LP Mishra.
Date...../...../.....

Page.....01.....

Applications of Automata :-

→ Mathematical model of computation to abstract away the specifics of a particular problem.

→ String and regular expression arising in computer language.

→ Compiler specification.

→ Timed automation can be used to model certain real time systems.

→ Modelling systollic arrays → Cellular automata.

→ Natural language processing.

→ Compiler development
identify keywords, identifiers & all other programming constructs.

→ text processing.

→ Hardware design.

Q1. W.A.L. consisting of all strings starting with 4 and any no. of 6.

$$L = \{4.6^n \mid n \geq 1\}$$

Q1. Starting with ab and any no of c

$$L = \{ ab.c^n \mid n \geq 1 \}$$

Q1. Consisting of any no. of p and q and in the end a single r.

$$L = \{ p^n q^m . r \mid n, m \geq 1 \}$$

Q1. Staring with ab and ending with even multiples.

$$L = \{ ab.2^n \mid n \geq 1 \}$$

Q1. Starting with same no of pqr and ending with same no. of abc.

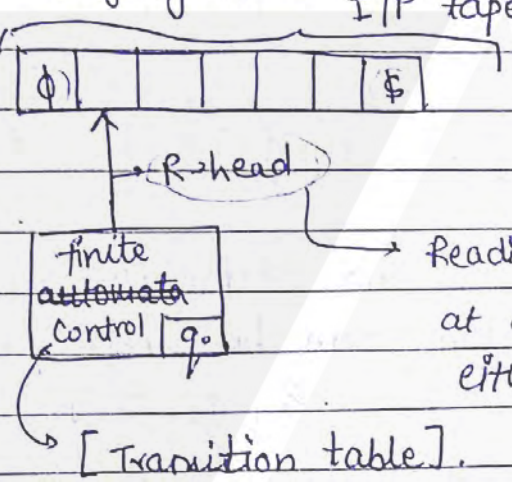
$$L = \{ (pqr)^n . (abc)^m \mid n, m \geq 1 \}$$

Date...../...../.....

Page.....03.....

Finite automata :-

It is a mathematical model of computers and algorithms or it is a powerful model of computation for accepting & generating languages.



Input tape :-

It is divided into squares, each square containing a single symbol from the input alphabet Σ. The first & last square indicates start & end of the tape. Absence of these indicate that tape is of infinite length.

R-head (reading head) :-

It examines only one square at a time & can move one square either to left or right.

Finite Control :-

The input to the finite control will be usually symbol under R-head say a and the present state of the machine q and will give the following:-

(i) movement of the read head along the tape to the next square.

(ii) the next state of the machine.

$$\delta(q, a) \vdash q_n$$

state input next state

$(q_3, b) \vdash (q_4, L)$ → moves left the R-head

formal definition of finite automata :-

A finite automaton can be represented by 5 tuples $F.A = \{Q, \delta, \Sigma, q_0, F\}$

where

Q is the non-empty finite set of states

Σ, it is a finite non-empty set of inputs.

q_0, belongs to Q & it is the initial state.

F set of final states. and $(F \subseteq Q)$

δ, it is a transition fun which maps $(Q \times \Sigma$ into $Q)$

state input next state

Representation of finite automata :→

Transition graph :

It is a collection of following

(i) Finite set of states including initial state and the end states. The state will be represented by a bubble ◯ with a free flowing arrow from the external environment.

→ (State)

* initial state will always be one for finite automata.

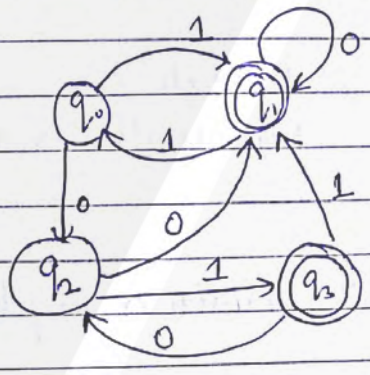
Date...../...../.....

Page.....05.....

State final state

(ii) Σ → possible inputs

(iii) A finite set of transitions that show how to go from one state to another state.



w = 1000101 (accepted)

w = 10001010 (not acc.) (does not reach the end state)

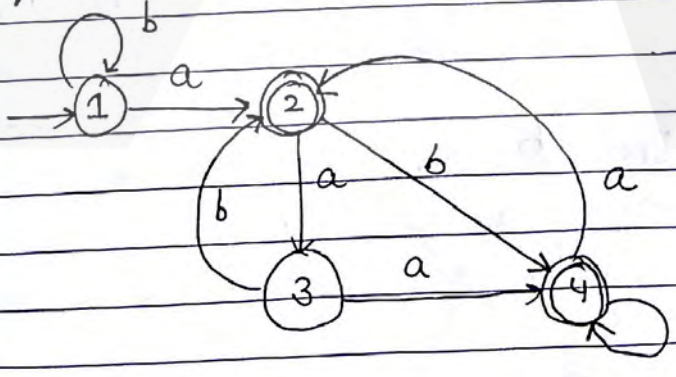
Transition table :⇒

It is a tabular representation of a transition function

δ	0	1
→ q_0	q_2	q_1
(*) q_1	q_1	q_0
(*) q_2	q_1	q_3
q_3	q_2	q_1

final set

Q.



