Fifth Semester B.E. Degree Examination

CBCS - Model Question Paper - 2

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

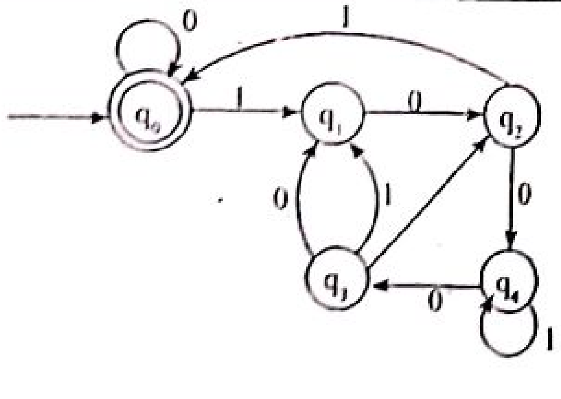
1. a. **Construct a DFA which accepts strings of 0's and 1's where the value of each string is represented as a binary number only the strings representing zero modulo five should be accepted.** (08 Marks)

Ans.

$\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$ with $r = 2$ and $k = 5$

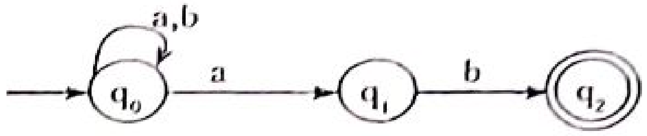
So, $j = (2i + d) \bmod 5$

Remainder	d	$(2 * i + d) \bmod 5 = j$	$\delta(q_i, d) = q_j$
$i = 0$	0	$(2 * 0 + 0) \bmod 5 = 0$	$\delta(q_0, 0) = q_0$
	1	$(2 * 0 + 1) \bmod 5 = 1$	$\delta(q_0, 1) = q_1$
$i = 1$	0	$(2 * 1 + 0) \bmod 5 = 2$	$\delta(q_1, 0) = q_2$
	1	$(2 * 1 + 1) \bmod 5 = 3$	$\delta(q_1, 1) = q_3$
$i = 2$	0	$(2 * 2 + 0) \bmod 5 = 4$	$\delta(q_2, 0) = q_4$
	1	$(2 * 2 + 1) \bmod 5 = 0$	$\delta(q_2, 1) = q_0$
$i = 3$	0	$(2 * 3 + 0) \bmod 5 = 1$	$\delta(q_3, 0) = q_1$
	1	$(2 * 3 + 1) \bmod 5 = 2$	$\delta(q_3, 1) = q_2$
$i = 4$	0	$(2 * 4 + 0) \bmod 5 = 3$	$\delta(q_4, 0) = q_3$
	1	$(2 * 4 + 1) \bmod 5 = 4$	$\delta(q_4, 1) = q_4$



δ	0	1
$\rightarrow *q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

b. Convert the following NFA to its equivalent DFA using lazy evaluation **(08 Marks)**



Ans. **Step 1 :** Identifying the start state of DFA : Since q_0 is the start state of NFA, $[q_0]$ is the state of DFA.

Step 2 : Identify the alphabets of DFA : The input alphabets of NFA are the input alphabets of DFA. So $\Sigma = \{a,b\}$

Step 3 : Identify 2D which are the states of DFA : Start from the start state q_0 and find the transition as shown below :

For state q_0 :-

Input symbol = a

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

Input symbol = b

$\delta_D(\{q_0\},b) = \{q_0\}$

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

Input symbol = a

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

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

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

$= \{q_0,q_1\}$

Input symbol = b

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

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

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

$= \{q_0,q_2\}$

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

Input symbol = a

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

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

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

$= \{q_0,q_1\}$

Input symbol = b

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

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

$= \{q_0\} U \phi$

$= \{q_0\}$

Since, no new state is generated the procedure is terminated.

