be generated as they have alternating 0's and 1's.

Q3) What is a context-free grammar? Construct a context-free grammar to generate the set of strings over $\{0,1\}$ containing twice as many 0's as 1's.

Ans) In formal language theory, a context-free grammar is a formal grammar in which every production rule is of the form:

$A \to \alpha$

where $A \in V$ (Non-terminal / Variable) A is a non-terminal

symbol and $\alpha \in (\Sigma \cup V)^*$ - combination of terminal symbols and non-terminal symbols. A grammar is considered context-free when its production rules can be applied regardless of the context of a non-terminal.

The automata/grammar that generates a string with 2 0's for every one (1) will be:-

$$S \rightarrow 00IS \mid 0I0S \mid I00S \mid \varepsilon$$

Q4) Prove that any set accepted by a finite automaton M is represented by a regular expression.

Ans) We are given that there exists some regular language that (L) that is accepted by some DFA (Deterministic finite state automata) M_1 such that $L = L(M_1)$.

Now, we will generate a NFA (Non-deterministic Finite State Automata) from it following the steps:-

i) There should be only one start state with outdegree 1 and indegree 0. and it shouldn't be the final state

ii) There should must be one final state with indegree ≥ 1 and outdegree = 0.

iii) There shouldn't be multiple final states.

iv) Eliminate multiple edges between the same states and have edge between every state with all options delimited with comma (,).

Now, we will write a regular expression from the NDFA using process of simplification.

We will select any state that is not the start state or final state and remove it by connecting / creating all edges that were joining it. eg.

Q5) Represent the following sets by regular expressions :-

a) $\{a^2, a^5, a^8 \ldots\} = \{a^{2+3n}, n \geq 0\}$

$R = aa(aaa)^*$

b) $\{w \in \{a,b\}^* \mid w$ has only $a\}$

$R = b^*ab^*$

c) $\{a^n / n$ is divisible by 2 or 3 or $n=5\}$

$S_2 = (aa)^*$ $S_3 = (aaa)^*$ $S_5 = aaaaa$

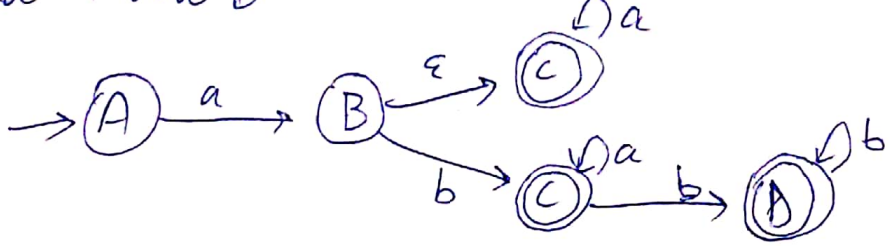
$S = (aa)^* | (aaa)^* | aaaaa$

d) The set of all strings over $\{a,b\}$ beginning and ending with a

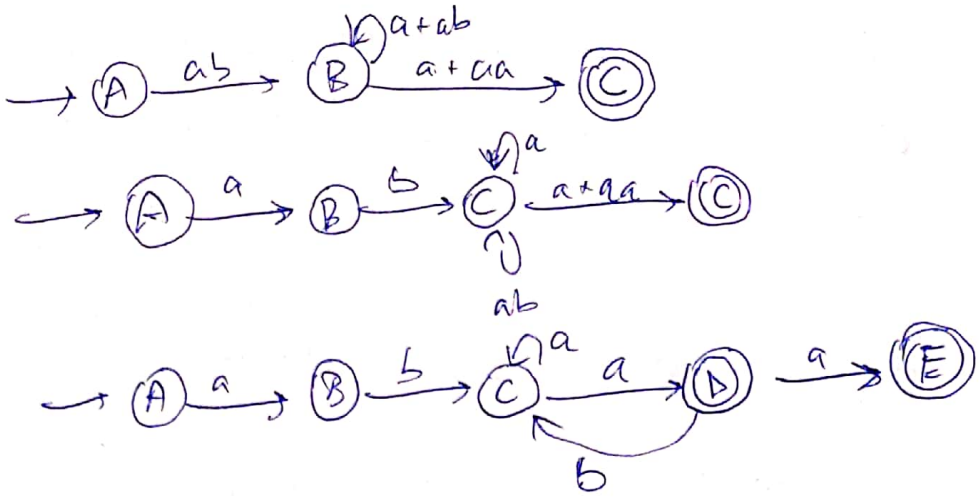
$S = a(a|b)^*a$

Q6) Construct transition systems / finite automata equivalent to regular expressions representing sets :-

