Date 21st Sept

Day: Sunday

Page No. 3
DATE / /

FINITE STATE MACHINE

DEFINITION :- FSM consists of finite set of states (S), which alter on receiving the i/p set (I) to produce the o/p set (O).

It consists of two functions :-

a.) State function (STF) :- $S \times I \rightarrow S$ (which is the next state)

b.) Machine function (MAF) :- $S \times I \rightarrow O$ (what is the o/p)

Q.1 Design an FSM to check whether the given decimal number is divisible by 3.

Sol :-

STEP 1 :- Theory :- Def.

STEP 2 :- Logic :-

1.) The i/p is ~~the~~ a decimal no. hence we define

$I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

2.) Machine checks for divisibility of number. hence we define

$O = \{Y, N\}$

3.) We define one state as start state i.e q_s. (start state also called initial state)

4.) Whenever a number is divided by 3 there are 3 remainders possible i.e 0, 1, 2 and accordingly we define the state q_1, q_0, q_2

Hence $S = \{q_s, q_0, q_1, q_2\}$

STEP 3:-

Implementation

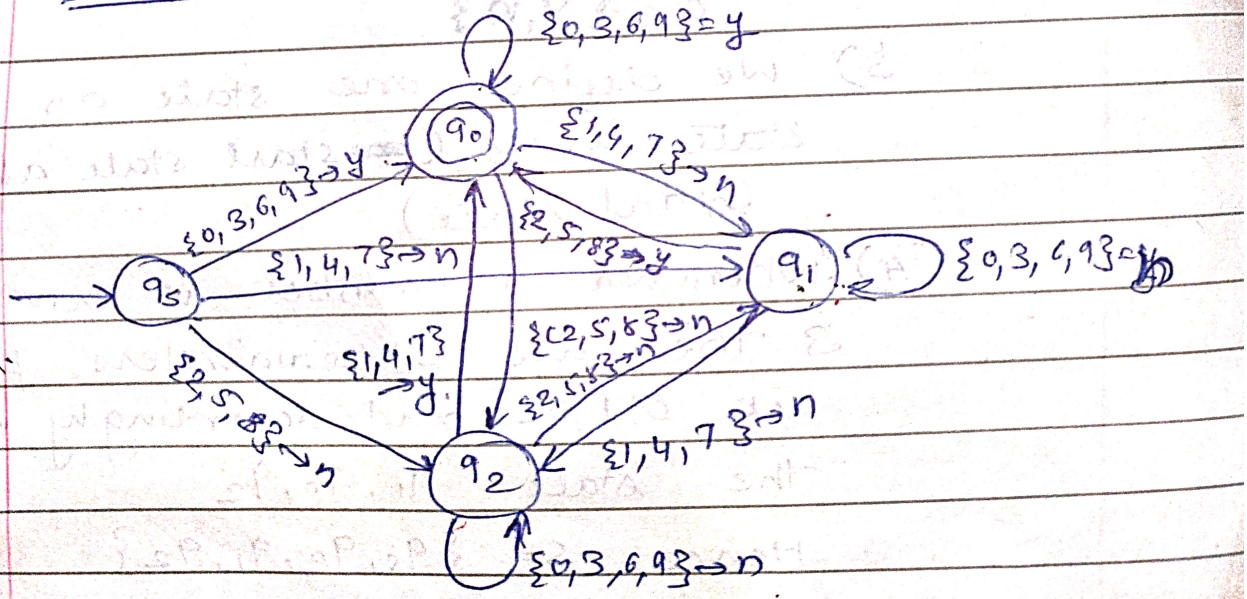
S \ I	{0,3,6,9}	{1,4,7}	{2,5,8}
→ q_s	q_0	q_1	q_2
q_0*	q_0	q_1	q_2
q_1	q_1	q_2	q_0
q_2	q_2	q_0	q_1

State function:- STF :- S x I → S

S \ I	{0,3,6,9}	{1,4,7}	{2,5,8}
→ q_s	y	n	n
q_0*	y	n	n
q_1	n	n	y
q_2	n	y	n

Machine function:- MAF :- S x I → O

State Transition Diagram



STEP 4 :- Example

(i) $(q_s, 3874)$
$\vdash (q_0, 874)$
$\vdash (q_2, 74)$
$\vdash (q_0, 4)$
$\vdash (q_1, \cdot) \to n$

(ii) $(q_s, 312)$
$\vdash (q_0, 12)$ $(q_0, 12)$
$\vdash (q_1, 2)$ $(q_1, 2)$
$\vdash (q_0) \to y$ $(q_0) \to y$

Q.2 Design an FSM to check whether the given decimal number is divisible by 4.
[may 06 / SM
may 07 / SM]

Sol

S \ I	{0,4,8}	{1,5,9}	{2,6}	{3,7}
→ q_s	q_0	q_1	q_2	q_3
q_0^*	q_0	q_1	q_2	q_3
q_1	q_2	q_3	q_0	q_1
q_2	q_0	q_1	q_2	q_3
q_3	q_2	q_3	q_0	q_1

STF :- $S \times I \to S$

S \ I	{0,4,8}	{1,5,9}	{2,6}	{3,7}
→ q_s	y	n	n	n
q_0^*	y	n	n	n
q_1	n	n	y	n
q_2	y	n	n	n
q_3	n	n	y	n

MAF :- $S \times I \to O$

Example

(i) $(q_s, 2984)$
$\vdash (q_2, 984)$
$\vdash (q_1, 84)$
$\vdash (q_2, 4)$
$\vdash (q_0) \rightarrow$ y.

Q.3 Design an FSM to check whether the given binary number is divisible by 4. [Dec 05/5M]

Sol. $I = \{0, 1\}$

$O = \{y, n\}$.

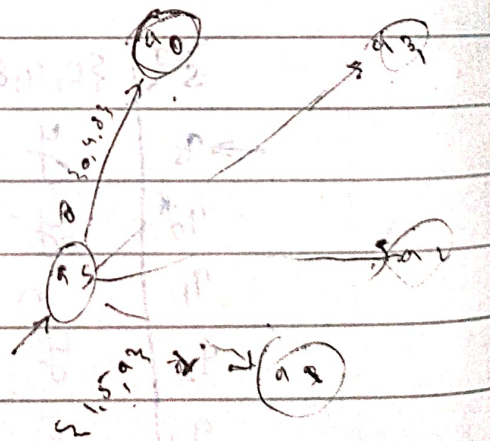
$S = \{q_s, q_0, q_1, q_2, q_3\}$.

S / I	0	1		
→ q_s	q_0	q_1	0/4	1/4
q_0*	q_0	q_1	1000/4	1001/4
q_1	q_2	q_3	1010/4	1011/4
q_2	q_0	q_1	1100/4	1101/4
q_3	q_2	q_3	1110/4	1111/4.

STF :- $S \times I \rightarrow S$.

S / I	0	1
start → q_s	y	n
q_0*	y	n
q_1	n	n
q_2	y	n
q_3	n	n

MAF :- $S \times I \rightarrow O$



discrepancy

Page No. 7
DATE / /

Example:- $(q_s, 1011101)$

$\vdash (q_1, 011101)$
$\vdash (q_2, 11101)$
$\vdash (q_1, 1101)$
$\vdash (q_3, 101)$
$\vdash (q_3, 01)$
$\vdash (q_2, 1)$
$\vdash (q_1) \rightarrow n.$

Q.4 Design an FSM in which the i/p is valid if it contains odd number of b's over (alphabet) $\Sigma = \{a, b\}$.

$n_b(x)$	S/I	a	b
$\rightarrow$	q_s	q_0	q_1
even	q_0^*	q_0	q_1
odd	q_1^*	q_1	q_0

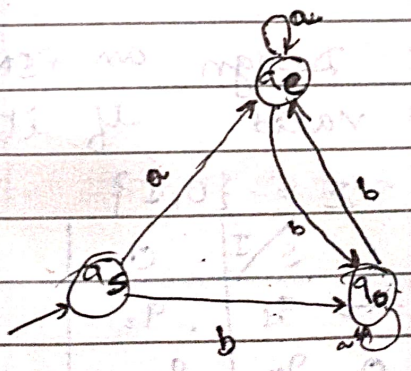
STF : $S \times I \rightarrow S$.

S/I	a	b
$\rightarrow q_s$	n	y
q_0	n	y
q_1^*	y	n

MAF : $S \times I \rightarrow O$

Example:- $(q_s, baabb)$

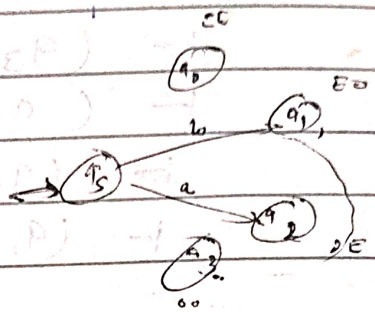
$\vdash (q_1, aabb)$
$\vdash (q_1, abb)$
$\vdash (q_0, bb)$
$\vdash (q_1, b) \rightarrow y.$



Page No.
DATE / /

Q.5 Design an FSM in which the i/p is valid if it contains even number of a's & and odd number of b's over

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



	S / I	a	b
$n_a(x) n_b(x)$ →	q_5	q_2	q_1
E E	q_0	q_2	q_1
E O	q_1^*	q_3	q_0
O E	q_2	q_0	q_3
O O	q_3	q_1	q_2

STF: $S \times I \to S$.

	S / I	a	b
→	q_5	n	y
	q_0	n	y
	q_1^*	n	n
	q_2	n	n
	q_3	y	n

NAF

$S \times I \to O$

Q.6 Design an FSM in which the i/p is valid if it ends in "100" over

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

	S / I	0	1
ends in →	q_5	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_3	q_1
100	q_3^*	q_0	q_1

~~0~~ ~~01~~

~~10~~ ~~11~~

~~100~~ ~~101~~

STF: $S \times I \to S$

S / I	0	1
start → q_s	n	n
q_0	n	n
q_1	n	n
q_2	y	n
q_3^*	n	n

MAF : S X I → O

Example (q_s, 10100)

⊢ (q_1, 0100)
⊢ (q_2, 100)
⊢ (q_1, 00)
⊢ (q_2, 0)
⊢ (q_3) → y

Q.7 Design an FSM in which the i/p is valid if it ends in "bab" over Σ = {a, b}

ends in	S / I	a	b
start →	q_s	q_0	q_1
a	q_0	q_0	q_1
b	q_1	q_2	q_1
ba	q_2	q_0	q_3
bab	q_3^*	q_2	q_1

STF : S X I → S.

S / I	a	b
q_s	n	n
q_0	n	n
q_1	n	n
q_2	n	y
q_3	n	n

MAF : S X I → O

Q.8 Design an FSM in which the i/p is valid if it contains substring "101" over $\Sigma = \{0,1\}$.

	S \ I	0	1	
contains →	q_s	q_0	q_1	similar to ends in.
0	q_0	q_0	q_1	
1	q_1	q_2	q_1	
10	q_2	q_0	q_3	
101	q_3^*	q_3	q_3	contains. [trap state]

STF : S X I → S

	S \ I	0	1
contains →	q_s	n	n
	q_0	n	n
	q_1	n	n
	q_2	n	y
	q_3^*	y	y

MAF : S X I → O

Q.9 Design an FSM in which the i/p is valid if it contains "aab" over $\Sigma = \{a, b\}$

	S \ I	a	b	
contains. Start →	q_s	q_0	q_1	
a	q_0	q_2	q_1	
b	q_1	q_0	q_1	
aa	q_2	q_2	q_3	
aab	q_3^*	q_3	q_3	(trap state)

STF : S X I → S.

S \ I	a	b.
contains → q_s	n	n
q_0	n	n
q_1	n	n
q_2	n	y
q_3	y	y

MAF : S X I → O'

Q.10 Design an FSM in which the i/p is valid if it does not contain any occurence of "bbb" over $\Sigma = \{a, b\}$.

	S \ I	a	b
does not contain (Start) →	q_s*	q_0	q_1
a	q_0*	q_0	q_1
b	q_1*	q_0	q_2
bb	q_2*	q_0.	q_3
bbb	q_3	q_3	q_3

STF : S X I → S

S \ I	a	b.	
→ q_s*	y	y	
q_0*	y	y	
q_1*	y	y	
q_2*	y	n	
q_3	n	n.	dead state

MAF : S X I → O.

NOTE :-

1.) There is alway one start state.
2.) There can be more than one final state.
3.) start state can also be a final state (in such a case blank input is also valid.)
4.) Dead state is a trap state which is non final.
5.) Dead state is always a trap state but a trap state may or may not be a dead state.
6.) The terms can be changed interchangably.

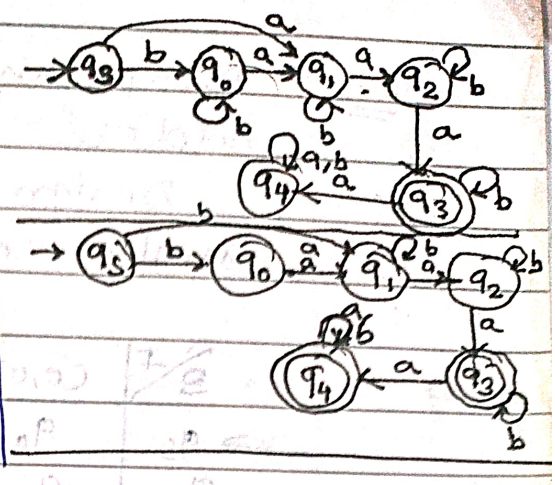
Start state	⟷	Initial state
final state	⟷	Accepting state.
Non-final state	⟷	Rejecting state.

Q.11 Design FSM in which ip is valid, if it contains.

(1.) exactly 3 a's.
(2.) atleast 3 a's
(3) atmost 3 a's.

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

S \ I	a	b
n a(x) → q_5	q_1	q_0
0 q_0	q_1	q_0
1 q_1	q_2	q_1
2 q_2	q_3	q_2
3 q_3	q_4	q_3
more than 3. q_4	q_4	q_4



(i) final states is q_3

(ii) final states are q_3 & q_4

(iii) final states are q_0, q_1, q_2 & q_3.

Q.12 Design an FSM to o/p. the remainder when decimal number is divided by 3.

$\Sigma = \{0,1,2,3 \cdots, 9\}$

$O = \{0,1,2\}$

S \ I	{0,3,6,9}	{1,4,7}	{2,5,8}
→ q_5	q_0	q_1	q_2
q_0	q_0	q_1	q_2
q_1	q_1	q_2	q_0
q_2	q_2	q_0	q_1

STF: S X I → S

*There is no final state

S \ I	{0,3,6,9}	{1,4,7}	{2,5,8}
→ q_5	0	1	2
q_0	0	1	2
q_1	1	2	0
q_2	2	0	1

MAF :- S X I → O

Q 13. Design an FSM to implement "Binary Adder". OR

Design an FSM to add two binary numbers of same length.

S \ I	(0,0)	(0,1)	(1,0)	(1,1)
→ q_s	q_0	q_0	q_0	q_1
NC q_0	q_0	q_0	q_0	q_1
C q_1	q_0	q_1	q_1	q_1

STF: S X I → S.

S \ I	(0,0)	(0,1)	(1,0)	(1,1)
→ q_s	0	1	1	0
NC q_0	0	1	1	0
C q_1	1	0	0	1

MAF: S X I → O.

BASIC CONCEPTS :-

1.] **Alphabet (Σ) :-** It is defined as th finite set of input symbols.

eg. $\Sigma = \{0, 1\}$ $\Sigma = \{a, b, c\}$

2.] **String / Sentence / Word :-** It is defined as finite seq. of symbols over the given Σ.

eg. 0, 1, 01, 00, 101, ...

3.] **String Length :-** It is defined as number of symbols present in given string.

eg: $x = 1101$

$|x| = 4$.

NOTE:- The string of length zero would be denoted by ϵ (Epsilon) (λ)

4.] **Language :-** It is defined as set of strings over the given alphabet (Σ).

eg:

$$L = \{ x \mid x \text{ ends in } 01 \text{ over } \Sigma = \{0, 1\} \}$$

$L = \{01, 001, 101, 0001 \ldots\}$.

5.] **Operations on Languages :-**

(i) **Union of two Languages :-** (no combination)

Let L_1 and L_2 be two languages.

$$L_1 \cup L_2 = \{x, y \mid x \in L_1 \text{ \& } y \in L_2\}$$

Page No.
DATE / /

(ii) Concatenation (combination) of two languages :-

Let L_1 and L_2 be two languages.

$$L_1 \cdot L_2 = L_1 L_2 = \{ xy \mid x \in L_1 \text{ & } y \in L_2 \}$$

Let $L_1 = \{101, 110, 10\}$

$L_2 = \{01, 10\}$

(1) $L_1 \cup L_2 = \{101, 110, 10, 01\}$

(2) $L_1 \cdot L_2 = \{10101, 11001, 1001, 10110, 11010, 1010\}$

(iii) Closure of a language :- [zero / more repeatition]

Let L be a language

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

(iv) Positive closure of a language :- [one / more repeatition]

Let L be a language

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

eg. $L = \{0,1\} = L^1$

$L^2 = L \cdot L = \{1010\}$

$L^3 = L \cdot L \cdot L = \{101010\}$

$L^0 = \{\epsilon\}$

$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$

$= \{\epsilon, 10, 1010 \dots\}$

$L^+ = L^1 \cup L^2 \cup L^3 \cup \dots$

$= \{10, 1010, 101010 \dots\}$

(v) Intersection of two lang.

(vi) Difference of two languages

(vii) complement of two languages.

REGULAR EXPRESSION (R.E.) :-

Definition:- Regular expression is used for specifying Regular language and is defined as follows:

(i) 'ϕ' is a R.E for $\{\}$.

(ii) 'ϵ' is a R.E for $\{\epsilon\}$.

(iii) 'a' is a R.E. for $\{a\}$.

(4) Let R & S be two R.E. for specifying the languages L_R & L_S respectively.

(a.) $(R)|(S)$ is a R.E for $L_R \cup L_S$

(b.) $(R)(S)$ is a R.E for $L_R \cdot L_S$.

(c.) $(R)^*$ is a R.E for L_R^*

Example :-

Regular Expression	Regular Language
[r]	[L(R)]
a	{a}
b	{b}
a+b or a|b	{a} ∪ {b} = {a, b} union
ab or a.b	{a} . {b} = {ab} concat
a^*	{ε, a, aa, aaa, ...} closure
a^+	{a, aa, aaa, ...} +ive closure
$(ab)^*$	{ε, ab, abab, ...}
$(a+b)^*$	{ε, a, b, aa, ab, ba, bb, ...}
$(00)^*$	{ε, 00, 0000, ...}
$0(00)^*$	{0, 000, 00000, ...}
$(00)^* \cdot 0$	{0, 000, 00000, ...}
$a \cdot a^*$ $= a^+$	{a, aa, aaa, ...} a, aa, aaa

NOTE :-

* $a \cdot \epsilon = \epsilon \cdot a = a$
* $\emptyset \cdot \epsilon = \epsilon \cdot \emptyset = \emptyset$
* $\emptyset \cdot \epsilon = \epsilon \cdot \emptyset = \emptyset$
* $a^* = a \cdot a^*$
* $L^+ = L \cdot L^*$

$L^* = L^+ \cup \{\epsilon\}$

Q. Write R.E for specifying the following

(1) set of all strings that begin with 'a' over $\Sigma = \{a, b\}$.