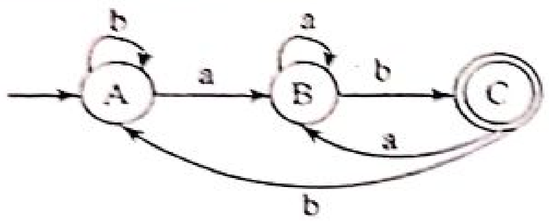
Step 4 : Identify the final states of DFA : Since q_2 is the final state of NFA in the above set, wherever q_2 is present as an element, the corresponding set is the final state of DFA so

$F_D = \{\{q_0, q_2\}\}$

	δ	a	b
→	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
*	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

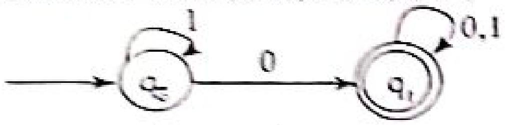
⇒ By renaming

	δ	a	b
→	A	B	A
	B	B	C
*	C	B	A



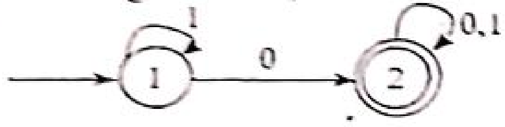
OR

2. a. **Obtain a regular expression for the FA shown below :**



What is the language corresponding to the regular expression. **(08 Marks)**

Ans. Let $q_0 = 1$ and $q_1 = 2$ by renaming the states, the above FA can be written as



Basis :

When k = 0

$R_{11}^{(0)} = \varepsilon + 1$

$R_{12}^{(0)} = 0$

$R_{21}^{(0)} = \phi$

$R_{22}^{(0)} = \varepsilon + 0 + 1$

Induction : The regular expression corresponding to the path from state i to state j through a state which is not higher than k is given by

$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}\left[R_{kk}^{(k-1)}\right]^{*} R_{kj}^{(k-1)}$

When k=1

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)}\left[R_{11}^{(0)}\right]^{*} R_{11}^{(0)}$$

$$= (\varepsilon + 1) + (\varepsilon + 1)(\varepsilon + 1)^{*}(\varepsilon + 1)$$

$$= 1^{*}$$

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}\left[R_{11}^{(0)}\right]^{*} R_{12}^{(0)}$$

$$= 0 + (\varepsilon + 1)(\varepsilon + 1)^{*} 0$$

$$= 0 + 1^{*}0$$

$$= 1^{*}0$$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}\left[R_{11}^{(0)}\right]^{*} R_{11}^{(0)}$$
$$= \phi + \phi(\varepsilon+1)^{*}(\varepsilon+1)^{*}0$$
$$= \phi$$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}\left[R_{11}^{(0)}\right]^{*} R_{12}^{(0)}$$
$$= (\varepsilon+0+1) + \phi(\varepsilon+1)^{*}0$$
$$= (\varepsilon+0+1)$$

When K = 2

$$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}\left[R_{22}^{(1)}\right]^{*} R_{21}^{(1)}$$
$$= 1^{*} + 1^{*}0(\varepsilon+0+1)^{*}\phi$$
$$= 1^{*}$$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}\left[R_{22}^{(1)}\right]^{*} R_{22}^{(1)}$$
$$= 1^{*}0 + 1^{*}0(\varepsilon+0+1)^{*}(\varepsilon+0+1)$$
$$= 1^{*}0 + 1^{*}0(0+1)^{*}(\varepsilon+0+1)$$
$$= 1^{*}0(0+1)^{*}$$

$$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)}\left[R_{22}^{(1)}\right]^{*} R_{21}^{(1)}$$
$$= \phi(\varepsilon+0+1)(\varepsilon+0+1)^{*}\phi$$
$$= \phi$$

$$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)}\left[R_{22}^{(1)}\right]^{*} R_{22}^{(1)}$$
$$= (\varepsilon+0+1) + (\varepsilon+0+1)(\varepsilon+0+1)^{*}(\varepsilon+0+1)$$
$$= (\varepsilon+0+1) + (0+1)^{*}$$
$$= (0+1)^{*}$$

So, $R_{12}^{2} = 1^{*}0(0+1)^{*}$

So, the regular expression for the given DFA is $1^{*}0(0+1)^{*}$ which is the language consisting of any number of is followed by a zero and then followed by string of 0's and 1's.

b. List and explain applications of Regular Expressions. **(08 Marks)**

Ans. 1) Regular expression in unix

2) Pattern Matching

3) Lexical Analysis

4) Unix editor

1) Regular expression in unix : Regular expressions are extensively used in UNIX operating system. But certain short hand notations are used in UNIX platform using which complex regular expressions are avoided. For example, the symbol '.' stands for any character, the sequence [bcde...] stands for regular expression "a + b + c + d...", the operator 1 is used in place of +, the operator? means "Zero or one of "etc most of the commands are in voked invariably uses regular expression. For example grep (Global Regular Expression and Print) used to search for a pattern of string.

