Fifth Semester B.E. Degree Examination

CBCS - Model Question Paper - 1

AUTOMATA THEORY AND COMPUTABILITY

Time: 3 hrs. Max. Marks: 80

Note : *Answer any FIVE full questions, selecting ONE full question from each module.*

MODULE - 1

1. a. Define the terms :

i. Kleene closure and kleene plus

ii. Language

iii. Alphabet

iv. Finite automata **(08 Marks)**

Ans. i. Kleene closure and kleene plus :

The kleene closure Σ^* is defined as follows :

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup .....$

Which is the set of words of any length. Each string is made up of symbols only from Σ.

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

$\Sigma^0 = \{\varepsilon\}$ $\Sigma^1 = \{0,1\}$, $\Sigma^2 = \{00,01,10,11\}$ ....

$\Sigma^* = \{\varepsilon, 0,1,00,01,10,11.....\}$

The kleene plus is a variation of kleene star operator. The kleene plus is denoted by Σ^+ is defined as follows :

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ....$

which is the set of words of any length except the null string.

Ex : $\Sigma^+ = \{0,1,00,01,10,11....\}$

ii. Language : A language can be defined as a set of strings obtained from Σ^* where Σ is set of alphabets of a particular language. Formally, a language 'L' over Σ is subset of Σ^* which is denoted by $L \subseteq \Sigma^*$

Ex : $L = \{\varepsilon,01,0011,000111,.....\}$

iii. Alphabet :

A language consist of various symbols from which the word s,statements can be obtained. These symbols are called alphabets.

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

$\Sigma = \{a,b,.....,z,A,B,.....,Z,0,1,.....9,.....\}$

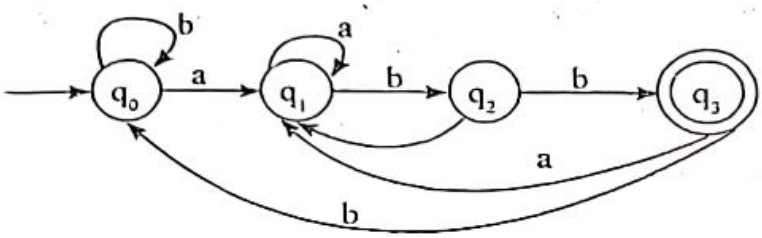
iv. Finite automata :

Finite automata (FA) are computing devices that accept/ recognize regular language. The FA acts as mathematical models and are used to study the abstract machines or abstract computing devices.

$L = \{W \in \Sigma^* \mid W$ is the language accepted by FA$\}$

b. Construct the DFA to accept string of a's and b's ending with the string abb, and also write transition table and transition function. (08 Marks)

Ans.



$\delta(q_0, a) = q_1$
$\delta(q_0, a) = q_0$
$\delta(q_1, a) = q_1$
$\delta(q_1, a) = q_2$
$\delta(q_2, a) = q_1$
$\delta(q_2, a) = q_3$
$\delta(q_3, a) = q_1$
$\delta(q_3, a) = q_0$

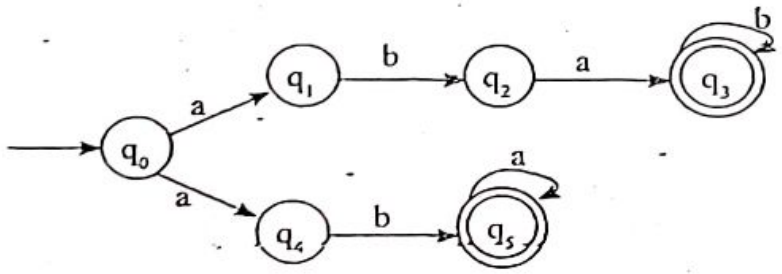
	δ	a	b
→	q_0	q_1	q_0
	q_1	q_1	q_2
	q_2	q_1	q_3
*	q_3	q_1	q_0

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
q_0 is start state
$F = \{q_3\}$
δ is transition function

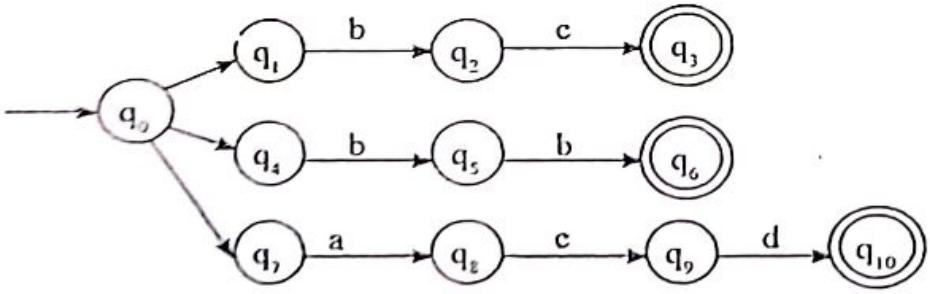
OR

2. a. Obtain the NFA to accept the language
i. $L = \{w \mid w \in abab^n \text{ or } aba^n \text{ where } n \geq 0\}$
ii. $L = \{w \mid w \in 0101 \text{ or } 101 \text{ or } 011\}$ (08 Marks)