Sol $r = a \cdot (a+b)^*$

$L(r) = \{a, aa, ab, aaa, aab, aba, abb\}$

(2) set of all strings that end with either 'b' or 'aa' over $\Sigma = \{a, b\}$

Sol :- $r = (a+b)^* b + (a+b)^* aa$.

$L(r) = \{b, ab, bb, aa, aaa, baa \ldots\}$

$r = (a+b)^* (b+aa)$

(3) set of all strings that begin with 'x' & end with 'xy' over $\Sigma = \{x, y\}$

$r = (x \cdot (x+y)^*) \cdot ((x+y)^* \cdot xy)$

$L(r) = x \cdot (x+y)^* \cdot xy$

$L(r) = \{xy, xxy, xxxy, xyxy \ldots\}$

Take care of combining pattern or overlapping.

(4.) Set of all strings that start and end with a different letter over $\{x, y\}$

Sol.

$$r = x(x+y)^*y + y(x+y)^*x$$

$$L(r) = \{xy, xxy, xyy, yx, yxx, yyx\}$$

(5.) set of all strings that contain atleast one occurance of "00" over $\Sigma = \{0, 1\}$.

Sol $r = (0+1)^* 00 (0+1)^*$

OR

$r = (0+1)^* (00)^+ (0+1)^*$

(6.) set of all strings that contain atleast two 0's over $\Sigma = \{0, 1\}$

Sol :- $r = (0+1)^* 0 (0+1)^* 0 (0+1)^*$

$L(r) = \{00, 000, 1001 \ldots\}$

(7.) set of all strings that contain atleast 'x' and atleast one 'y' over $\Sigma = \{x, y\}$.

Sol $r = (x+y)^* x (x+y)^* y (x+y)^* + (x+y)^* y (x+y)^* x (x+y)^*$

Page No.
DATE / /

8.) Set of all strings which contain atleast one 'x' atleast one 'y', atleast one 'z'. over $\Sigma = \{x, y, z\}$.

Sol $r = (x+y+z)^* x (x+y+z)^* y (x+y+z)^* z (x+y+z)^*$
$+ (x+y+z)^* x (x+y+z)^* z (x+y+z)^* y (x+y+z)^*$
$+ (x+y+z)^* y (x+y+z)^* x (x+y+z)^* z (x+y+z)^*$
$+ (x+y+z)^* y (x+y+z)^* z (x+y+z)^* x (x+y+z)^*$
$+ (x+y+z)^* z (x+y+z)^* x (x+y+z)^* y (x+y+z)^*$
$+ (x+y+z)^* z (x+y+z)^* y (x+y+z)^* x (x+y+z)^*$

(9.) $*L = \{a^n \mid n \geq 2\}$

Sol $r = aaa^*$
or $r = a \cdot a a^*$
or $r = a^2 \cdot a^*$

(10.) $*L = \{a^n \mid n \leq 2\}$

Sol $r = \{aa + a + \epsilon\}$

Same

(11.) $*L = \{a^m b^n \mid m \geq 4, n \leq 3\}$

Sol $r = aaaaa^* (bbb + bb + b + \epsilon)$

~~aaaab, aaaabb,~~

Same

(12.) $*L = \{a^m b^n \mid (m+n) \text{ is even}\}$

Sol :- $r = (aa)^* (bb)^* + a(aa)^* b(bb)^*$.

June 06 (13)

* $L = \{ w \mid w \in \{a,b\}^* , |w| \bmod 3 = 0 \}$.

Sol $r = ((a+b) \cdot (a+b) \cdot (a+b))^*$

$r = \{ aaa + aab + aba + abb + baa + bab + bba + bbb \}^*$

Dec 04 (14)

$* \; L = \{ w \mid w \in \{a,b\}^* \cdot n_a(w) \bmod 3 = 0 \}$

$r = (b^* a b^* a b^* a b^*)^* + b^*$

Q. any pair of adjacent zero's appears before any pair of adjacent 1's

May 2006

$(1+\epsilon) \cdot (0+01) \cdot (0+\epsilon) \; (1+10)^*$

0011

Q. $n = (a^n b^m \mid a \geq 4, m \leq 3)$

re $a^4 \cdot a^* (\epsilon + b + bb + bbb)$

Q. $(a^n b^m \mid n+m$ is even$)$

Q. re $(aa)^* (bb)^* + a(aa)^* b (bb)^*$

FINITE AUTOMATA (F.A) :-

Finite automata is considered to be a mathematical model of a system.

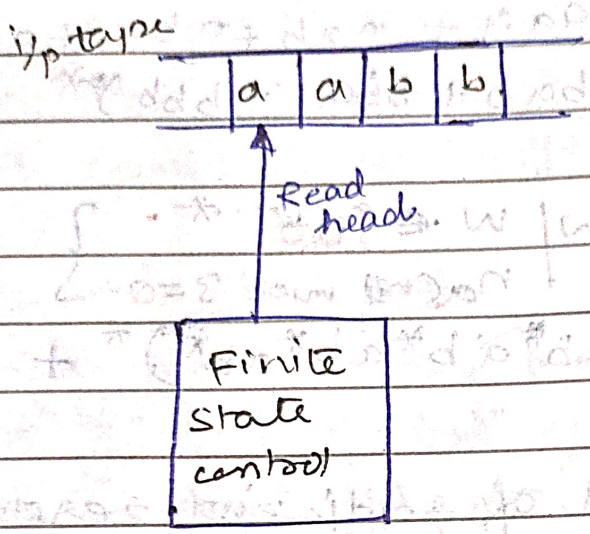


Fig :- Model of FA

Components of F.A :-

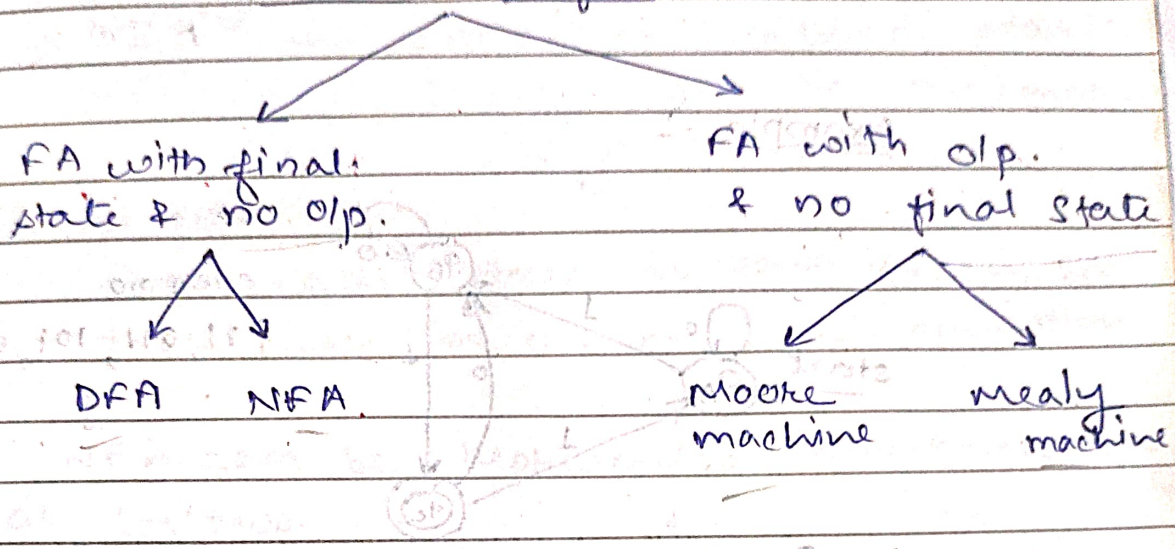
F.A consists of finite set of states, i/p tape & read head.

Working of F.A :-

Depending on the state & the i/p symbol

* FA can change the state/ remain in the same state.
* F.A moves the head to the Right of current cell.

Variations of FA



Deterministic finite ~~Ma~~ Automata (DFA):-

Definition:- DFA consists of finite set of states, one state is called start state & there can be one/more final states.

In DFA,

from each state on each i/p symbol there is exactly one transition.

DFA can be represented mathematically as follows.

$$M = (Q, \Sigma, \delta, q_0, F)$$

where

Q = finite set of states.

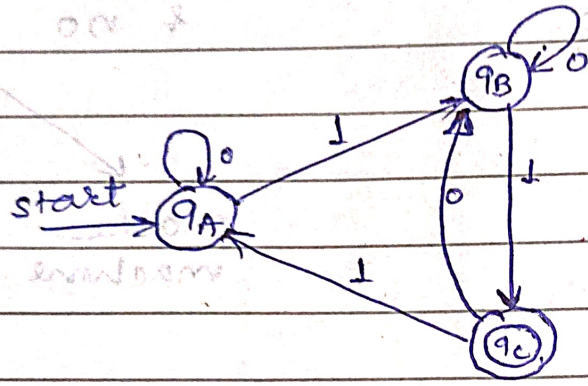
Σ = Input alphabet

δ = transition function.

$\delta :- Q \times \Sigma \to Q$.

q_0 = start state i.e $q_0 \in Q$
f = finite set of final states
$f \subseteq Q$.

Example :-



even no.
{11, 011, 101, 0011, 1001, ... }.

$Q = \{q_A, q_B, q_C\}$
$\Sigma = \{0, 1\}$.
$q_0 = q_A$
$f = \{q_C\}$.

δ =

Q \ Σ	0	1
→ q_A	q_A	q_B
q_B	q_B	q_C
q_C*	q_B	q_A

eg. $(q_A, 101)$
$\vdash (q_B, 01)$
$\vdash (q_B, 1)$
$\vdash (q_C, \epsilon)$
Accept

$(q_A, 110)$
$\vdash (q_B, 10)$
$\vdash (q_C, 0)$
$\vdash (q_B, \epsilon)$
Rejected.

Non-Deterministic F.A (NFA):-

Definition:- NFA consists of finite set of ~~sets~~ states, one state is called start state and there can be one/more final states.

In NFA,

from each state on each i/p symbol there can be 0, 1 or more transition.

NFA can be represented mathematically as follows.

$M = \{Q, \Sigma, \delta, q_0, F\}$

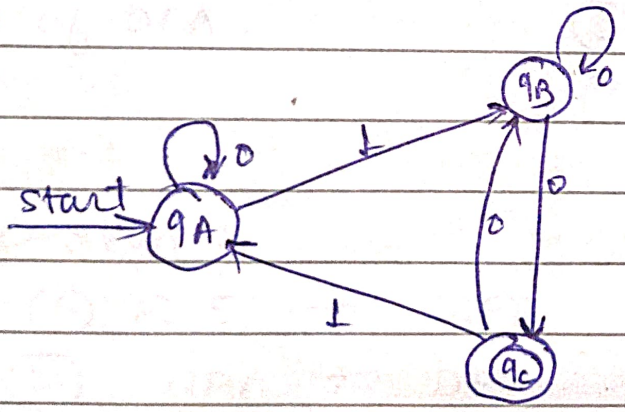
where Q = finite set of states.

Σ = i/p alphabet

δ = transition function $\delta :- Q \times \Sigma \to 2^Q$

q_0 = start state $q_0 \in Q$

F = finite set of final states $F \subseteq Q$.



NFA

$Q = \{q_A, q_B, q_C\}$ $q_0 = q_A$

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

DATE / /

$\delta =$

Q \ Σ	ε	0	1
q_A		q_A	q_B
q_B		$\{q_B, q_C\}$	{ }
q_C		q_B	q_A

eg. (i) $(q_A, 100)$

$\vdash (q_B, 00)$

$\vdash (q_B, 0)$ $\vdash (q_C, 0)$

$\vdash (q_B, \epsilon)$

$\vdash (q_B, \epsilon)$ $\vdash (q_C, \epsilon)$

Accept

(ii) $(q_A, 111)$

$\vdash (q_B, 11)$

Reject

Differentiate b/w DFA & NFA

DFA	NFA
(1.) In DFA from each state on each input symbol there is exactly one transition.	(1.) In NFA, from each state on each i/p symbol there can be zero, 1 or more transitions
(2.) In DFA, transition function defined as. $\delta :- Q \times \Sigma = Q$.	(2.) In NFA, transition function defined as. $\delta :- Q \times \Sigma = 2^Q$.
(3.) Implementation of DFA with the help of computer program is simple.	(3.) Implementation of NFA with the help of computer program is difficult.
(4.) Every DFA is always an NFA.	(4.) An NFA may or may not be a DFA.
(5.) e.g. of DFA	(5.) e.g of NFA.

Designs:-

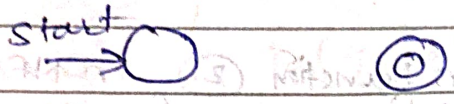
1. R.E to NFA
2. NFA to DFA
3. DFA to minimised DFA.

Page No.
DATE / /

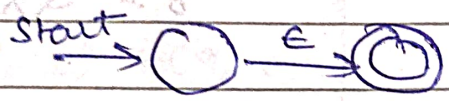
1:) R.E to NFA:-

* Divide the given R.E into smaller subexpressions & construct NFA for each using Rule 1, 2 & 3.

(a) Rule 1:- NFA for r='ϕ' empty set



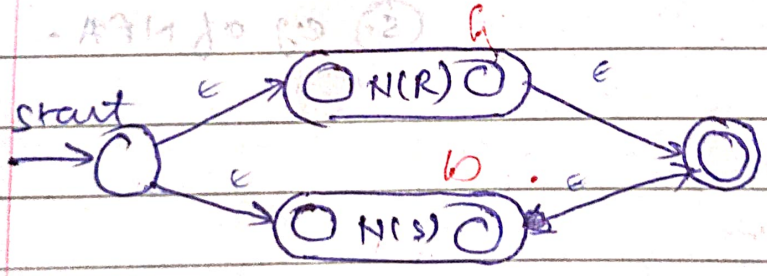
(b) Rule 2:- NFA for r='ϵ' empty string.



(c) Rule 3:- NFA for 'r='a'

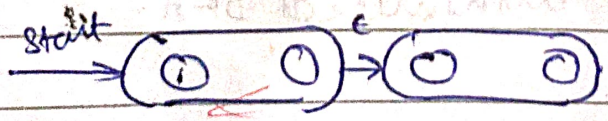
(d) Rule 4:-

(i) NFA for . r = (R)/(S) (R)/(S)

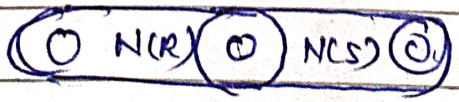


a b

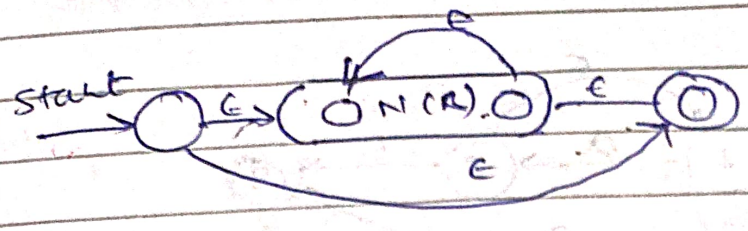
(ii) NFA for . r = (R) . (S)



or

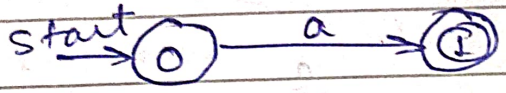


(iii) NFA for $r = (R)^*$



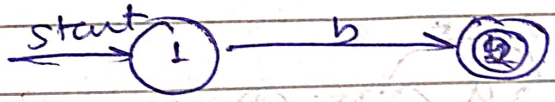
Q. Construct NFA for following:-

(i) $r = a$.



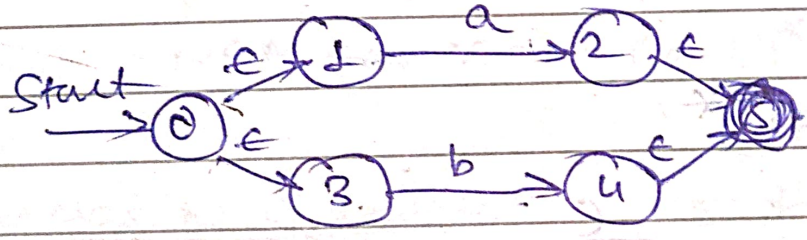
NFA for $r = a$.

(ii) $r = b$

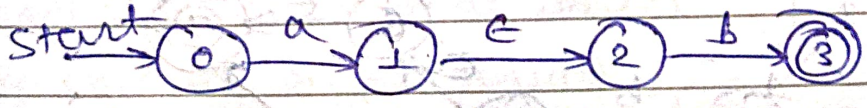


NFA for $r = b$

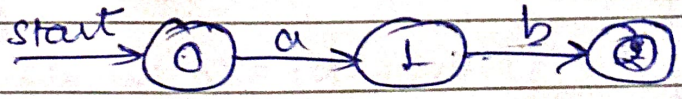
(iii) $r = a + b$



(iv) $r = a \cdot b$.