2) Pattern Matching : Refers to set of objects with some common properties. We can match an identifier or a decimal number or we can search for a string in the text.

3) Lexical Analysis : Regular expressions are extensively used in the design of lexical analyzer phase. This phase scans the source program and recognizer the tokens which are logically together. The UNIX commands such as lex accepts regular expressions as the input and produces the lexical analyzer generator. This generator takes a high level description of a lexical analyzer as the input and produces lexical analyzer.

4) Unix editor : In UNIX operating system, we can use the editor ed to search for a specific pattern in the text. For example, if the command specified is /abc*c/ then the editor searches for a string which starts with ab followed by zero or more c'x and followed by the symbol C.

Module - 2

3. a. Shew that $L = \{a^ib^j \mid i>j\}$ is not regular. (08 Marks)

Ans. **Step 1 :** Let L is regular and n be the number of states in FA.

Consider the string $x = a^{n+1}b^n$

Step 2 : Since $|x| = 2n + 1 \geq n$, we can split x into uvw such that $|uv| \leq n$ and $|v| \geq 1$ as shown below.

$$X = a^{n+1}b^n = \underset{u}{a^j}\,\underset{v}{a^k}\,\underset{w}{ab^n}$$

$$\text{Where } |u| = j \text{ and } |v| = k \geq 1 \text{ and so that } |uv| = |u| + |v| = j + j \leq n$$

Step 3 : According to pumping lemma, $uv\,w \in L$ for $i \geq 0$

$$\text{i..e, } a^j\left(a^k\right)^i ab^n \text{ÎL for i³0}$$

Now, if we choose i = 0, number of a's in string u will bot be less than the number of b's in w which is contradiction to the assumption that number of a's are more than the number of b's.

So, the language $L = \{a^ib^j \mid i>j|\}$ is not regular

b. Show that if L_1 and L_2 are regular, then $L_1 \cup L_2$, $L_1 \cdot L_2$ and L_1^* are also regular. (08 Marks)

Ans. **Theory :** If L_1 and L_2 are regular, then $L_1 \cup L_2$, $L_1 \cdot L_2$ and L_1^* also denote the regular languages.

Proof : It is given that L_1 and L_2 are regular languages , so there exists regular expression R_1 and R_2 such that

$L_1 = L(R_1)$

$L_2 = L(R_2)$

By the definition of regular expressions, we have

1. $R_1 + R_2$ is a regular expression denoting the language $L_1 \cup L_2$.
2. $R_1 . R_2$ is a regular expression denoting the language $L_1 . L_2$
3. $R_1{}^*$ is a regular expression denoting the language $L_1{}^*$

So the regular languages are closed under union, concatenation and star operations.

OR

4. a. Obtain the minimized DFA for the following. **(10 Marks)**

δ	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Ans. Vertical and horizontal marking

B							
C							
* D	X	X	X				
E				X			
F				X			
G				X			
H				X			
	A	B	C	D	E	F	G
				*			

δ	a	b
(A,B)	(A,B)	(A,C)
(A,C)	(B,D)	(A,B)
(A,E)	(B,D)	(A,F)
(A,F)	(B,G)	(A,E)
(A,G)	(B,F)	(A,G)
(A,H)	(B,G)	(A,D)
(B,C)	(A,D)	(C,B)
(B,E)	(A,D)	(C;F)
(B,F)	(A,G)	(C,E)
(B,G)	(A,F)	(C,G)
(B,H)	(A,G)	(C,D)
(C,E)	(D,D)	(B,F)
(C,F)	(D,G)	(B,E)
(C,G)	(D,F)	(B,G)
(C,H)	(D,G)	(B,D)
(E,F)	(D,G)	(F,E)
(E,G)	(D,F)	(F,G)
(E,H)	(D,G)	(F,D)
(F,G)	(G,F)	(E,G)
(F,H)	(G,G)	(E,D)
(G,H)	(F,G)	(D,G)