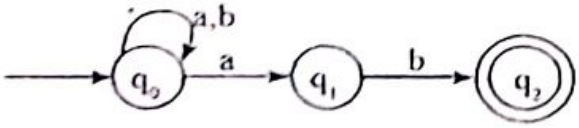
Ans. i.



ii.



b. Convert the following NFA to its equivalent DFA using subset construction method. (08 Marks)



Ans. Step 1 : q_o is the start state

Step 2 : $\Sigma = \{a,b\}$

Step 3 : $Q_N = \{q_o, q_1, q_2\}$

$Q_D = \{\{\}, \{q_o\}, \{q_1\}, \{q_2\}, \{q_o,q_1\}, \{q_o,q_2\}, \{q_1,q_2\}, \{q_o,q_1,q_2\}\}$

Step 4: Final states from the above subset

$F_D = \{\{q_2\}, \{q_o,q_2\}, \{q_1,q_2\}, \{q_o,q_1,q_2\}\}$

Step 5 : Identify transition function

For state ϕ :

Input symbol = a

$\delta_D(\phi,a) = \phi$

Input symbol = b

$\delta_D(\phi,b) = \phi$

For state q_o :

Input symbol = a

$\delta_D(\{q_o\}, a) = \{q_o,q_1\}$

Input symbol = b

$\delta_D(\{q_o\},b\} = \{q_D\}$

For State q_1 :

Input symbol = a

$\delta_D(\{q_1\},a) = \phi$

Input symbol = b

$\delta_D(\{q_1\}, b) = \{q_2\}$

For State q_2 :

Input symbol = a

$\delta_D(\{q_2\},a) = \phi$

Input symbol = b

$\delta_D(\{q_2\},b) = \phi$

For state $\{q_o,q_1\}$:

Input symbol = a

$$\delta_D(\{q_o,q_1\},a) = \delta_N(\{q_o,q_1\},a)$$
$$= \delta_N(\{q_o\},a) \cup \delta_N(\{q_1\},a)$$
$$= \{q_o,q_1\} \cup \phi$$
$$= \{q_o,q_1\}$$

Input symbol = b

$$\delta_D(\{q_o,q_1\},b) = \delta_N(\{q_o,q_1\},b)$$
$$= \delta_N(\{q_o\},b) \cup \delta_N(\{q_1\},b)$$
$$= \{q_o\} \cup \{q_2\}$$
$$= \{q_o,q_2\}$$

For state $\{q_o,q_2\}$:

Input symbol = a

$$\delta_D(\{q_o,q_2\},a) = \delta_N(\{q_o,q_2\},a)$$
$$= \delta_N(\{q_o\},a) \cup \delta_N(\{q_2\},a)$$
$$= \{q_o,q_1\} \cup \phi$$
$$= \{q_o,q_1\}$$

Input symbol = b

$$\delta_D(\{q_o,q_2\},b) = \delta_N(\{q_o,q_2\},b)$$
$$= \delta_N(\{q_o\},b) \cup \delta_N(\{q_2\},b)$$
$$= \{q_o\} \cup \phi$$
$$= \{q_o\}$$

For state $\{q_1,q_2\}$:

Input symbol = a

$$\delta_D(\{q_1,q_2\},a) = \delta_N(\{q_1,q_2\},a)$$
$$= \delta_N(\{q_1\},a) \cup \delta_N(\{q_2\},d)$$
$$= \phi \cup \phi$$
$$= \phi$$

Input symbol = b

$$\delta_D(\{q_1,q_2\},b) = \delta_N(\{q_1,q_2\},b)$$
$$= \delta_N(\{q_1\},b) \cup \delta_N(\{q_2\},b)$$
$$= \{q_2\} \cup \phi$$
$$= \{q_2\}$$

For state $\{q_0,q_1,q_2\}$:

Input symbol = a

$\delta_D(\{q_0,q_1,q_2\},a) = \delta_N(\{q_0,q_1,q_2\},a)$

$= \delta_N(\{q_0\},a) U \delta_N(\{q_1\},b) U \delta_N(\{q_2\},a)$

$= \{q_0,q_1\} U \phi U \phi$

$= \{q_0,q_1\}$

Input symbol = b

$\delta_D(\{q_0\},b) U \delta_N(\{q_1\},b) U \delta_N(\{q_2\},b)$

$= \delta_N(\{q_0,q_1,q_2\},b) = \delta_N(\{q_0,q_1,q_2\},b)$

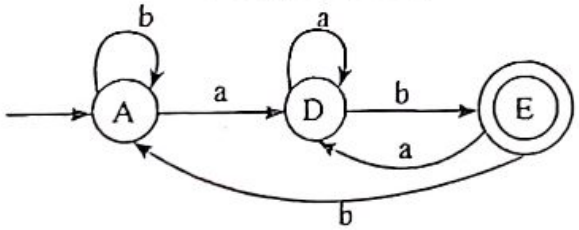
$= \{q_0\} U \phi U \{q_2\}$

$= \{q_0,q_2\}$

δ	a	b
ϕ(	ϕ	ϕ
$\{q_0\}$(A)	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_1\}$(B)	ϕ	$\{q_2\}$
*$\{q_2\}$(C)	ϕ	ϕ
$\{q_0,q_1\}$(D)	$\{q_0,q_1\}$	$\{q_0,q_2\}$
*$\{q_0,q_1\}$(E)	$\{q_0,q_1\}$	$\{q_0\}$
*$\{q_1,q_2\}$(F)	ϕ	$\{q_2\}$
*$\{q_0,q_1,q_2\}$ (G)	$\{q_0q_1\}$	$\{q_0,q_2\}$

Even though we have 8 states observing from the table that 'A' is start state. The states reachable from A are A,D,E rest of the states are not reachable hence can be eliminated.

	δ	a	b
→	A	D	A
	D	D	E
*	E	D	A



Module - 2

3. a. What is regular expression ? Obtain a regular expression to a language consisting of strings of 0's and 1's with almost one point of consecutive 0's. (08 Marks)

Ans. A regular expression is recursively defined as follows :

1. ϕ is a regular expression denoting an empty language.
2. ε is a regular expression indicates the language containing an empty string.

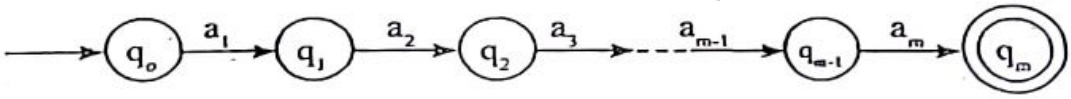
3. a is a regular expression which denotes the language containing only {a}.
4. If R is a regular expression denoting the language f_R and S is a regular expression denoting the language L_s, then
a. R + S is a regular expression corresponding to the language $L_R \cup L_S$.
b. R - S is a regular expression corresponding to the language $L_R . L_S$.
c. R* is a regular expression corresponding to the language L_R.
5. The expression obtained by applying any of the rules from 1 to 4 are regular expressions.
R.E = 1*+(1+01)* 0(1+01)* +(1+01)* 00(1+01)*

b. State and prove pumping lemma for regular languages. (08 Marks)

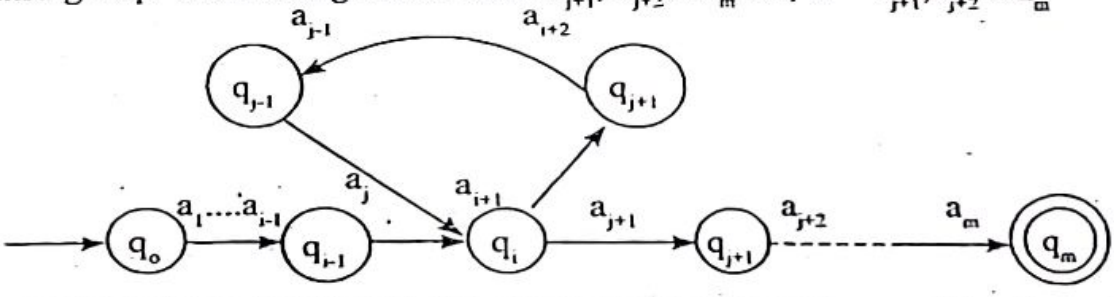
Ans. Let M = (Q,Σ,δ,q_o ,F) be an FA and has 'η' number of states. Let L be the regular language accepted by M. Let frequency string x can be broken into three substrings u,v and w such that

$X = uvw$

Satisfying the following constraints :
$V \neq \varepsilon$ ie., $|V| \geq 1$
$|uv| \leq \eta$
Then $uv^i w$ is in L for $i \geq 0$
Let $X = a_1a_2a_3 \ldots a_m$ where $m \geq n$ and each a_2 is in Σ. Here , n represent the states of DFA. Since we have m input symbols, naturally we should have m+1 states in the sequence $q_o,q_1,q_2 \ldots q_m$, where q_o will be the start state and q_m will be the final state as shown below.



Since $|x| \geq n$, by the pigeon hole principle it is not possible to have distinct transitions. Once of the state can have a loop. Let the string x is divided into three substrings as shown below.
The first group is the string prefix from $a_1a_2 \ldots a_i$ i.e., $u = a_1a_2 \ldots a_i$
The second group is the loop string from $a_{i+1},a_{i+2} \ldots a_{j-1}$ i.e., $V = a_{i+1}, a_{i+2} \ldots a_{j-1}a_j$
The third group is the string suffix from $a_{j+1}, a_{j+2} \ldots a_m$ i.e., $w = a_{j+1},a_{j+2} \ldots a_m$



Observe from above figure that, the prefix string u taxes the machine from q_o to q_i , the loop string v takes the machine from q_i to q_i (Note $q_i = q_j$) and suffix string w toches the machine from q_j to q_f . The minimum string that can be accepted by the above FA is uw with i = 0.