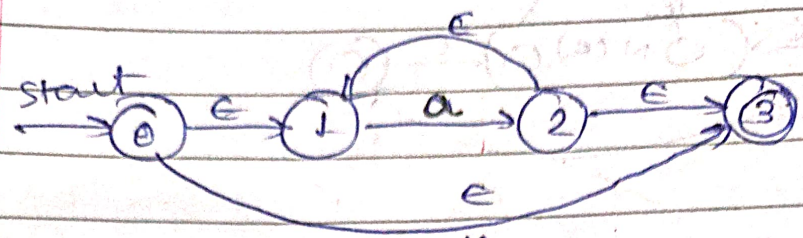


or



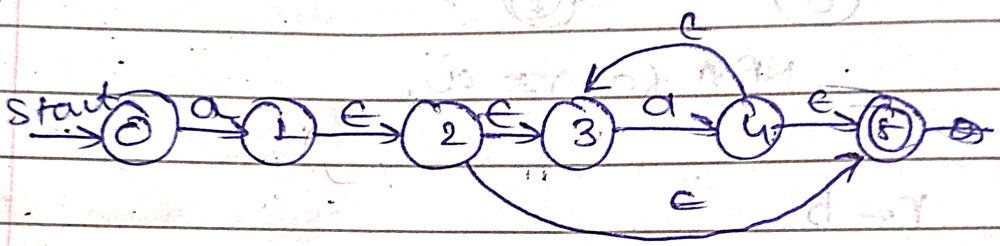
NFA for $r = a \cdot b$

(5.) $r = a^*$

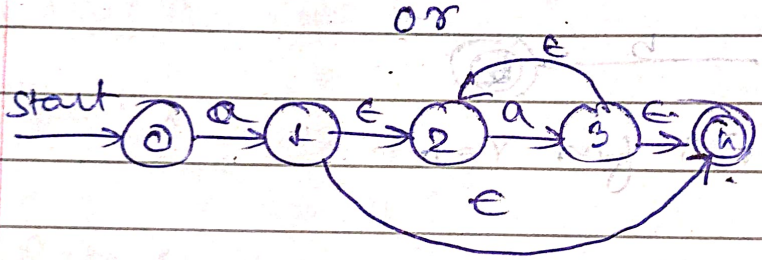


NFA for $r = a^*$

(6.) $r = a \cdot a^*$

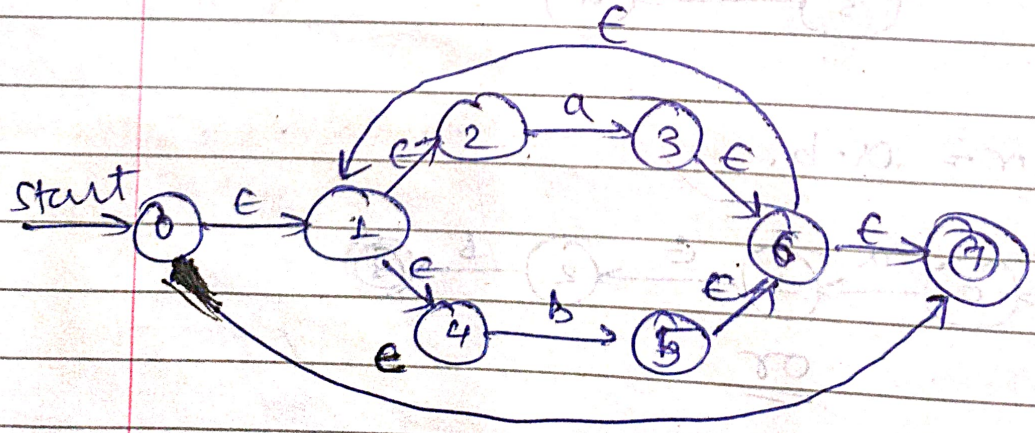


or

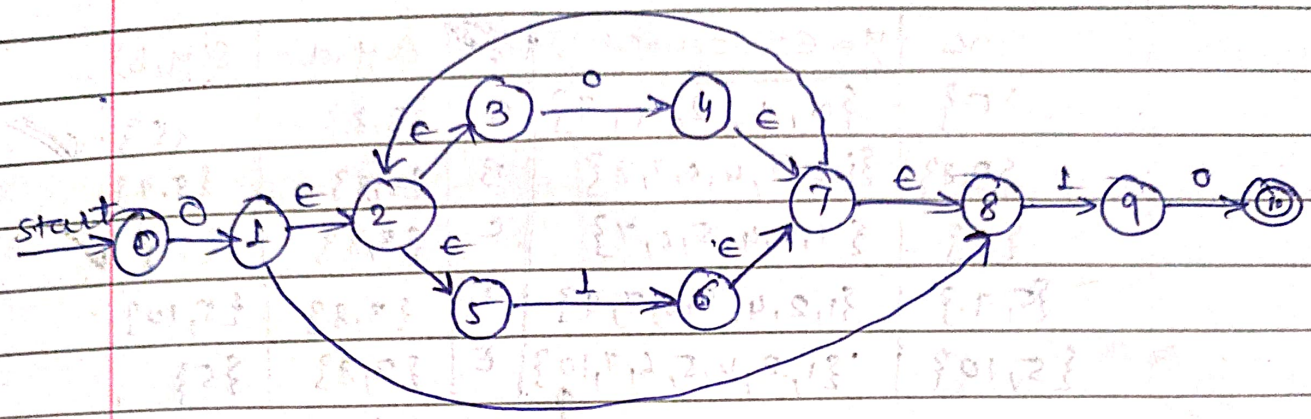


NFA for $r = a+$

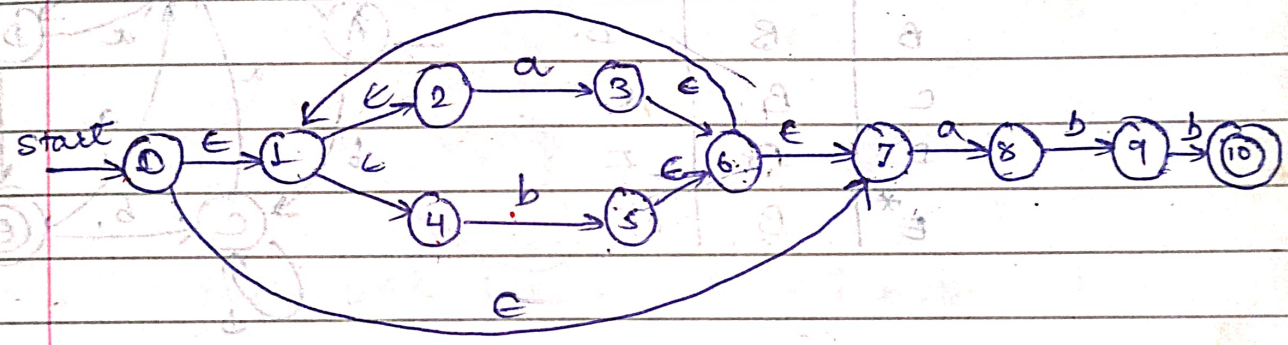
(7.) $r = (a+b)^*$



(8) $r = 0(0+1)^* 10$.



NFA for $r = 0(0+1)^* 10$



NFA for $r = (a+b)^* abb$.

ε-closure of a state

It is defined as the set of states that are reachable from that state by walking on ε-transitions (including the state).

eg.

ε-closure(0) = {0, 1, 2, 4, 7}

ε closure(5) = {1, 2, 4, 5, 6, 7}.

ε-closure of set of states

It is defined as the union of ε-closure of each state of the set

eg ε-closure({3, 8})

= ε-closure(3) ∪ ε-closure(8)

= {1, 2, 3, 4, 6, 7} ∪ {8}

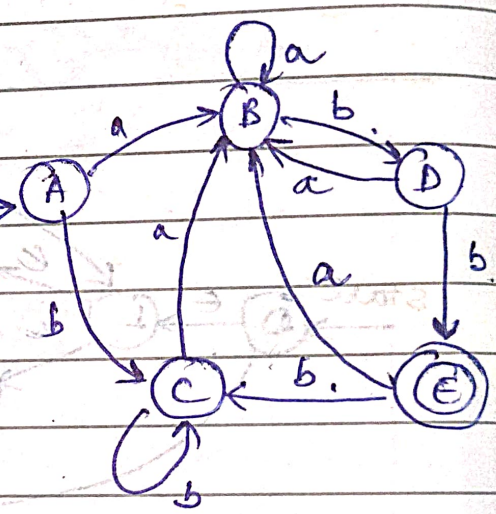
= {1, 2, 3, 4, 6, 7, 8}

2.) NFA TO DFA :-

x	y = ε-closure(x)	new states	δ(y, a)	δ(y, b)
{0}	{0, 1, 2, 4, 7}	A	{3, 8}	{5}
{3, 8}	{1, 2, 3, 4, 6, 7, 8}	B	{3, 8}	{5, 9}
{5}	{1, 2, 4, 5, 6, 7}	C	{3, 8}	{5}
{5, 9}	{1, 2, 4, 5, 6, 7, 9}	D	{3, 8}	{5, 10}
{5, 10}	{1, 2, 4, 5, 6, 7, 10}	E	{3, 8}	{5}

δ :-

Q / Σ	a	b
start → A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C



DFA for r = (a+b)* abb.

3.) DFA TO Min DFA :-

Minimisation Rule :-

states can be merged if

(all states have same transitions) and (all are final or non-final)

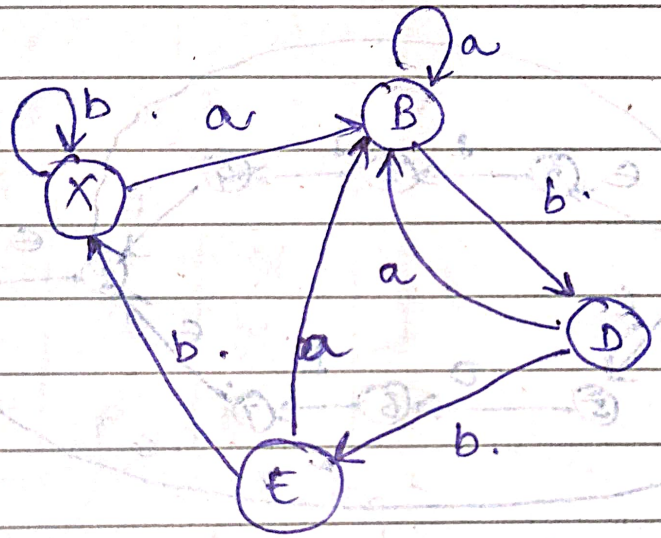
DATE / /

S:-

X = A & C

Q \ Σ	a	b
→ X	B	X
B	B	D
D	B	E
E*	B	X.

A = C = ~~D~~.



Min DFA for r = (a+b)* abb.

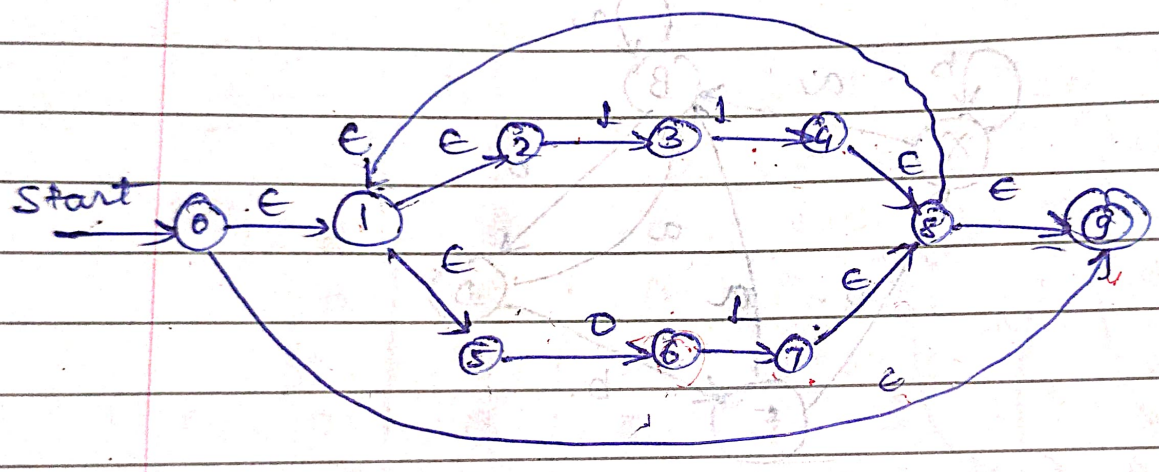
Q. Design DFA / FA for the following. (Dec 06/Sm)

Sol: $r = (11 + 01)^*$

Sol

Step 1: R·E to NFA

Step 2 ~~NF~~



Step 2:- NFA TO DFA

x	y = ε-closure(x)		δ(y, 0)	δ(y, 1)
{0}	{0, 1, 2, 5, 9}	A	{6}	{3}
{6}	{6}	B	{}	{7}
{3}	{3}	C	{}	{4}
{4}	{1, 2, 5, 7, 8, 9}	D	{6}	{3}
{4}	{1, 2, 4, 5, 8, 9}	E	{6}	{3}
{}	{}	F	{}	{}

δ :-

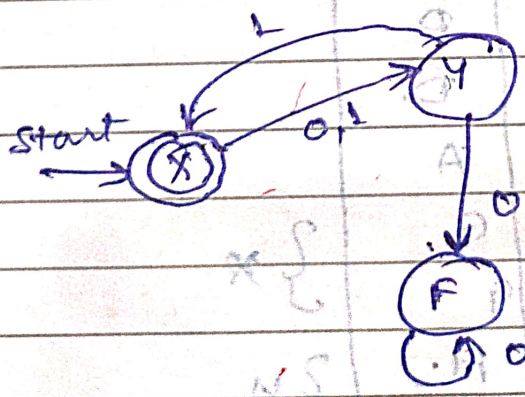
Q / Σ	0	1
→ A*	B	C
B	F	D
C	F	E
D*	B	C
E*	B	C
F	F	F

Step 3 :- DFA to min. DFA

Q / Σ	0	1
X*	B	C
B	F	~~X~~
C	F	~~X~~
F	F	F

Y.

Q / Σ	0	1
→ X*	Y	Y
Y	F	X
F	F	F



Min DFA for $r = (11 + 01)^*$

$L(r) = \{\epsilon, 11, 01, 1101, 0111 \ldots\}$

Bhatgraphics@gmail.com

Q.2 Construct DFA equivalent to given NFA (DEC .05/10)

$(\{P, q, r, s\}, \{0, 1\}, \delta, p, \{s\})$
$Q \qquad \Sigma$

Q / Σ	0	1
→ P	P, q	P
q	r	r
r	s	-
*s	s	s

Note :- NFA with ε or without ε, solution remains the same.

Sol

Step 1 :-

x	y = ε-closure(x)		δ(y, 0)	δ(y, 1)
{P}	{P}	A	{P, q}	{P}
{P, q}	{P, q}	B	{P, q, r}	{P, r}
{P, q, r}	{P, q, r}	C	{P, q, r, s}	{P, r}
{P, r}	{P, r}	D	{P, q, s}	{P}
{P, q, r, s}	{P, q, r, s}	E	{P, q, r, s}	{P, r, s}
{P, q, s}	{P, q, s}	F	{P, q, r, s}	{P, r, s}
{P, r, s}	{P, r, s}	G	{P, q, r, s}	{P, s}
{P, s}	{P, s}	H	{P, q, s}	{P, s}

Step 2 NFA to DFA

δ :-

Q / Σ	0	1	
→ A	B	A	
B	C	D	
C	E	D	
D	F	A	
E*	E	G	} x
F*	E	G	
G*	E	H	} y
H*	F	H	

Page No. 37
DATE / /

Step 3 DFA to min. DFA

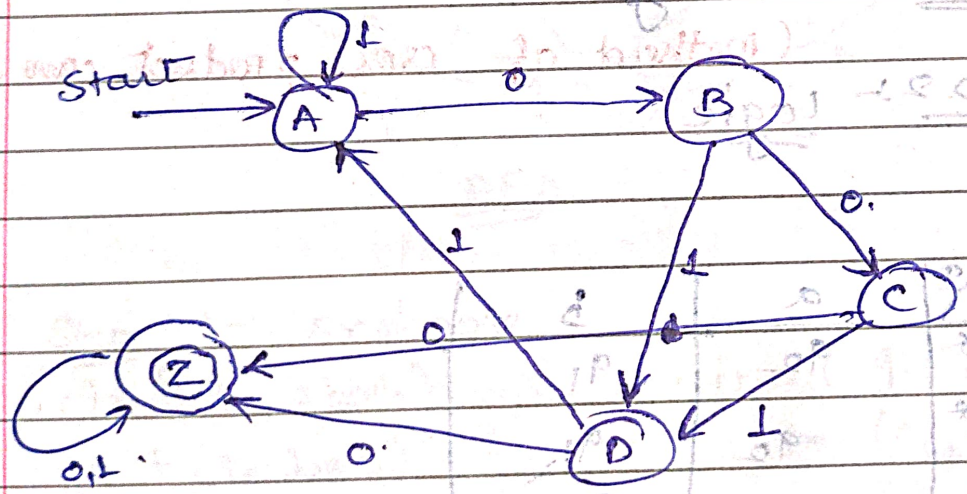
δ :-

Q / Σ	0	1	
→ A	B	A	
B	C	D	
C	X	D	
D	X	A	
X*	X	Y	Z
Y*	X	Y	

δ

Q / Σ	0	1
→ A	B	A
B	C	D
C	Z	D
D	Z	A
Z*	Z	Z

Step 4 :-



Q.:- Design a DFA for recognizing the foll. language:

$$L = \{ x \mid x \text{ contains even no. of a's and even no. of b's} \}$$

Sol Note:- If regular exp. is not given & they have asked for DFA then we can construct direct DFA using concept of FSM.

Step 1:- Theory

(Method of cross product can be used)

Step 2:- Logic

$n_a(x)$	$n_b(x)$	S / Σ	a	b
		→ q_s*	q_2	q_1
E	E	q_0*	q_2	q_1
E	O	q_1	q_3	q_0
O	E	q_2	q_0	q_3
O	O	q_3	q_1	q_2

Step 3:-

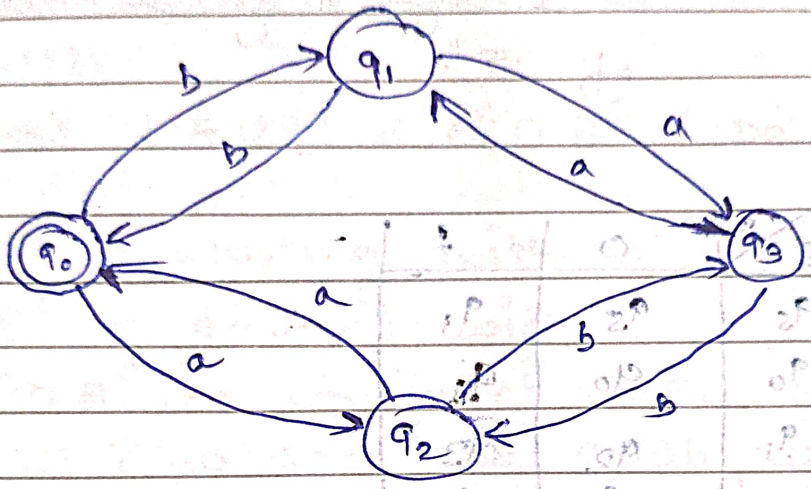
DFA $M = (Q, \Sigma, \delta, q_0, F)$

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

$\Sigma = \{a, b\}$ $F = \{q_0\}$

δ =

Q / Σ	a	b
q_0*	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2



DFA

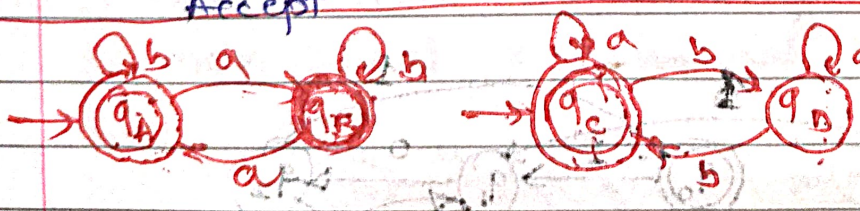
Step 4 :- Example

(i) $(q_0, abaab)$
$\vdash (q_2, baab)$
$\vdash (q_3, aab)$
$\vdash (q_1, ab)$
$\vdash (q_0, b)$
$\vdash (q_1, \epsilon)$

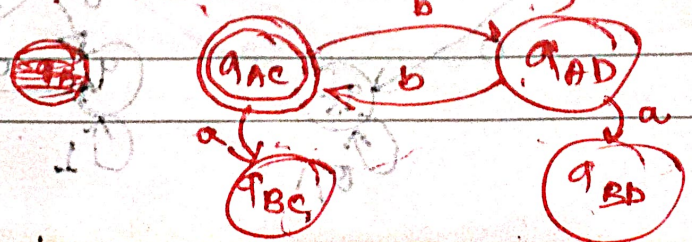
Accept

(ii) $(q_0, bbba)$
$\vdash (q_1, bba)$
$\vdash (q_0, a)$
$\vdash (q_2, \epsilon)$

Reject



$q_A q_B \times q_C q_D = \{q_A q_C, q_A q_D, q_B q_C, q_B q_D\}$



Prob

Q. Give the DFA for recognizing the following language over $\Sigma = \{0,1\}$ such that

L = "set of all strings that begin with "1" and when interpreted as binary integes it multiple of 5".