$\Longrightarrow$

	B							
	C	X	X					
*	D	X	X	X				
	E	X	X		X			
	F			X	X	X		
	G			X	X	X		
	H	X	X	X	X	X	X	X
	-	A	B	C	D	E	F	G
					*			

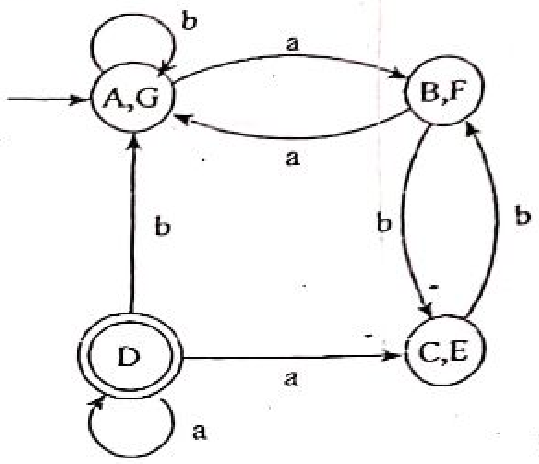
δ	a	b
(A,B)	(B,A)	(A,C)
(A,F)	(B,G)	(A,E)
(A,G)	(B,F)	(A,G)
(B,F)	(A,G)	(C,E)
(B,G)	(A,F)	(C,G)
(C,E)	(D,D)	(B,F)
(F,G)	(G,F)	(E,G)

$\Longrightarrow$

	B	*						
	C	X	X					
*	D	X	X	X				
	E	X	X		X			
	F	X		X	X	X		
	G		X	X	X	X		
	H	X	X	X	X	X	X	X
		A	B	C	D	E	F	G
					*			

(A,G)(B,F) & (C,E) are indistinguishable
(D,H) are distinguishable

δ	a	b
→(A,G)	(B,F)	(A,G)
(B,F)	(A,G)	(C,E)
(C,E)	D	(B,F)
*D	D	(A,G)
H	(A,G)	D


$$\begin{tikzcd}
\arrow[r] & (A,G) \arrow[loop above, "b"] \arrow[rr, bend left, "a"] & & (B,F) \arrow[ll, bend left, "a"] \arrow[d, bend right, "b"'] \\
& D \arrow[u, "b"] \arrow[rr, "a"] \arrow[loop below, "a"] & & (C,E) \arrow[u, bend right, "b"']
\end{tikzcd}$$

b. What are the limitations of finite automation. **(06 Marks)**

Ans. 1. An FA has finite number of states and so it does not have the capacity to remember arbitrary long amount of information.

2. Since it does not have memory, FA cannot remember a long string. For example, to check for matching parenthesis, check whether the string is a palindrone or not etc, are not possible using FA.

3. Finite autoyata or finite state machine have troube recognizing various types of languages involving counting, calculating, storing the string.

Module - 3

5. a. Obtain a grammer to generate the language **(10 Marks)**

i. $L=\{q^n b^{n-3} | \eta \geq 3\}$

ii. $L=L_1L_2$ where $L_1=\{a^n b^m | n \geq 0, m > n\}$ $L_2=\{0^n, {}^{2n} | n \geq 0\}$

Ans. **i. $L=\{a^n b^{n-3} \{ n \geq 3\}$**

It is clear from the above statement that the set of strings that can be generated by this language can be represented as

L = {aaa, aaaab, aaaaabb, ...}

$S \rightarrow aaaA$

$A \rightarrow aAb \mid \varepsilon$

So the final grammar that can be generated is

V = {S,A}

T = {a,b}

P = {

$S \rightarrow aaA$

$A \rightarrow aAb|\varepsilon$

}

S - is start symbol

ii) $L = L_1 L_2$

$L_1 = \{a^n b^m \mid n \geq 0, m > n\}$

$L_2 = \{0^n 1^{2n} \mid n \geq 0\}$

$L_1 = S_1 \rightarrow aS_1b \mid bB$

$B \rightarrow bB \mid b$

$L_2 = S_2 \rightarrow aS_2 11 \mid \varepsilon$

$S \rightarrow S_1 S_2$

$V = \{S, S_1, S_2, B\}$

$T = \{a, b, 0, 1\}$

$P = \{$

$S \rightarrow S_1 S_2$

$S_1 \rightarrow aS_1 b \mid dB$

$S \rightarrow S_2 11 \mid \varepsilon$

$B \rightarrow bB \mid b$

$\}$

S_1 is start symbol.

b. Obtain the string ibtib taxa from the grammar shown below and verify whether the grammar is ambiguous or not (06 Marks)