But, when i = 1, the string uvw is accepted by DFA when i = 2, it is accepted. So if i > 0 the machine goes from qo to qj on input string u, circles from qi to qi based on the value of i and then goes to accepting state on input string w. The machine will be in state qi. Since qi and qj are same we can input the string a_{i+1} a_{i+2} ....a_j any number of times and the machine will stay in qj only. Finally, if the input string is a_{j+1} a_{j+2}.... a_m the machine enters into final state q_m.

4. a. **Show that L= {ww^R } w∈(0+1)*} is not regular.** **(08 Marks)**

Ans. Let L is regular and n be the number of states in FA. Consider the string

$$X = \overbrace{1.....1}^{u}\overbrace{0.....0}^{v}\overbrace{0.....0}^{w}\overbrace{1.....1}^{w^R}$$

Where 'n' is the number of the states of FA, w = 1....10....0 and reverse of w is given by w^R = 0....01.....1

Since |x| ≥ n we can split the string x into uvw such that |uv| ≤ n and |V| ≥ 1 as shown below

$$X = 1....\underbrace{1}_{v}\underbrace{0....00.....01.....1}_{w}$$

Where |u| = n-1 and |v| = 1 so that |uv| = |u| + |v| = η -1 + 1 = η which is true according to pumping, lemma, uvw ∈ L for i = 0,1,2.....

If i is 0 i.e., v does not appear and so the number of is on the left of x will be less than the number of is on the right of x and so the string is not of the form ww^R. So uvw L when i = 0. This is a contradiction to the assumption that the language is regular.

So, the language L = {ww^R | w ∈ {0+1}*} is not regular.

b. **What is table filling algorithm? Explain the procedure to minimize DFA.** **(08 Marks)**

Ans. The table filling algorithm is used to find the set of states that are distinguishable and indistinguishable states. The algorithm is recursively defined as shown below

Step 1 : Identify the initial markings : for each pair (p,q) where p ∈ Q and q ∈ Q, if p ∈ F and p ∈ F or vice versa then, the pair (p,q) is indistinguishable and mark the pair (p,q).

Step 2 : Identify the subsequent markings : For each pair (p,q) and for each a ∈ Σ, final δ(p,a) = r and (q,a) = a. If the pair (r,s) is already marked as distinguishable than the pair (p,q) is also distinguishable and mark it as say 'x'. Repeat step 2 until no previously unmarked pairs are marked

Algorithm to minimize the DFA is shown below :

Step1 : Find the distinguishable and indistinguishable pairs : using the table filling algorithm.

Step 2 : Obtain the states of minimized DFA : These groups consist of indistinguishable pairs obtained in previous step and individual distinguishable pairs obtained in previous step and individual distinguishable states. The groups obtained are the states of minimized DFA.

Step 3 : Compute the transition table if [p_1,p_2,—p_k] is a group and if δ[p_1,p_2,—.p_k],

a) = $[r_1, r_2, ...r_m]$ and label the edge with the symbola. Follow this procedure for each group obtained in step 2 and for each at Σ.

Step 4 : Identify the start state : if one of the component in the group $[p_1, p_2 ... p_k]$ consist of a start state of given DFA then $[p_1, p_2, ...p_k]$ is the star state of minimized DFA.

Step 5 : Identify the final state if the group $[p_1, p_2, ...p_k]$ contains a final state of given of A then the group $[p_1, p_2, ...p_k]$ is final state of minimized DFA.

Module - 3

5. a. Explain the classification of grammars with example. **(08 Marks)**

Ans.

Types of grammars
- Type0 grammar (phrase structured grammer)
- Type1 grammar (Context sensitive grammer)
- Type2 grammar (Context free grammer)
- Type3 grammar (Regular grammer)

Type 0 : A grammar G = (V,T,P,S) is said to be type 0 grammer or unrestricted grammer or propose structured grammer if all the production are of the of the form $\alpha - \beta$ where $\alpha \in (VUT)^* \alpha v \delta \ \beta \in (V \cup T)^*$

Ex : $S \rightarrow a\,Ab \mid \varepsilon$
$aA \rightarrow bAA$
$bA \rightarrow a$

Type 1 : A grammer G = (V,T,P,S) is said to be type 1 grammar or context sensitive grammer if all the productions are of the form $\alpha \rightarrow \beta$ as in type grammar. But, there is a restriction on length of β. The length of β must be at least as much as the length of α ie., $|\beta| \geq |\alpha|$ and α and $\beta \in (V \cup T)^+$ i.e., ε can not appear on the left-hand or right hand side of any production.