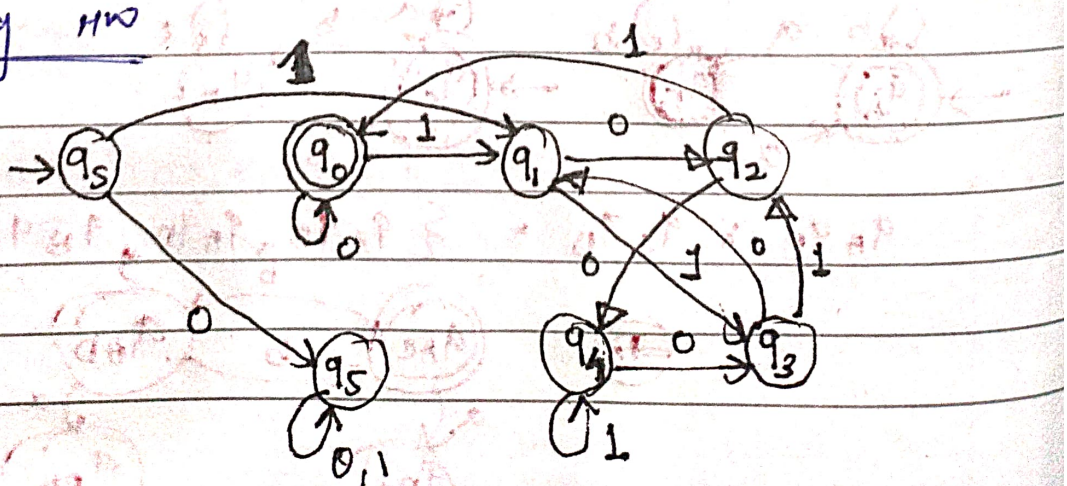
Sol

S / I	0	1
remainder → q_5	q_5	q_1
0 q_0	q_0	q_1
1 q_1	q_2	q_3
2 q_2	q_4	q_0
3 q_3	q_1	q_2
4 q_4	q_3	q_4
dead state q_5	q_5	q_5

↳ starting with 0

Note if string starts with 0, then that string will enter in dead state.

Remaining HW



1101

101 = 5
1010 = 10
1111 = 15 Dead state

Q. Construct a DFA which accepts a language of all strings not starting with 'a' or not ending with 'b' over the alphabet $\Sigma = \{a, b\}$.

11 - q_3
100 - q_4
101 - q_5

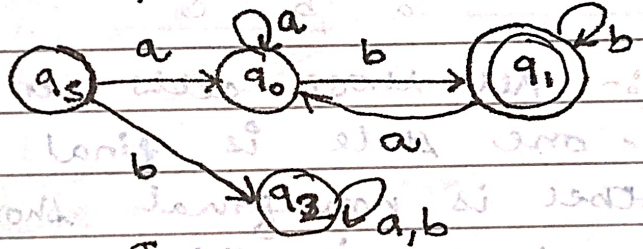
Solution $\Sigma = \{a, b\}$

not starting with 'a' → A
not ending with 'b' → B
A ∪ B

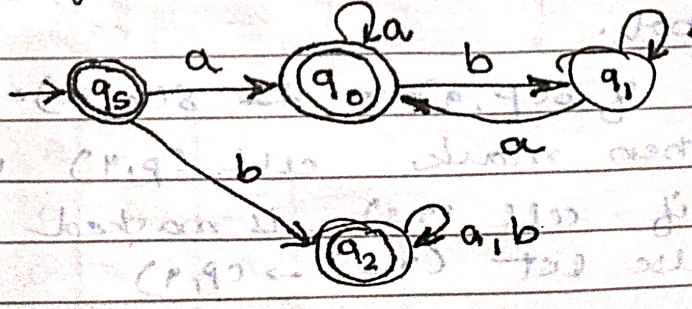
Now, $(A \cup B)^c = A^c \cap B^c$ (De morgan's Law)

A^c → starting with 'a'
B^c → ending with 'b'
$A^c \cap B^c$ → construct DFA for this

L = { ab, aab, abb, aaab, abab, abbb ... }



Now, this is the DFA for $A^c \cap B^c$ but we want for $(A \cup B)$, complement the above DFA by making final state to non final & nonfinal to final.

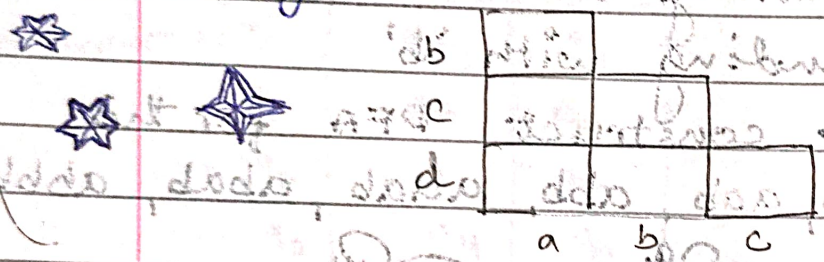


DFA Minimisation (Box Method) (Based on Mybill-Nerode theorem)

Step 1:- Remove all those states which are not reachable from the start state.

Step 2:- Create a table such that the rows and the columns can be identified by the state number.

eg.



Step 3:- All those cells in which one state is final and other is non final should be marked with "1".

Step 4:- For remaining unmarked cells perform following check:

Let (p, q) be an unmarked cell and let 'a' be some ip symbol.

* if $\delta(p, a) = r$ & $\delta(q, a) = s$.

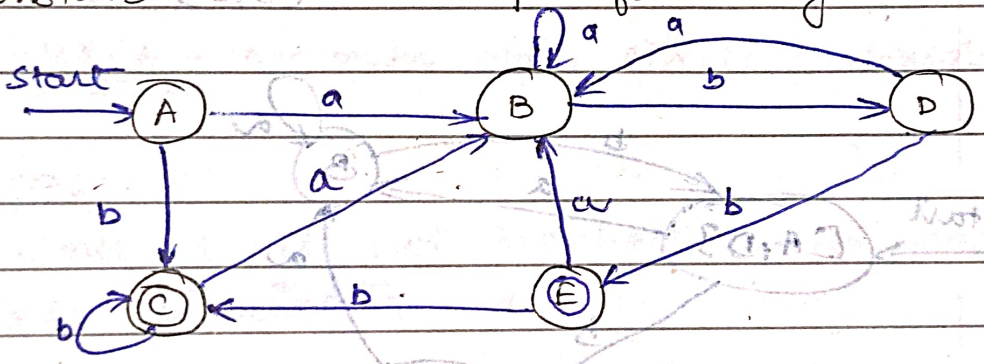
then mark cell (p, q) with "2" if cell (r, s) is marked else list (r, s) → (p, q)

* if $\delta(p,a) = \delta(q,a) = r$.
then no decision can be taken.

If cell (p,q) is still unmarked w.r.t 'a' then repeat step 4 for cell (p,q) with another i/p if available else don't repeat.

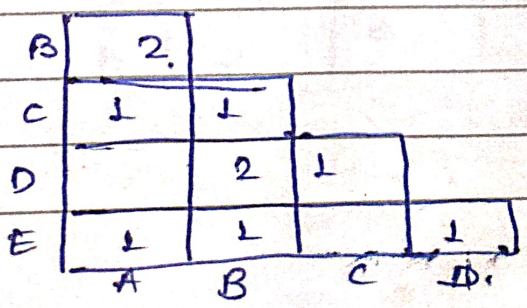
Step 5 :- All those cells which are unmarked can be merged into single step respectively.

Q. Construct min DFA for following.



δ :-

Q / Σ	a	b
→ A	B	C
B	B	D
C*	B	C
D	B	E
E*	B	C



(C,E) : $\delta(C,a) = B$ $\quad \delta(E,a) = B$

$\delta(C,b) = C$ $\quad \delta(E,b) = C$

(B,D) : $\delta(B,a) = B$ $\quad \delta(D,a) = B$

$\delta(B,b) = D$ $\quad \delta(D,b) = E$

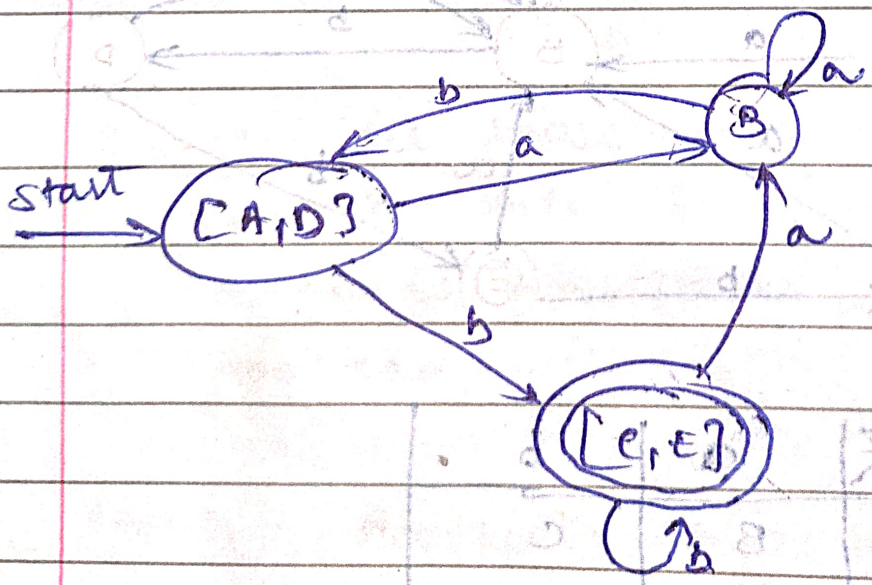
(A,D) : $\delta(A,a) = B$ $\quad \delta(D,a) = B$

$\delta(A,b) = C$ $\quad \delta(D,b) = E$

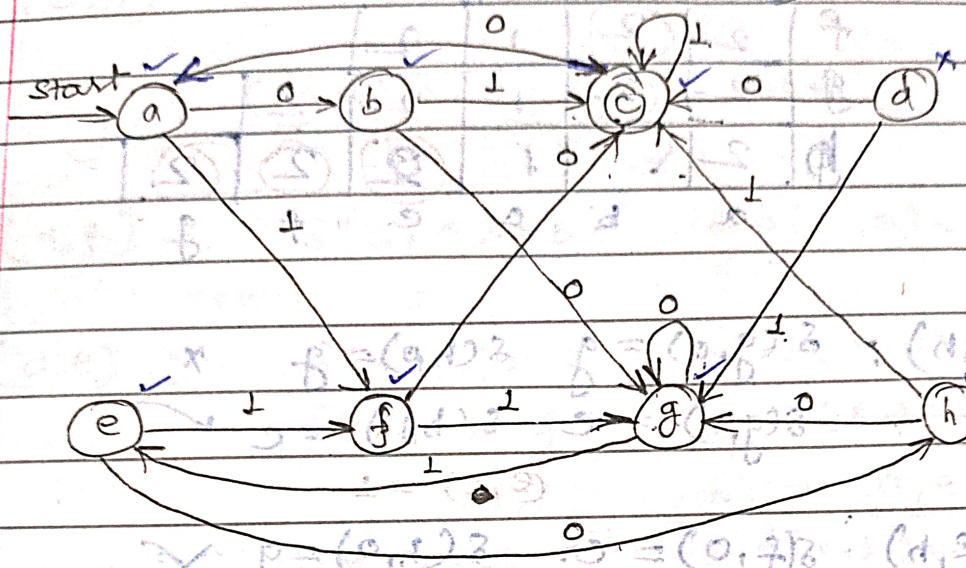
list $(C,E) \rightarrow (A,D)$

(A,B) : $\delta(A,a) = B$ $\quad \delta(B,a) = B$

$\delta(A,b) = C$ $\quad \delta(B,b) = D$



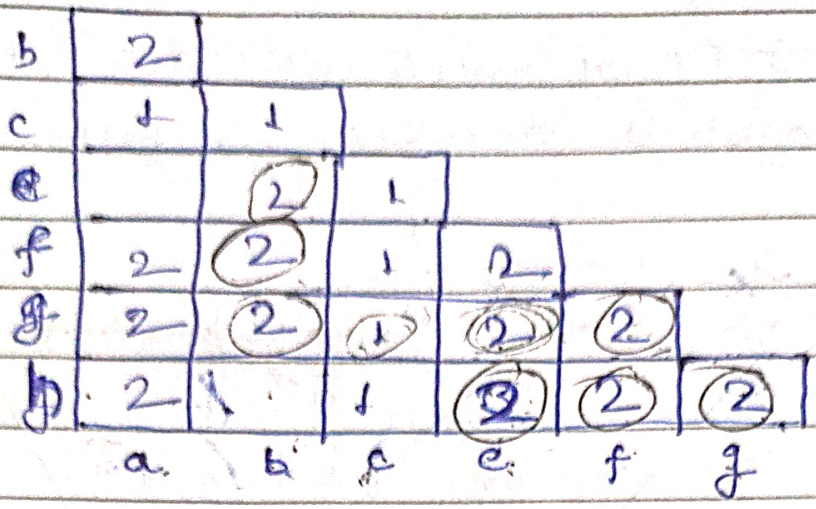
Q. (Min FA of min A OR MSFA.)
Construct min DFA for following:



MSFA:— minimum state finite automata.

Step 1
Since d is not reachable from start state we eliminate 'd'.

δ	0	1
a	b	f
b	g	c
c	a	c
e	h	f
f	c	g
g	g	e
h	g	c



(g,h) : $\delta(g,0) = g$ $\delta(h,0) = g$ ×

$\delta(g,1) = e$ $\delta(h,1) = c$ ✓

(e,c) = 1.

(f,h) : $\delta(f,0) = c$. $\delta(h,0) = g$ ✓

(c,g) = 1.

(f,g) : $\delta(f,0) = c$ $\delta(g,0) = g$ ✓

(c,g) = 1

(e,h) : $\delta(e,0) = h$ $\delta(h,0) = g$ ✓

(e,g) = $\delta(e,0) = h$ $\delta(g,0) = g$ ✓

(e,f) : $\delta(e,0) = h$ $\delta(f,0) = c$ ✓

(b,h) : $\delta(b,0) = g$ $\delta(h,0) = g$ ×

$\delta(b,1) = c$ $\delta(h,1) = c$ ×

(b,g) : $\delta(b,0) = g$ $\delta(g,0) = g$ ×

$\delta(b,1) = c$ $\delta(g,1) = e$ ✓

(b,f) = $\delta(b,0) = g$ $\delta(f,0) = c$ ✓

(b,e) : $\delta(b,0) = g$ $\delta(e,0) = b$ ~

(a,b) : $\delta(a,0) = b$ $\delta(b,0) = g$ ~

(a,g) : $\delta(a,0) = b$ $\delta(g,0) = g$ ~

(a,f) : $\delta(a,0) = b$ $\delta(f,0) = e$ ~

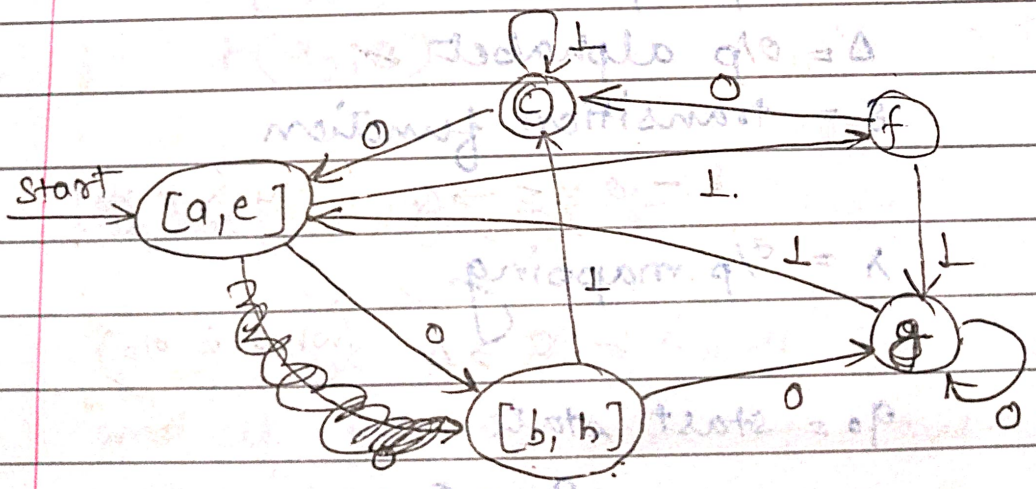
(a,e) : $\delta(a,0) = b$ $\delta(e,0) = b$ ✗

~~$\delta(a,1) = f$~~ ~~$\delta(e,1) = f$~~ ✗

list (b,b) → (a,e).

$\delta(a,1) = f$ $\delta(e,1) = f$ ✗

(a,b) : $\delta(a,0) = b$ $\delta(b,0) = g$ ~



MOORE MACHINE :-

Definition: It is a FA with no final state & it produces the o/p sequence for the given i/p sequence.

In moore machine, there is a symbol along with each state & such a symbol is called o/p symbol.

Mathematical Expression:

It is a 6 tupple representation.

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where Q = finite set of state.

Σ = i/p alphabet.