$S \rightarrow iCts \mid ictSeS|a$

$C \rightarrow b$

Ans. $S \Rightarrow iCtS \Rightarrow ibtS$

$\Rightarrow ibtiCtSeS$

$\Rightarrow ibtibtSeS$

$\Rightarrow ibtibtaeS$

$\Rightarrow ibtibtaca$

$\Rightarrow iCtSeS$

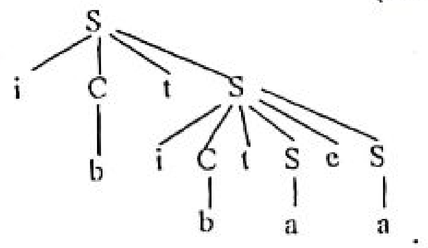
$\Rightarrow ibtSeS$

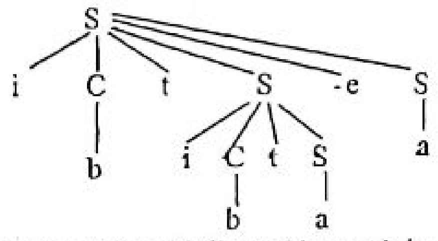
$\Rightarrow ibtiCtSeS.$

$\Rightarrow ibtibtSeS$

$\Rightarrow ibtibtacS$

$\Rightarrow ibtibtaea$





Since there are two different parse trees for the string 'i btibtaca' by applying atmost derivaton the given grammer is ambiguous.

6. a. For the grammer

$S \rightarrow aABC$

$A \rightarrow aB/a$

$B \rightarrow bA / b$

$C \rightarrow a$

Obtain the corresponding PDA (08 Marks)

Ans. **Step 1 :** Push the start symbol S on to the stack and change the state to q_1, $\delta(q_0, \varepsilon, Z_0) = (q_1, SZ_0)$

Step 2 :

Production	Transition
$S \rightarrow aABC$	$\delta(q_1,a,S) = (q_1, ABC)$
$A \rightarrow aB$	$\delta(q_1,a,A) = (q_1,B)$
$A \rightarrow a$	$\delta(q_1,a,A) = (q_1,\varepsilon)$
$B \rightarrow bA$	$\delta(q_1,b,B) = (q_1,A)$
$B \rightarrow b$	$\delta(q_1,b,B) = (q_1,\varepsilon)$
$C \rightarrow a$	$\delta(q_1,a,C) = (q_1, \varepsilon)$

Step 3 : Finally in state q_1, without consuming any input change the state to q_f which is an accepting state i.e., $\delta(q_1, \varepsilon, Z_o) = (q_f, Z_o)$

$Q = \{q_o, q_1, q_f\}$

$\Sigma = \{a, b\}$

$\Gamma = \{S, A, B, C, Z_o\}$

δ:isshown below

$\delta(q_o, \varepsilon, Z_o) = (q_1, SZ_o)$

$\delta(q_1, a, S) = (q_1, ABC)$

$\delta(q_1, a, A) = (q_1, B)$

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

$\delta(q_1, b, B) = (q_1, A)$

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

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

$\partial(q_1, \varepsilon, Z_o) = (q_f, Z_o)$

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

$Z_o \in \Gamma$ is the initial symbol on the stock

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

b. Obtain a CFG that generates the language accepted by PDA (08 Marks)

$M = (\{q_o, q_1\}, \{a, b\}, \{A, z\}, \delta, q_o, Z, \{q_f\})$ with the transitions

$\delta(q_o, a, z) = (q_o, AZ)$

$\delta(q_o, b, A) = (q_o, AA)$

$\delta(q_o, a, A) = (q_f, \varepsilon)$

Ans. $\delta(q_o, a, A) = (q_1, \varepsilon)$

For δ of the form $\delta(q_i, a, z) = (q_j, \varepsilon)$	Resulting production
$\delta(q_o, a, A) = (q_1, \varepsilon)$	$(q_o A q_1) = a$

The transitions

$\delta(q_0, a, z) = (q_0, Az)$

$\delta(q_0, b, A) = (q_0, AA)$

For δ of the form $\delta(q_i, a, z) = (q_i, AB)$	Resulting production
$\delta(q_0, a, z) = (q_0, AZ)$	$(q_0Zq_0) \rightarrow a(q_0Aq_0)(q_0Zq_0) \mid a(q_0Aq_1)(q_1Zq_0)(q_0Zq_1)$ $\rightarrow a(q_0Aq_0)(q_0Zq_1) \mid a(q_0Aq_1)(q_1Zq_1)$
$\delta(q_0, b, A) = (q_0, AA)$	$(q_0Zq_0) \rightarrow b(q_0Aq_0)(q_0Aq_0) \mid b(q_0Aq_1)(q_1Aq_0)$ $(q_0Zq_1) \rightarrow b(q_0Aq_0)(q_0Aq_1) \mid b(q_0Aq_1)(q_1Aq_1)$

Module - 4

7. a. Eliminate the useless symbols in the grammar.