Ex : $S \rightarrow a\,Ab$
$aA \rightarrow bAA$
$bA \rightarrow aa$

Type 2 : A grammer G = (V,T,P,S) is said to be type 2 grammar or context free grammar if all the productions are of the form $A \rightarrow \alpha$ where $\alpha \in (V \cup T)^*$ and A is non-terminal.

Ex : $S \rightarrow aB \mid bA \mid \varepsilon$
$A \rightarrow aA \mid b$
$B \rightarrow bB \mid a \mid \varepsilon$

Type 3 : The grammer G = (V,T,P,S) is said to be type 3 grammar or regular grammar if the grammar is right linear or left linear. A grammer g is said to be right linear if all production are the form.

$A \rightarrow wB$ or $A \rightarrow w$

If it is left linear, all production of the form

$A \rightarrow Bw$ or $A \rightarrow w$

Ex : $S \rightarrow aaB \mid bbA \mid \varepsilon$

$A \rightarrow aA \mid b$
$B \rightarrow bB \mid a \mid \varepsilon$

b. Obtain a grammar to generate the following language **(08 Marks)**

$L = \{wwR \mid w \in \{a,b\}^*\}$

Ans. The language can be written as

$L = \{aa,bb,abba,baab,....\}$

Observe that the given string is a palindrome of even length. This achieved by deleting the production $S \rightarrow a/b$. So the final grammar is given by

$S \rightarrow \varepsilon$
$S \rightarrow a\ Sa \mid bSb$

$G = (V,T,P,S)$ can be defined as

$V = \{S\}$
$T = \{a,b\}$
$P = \{$
$S \rightarrow asa \mid bsb$
$S \rightarrow \varepsilon$
$\}$

S is start symbol.

OR

6. a. Is the following grammar ambiguous? **(08 Marks)**

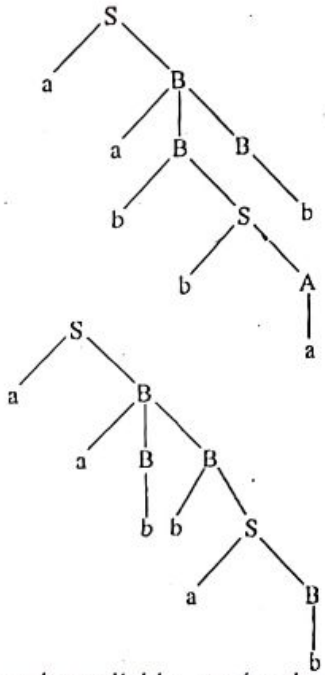
$S \rightarrow aB \mid bA$
$A \rightarrow as \mid bAA|a$
$B \rightarrow bS|aBB|b$

Ans. Left most derivation is

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aabSB$
$\Rightarrow aabbAB$
$\Rightarrow aabbaB$
$\Rightarrow aabbab$

Left most derivation

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aabB$
$\Rightarrow aabbS$
$\Rightarrow aabbaB$
$\Rightarrow aabbab$



For string aabbab, more than one parse tree is available , so the given grammar is ambiguous.

b. **Obtain a PDA to accept the language L(m) = {wCw^R | w∈(a+b)*}.** **(08 Marks)**

Ans. M = (Q,Σ, Γ,δ,Q$_0$,Z$_0$,f]

Where Q = {q$_o$,q$_1$,q$_2$}

Σ = {a,b,c}

Γ = {a,b,Z$_o$}

δ:{

$\delta(q_o,a,z_o)=(q_o,aZ_o)$

$\delta(q_o,b,Z_o)=(q_o,b,Z_o)$

$\delta(q_o,a,a)=(q_o,aa)$

$\delta(q_o,b,a)=(q_o,ba)$

$\delta(q_o,a,b)=(q_o,ab)$

$\delta(q_o,c,Z_o)=(q_1Z_o)$

$\delta(q_o,c,a)=(q_1,a)$

$\delta(q_o,c,a)=(q_1,a)$

$\delta(q_1,a,a)=(q_1,\varepsilon)$

$\delta(q_1,b,b)=(q_1,\varepsilon)$

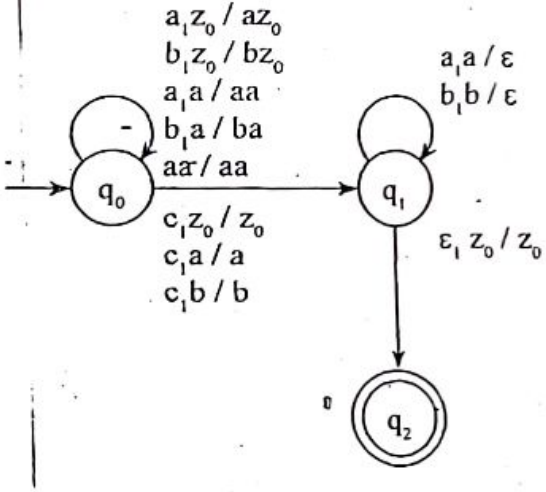
$\delta(q_1,\varepsilon,Z_o)=(q_1Z_o)$

}

$q_0 \in Q$ is start state

$z_0 \in \Gamma$ is the initial stack symbol

F = {q$_2$} is final state



Module - 4

7. a. **What is left recursion? Eliminate a left recursion from the following grammar.**