Δ = o/p alphabet

δ = transition function

$\delta :- Q \times \Sigma \to Q$. (which is next state)

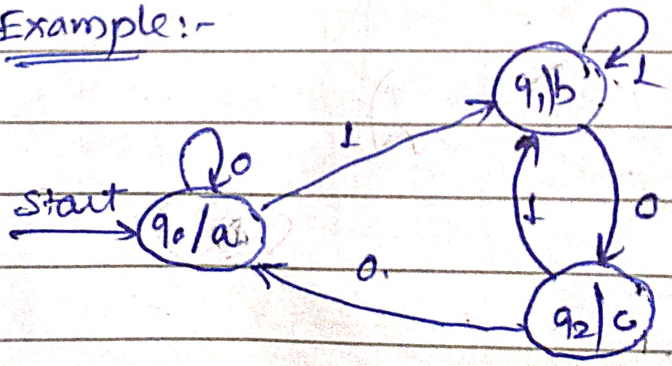
λ = o/p mapping

$\lambda :- Q \to \Delta$ (what is o/p)

q_0 = start state

$q_0 \in Q$.

Example:-



Moore Machine

$Q = \{q_0, q_1, q_2\}$ $\Delta = \{a, b, c\}$

$\Sigma = \{0, 1\}$ $q_0 = q_0$

$\delta =$

Q \ Σ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_1

$\lambda =$

$\lambda(q_0) = a$

$\lambda(q_1) = b$

$\lambda(q_2) = c.$

Example :-

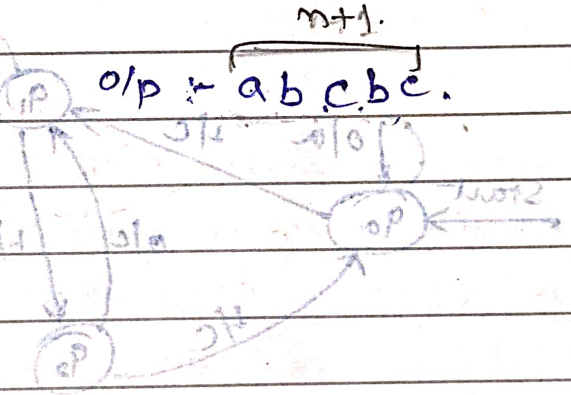
$(q_0, \overbrace{1010}^{n})$ o/p :- $\overbrace{abcbc}^{n+1}$.

$\vdash (q_1, 010)$

$\vdash (q_2, 10)$

$\vdash (q_1, 0)$

$\vdash (q_2, \epsilon)$



MELAY MACHINE :-

It is a FA. with no final state and it produces the o/p sequence for given i/p sequence.

In Mealy machine, there is a symbol along with each transition and such a symbol is called o/p symbol.

Mathematical Expression:-

It is a 6 tuple representation.

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where

Q = finite set of states.

Σ = i/p alphabet

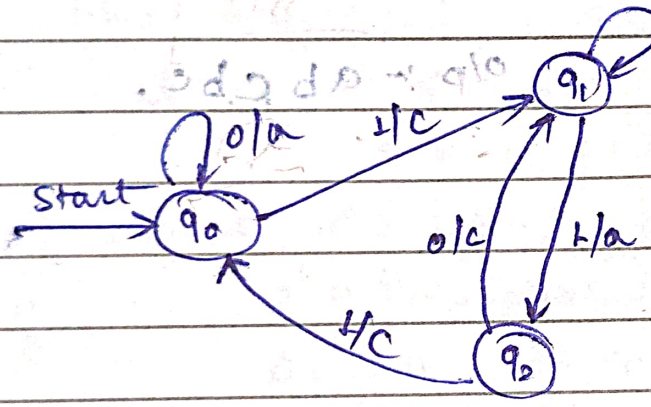
Δ = o/p alphabet

δ = transition function

$\delta : Q \times \Sigma \to Q$.

λ = o/p mapping. $\lambda : Q \times \Sigma \to \Delta$

q_0 = ~~state~~ start state



Mealy Machine.

$Q = \{q_0, q_1, q_2\}$ $\quad \Delta = \{a, b, c\}$

$\Sigma = \{0, 1\}$ $\quad q_0 = q_0$.

$\delta =$

Q \ Σ	0	1
→ q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_0

$\lambda =$

Q \ Σ	0	1
→ q_0	a	c
q_1	b	a
q_2	c	c

Example :-

$(q_0, 1010)$ o/p :- c b a c.

$\vdash (q_1, 010)$

$\vdash (q_1, 10)$

$\vdash (q_2, 0)$

$\vdash (q_1, \epsilon)$

Page No.
DATE / /

	MOORE MACHINE	MEALY MACHINE
1.	In moore machine state gives the o/p	In mealy machine transition gives the o/p
2.	In moore machine o/p mapping is defined as $\lambda :- Q \to \Delta$	In Mealy machine, o/p mapping is defined as. $\lambda : Q \times \Sigma \to \Delta$
3.	In Moore machine o/p is dependent on state.	In Mealy machine o/p is dependent on state & the input.
4.	In Moore machine if length of i/p sequence is n then length of o/p sequence is $n+1$	In Mealy machine, if length of i/p sequence is n then length of o/p sequence is also n.
5.	In Moore, machine we can get o/p on ϵ.	In Mealy machine we cannot get o/p on ϵ.
6.	Example of Moore machine	Example of Mealy machine.

Minimisation of Moore Machine

States can be merged if they satisfy following two conditions:

1. All the states have same transitions. &
2. O/p symbols along with the states are same.

Q. Design Moore machine to O/p
'A' if i/p ends in "101".
'B' if i/p ends in "110".
'C' otherwise.
over $\Sigma = \{0,1\}$

Sol:- Step 1 : Theory

Step 2 :- Logic.

S / I	0	1		O/P (Δ)
→ q_1	q_0	q_1	q_0	C
0 q_0	q_0	q_1		C
1 q_1	q_2	q_4		C
10 q_2	q_0	q_3		C
101 q_3	q_2	q_4		A
11 q_4	q_5	q_4		C
110 q_5	q_0	q_3		B

Step 3:- Implementation.

$M = (Q, \Sigma, \Delta, \delta, q_0)$.

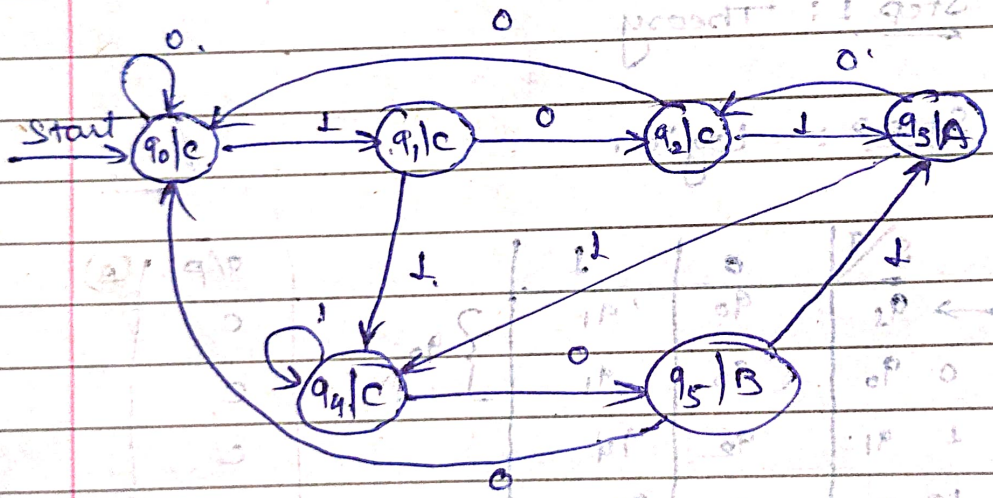
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$.

$\Delta = \{A, B, C\}$.

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

$q_0 = q_0$

δ : Q \ Σ	0	1	o/p
q_0	q_0	q_1	$\lambda(q_0) = C$
q_1	q_2	q_4	$\lambda(q_1) = C$
q_2	q_0	q_3	$\lambda(q_2) = C$
q_3	q_2	q_4	$\lambda(q_3) = A$
q_4	q_5	q_4	$\lambda(q_4) = C$
q_5	q_0	q_3	$\lambda(q_5) = B$



Moore Machine

Step 4:- Example :- $(q_0, 1101)$!

$\vdash (q_1, 101)$
$\vdash (q_4, 01)$
$\vdash (q_5, 1)$
$\vdash (q_3, e)$

o/p CCCBA

$(q_0, 10110)$
$\vdash (q_1, 0110)$
$\vdash (q_2, 110)$
$\vdash (q_3, 10)$
$\vdash (q_4, 0)$ $\vdash (q_5, e)$

CCCACB

Q. Design Moore machine for the following folding process:
"print residue modulo 3 for binary i/p"

Sol:- Step 1 :- Theory

Step 2 :- Logic

S / I	0	1		O/P.
q_3	q_0	q_1	q_0	0
q_0	q_0	q_1		0
q_1	q_2	q_0		1
q_2	q_1	q_2		2

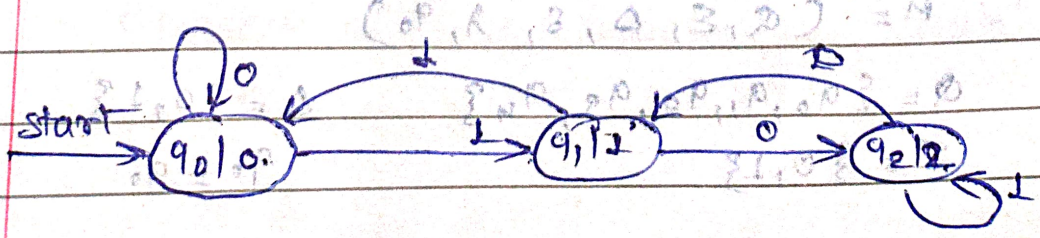
Step 3 :- Implementation

$M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

$Q = \{q_0, q_1, q_2\}$ $\quad \Delta = \{0, 1, 2\}$

$\Sigma = \{0, 1\}$ $\quad q_0 = q_0$

δ (Q \ Σ)	0	1	
→ q_0	q_0	q_1	$\lambda(q_0) = 0$
q_1	q_2	q_0	$\lambda(q_1) = 1$
q_2	q_1	q_2	$\lambda(q_2) = 2$



NOTE: In moore & mealy machine designs only last bit can be changed.

Page No. DATE / /

Step 4:- Example.

$(q_0, 11.01)$
$\vdash (q_1, 101)$
$\vdash (q_0, 01)$
$\vdash (q_0, 1)$
$\vdash (q_1, \epsilon)$

0 1 0 0 (1.)
↓ δ(q0, 1)

$(q_0, 1000)$
$\vdash (q_1, 001)$
$\vdash (q_2, 01)$
$\vdash (q_1, 1)$
$\vdash (q_0, \epsilon)$

0 1 2 1 (0)
↓
final o/p:

Q. Design Moore machine to change each occurance of substring '1000' to "1001". over $\Sigma = \{0,1\}$

(Type:- Search & Replace) (June 05/10 M)

Sol Step 1 :- Theory.

Step 2 :- Logic

I / S			O/P
→ q_0	q_0	q_1	0
0 q_0	q_0	q_1	0
1 q_1	q_2	q_1	1
10 q_2	q_3	q_1	0
100 q_3	q_4	q_1	0
1000 q_4	q_0	q_1	1

Step 3:- Implementation:

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

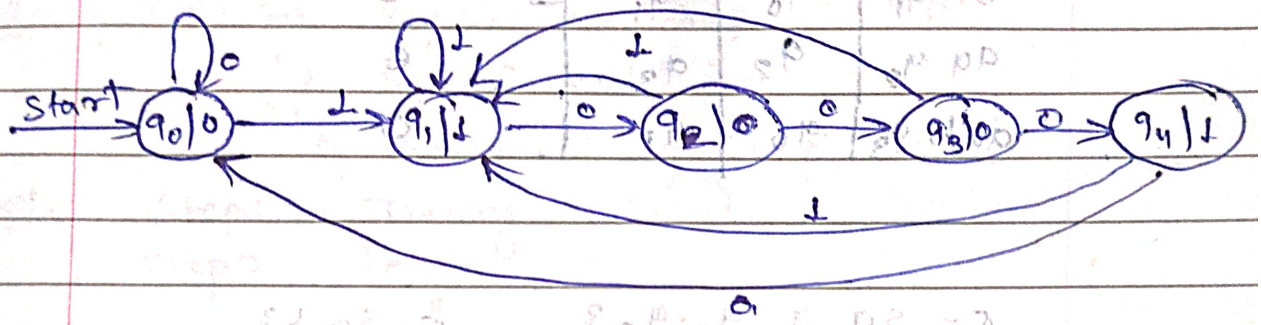
$Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{0,1\}$
$\Delta = \{0,1\}$
$q_0 = q_0$

DATE 7/7

δ:

Q \ Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_3	q_1
q_3	q_4	q_1
q_4	q_0	q_1

$\lambda(q_0) = 0$
$\lambda(q_1) = 1$
$\lambda(q_2) = 0$
$\lambda(q_3) = 0$
$\lambda(q_4) = 1$



Step 4 :- Example

$(q_0, 0100000)$
$\vdash (q_0, 100000)$
$\vdash (q_1, 00000)$
$\vdash (q_2, 0000)$
$\vdash (q_3, 000)$
$\vdash (q_4, 00)$
$\vdash (q_0, 0)$
$\vdash (q_0, \epsilon)$

O/P :- 00 100 10

Q:- Design moore machine to count each occurance of substring "aab" in long I/P string over $\Sigma = \{a, b\}$.

Page No.
DATE 1 / 1

Step 1 :-
Theory :-

Step 2 :- Logic

Σ / I	a	b	
$\rightarrow q_3$	q_0	q_1	ϵ
a q_0	q_2	q_1	q_1 ϵ
b q_1	q_0	q_1	ϵ
aa q_2	q_2	q_3	ϵ
aab q_3	q_0	q_1	

$Q = \{q_0, q_1, q_2, q_3\}$ $\quad \Sigma = \{a, b\}$

$\Delta = \{$

Minimisation of Mealy Machine

States can be merged if they satisfy following conditions.

1.) All the states have same transitions.
2.) O/p symbols along with transitions are same.

Q. Design mealy machine to o/p.
'A' if i/p ends in '101'
'R' otherwise over $\Sigma = \{0, 1\}$

Sol:- Step1 :- Theory
Step2 :- Logic

S/I	0	1
→ q_0	q_0	q_1
0 q_0	q_0	q_1
1 q_1	q_2	q_1
10 q_2	q_0	q_3
101 q_3	q_2	q_1

S/I	0	1	
→ q_0	R	R	q_0
q_0	R	R	
q_1	R	R	q_1
q_2	R	A	
q_3	R	R	

Step 3 :- Implementation.

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$Q = \{q_0, q_1, q_2\}$ $\quad \Delta = \{A, R\}$

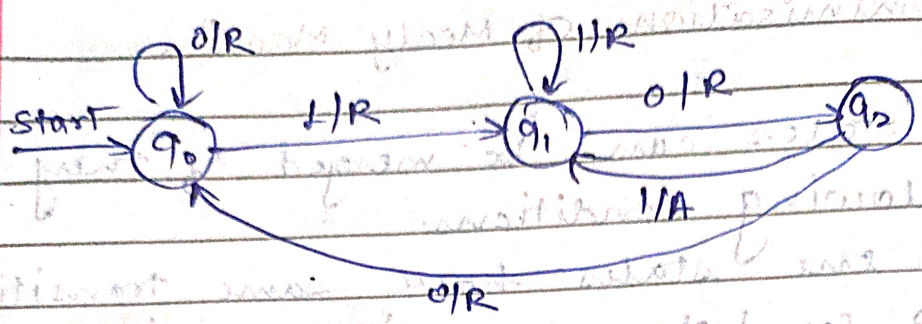
$\Sigma = \{0, 1\}$ $\quad q_0 = q_0$

$\delta =$

Q/Σ	0	1
→ q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_1

$\lambda =$

Q/Σ	0	1
→ q_0	R	R
q_1	R	R
q_2	R	A



Step 4:-

$(q_0, 1101)$ o/p R R R (A).

$\vdash (q_1, 101)$

$\vdash (q_1, 01)$

$\vdash (q_2, 1)$

$\vdash (q_1, \epsilon)$

Q. Construct Mealy machine to o/p EVEN and ODD according to the total no. of 1's encountered are even and odd. The o/p symbols are 0's & 1's. $\Sigma = \{0, 1\}$.

Sol

~~Step 1~~ :- Theory

Step 2 :- Logic

S \ I	0	1
→ q_s	q_0	q_1
q_0	q_0	q_1
q_1	q_1	q_0

S \ I	0	1
→ q_s	EVEN	ODD
q_0	EVEN	ODD
q_1	ODD	EVEN

q_0

Step 3 : Implementation.

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

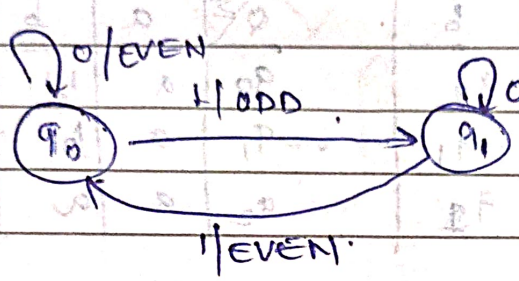
$Q = \{q_0, q_1\}$ $\quad \Delta = \{EVEN, ODD\}$

$\Sigma = \{0, 1\}$ $\quad q_0 = q_0$

$\delta =$

Q \ Σ	0	1
→ q_0	q_0	q_1
q_1	q_1	q_0

Q \ Σ	0	1
→ q_0	EVEN	ODD
q_1	~~ODD~~	EVEN



Step 4 :- $(q_0 ; 1011)$

$\vdash (q_1, 011)$ $\quad$ O/P :- ODD ODD EVEN ODD.

$\vdash (q_1, 11)$

$\vdash (q_0, 1)$.