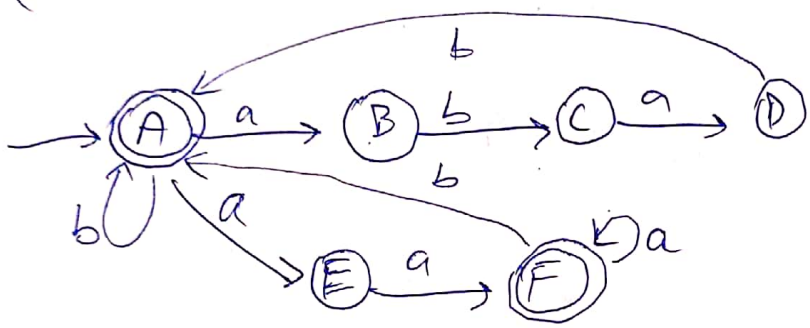
a) $aa^* + aba^*b^*$



b) $ab(a+ab)^*(a+aa)$

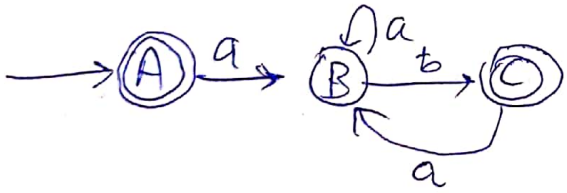


c) $(abab)^* + (aaa^* + b)^*$



d) $((aa^*)^*b)^*$

$(a+b)^+ = \{a, a^+b, a^+ba^+b, a^+ba^+ba^+b, \ldots\}$

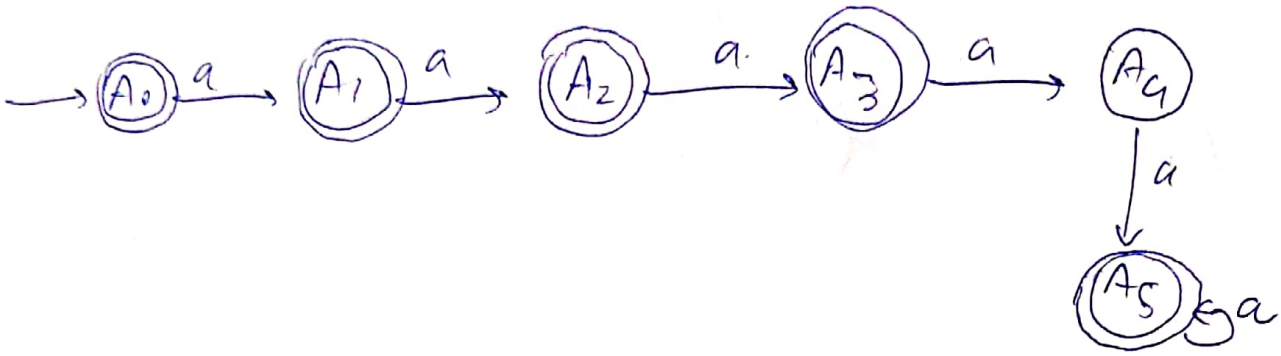


97) Show that the following sets are regular:

Ans; TO

TO show that any RL is regular we can generate a finite automata for it, as the language recognized by finite automata is a regular language.

i) $\{a^n ; n \geq 0, n = 4\}$

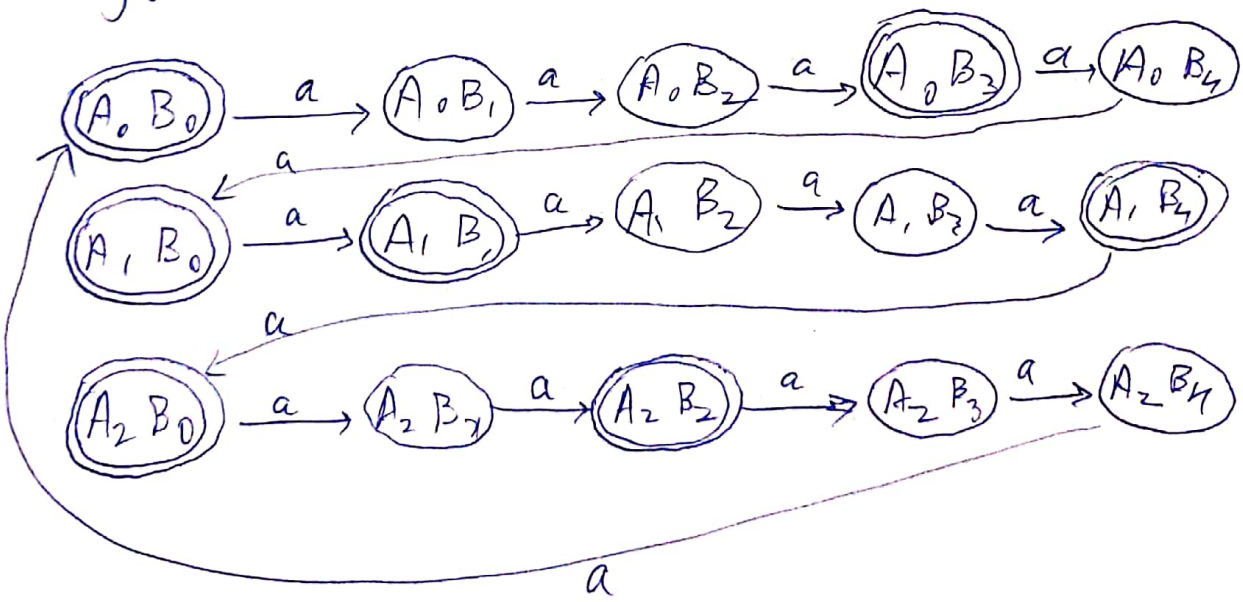


ii) $\{a^n ; n$ is either multiple of 3 or multiple of 5$\}$

$\{a^n ; n \bmod 3 = 0, n \bmod 5 = 0\}$

$\{A_i\}$ $n \bmod 3 = i$

$\{B_j\}$ $n \bmod 5 = j$



c) $\{a^n : n \text{ is a multiple of } 3 \text{ but not of } 5\}$

$\{a^n : n \bmod 3 = 0,\ n \bmod 5 \neq 0\}$