E → E + T | T

T → T * F|F

F → (E) | id **(08 Marks)**

Ans. A grammer G is said to be left recursive if there is some non terminal A such that $A \overset{+}{\Rightarrow} A\alpha$

The left recursion in a grammer G can be eliminated as shown below

$A \to A\alpha_1 | A\alpha_2 | A\alpha_3 ..... A\alpha_n | \beta_1 | \beta_2 | .... | \beta_m$

Where β_1 do not start with A

$A \to \beta_1 A' | \beta_2 A' | \beta_3 A' | .....\beta_m A |$

$A' \to \alpha_1 A' | \alpha_2 A' | \alpha_3 A' | ..... | \alpha_n A' | \varepsilon$

$A \rightarrow A\alpha_i / \beta_i$	Sbstitution	Without left recursion
$\in \rightarrow \in +T/T$	$A = \in$ $\alpha_1 = +T$ $\beta_1 = T$	$\in \rightarrow TE'$ $E' \rightarrow +T\in' / \varepsilon$
$T \rightarrow T*F/F$	$A = T$ $\alpha_1 = *F$ $\beta_2 = F$	$T \rightarrow FT'$ $T' \rightarrow *FT' / \varepsilon$
$F \rightarrow (\in)/id$	Not applicable	$F \rightarrow (\in) \mid id$

$E \rightarrow TE'$

$E' \rightarrow +T\in' \mid \varepsilon$

$T \rightarrow FT' \mid *FT' \mid \varepsilon$

$F \rightarrow (E) \mid id$

b. **Eliminate all ε - production from the grammer.** **(08 Marks)**

$S \rightarrow ABCa \mid bD$

$A \rightarrow BC \mid b$

$C \rightarrow c/\varepsilon$

$D \rightarrow d$

Ans. **Step 1 : Obtain set of nulable variables from the grammer**

ov	nv	Production
ϕ	B,C	$B \rightarrow \varepsilon$ $C \rightarrow \varepsilon$
B,C	B,C,A	$A \rightarrow BC$
B,C,A	B,C,A	--

V = {B,C,A} are nullable variables

Step 2 : Construction of production p1

Productions	Resulting productions (p)
$S \rightarrow ABCa$	$S \rightarrow ABC_a \mid BCa \mid ACa \mid ABa \mid Ca \mid Aa \mid Ba \mid a$
$S \rightarrow bB$	$S \rightarrow bD$
$A \rightarrow BC \mid b$	$A \rightarrow BC \mid B \mid C \mid b$
$B \rightarrow b \mid \varepsilon$	$B \rightarrow b$
$C \rightarrow C/\varepsilon$	$C \rightarrow C$
$D \rightarrow d$	$D \rightarrow d$

$G^1 = (V^1, T^1, p^1, S)$

$V^1 = \{S, A, B, C, D\}$

$T^1 = \{a, b, c, d\}$

$p^1 = \{$

$S \rightarrow ABCa|BCa|ACa|ABa|Ca|Aa|Ba|a|bD$

$A \rightarrow BC|CC|b$

$B \rightarrow b, C \rightarrow c, D \rightarrow d$

$\}$

S is start symbol.

OR

8. a. Explain two forms of normal forms. **(08 Marks)**

Ans. There are two different types of normal forms.

1. Chomsky normal form (CNF)
2. Greibach normal form (GNF)

1. Chomsky Normal Form (CNF) : Let G = (V,T,P,S) be a CFG. The grammar G is said to be in CNF if all production are of the form.

$A \rightarrow BC$ or $A \rightarrow a$

Where A,B and $C \in V$ and $a \in T$

Note that if a grammar is in CNF, the right hand side of the production should contain two symbols or one symbol. If there are two symbols on the right hand side those two symbols must be non - terminals and if there is only one symbol, that symbol must be a terminal.

2. Greibach normal form (GNF) : In CNF there is restriction on the number of symbols on the right hand side of teh production. Note that in CNF not more than two symbols on RHS of the production are permitted. If there is only symbol that symbol must be a terminal and if there are two symbols, those symbols must be variables.

In GNF there is no restriction the number of symbols on the right hand side but there is restriction on the terminals and variables appear on the right hand side of the production.

G = (V,T,P,S). The 'G; is said to be in GNF if all the productions are of the form $A \rightarrow a\alpha$.

Where $a \in T$ and $\alpha \in V^*$ i.e., the first symbol on the right hand side of the production must be terminal and it can be followed by zero or more variables.

b. Consider the grammar

$S \rightarrow 0A \mid 1B$

$A \rightarrow 0AA \mid 1S \mid 1$

$B \rightarrow 1BB \mid 0S \mid 0$

Obtain the grammar in CNF **(08 Marks)**

Ans.