$\vdash (q_1, \epsilon)$

Q: Construct Mealy machine to change each occurance of substring "abb" to "aba" over $\Sigma = \{a, b\}$

Sol Step 1 :- Theory

Step 2 :- Logic

$q_3, q_1, q_3 = q_1$

S \ I	a	b
→ q_s	q_0	q_1
a q_0	q_0	q_2
b q_1	q_0	q_1
ab q_2	q_0	q_3
abb q_3	q_0	q_1

S \ I	a	b
→ q_s	a	b
q_0	a	b
q_1	a	b. → q_1
q_2	a	a
q_3	a	b

DATE / /

Step3: Implementation

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

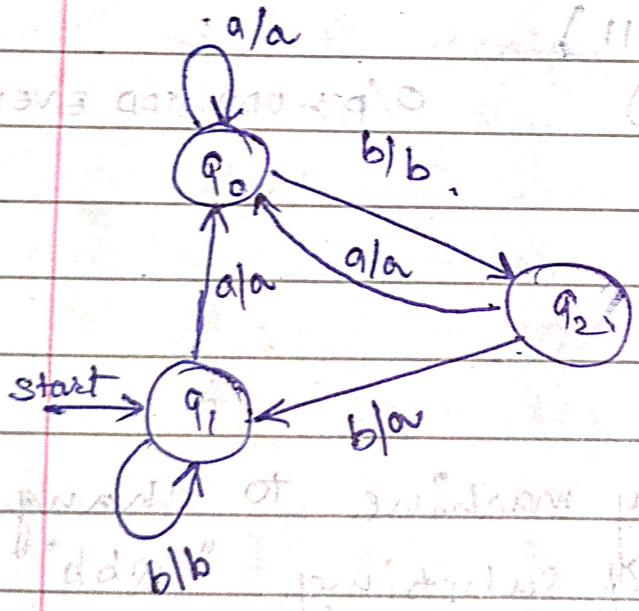
$Q = \{q_0, q_1, q_2\}$ $\quad \Delta = \{a, b\}$

$\Sigma = \{a, b\}$ $\quad q_0 = q_1$

$\delta =$

Q \ Σ	a	b
q_0	q_0	q_2
→ q_1	q_0	q_1
q_2	q_0	q_2

Q \ Σ	a	b
q_0	a	b
→ q_1	a	b
q_2	a	a



eg $(q_1, abba)$ $\quad$ o/p: abaa

$\vdash (q_0, bba)$

$\vdash (q_2, ba)$

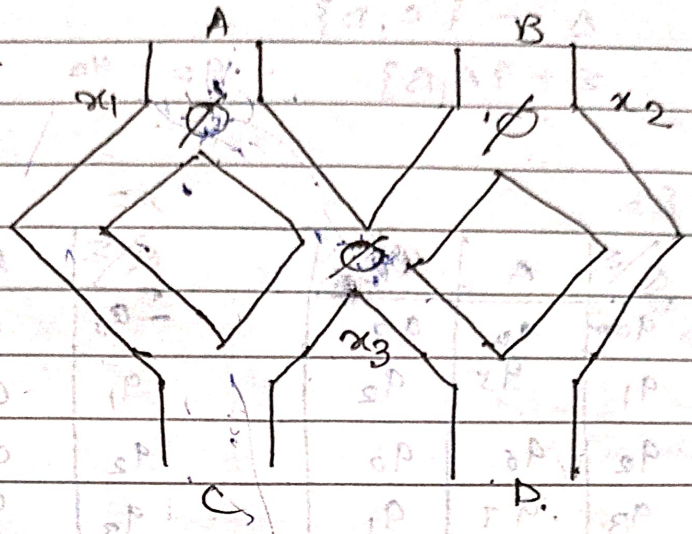
$\vdash (q_1, a)$

$\vdash (q_0, \epsilon)$

Q: Model the toy with Mealy machine.

Ø - Left

Q - Right



x_1	x_2	x_3	
L	L	L	q_0
L	L	R	q_1
L	R	L	q_2
L	R	R	q_3
R	L	L	q_4
R	L	R	q_5
R	R	L	q_6
R	R	R	q_7

A

x_1	x_2	x_3	S	o/p
R	L	L	q_4	C
R	L	R	q_5	C
R	R	L	q_6	C
R	R	R	q_7	C
L	L	L	q_0	D
L	L	R	q_1	C
L	R	L	q_2	C
L	R	R	q_3	C

B

x_1	x_2	x_3	S	o/p
L	R	R	q_3	C
L	R	L	q_2	D
L	L	L	q_0	D
L	L	R	q_1	D
R	R	R	q_7	C
R	R	L	q_6	D
R	L	L	q_4	D
R	L	R	q_5	D

* for I/p = A

if $x_1 = x_3 = L$ then o/p = D

else o/p = C

* for I/p = B

if $x_2 = x_3 = R$ then o/p = C

else o/p = D

Step 3 :-

$M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$

$\Delta = \{C, D\}$

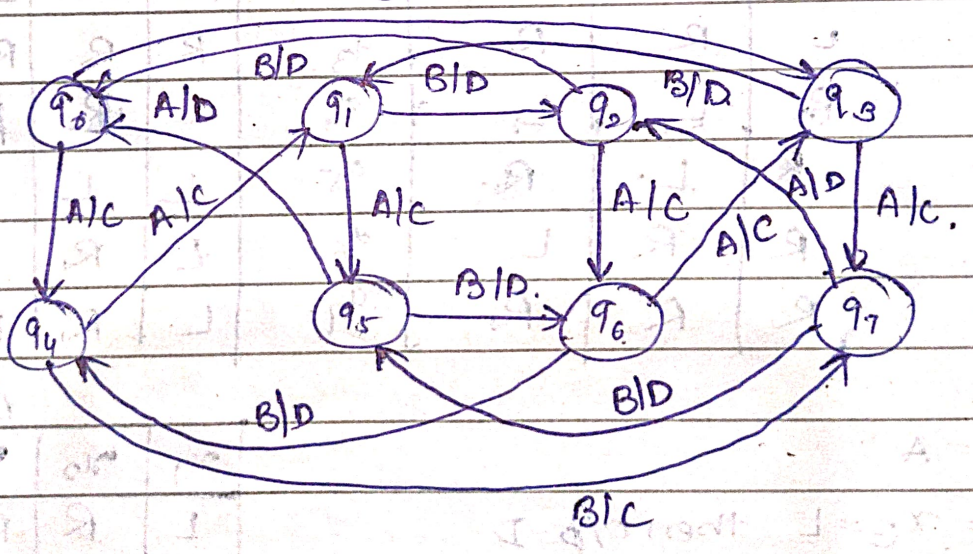
$\Sigma = \{A, B\}$ $\quad q_0 = q_a$

$\delta =$

	$Q \backslash \Sigma$	A	B
LLL	$\rightarrow q_0$	q_4	q_3
LLR	q_1	q_5	q_2
LRL	q_2	q_6	q_0
LRR	q_3	q_7	q_1
RLL	q_4	q_1	q_7
RLR	q_5	q_0	q_6
RRL	q_6	q_3	q_4
RRR	q_7	q_2	q_5

$\lambda =$

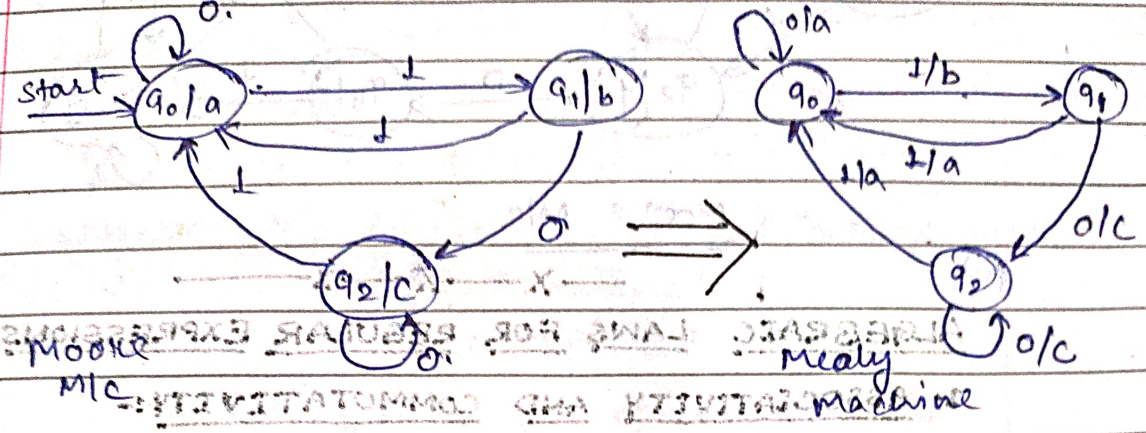
$Q \backslash \Sigma$	A	B
$\rightarrow q_0$	C	C
q_1	C	D
q_2	C	D
q_3	C	D
q_4	C	C
q_5	D	D
q_6	C	D
q_7	D	D



Mealy Machine

Moore Machine to Mealy Machine :-

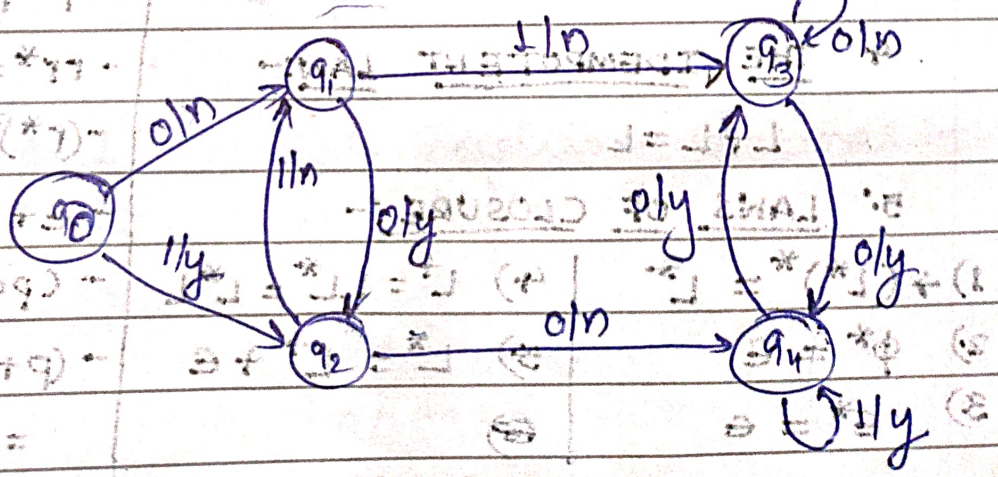
→ Assign the o/p symbol along with the state to all of its incoming transitions.



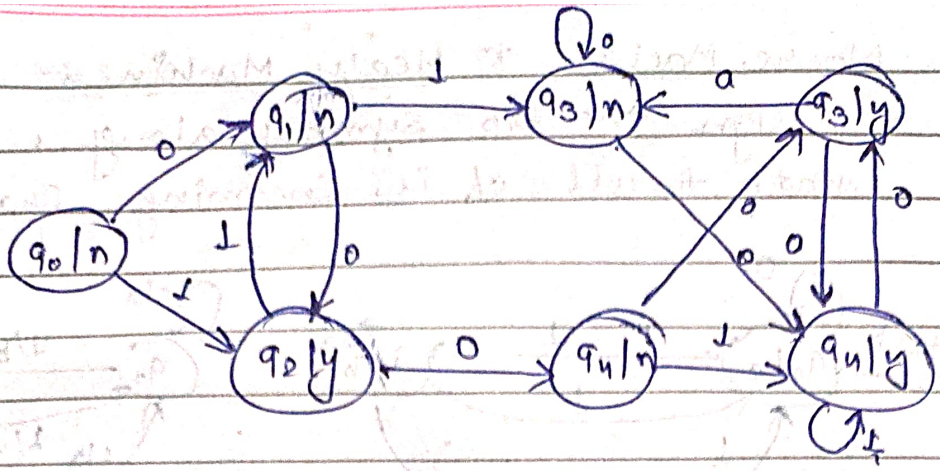
* Mealy M/c to Moore M/c

- If the o/p symbols with the incoming transitions to a state are same, then assign that symbol to that state.
- If the o/p symbols with incoming transitions to a state are not same then split that state as many times as the o/p symbols with each state producing a different o/p symbol.
- If there are no incoming transitions to a state, then any o/p symbol can be assigned to that state.

Ex-



Mealy machine



Moore M/C

ALGEBRAIC LAWS FOR REGULAR EXPRESSIONS :-

1. ASSOCIATIVITY AND COMMUTATIVITY :-

☑ $P+Q = Q+P$ ☒ $PQ \neq QR$

☑ $(P+Q)+R = P+(Q+R)$ ☑ $P(QR) = (PQ)R$

2. IDENTITIES :-

→ $\phi + L = L + \phi = L$ { ϕ is the identity for union}

→ $\epsilon L = L\epsilon = L$ { ϵ is the identity for concatenation}

→ $\phi L = L\phi = \phi$

3. DISTRIBUTIVE LAW :-

$x(y+z) = xy + xz$

$P(Q+R) = PQ + PR$

$(Q+R)P = QP + RP$

4. THE IDEMPOTENT LAW :-

$L+L=L$

5. LAWS OF CLOSURE :-

1) → $(L^*)^* = L^*$

2) $\phi^* = \epsilon$

3) $\epsilon^* = \epsilon$

4) $L^+ = LL^* = L^*L$

5) $L^* = L^+ + \epsilon$

6. IDENTITIES :-

- $\phi + r = r$
- $\phi r = r\phi = \phi$
- $\epsilon r = r\epsilon = r$
- $\epsilon^* = \epsilon$ & $\phi^* = \epsilon$
- $r + r = r$
- $r^* r^* = r^*$
- $rr^* = r^*r = r^+$
- $(r^*)^* = r^*$
- $\epsilon + rr^* = r^*$
- $(pq)^* p = p(qp)^*$
- $(p+q)^* = (p^*q^*)^*$
  $= (p^* + q^*)^*$
- $(p+q)r = pr + qr$
  $r(p+q) = rp + rq$

$\alpha + \beta$ are in some sentential form
ie any combination of variables & terminals.

DATE / /

GRAMMARS :-

Grammars is used for specifying Syntax of a language & is defined as follows :-

Def: $G = (V, T, P, S)$

where V: finite set of variables or Non-Terminals. (capital letters)

T: finite set of terminals (small letters / operators)

P: finite set of production OR Rewritting rules:

S: start variable

Types of Grammar :-

Chomsky's Hierarchy :-

① Type 0 / Unrestricted Grammer :-

R :- $\alpha \rightarrow \beta$ - α is $(V+T)^* V (V+T)^*$

eg: $S \rightarrow AB$ - β is $(V+T)^*$

$AB \rightarrow AC$ - These must be atleast one variable on left side of production Rule.

$A \rightarrow a$

$B \rightarrow b$.

$C \rightarrow c$

② Type 1 / Context Sensitive Grammar :-

R :- $|\alpha| \leq |\beta|$

eg :- $S \rightarrow aAb$.

$aA \rightarrow aaA$

$aA \rightarrow bA$

$A \rightarrow b$

(3.) Type 2 / context free Grammar :-

R:- $A \rightarrow \alpha$

eg: $S \rightarrow aAb \mid aSb \mid a$

$A \rightarrow SbA \mid a.$

(4.) Type 3 / Regular Grammer :-

R:-

Left Linear G Right Linear Grammar

LLG	RLG
$S \rightarrow Aa$	$S \rightarrow aA$
$A \rightarrow Bb \mid a$	$A \rightarrow aA \mid a$
$B \rightarrow Cg$	

Def. of context Free Grammer

Grammar is said to be CFG if all the productions are of the form $A \rightarrow \alpha$ (A can take the form of α). where A is variable & α is in some sentential form.

Derivation :-

It is defined as the process to check whether the given grammar can derive/ generate the given sentence using any combination of production rules starting from start variables production rule.

* Left Most Derivation (LMD) :-

Derivation is LMD if at every step we select and replace the left most variable by its production rule.

* Right Most Derivation (RMD) :-

Derivation is RMD if at every step we select and replace the rightmost variable by its production rule.

Q. $S \to aAS \mid a$
$A \to SbA \mid SS \mid ba$

Derive using LMD & RMD "aabbaa".

Sol. **LMD :-**

$S \underset{lm}{\Rightarrow} aAS$ using $S \to aAS$

$\underset{lm}{\Rightarrow} asbAS$ using $A \to SbA$

$\underset{lm}{\Rightarrow} aabAS$ using $S \to a$

$\underset{lm}{\Rightarrow} aabbaS$ using $A \to ba$.

$\underset{lm}{\Rightarrow} aabbaa$ using $S \to a$

RMD :- $S \underset{rm}{\Rightarrow} aAS$ using $S \to aAS$.

$\underset{rm}{\Rightarrow} aAa$ using $S \to a$

$\underset{rm}{\Rightarrow} asbAa$ using $A \to SbA$.

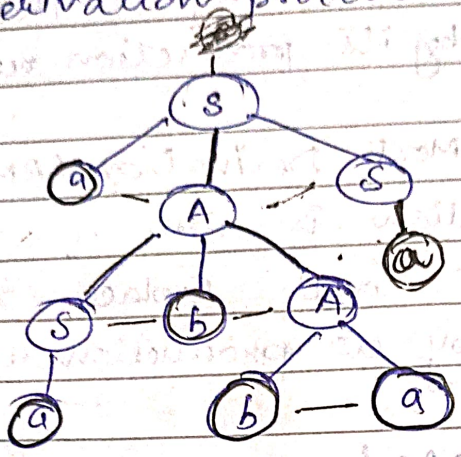
$\underset{rm}{\Rightarrow} asbbaa$ using $A \to ba$

$\underset{rm}{\Rightarrow} aabbaa$ using $S \to a$

Page No.
DATE / /

Derivation | Rule | Parse tree :-

It is graphical representation of the derivation process.



LMD

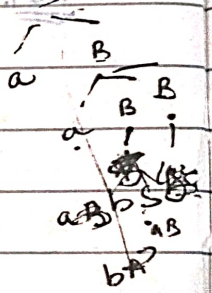
aabbaa

Q. $S \rightarrow aB \mid bA$
$A \rightarrow a \mid aS \mid bAA$
$B \rightarrow b \mid bS \mid aBB$

Derive using LMD & RMD

(1) aaabbba

(2) aabaa



(1) LMD

$S \underset{lm}{\Rightarrow} aB$ using $S \rightarrow aB$

$S \underset{lm}{\Rightarrow} aaBB$ using $B \rightarrow aBB$

$S \underset{lm}{\Rightarrow} aaaBBB$ using $B \rightarrow aBB$

$S \underset{lm}{\Rightarrow} aaabBB$ using $B \rightarrow b$

$S \underset{lm}{\Rightarrow} aaabbB$ using $B \rightarrow b$

$S \underset{lm}{\Rightarrow} aaabbb$ using $B \rightarrow b$.

RMD

$S \underset{rm}{\Rightarrow} aB$ using $S \to aB$.

$S \underset{rm}{\Rightarrow} aaBB$ using $S \to aBB$

$S \underset{rm}{\Rightarrow} aaBb$ using $B \to b$

$S \underset{rm}{\Rightarrow} aaaBBb$ using $B \to aBB$

$S \underset{rm}{\Rightarrow} aaaBbb$ using $B \to b$

$S \underset{rm}{\Rightarrow} aaabbb$ using $B \to b$.

(2.) a a b a

RMD :-

$S \underset{rm}{\Rightarrow} aB$. using $S \to aB$.