$S \rightarrow aA \mid bB$

$A \rightarrow aA \mid a$

$B \rightarrow bB$

$D \rightarrow ab \mid Ea$

$E \rightarrow aC \mid d$ **(08 Marks)**

Ans.

Old variable	New variable	Productions
ϕ	A,D,E	$A \rightarrow a$ $D \rightarrow ab$ $E \rightarrow o\mid$
A,D,E	A,D,E,S	$S \rightarrow aA$ $A \rightarrow aA$ $D \rightarrow Ea$
A,D,E,S	A,D,E,S	

The resulting grammar $G_1 = (V_1, T_1, P_1, S)$ where

$V_1 = \{A, D, E, S\}$ S is start symbol

$T_1 = \{a, b, d\}$

$P_1 = \{$

$A \rightarrow a \mid aA$

$D \rightarrow ab \mid Ea$

$\in \rightarrow d$

$S \rightarrow aA$

$\}$

P^1	T^1	V^1
	–	S

$S \rightarrow aA$	a	S,A
$A \rightarrow a \mid aA$	a	S,A

$G' = (V', T', P', S)$

$V' = \{S, A\}$

$T' = \{a\}$

$P' = \{$

$S \rightarrow aA$

$A \rightarrow a \mid aA$

$\}$

Sisstartsymbol

b. Eliminate all unit productions from the grammar.

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow C \mid b$

$C \rightarrow D$

$D \rightarrow E \mid bC$

$E \rightarrow d \mid Ab$

Ans. The non unit productions of the grammar G are shown below **(08 Marks)**

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

$D \rightarrow bC$

$E \rightarrow d \mid Ab$

Unit productions of the grammar G are

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow E$

$$\begin{tikzcd} B \arrow[r] & C \arrow[r] & D \arrow[r] & E \end{tikzcd}$$

It is clear from above graph D => E, so all non unit production generated from E can also be generated from D.

The non unit production from E are

$E \rightarrow d \mid Ab$

Also be obtained from D

$D \rightarrow d \mid Ab$

The resulting produciton are

$D \rightarrow bC$

$D \rightarrow d \mid Ab$

Similarly

$C \rightarrow d|Ab$

$C \rightarrow bC$

$C \rightarrow d|Ab$

$B \rightarrow b$

$B \rightarrow d|Ab$

$B \rightarrow bC$

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

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

$p^1 = \{$

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b \mid d \mid Ab \mid bC$

$C \rightarrow bC \mid d \mid Ab$

$D \rightarrow bC \mid d|Ab$

$E \rightarrow d \mid Ab$

$\}$

S is the start symbol

OR

8. a. Convert the following grammar G to GNF **(08 Marks)**

$G = \{(A_1, A_2, A_3), (a, b), P, A_1\}$ where, p consists of the following

Productions :

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow A_1A_2 \mid a$

Ans. We observe $A_1 \rightarrow A_2A_3$ is the only production with A_1 on the hand side. Let us substitute A_2A_3 for A_1 in production $A_3 \rightarrow A_1 A_2$

The resulting set of production is :

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow A_2A_3A_2 \mid a$

Similarly A_2 in the production rule for A_3 with A_3A_1 and b

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow A_3A_1A_3A_2 \mid bA_3A_2 \mid a$

$A_3 \rightarrow bA_3A_2B_3$

$A_3 \rightarrow aB_3$

$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$

Resulting

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a$

$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$

Now all the production for A_3 are all in GNF

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b$

$A_3 \rightarrow bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a$

$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$.