Given productions	Action	Resulting production
$S \rightarrow 0A \mid 1B$	Replace 0 by B_0 and introduce production $B_0 \rightarrow 1$ Replace 0 by B_1 and introduce the production $B_1 \rightarrow 1$	$S \rightarrow B_0A \mid B_1B$ $B_0 \rightarrow 0$ $B_2 \rightarrow 1$
$A \rightarrow 0AA / 1S$	Replace 0 by B_0 and introduce $B_0 \rightarrow 0$ Replace 1 by B_1 and introduce $B_1 \rightarrow 1$	$A \rightarrow B_0AA \mid B_1S$ $B_0 \rightarrow 0$ $B_1 \rightarrow 1$
$B \rightarrow 1BB/0S$	Replace 0 by B_0 and introduce $B_0 \rightarrow 0$ Replace 1 by B_1 and introduce $B_1 \rightarrow 1$	$B \rightarrow B_1BB / B_0S$ $B_1 \rightarrow 1$ $B_0 \rightarrow 0$

$G=(V_1,T_1,P_1,S)$

$V_1 = \{S, A, B, B_0, B_1\}$

$T_1 = \{0, 1\}$

$P_1 = \{$

$S \rightarrow B_0A \mid B_1B$

$A \rightarrow B_0AA \mid B_1S \mid 1$

$B \rightarrow B_1BB \mid B_0S \mid 0$

$B_0 \rightarrow 0$

$B_1 \rightarrow 1$

$\}$

S is the start symbol

Finally

$G^1 = (V^1, T^1, P^1, S)$ in CNF

$V_1 = \{S, A, B, B_0, B_1, D_1, D_2\}$

$T_1 = \{0, 1\}$

$P_1 = \{$

$S \rightarrow B_0A \mid B_1B$

$A \rightarrow B_1S \mid 1 \mid B_0D_1$

$B \rightarrow B_0S \mid 0 \mid B_1D_2$

$B_0 \rightarrow 0$

$B_1 \rightarrow 1$

$D_1 \rightarrow AA$

$D_2 \rightarrow BB$

$\}$

S is the start symbol

Module - 5

9. a. Obtain a turning machine to accept the language $L = \{0^n 1^n \mid n \geq 1\}$ (08 Marks)

Ans. $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where $Q = \{q_0, q_1, q_2, q_3\}$

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

$\Gamma = \{0, 1, x, y, B\}$

$q_0 \in Q$ is the start state of machine

$B \in \Gamma$ is the blank symbol

$F = \{q_4\}$ is the final state

δ is shown below:

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, y) = (q_1, y, R)$

$\delta(q_1, 1) = (q_2, Y, L)$

$\delta(q_2, Y) = (q_2 Y, L)$

$\delta(q_2, 0) = (q_2, 0, L)$

$\delta(q_2, X) = (q_0, X, R)$

$\delta(q_0, Y) = (q_3, Y, R)$

$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, B) = (q_4, B, R)$

S	Tape symbols (Γ)				
States	0	1	X	Y	B
q_0	(q,X,R)	–		(q_3,Y,R)	
q_1	$(q_1,0,R)$	(q_2,Y,L)		(q_1,Y,R)	
q_2	(q_2,o,L)	–	(q_0,X,R)	(q_2,Y,L)	
q_3				(q_3,Y,L)	(q_4,B,R)
q_4					

b. Explain

I. turing machine as multi-stack machines

II. semi-infinite tape **(08 Marks)**

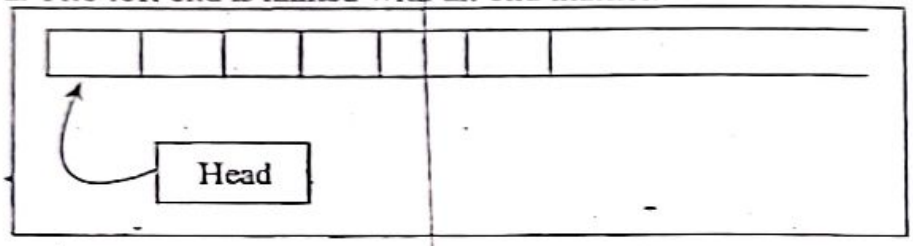
Ans. **I. Turing machine as multi-stack machines:-** Multi-stack/track Turing machines, a specific type of Multi-tape Turing machine, contain multiple tracks but just one tape head reads and writes on all tracks. Here, a single tape head reads n symbols from n tracks at one step. It accepts recursively enumerable languages like a normal single-track single-tape Turing Machine accepts.

A Multi-track Turing machine can be formally described as a 6-tuple $(Q, X, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- X is the tape alphabet
- Σ is the input alphabet
- δ is a relation on states and symbols where
  $\delta(Qi, [a_1, a_2, a_3, ...]) = (Q_j, [b_1, b_2, b_3, ...], \text{Leftshift or Rightshift})$
- q_0 is the initial state
- F is the set of final states

Note - For every single-track Turing Machine S, there is an equivalent multi-track Turing Machine M such that L(S) = L(M).