$S \underset{rm}{\Rightarrow} aabS$ using $B \to bS$

$S \underset{rm}{\Rightarrow} aab$

Sentence is not derivable.

Note:- If a sentence is derivable then it can be derived using LMD + RMD both.

Page No.
DATE / /

Examples of Context Free Grammer :-

r.e	L(r)/CFL	CFG
a	{a}	S → a
b	{b}	S → b
a|b or a+b	{a, b}	S → a|b
a.b	{ab}	S → ab
a*	{ε, aa, aaa ...}	S → aS|ε
a+	{a, aa, aaa ...}	S → aS|a
(a.b)*	{ε, ab, abab ...}	S → abS|ε
(a+b)*	{ε, a, b, aa, ab, ba, bb}	S → aS|bS|ε.

r - regular expression.
CFL - CF language
CFG - CF Grammar

Q. Write CFG to generate following.
* strings that start with 'a' over $\Sigma = \{a, b\}$

Sol P = $S \to aX$
$X \to aX|bX|\epsilon$
$G = (\{S, X\}, \{a, b\}, P, S)$
$L(G) = \{a, aa, ab, aaa, aab \cdots\}$

Q. * strings that start & end with different letter over $\{a, b, c\}$.

Sol P = $S \to aXb|aXc|bXa|bXc|cXa|cXb$
$X \to aX|bX|cX|\epsilon$
$G = (\{S, X\}, \{a, b, c\}, P, S)$

* Strings that contain atleast one occurance of "00" over $\Sigma = \{0,1\}$.

P = $S \to X00X$.
$X \to 0X \mid 1X \mid \epsilon$.

* Strings that contain atleast two 0's over $\Sigma = \{0,1\}$ (Dec 06/5M)

P = $S \to X0X0X$
$X \to 0X \mid 1X \mid \epsilon$

* Strings that contain alternate sequence of 0's & 1's. over $\Sigma = \{0,1\}$ (Dec 06/5M)

P = $S \to 0X \mid 1Y$
$X \to 1Y \mid \epsilon$
$Y \to 0X \mid \epsilon$

Q. Write CFG to generate the following.
* $L = \{a^n b^n \mid n \geq 1\}$

$n=1$ ab
$n=2$ aabb
$n=3$ aaabbb

Ans 1:- $S \to aSb \mid ab$

Ans 2:- $S \to aXb$
$X \to aXb \mid \epsilon$.

Page No.
DATE / /

* $L = \{a^n b^{3n} \mid n \geq 1\}$

$n=1$ abbb

$n=2$ aa bbb bbb.

(1) $S \rightarrow aSbbb \mid abbb.$

(2.) $S \rightarrow aXbbb.$

$X \rightarrow aXbbb \mid \epsilon$

* Write CFG to generate the foll.

$L = \{a^n b^{n+1} \mid n \geq 1\}$

$n=1$ abb

$n=2$ aabbb.

(1) $S \rightarrow aXbb.$

$X \rightarrow aXb \mid \epsilon$

(2.) $S \rightarrow aXbb.$

$X \rightarrow aXb \mid \epsilon$

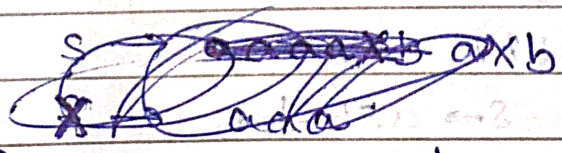
(3.) $S \rightarrow asb \mid abb.$

* $L = \{a^n b^{2n} \mid n \geq 1\}$

$S \rightarrow aSbb \mid abb$

* $L = \{a^m b^{m-3} \mid m \geq 3\}.$

asa|bsb|aa.



(1) $S \rightarrow asb \mid aaa$

(2.) $S \rightarrow aaaX$

$X \rightarrow aXb \mid \epsilon.$

Q. Write a CFG to generate the following. (May-06 Dec)

$L_1 = \{0^n 1^n 2^i \mid n \geq 1, i \geq 1\}$

$S \rightarrow XY$

$X \rightarrow 0X1 \mid 01$

$Y \rightarrow 2Y \mid 2$

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

$S \rightarrow XY$

$X \rightarrow 0X \mid 0$

$Y \rightarrow 1Y2 \mid 12$

Q. even palindrome over $\Sigma = \{a, b\}$

$S \rightarrow aSa \mid bSb \mid \epsilon$

odd palindrome over $\Sigma = \{a, b, c\}$

$S \rightarrow aSa \mid bSb \mid cSc \mid a \mid b \mid c$

Pg 87 Q. 5 L is binary string divisible by 4.

$L = \{0, 0100, 1000, 1100, 10000, \ldots\}$

P:- $S \rightarrow A00 \mid 0$

$A \rightarrow 0AB \mid 10B \mid 11B$

$B \rightarrow 0B \mid 1B \mid \epsilon$

01000

01100

Simplification of CFG.

Simplification of CFG implied the rewritting CFG after eliminating

1. Useless productions
2. Unit productions

Such that $L(G') = L(G)$

where G is given grammer & G' is the simplified grammer.

Elimination of Useless Productions:-

Definition of Useless Variable :-

A variable 'X' is useful if

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

otherwise 'X' is useless.

Definition of Useless Production :-

A production where the useless variables are present is called useless production.

Elimination Procedure :-

Given $G = (V, T, P, S)$

Define: $G' = (V', T', P', S)$ to be the CFG that does not contain any useless procedure such that $L(G') = L(G)$

Step 1: Initialise P' to P.

Step 2: Find useless variables:

(*) variable is useless if it cannot derive any sentence

(*) Variable is useless if if does not occur in any derivation for the string derivable from start ~~state~~ symbol S.

Q Eliminate useless production

$S \to aSb \mid a \mid aAB$

$A \to bB \mid c$.

Sol Given: $G = (\{S, A, B\}, \{a, b, c\}, P, S)$

$P \Rightarrow S \to aSb \mid a \mid aAB$

$A \to bB \mid c$

Define $G'(V', T', P', S)$ be the CFG that does not contain any useless production such that $L(G') = L(G)$

$P' =$

$S \to aSb \mid a \mid aAB$

$A \to bB \mid c$.

∵ B cannot derive any sentence

∴ B is useless.

$S \to aSb \mid a$

~~$A \to c$~~.

∵ A is not reachable from start symbol

A is useless.

$P' = S \to aSb \mid a$.

$G' = (\{S\}, \{a, b\}, P', S)$

Elimination of Unit Production:-

* Definition of Unit Production:-
  A production of form
  'A → B'
  where A & B are variables is called unit production.

* Definition of Non-Unit Production:-
  A production not of form 'A → B'. where A & B are variables is called non unit production.

Elimination Procedure:-
Given - G = (V, T, P, S)
Define G' = (V', T', P', S') be the CFG that does not contain any unit production such that L(G') = L(G).

Step 1:- Initialise P' to P.
Step 2:- Find unit production.
* Say 'A → B' is a unit production of A, then add to A, the non-unit production of B which are not present in A
Step 3:- Delete Unit production

Q. Eliminate unit production

$S \to aSb \mid a \mid A$

$A \to Ab \mid a \mid b$

Sol

production	new production
$A \to Ab$	—
$A \to a$	—
$A \to b$	—
$S \to A$	$S \to Ab, S \to b, S \to a$
$S \to a$	—
$S \to aSb$	—

Production + new production.

$S \to aSb \mid a \mid Ab \mid b$.

$A \to Ab \mid a \mid b$

After deleting unit production

$S \to aSb \mid a \mid Ab \mid b$.

$A \to Ab \mid a \mid b$

Q. Simplify / Reduce the given G.

$S \to aS \mid A \mid C$

$A \to a$

$B \to aa$

$C \to aCb$

Sol :- Step 1 :- Elimination of unit production

Page No.
DATE / /

Production	new production
$C \to aCb$	—
$B \to aa$	—
$A \to a$	—
$S \to C$	$S \to aCb$.
$S \to A$	$S \to a$
$S \to aS$	—

$S \to C$

$C \to \epsilon$

$S \to \epsilon$

Production to new production

$S \to aS \mid A \mid B \mid a \mid aCb$.

$A \to a$

$B \to aa$

$C \to aCb$

After eliminating unit prod.

$S \to aS \mid a \mid aCb$

$A \to a$

$B \to aa$

$C \to aCb$

$A \to B$

$B \to b$

Step 2: Elimination of useless production.

∵ C cannot derive any sentence

C is useless.

∵ A & B are not reachable from S

A & B are useless.

$S \to aS \mid a$.

Elimination of Null Productions

Definition of Null Production

A production of form

'$A \to \epsilon$' or '$A \to \lambda$'

where A is a variable is called Null production.

Definition of Nullable Variable :-

A variable 'X' is nullable if

'$X \to \epsilon$' or '$X \overset{*}{\Rightarrow} \epsilon$'.

Elimination Procedure :-

Given $G = (V, T, P, S)$

Define $G' = (V', T', P', S)$ be th CFG that does not contain any null productin such that $L(G') = L(G) - \{\epsilon\}$

Step 1 :- Initialise P' to P.

Step 2 :- Find Nullable variables.

* Say '$A \to \alpha$' is a prod. of A, where α contains some nullable variable. Then add to A, The prod. obtained by deleting all posible subset of nullable variables from α if not present.

Step 3 : Delete all Nullable productions.

Q. Eliminate Null production

$S \to aSb \mid a \mid AB$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

Sol

Production	New production
$B \to bB$	$B \to b$
$B \to \epsilon$	-
$A \to aA$	$A \to a$
$A \to \epsilon$	-
$S \to AB$	$S \to B, S \to A, S \to \epsilon$
$S \to a$	-
$S \to aSb$	$S \to ab$

Productions → new production

$S \to aSb \mid ab \mid a \mid AB \mid A \mid B \mid \cancel{\epsilon}$

$A \to aA \mid a \mid \cancel{\epsilon}$

$B \to bB \mid b \mid \cancel{\epsilon}$

After deleting Null productions

$S \to aSb \mid ab \mid a \mid AB \mid A \mid B$

$A \to aA \mid a$

$B \to bB \mid b$

$S \to ABCDCDA$

$A \to CD$

$B \to Cb$

$C \to a \mid \epsilon$

$D \to bD \mid \epsilon$

$S \to ABCBCDA$

$A \to CD$

$C \to a$

$C \to \epsilon$

$D \to bD$

$D \to \epsilon$

$B \to Cb$

$A \to \epsilon D$

$S \to ABCBCDA$

$C \to B$

ϵ

$D \to b$

-

$B \to b$

$A \to C, A \to D, A \to \epsilon$

$S \to BCBCD$

$S \to AB BDA$

$S \to ABCBCA$.

8 -

Normal Forms :-

Normal forms implies the rewritting of CFG as per given conditions.

Chomsky Normal Forms :-

Def:- Any ϵ free CFL can be generated by a CFG in which all the production are of the form "$A \to BC$" or "$A \to a$" where A, B, C are variables and a is terminal.

Such a CFG is said to be in CNF.

* Conversion Procedure from CFG to CNF

Step 1: Simplify the given CFG i.e elimination of Null, unit and useless productions.

Step 2:- Add to the sol. the production which are already in CNF.

Step 3:- For the remaining non-CNF prod.
* Replace the terminals by some variables
* List no. of variables on R.H.S to 2.

Q.1 Express CFG in CNF.

$S \to aSbb \mid bSa \mid aa \mid a \mid bb \mid b$

Sol:

Production	Sol.
$S \to a$	$S \to a$ ✓
$S \to b$	$S \to b$ ✓
$S \to aa$	$C_1 \to a$ ✓

Production	solution
(R) $S \to C_1C_1$	$S \to C_1C_1$
$S \to bb$	$C_2 \to b$
(R) $S \to C_2C_2$	$S \to C_2C_2$
$S \to bSa$	
(R) $S \to C_2SC_1$	$S \to C_2C_2$
(R) $C_3 \to SC_1$	$C_3 \to SC_1$
$S \to aSbb$	
(R) $S \to C_1SC_2C_2$	$S \to C_1C_4$
(R) $C_4 \to SC_2C_2$	$C_4 \to SC_5$
(R) $C_5 \to C_2C_2$	$C_5 \to C_2C_2$

CFG in CNF is

$S \to a \mid b \mid C_1C_1 \mid C_2C_2 \mid C_2C_3 \mid C_1C_4$

$C_1 \to a$

$C_2 \to b$

$C_3 \to SC_1$

$C_4 \to SC_5$

$C_5 \to C_2C_2$

Q.2 Express CFG in CNF

$S \to 11S \mid 1 \mid 00S \mid 0 \mid \lambda$

Sol:- Step I:- Elimination of Null Production

Production	new production
$\{S\} = \{\{S\}, \{\}\}$ $S \to 11S$	$S \to 11$
$S \to 1$	
$S \to 00S$	$S \to 00$
$S \to 0$	
$S \to \lambda$	

DATE

Production → new production.

$S \to 11S \mid 1 \mid 00S \mid 0 \mid 11 \mid 00 \mid \lambda$

After deleting null production.

$S \to 11S \mid 1 \mid 00S \mid 0 \mid 11 \mid 00$

Step 2 & 3 :-

Production	Solution.
$S \to 0$	$S \to 0$
$S \to 1$	$S \to 1$
$S \to 00$	$C_1 \to 0$
(R) $S \to C_1C_1$	$S \to C_1C_1$
$S \to 11$	$C_2 \to 1$
(R) $S \to C_2C_2$	$S \to C_2C_2$
$S \to 00S$	
(R) $S \to C_1C_1S$	$S \to C_1C_3$
(R) $C_3 \to C_1S$	$C_3 \to C_1S$
$S \to 11S$	
(R) $S \to C_2C_2S$	$S \to C_2C_4$
$C_4 \to C_2S$	$C_4 \to C_2S$

CFG in CNF is

$S \to 0 \mid 1 \mid C_1C_2 \mid C_1C_3 \mid C_2C_4$

$C_1 \to 0$

$C_2 \to 1$

$C_3 \to C_1S$

$C_4 \to C_2S$

Q.8 Express CFG in CNF

$S \to \sim S$

$S \to [S \, \& \, S]$

$S \to P$

$S \to q$

In Exam:- ① Write def. of CNF
② conversion Procedure.
③ Solve problem.
} For getting full marks.

Sol:- Step I:- Grammer is already in simplified form.

Step II:- Production	Solution.
$S \to P$	$S \to P$
$S \to q$	$S \to q$
$S \to \sim S$	$C_1 \to \sim$
(E) $S \to C_1 S$	$S \to C_1 S$
$S \to [S\ )\ S]$	$C_2 \to [, C_3 \to), C_4 \to]$
(R) $S \to C_2 S C_3 S C_4$	$S \to C_2 C_5$
(L) $C_5 \to S C_3 S C_4$	$C_5 \to S C_6$
(L) $C_6 \to C_3 S C_4$	$C_6 \to C_3 C_7$
(L) $C_7 \to S C_4$	$C_7 \to S C_4$

CFG in CNF is

$S \to p \mid q \mid C_1 S \mid C_2 C_5$

$C_1 \to \sim$

$C_2 \to [$

$C_3 \to)$

$C_4 \to]$

$C_5 \to S C_6$

$C_6 \to C_3 C_7$

$C_7 \to S C_4$.

Greibach Normal Theory (GNF)

Definition:- Any ϵ free CFG can be generated by a CFG in which all the productions are of the form

$A \to a\gamma$ & A_i

where $A \to$ variable

$a \to$ terminal

$\gamma \to$ string of variables. (V^*)

(can be empty)

Such a CFG is said to be in GNF

Conversion Procedure From CFG to GNF

Step I :- Simplify the given CFG

Step II :- Use any combination of Rule-I and Rule-2 to obtain CFG in GNF

Rule-1 :- Let $A \to B\alpha$ be some A production and

Let $B \to \beta_1 | \beta_2 | \cdots | \beta_n$ be the B-productions then we can write A productions as

$$A \to \beta_1\alpha | \beta_2\alpha | \cdots | \beta_n\alpha$$

Rule 2 :- Let $A \to A\alpha_1 | A\alpha_2 | \cdots | A\alpha_r$ be some A-production and Let $A \to \beta_1 | \beta_2 | \cdots | \beta_s$ be the remaining A-production. then introduce a new variable B. B-production, $B \to \alpha_i | \alpha_i B$ $1 \le i \le r$. A-production $A \to \beta_i | \beta_i B$ $1 \le i \le s$.

Q.. Express CFG in CNF

1.) $S \to AA | AB | BA$

$A \to aA | a$

$B \to bB | b$.

α implies the production that starts with the LHS variable.
β implies the production that do not starts with the LHS variable.

Sol:- $B \to bB \mid b$.
$A \to aA \mid a$.
$S \to BA \Rightarrow S \to bBA \mid bA$ ✓ (R_1)
$S \Rightarrow AB \Rightarrow S \to aAB \mid aB$ ✓ (R_1)
$S \to AA \Rightarrow S \to aAA \mid aA$

2.) $A_1 \to A_2A_2 \mid 0$.
$A_2 \to A_1A_1 \mid 1$

Sol:- $A_2 \to A_1A_1 \mid 1$
$A_2 \to A_2A_2A_1 \mid 0A_1 \mid 1$ (R_1)

$A \to A\alpha \mid \beta_1 \mid \beta_2$ (R_2)

Sol. $\beta \to \alpha_1 \mid \alpha_1 B$ $B \to A_2A_1 \mid A_2A_1B$
$A \to \beta_1 \mid \beta_1 B$ $A_2 \to 0A_1 \mid 0A_1B \mid 1 \mid 1B$ (R_2)

$A_1 \to 0A_1A_2 \mid 0A_1BA_2 \mid 1A_2 \mid 1BA_2 \mid 0$.

$B \to 0A_1A_1 \mid 0A_1BA_1 \mid 1A_1 \mid 1BA_1 \mid 0A_1A_1B \mid 0A_1BA_1B \mid 1A_1B \mid 1BA_1B$

Q.3

$S \to AA \mid a$
$A \to SS \mid b$

Sol:- $A \to SS \mid b$.
$A \to AAS \mid aS \mid b$. (R_1)
$B \to \alpha_1 \mid \alpha_1 B$ $B \to AS \mid ASB$
$A \to \beta_1 \mid \beta_1 B$ $A \to aS \mid aSB \mid b \mid bB$ (R_2)

$S \rightarrow aSA \mid aSBA \mid bA \mid bBA \mid a$ (R1)

$B = aSS \mid aSBS \mid bS \mid bBS \mid aSSB \mid aSBSB \mid bSB \mid bBSB$ (R1).

Q.4 Generate and Reduce the following to CNF to GNF

$L = \{ a^n b^n \}$.

$L = \{a^n b^n\}$. $n \geq 0$.