An equivalent grammar in GNF thus can be written as

$A_1 \rightarrow bA_3A_2B_3A_1A_3 \mid aB_3A_1A_3 \mid bA_3A_2A_1A_3 \mid A_1A_3 \mid bA_3$

$A_2 \rightarrow bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b$

$A_3 \rightarrow bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a$

$B_1 \rightarrow bA_3A_2B_3A_1A_3A_3A_2 \mid bA_3A_2B_3A_1A_3A_3A_2B_3$

$aB_3A_1A_3A_3A_2 \mid aB_3A_1A_3A_3A_2B_3 \mid bA_3A_2A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2B_3 \mid b$

$aA_1A_3A_3A_2 \mid aA_1A_3A_3A_2B_3 \mid bA_3A_3A_2 \mid bA_3A_3A_2B_3$

b. **Show that L= {w|w∈{a,b,c}}* where $n_a(w) = n_b(w) = n_c(w)$} is not context free.** **(08 Marks)**

Ans. The language $L_1 = \{a^n b^n c^n \mid n \geq 0\}$ is obtained by the intersection Land the regular language represented by the regular a*b*c* i.e.,

$$\{a^nb^nc^n \mid n \geq 0\} = \{a^*b^*c^* \cap \{w \mid w \in \{a,b,c\}\}^* \text{where} \eta_a(w) = \eta_b(w) = \eta_c(w)\}$$

We know that intersection of context free language and regular language is also a context free. But its already known that

$L_1 = \{a^nb^nc^n \mid n \geq 0\}$ is not context free. Since L_1 is not context free, it implies that the given language

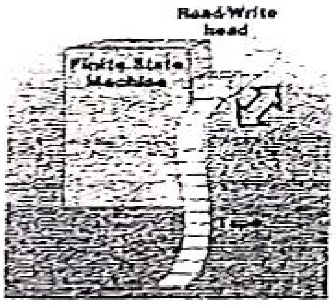
$L = \{w \mid w \in \{a,b,c\}^* \text{where} \eta_a(w) = \eta_b(w) = \eta_c(w)\}$ is not context free and not context free grammer.

Module - 5

9. a. **Explain the concept of turing machine in detail.** **(08 Marks)**

Ans. A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of the Turing machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left. If the TM reaches the final state, the input string is accepted, otherwise rejected. A TM can be formally described as a 7-tuple

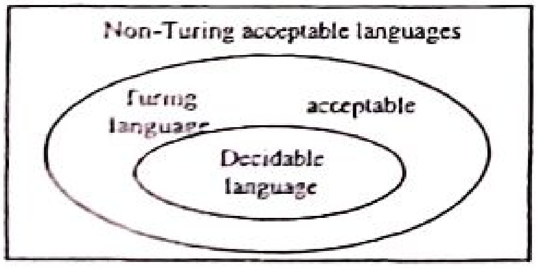
(Q, X, Σ, δ, qo, B, F) where -
Q is a finite set of states
X is the tape alphabet
Σ is the input alphabet
δ is a transition function; $\delta: Q \times X \rightarrow Q \times X \times$ {Left_shift, Right_shift}.
q_0 is the initial state
B is the blank symbol
F is the set of final states
A Turing machine is a finite state machine that has an unlimited supply of paper tape that it can write on and read back. There are many formulations of a Turing machine, but essentially the machine reads a symbol from the tape, which is used as an input to the finite state machine. This takes the input symbol and according to it and the current state does three things:
1. It prints something on the tape
2. Moves the tape right or left by one cell
3. Changes to a new state



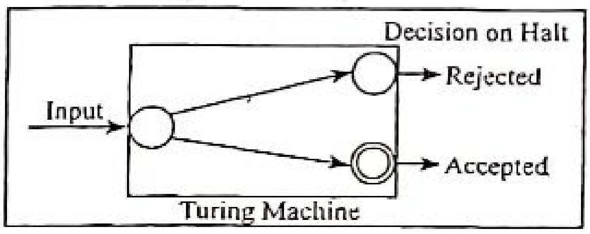
A Turing machine can also perform a special action - it can stop or halt - and surprisingly it is this behavior that attracts a great deal of attention.
For example, a Turing machine is said to recognize a sequence of symbols written on the tape if it is started on the tape and halts in a special state called a final state. What is interesting about this idea is that there are sequences that a Turing machine can recognize that a finite state machine can't.

b. **Explain in detail language decidability in the context of theory of computation.** **(08 Marks)**

Ans. A language is called **Decidable** or **Recursive** if there is a Turing machine which accepts and halts on every input string w. Every decidable language is Turing-Acceptable.