II. **Semi- Infinite tape:-**A Turing Machine with a semi-infinite tape has a left end but no right end. The left end is limited with an end marker.



It is a two-track tape -

Upper track - It represents the cells to the right of the initial head position.

Lower track - It represents the cells to the left of the initial head position in reverse order.

The infinite length input string is initially written on the tape in contiguous tape cells. The machine starts from the initial state qo and the head scans from the left end marker 'End'.

In each step, it reads the symbol on the tape under its head. It writes a new symbol on that tape cell and then it moves the head either into left or right one tape cell. A transition function determines the actions to be taken.

It has two special states called **accept state** and **reject state.** If at any point of time it enters into the accepted state, the input is accepted and if it enters into the reject state, the input is rejected by the TM. In some cases, it continues to run infinitely without being accepted or rejected for some certain input symbols.

Note - Turing machines with semi-infinite tape are equivalent to standard Turing machines.

OR

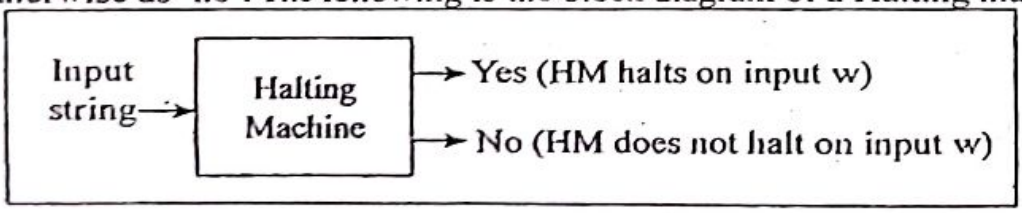
10. a. Explain Halting Problem **(08 marks)**

Ans. Halting Problem:

Input - A Turing machine and an input string w.

Problem - Does the Turing machine finish computing of the string win a finite number of steps? The answer must be either yes or no.

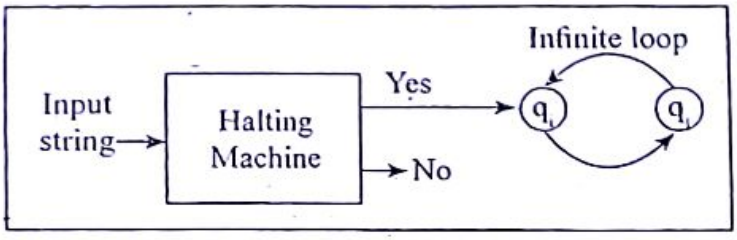
Proof - At first, we will assume that such a Turing machine exists to solve this problem and then we will show it is contradicting itself. We will call this Turing machine as a **Halting machine** that produces a 'yes' or 'no' in a finite amount of time. If the halting machine finishes in a finite amount of time, the output comes as 'yes', otherwise as 'no'. The following is the block diagram of a Halting machine -



Now we will design an **inverted halting machine (HM)'** as -

- If H returns YES, then loop forever.
- If H returns NO, then halt.

The following is the block diagram of an 'Inverted halting machine' -



Further, a machine $(HM)_2$ which input itself is constructed as follows -

- If $(HM)_2$ halts on input, loop forever.
- Else, halt.

Here, we have got a contradiction. Hence, the halting problem is **undecidable.**

b. Explain Linear bounded automata with respect to hiring machines (08 Marks)

Ans. A linear bounded automaton is a multi-track non-deterministic Turing machine with a tape of some bounded finite length.

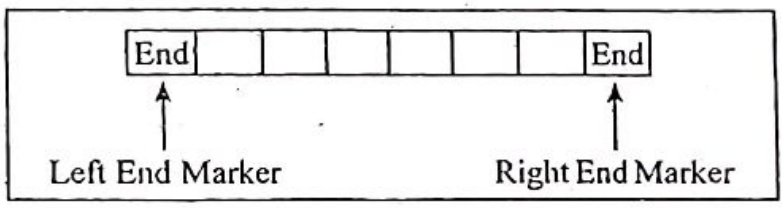
Length = function (Length of the initial input string, constant c) Here,

Memory information ≤ c * Input information

The computation is restricted to the constant bounded area. The input alphabet contains two special symbols which serve as left end markers and right end markers which mean the transitions neither move to the left of the left end marker nor to the right of the right end marker of the tape.

A linear bounded automaton can be defined as an 8-tuple (Q, X, X, Qo, ML, MR, δ, F) where -

- Q is a finite set of states
- X is the tape alphabet
- Σ is the input alphabet
- q_0 is the initial state
- M_L is the left end marker
- M_R is the right end marker where $M_R \wedge M_L$
- δ is a transition function which maps each pair (state, tape symbol) to (state, tape symbol, Constant 'c') where c can be 0 or +1 or -1
- F is the set of final states



A deterministic linear bounded automaton is always **context-sensitive** and the linear bounded automaton with empty language is **undecidable.**