$= \{\epsilon, ab, aabb, aaabbb, aaaabbbb\}$

$n = 0; n = 1; n = 2, n = 3; n = 4$

Q.4 Generate and Reduce the following CNF to GNF.

$L = \{a^n b^n\}$.

Sol:- CFG: $S \rightarrow aSb \mid \epsilon$

Step I :- Elimination of Null production.

Production	New production
$S \rightarrow \epsilon$	—
$\{S\} = \{\{S\}, \{\}\}$ $S \rightarrow aSb$	$S \rightarrow ab$

Production + new production.

$S \rightarrow aSb \mid \epsilon \mid ab$.

After deleting Null production

$S \rightarrow aSb \mid ab$.

Page No.
DATE / /

Simplified CFG: $S \rightarrow asb \mid ab$

$(A \rightarrow aY)$

$(A \rightarrow BC,\ A \rightarrow a)$

CNF

$C_1 \rightarrow a$

$C_2 \rightarrow b$

$S \rightarrow C_1C_2$

$S \rightarrow C_1C_3$

$C_3 \rightarrow SC_2$

GNF

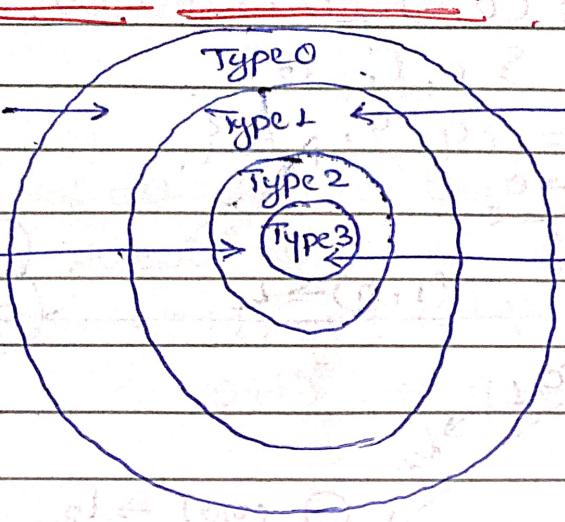
$S \rightarrow aSC_1$

$C_1 \rightarrow b$

$S \rightarrow aC_1$

Page No.
DATE / /

CHOMSKY'S HIERARCHY



1) FSM

Q. Design an FSM to check whether the given unary number is divisible by 4

I / S	0	
→ q_0^*	q_1	0
q_0^*	q_1	00000/4
q_1	q_2	
q_2	q_3	
q_3	q_0	

Q2. Design an FSM that compare two binary numbers to determine whether are equal or which is larger.

May 06
Dec 07/10M

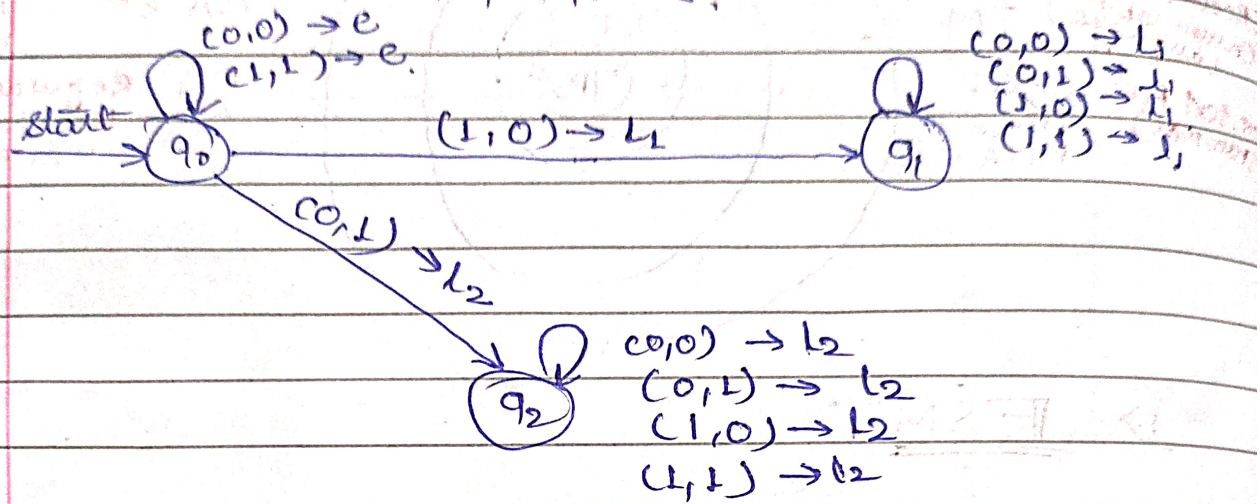
1011	1101	1011
1011	1011	1101
e e e (e)	e l_1 l_1 (l_1)	e l_2 l_2 (l_2)

Page No.
DATE / /

Sol:- $I = \{(0,0), (0,1), (1,0), (1,1)\}$

$O = \{e, l_1, l_2\}$

$S = \{q_0, q_1, q_2\}$



S \ I	(0,0)	(0,1)	(1,0)	(1,1)
→q_0	q_0	q_2	q_1	q_0
q_1	q_1	q_1	q_1	q_1
q_2	q_2	q_2	q_2	q_2

STF : $S \times I \to S$

S \ I	(0,0)	(0,1)	(1,0)	(1,1)
→q_0	e	l_2	l_1	e
q_1	l_1	l_1	l_1	l_1
q_2	l_2	l_2	l_2	l_2

MAF : $S \times I \to O$

Regular Expression

Dec 06 | 2M

* Set of all strings that contain both '11' & "00" as substring over $\Sigma = \{0,1\}$

$$r = (0+1)^* 11 (0+1)^* 00 (0+1)^* + (0+1)^* 00 (0+1)^* 11 (0+1)^*$$

May 06 | 2M

* Set of all strings that does not contain consecutive 0's over $\Sigma = \{0,1\}$.

$$r = (1+01)^* (0+\epsilon)$$

$$L(r) = \{\epsilon, 0, 1, 01, 11, 010, 10, \cdots\}$$

* Set of all strings that does not contain consecutive 1's over $\Sigma = \{0,1\}$.

$$r = (0+10)^* (1+\epsilon)$$

May 06 | 2M

* Set of all strings that every pair of consecutive (~~adjustment~~ (adjacent) 0's appear before any pair of consecutive 1's over $\Sigma = \{0,1\}$.

$$r = (0+10)^* (1+\epsilon) \cdot (1+01)^* (0+\epsilon)$$

If there is counting relation b/w a & b then for such language regular expression is not possible -

Note:- For non-regular languages, we cannot write regular expression neither can be design DFA.

* $L = \{a^3 b^2\}$
  $r = a^3 b^2$ ie aaabb.

* $L = \{ ab^i \mid i \geq 1 \}$
  $r = ab^+$

* $L = \{ a^i b \mid i \geq 1 \}$
  $r = a^+ b$.

* $L = \{ a^i b^i \mid i \geq 1 \}$

Sol:- Regular expression cannot be written for the given language because it is not a regular language. which can be proved by using pumping lemma for regular language.

Steps to prove that the language is not regular.

Step I:- Assume that the given language is regular.

Step II:- Make a selection of the string as per the given criteria

Step III:- As per pumping lemma. obtained the required equations.

Step IV:- Prove that your assumption is wrong.

PUMPING LEMMA FOR REGULAR LANGUAGE:-

Let L be a regular language & let z be a word of L such that $|z| \geq n$ where n is the min number of DFA states required for recognizing L. Then we can write $z = uvw$ where $|uv| \leq n$ and $1 \leq |v| \leq n$. such that all the strings of the form uv^iw for $i \geq 0$ would belong to L.

1.] Prove using pumping lemma.

$L = \{a^i b^i \mid i \geq 1\}$ is not regular.

Proof :-

Step I :- Let L be a regular language

Step II :- Let $z = a^n b^n$ $|z| \geq n$ where n is pumping lemma. constant. (pumping length)

&

Step III :- Then as per pumping lemma we can write $z = uvw$ where $|uv| \leq n$ and $1 \leq |v| \leq n$. such that all the strings of the form uv^iw for $i \geq 0$ would belong to L.

$i = 2$ $\quad |uv^2w| = |uvw| + |v|$

$1 \leq |v| \leq n$

Add $|uvw|$ on all the sides.

$$1 + 2n \leq |uv^2w| \leq n + 2n.$$

$$\boxed{2n < |uv^2w| < 3n+1} \quad n \geq 1$$

Step IV :-

$n = 1$

$a^i b^i$

$2 < |uv^2w| < 4$

$i = 1 ; |ab| = 2$

$i = 2 ; |aabb| = 4$

$i = 3 ; |aaabbb| = 6$

$\because$ L cannot contain any string of length 3

$\therefore$ L is not regular.

2.] Prove using Pumping Lemma.

$L = \{a^i b^{2i} \mid i \geq 1\}$ is not regular.

Proof :-

Step I :- Let L be a Regular language.

Step II :- Let $z = a^n b^{2n}$ $|z| \geq n$.

where n is pumping lemma constant

$|a^n b^{2n}| = 3n$

Step III :- Then as per pumping lemma, we can write,

$z = uvw$

where $|uv| \leq n$ & $1 \leq |v| \leq n$

Such that all the string of the form uv^iw for $i \geq 0$ would belong to L.

$i = 2$. $|uv^2w| = |uvw| + |v|$

$1 \leq |v| \leq n$. — (1) $|uvw| = i$

$1 + |uvw| \leq |uvw| + |v| \leq n + |uvw|$

Add $|uvw|$ on all the sides ie $3n$

$1 + 3n \leq |uv^2w| \leq n + 3n$.

$\boxed{3n < |uv^2w| \leq 4n + 1}$ $n \geq 1$

Step IV :-

$a^i b^{2i}$ $n = 1$.

$i = 1$; $|abb| = 3$.

$i = 2$; $|aabbbb| = 6$

$i = 3$; $|aaabbbbbb| = 9$.

$3 < |uv^2w| \leq 5$

$\therefore$ L cannot contain any string of length 4.

$\therefore$ L is not regular.

Finite Automata (F.A.)

- FA with final state & no. output
  - DFA
  - NFA
- FA with o/p and no final state.
  - Moore machine
  - Mealy Machine

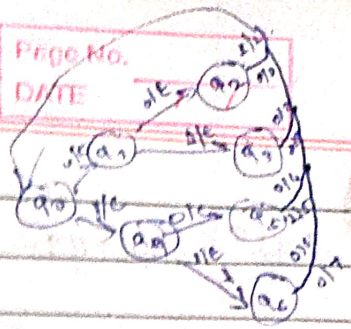
octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

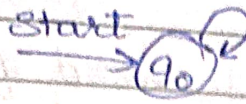
Q. Design Mealy machine to convert
1. Octal to binary
2. Binary to octal
3. Hexa to binary
4. Binary to hexa
5. Octal to Hexa [① O2B ② B2H
6. Hexa to octal [① H2B ② B2O

Sol

Octal → Binary

2	7	4
010	111	100



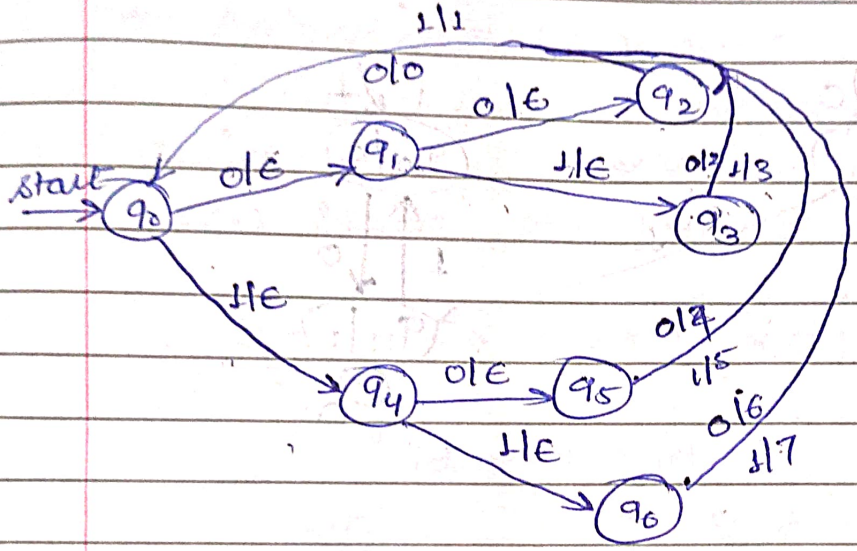


Octal to Binary

Binary → octal

010 101 110

2 5 0

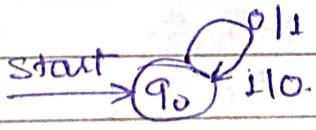


Binary to octal

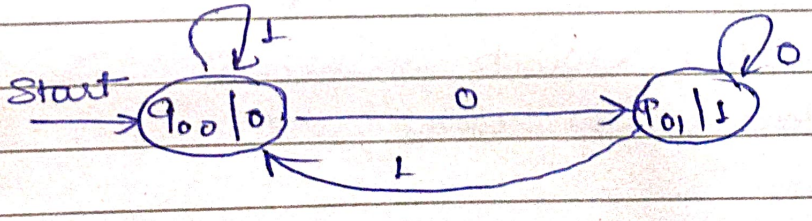
Q. Design Moore & Mealy machine to find
(1) 1's complement of binary number.
(2) 2's complement of binary number.

Sol:-

(1.)



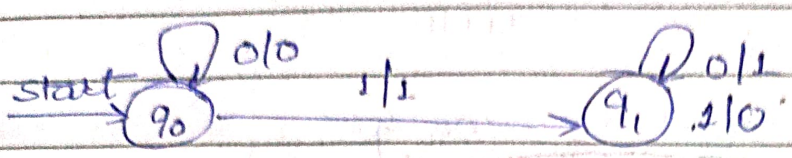
mealy machine



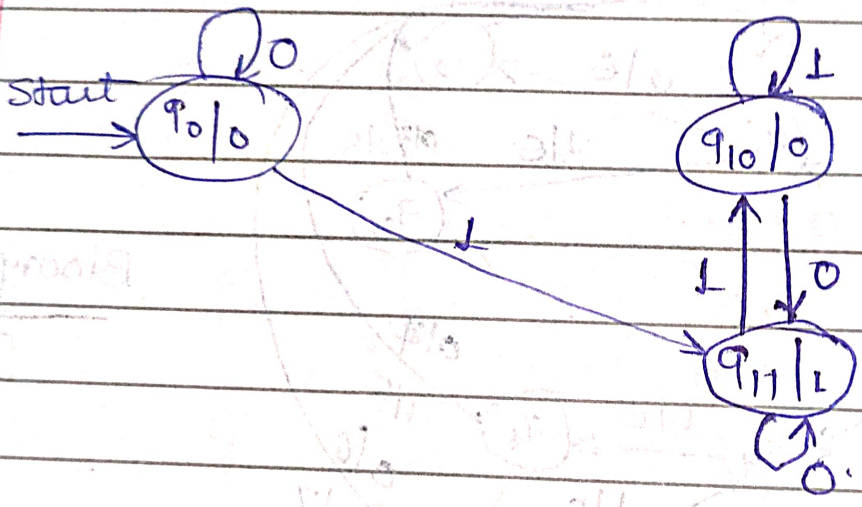
Moore Machine

(2) Note:- I/p would be scanned from R to L

10110
01010
2's



Mealy Machine



Grammers

Ambiguous & Unambiguous Grammer :-

* **Unambiguous Grammer**

Grammer is unambiguous if it can derive all the sentences using exactly one LMD or RMD.

* **Ambiguous Grammer**

Grammer is ambiguous if it can derive atleast one sentence using more than one LMD or RMD.

eg. $E \rightarrow E+E \mid E*E \mid id$.

"id + id * id"

LMD :- $E \Rightarrow E+E$
$\Rightarrow id+E$
$\Rightarrow id+E*E$
$\Rightarrow id+id*E$
$\Rightarrow id+id*id$

LMD :- $E \Rightarrow E*E$
$\Rightarrow E+E*E$
$\Rightarrow id+E*E$
$\Rightarrow id+id*E$
$\Rightarrow id+id*id$

∵ G can derive a sentence using more than one LMD, G is ambiguous.

Ex. Unambiguous Grammer

$E \rightarrow E+T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow id$

"id + id * id"

LMD :

$E \Rightarrow E+T$

$\Rightarrow T+T$

$\Rightarrow F+T$

$\Rightarrow id+T$

$\Rightarrow id+T*F$

$\Rightarrow id+F*F$

$\Rightarrow id+id*F$

$\Rightarrow id+id*id$

For CFL we can write CFG but neither can design PDA

Pumping Lemma for CFL

Let L be a CFL and let z be a word of L such that $|z| \geq n$. where n is pumping lemma constant then we can write:

$$z = uvwxy$$

where $|vwx| \leq n$ & $1 \leq |vx| \leq n$.

Such that all the strings of the form uv^iwx^iy for $i \geq 0$ would belong to L.

Q: Prove using pumping lemma.

$L = \{a^i b^i c^i \mid i \geq 1\}$ is not CFL.

Sol

Step I:- Let L be a CFL

Step II:- Let $z = a^n b^n c^n$ $|z| \geq n$ where n is pumping lemma constant.

Step III:- Then, as per pumping lemma we can write $z = uvwxy$., where $|vwx| \leq n$ & $1 \leq |vx| \leq n$, such that all the strings of the form uv^iwx^iy for $i \geq 0$ would belong to L.

$i = 2$ $|uv^2wx^2y| = |uvwxy| + |vx|$.

$1 \leq |vx| \leq n$

Add $|uvwxy|$ on all the sides

$$1 + 3n \leq |uv^2wx^2y| \leq n + 3n.$$

$$\boxed{3n < |uv^2wx^2y| < 4n+1.} \quad n \geq 1$$

Step 4:-

$i=1$; $|abc| = 3$

$i=2$; $|aabbcc| = 6$.

$i=3$; $|aaabbbccc| = 9$

$n=1$

$3 < |uv^2wx^2y| < 5$

∴ L cannot contain any string of length 4

L is not a CFG.

Page No.
DATE / /

DFA / Direct DFA

Q. Design DFA for recognising a string that start & ends with a different letter $\Sigma = \{x, y, z\}$

Q \ Σ	x	y	z	
→ q_5	q_0	q_4	q_8	
q_0	q_0	q_1	q_2	
q_1	q_0	q_1	q_2	q_0
q_2	q_0	q_1	q_2	
q_3	q_3	q_4	q_5	
q_4	q_3	q_4	q_5	q_3
q_5	q_3	q_4	q_5	
q_6	q_6	q_7	q_8	
q_7	q_6	q_7	q_8	q_8
q_8	q_6	q_7	q_8	