A decision problem P is decidable if the language L of all yes instances to P is decidable. For a decidable language, for each input string, the TM halts either at the accept or the reject state as depicted in the following diagram -



Example 1

Find out whether the following problem is decidable or not -

Is a number 'm' prime?

Solution

Prime numbers = {2,3, 5, 7, 11, 13, }

Divide the number 'm' by all the numbers between '2' and 'Vm' starting from '2'.

If any of these numbers produce a remainder zero, then it goes to the "Rejected state", otherwise

it goes to the "Accepted state". So, here the answer could be made by 'Yes' or 'No'. Hence, it is a decidable problem.

10. a. State and prove the Rice theorem **(08 Marks)**

Ans. Theorem

Rice's theorem: Any nontrivial property about the language recognized by a Turing machine is undecidable.

A property about Turing machines can be represented as the language of all Turing machines, encoded as strings, that satisfy that property. The property P is about the language recognized by Turing machines if whenever L(M)=L(N) then P contains (the encoding of) M iff it contains (the encoding of) N. The property is non-trivial if there is at least one Turing machine that has the property, and at least one that hasn't.

Proof: Without limitation of generality we may assume that a Turing machine that recognizes the empty language does not have the property P. For if it does, just take the complement of P. The undecidability of that complement would immediately imply the undecidability of P.

In order to arrive at a contradiction, suppose P is decidable, i.e. there is a halting Turning machine B that recognizes the descriptions of Turing machines that satisfy P. Using B we can construct a Turning machine A that accepts the language {(M,w)| M is the description of a Turing machine that accepts the string w}. As the latter problem is undecidable this will show that B cannot exists and P must be undecidable as well.

Let MP be a Turing machine that satisfies P (as P is non-trivial there must be one). Now A operates as follows:

On input (M,w), create a (description of a) Turing machine C(M,w) as follows:

On input x, let the Turing machine M run on the string w until it accepts (so if it doesn't accept C(M,w) will run forever).

Next run MP on x. Accept iff MP does.

Note that C(M,w) accepts the same language as MP if M accepts w; C(M,w) accepts the empty language if M does not accept w.

Thus if M accepts w the Turing machine C(M,w) has the property P, and otherwise it doesn't. Feed the description of C(M,w) to B. If B accepts, accept the input (M,w); if B rejects, reject.

b. Define

i) Post Correspondence problem & ii) Quantum computation (08 Marks)

Ans. i) Post Correspondence problem

The Post Correspondence Problem (PCP), introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet £ is stated as follows -

- Given the following two lists, M and N of non-empty strings over £ ~

$M = (x_1, x_2, x_3, \quad , x_n)$

$N = (y_1, y_2, y_3, \quad , y_n)$

We can say that there is a Post Correspondence Solution, if for some $i_1, i_2 \ldots\ldots i_k$ ik, where $1 < i_j < n$, the condition $X_{i1} \ldots\ldots X_{ik} = Y_{i1} \ldots\ldots Y_{ik}$.

Find whether the lists

M = (abb, aa, aaa) and N = (bba, aaa, aa) have a Post Correspondence Solution?

	X_1	X_2	X_3
M	Abb	aa	aaa
N	Bba	aaa	aa

Here, $X_2X_1X_3$ = 'aaabbaaa' and $y_2y_1y_3$ = 'aaabbaaa'

We can see that $X_2X_1X_3 = y_2y_1y_3$ Hence, the solution is i = 2, j = 1, and k = 3

ii) Quantum computation

Quantum computing studies theoretical computation systems (quantum computers) that make direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data. Quantum computers are different from binary digital electronic computers based on transistors. Whereas common digital computing requires that the data be encoded into binary digits (bits), each of which is always in one of two definite states (0 or 1), quantum computation uses quantum bits, which can be in superpositions of states. A quantum Turing machine is a theoretical model of such a computer, and is also known as the universal quantum computer. The field of quantum computing was initiated by the work of Paul Benioff and Yuri Man in in 1980, Richard Feynman in 1982, and David Deutsch in 1985. A quantum computer with spins as quantum bits was also formulated for use as a quantum spacetime in 1968.

As of 2017, the development of actual quantum computers is still in its infancy, but experiments have been carried out in which quantum computational operations were executed on a very small number of quantum bits. Both practical and theoretical research continues, and many national governments and military agencies are funding quantum computing research in an effort to develop quantum computers for civilian, business, trade, environmental and national security purposes, such as cryptanalysis.