δ	a	b
1	2	1
* 2	3	4
3	4	2
* 4	2	4

Transition system :-
A transition system can be defined as 5 tuples
T.S. $= \{Q, \delta, \Sigma, Q_0, F\}$

$Q_0 \rightarrow$ is a subset of Q and it is the set of intial States.

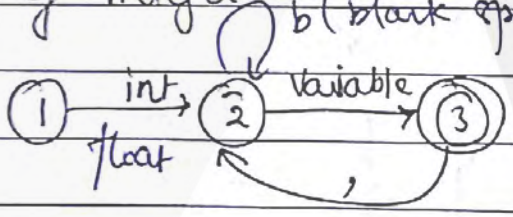
$F \rightarrow$ non-empty set of final State

$\delta \rightarrow$ it is a transition fun. which maps $Q \times \Sigma^* \text{ into } Q$

Q. diff. b/w following
Finite automata and Transition system.

Acceptability of a string using finite automata.
A string w is accepted by finite automata.
$M = \{Q, \Sigma, \delta, q_0, F\}$ if and only if δ of q_0, w
one or more than one transition
$\delta(q_0, w) \overset{*}{\vdash} q$ for some $q \in F$

declaration of integer



int a, b
accepted.

black one
int a, b
↳ not accepted because space

w_1 = float ---a,---b
accepted

Date...../...../.....

Page.....(01)

DFA NDFA or NFA (padh kar aana)

Types of finite automata

(i) DFA :- Deterministic finite automata.

(ii) NDFA or NFA :- Non deterministic finite automata.

DFA :-

In DFA, for each input symbol, one can determine the state to which the machine will move.

formal definition :-

A DFA can be represented by 5-tuple $(Q, \Sigma, \delta, q_0, F)$

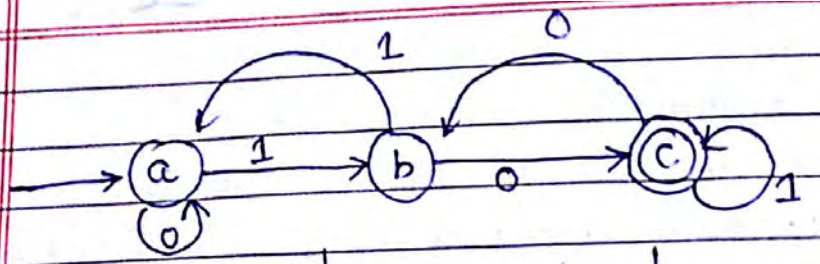
Q is a finite set of states.
Σ is a finite set of symbol called inputs
δ is a transition fun. where $\delta: Q \times \Sigma \to Q$
q_0 is the intial state. $(q_0 \subseteq Q)$
F is the final state / states of Q $(F \subseteq Q)$

Graphical representation by digraphs called state diagram

→ The vertices represents the states.
→ The arcs labelled with an input alphabet show the transition.
→ The intial ~~set~~ state is denoted by an empty single incoming arc.
→ The final state is indicated by double circles.

Example :-

$Q = \{a, b, c\}$ $F = \{c\}$
$\Sigma = \{0, 1\}$
$q_0 = \{a\}$



Present State	Input 0	Input 1
a	a	b
b	c	a
c	b	c

NDFA :→

In NDFA, for a particular input symbol, the machine can move to any combination of states in the machine. In other words, the exact state to which the machine moves cannot be determined. Hence, it is called Non-deterministic Automation. As it has finite no. of states, the machine is called Non-deterministic finite Machine

formal definition :→

As NDFA can be represented by a 5-tuples $(Q, \Sigma, \delta, q_0, F)$ where :→

→ Q is a finite set of states.

→ Σ is a finite set of symbols called the alphabets.

→ δ is the transition fun. where $\delta: Q \times \{\Sigma \cup \varepsilon\} \to 2^Q$

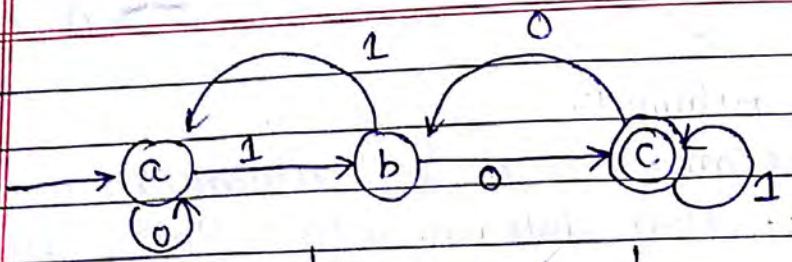
Here the power set of Q (2^Q) has been taken because in case of NDFA, from a state, transition can occur to any combination of Q states).

→ q_0 is the initial state where any input is processed ($q_0 \in Q$)

→ F is the final state / states of Q ($F \subseteq Q$).

Date...../...../.....

Page.....08.....



Present State	Input 0	Input 1
a	a	b
b	c	a
c	b	c

NDFA :→

In NDFA, for a particular input symbol, the machine can move to any combination of states in the machine. In other words, the exact state to which the machine moves cannot be determined. Hence, it is called Non-deterministic Automation. As it has finite no. of states, the machine is called Non-deterministic finite Machine.

formal definition :→

An NDFA can be represented by a 5-tuples $(Q, \Sigma, \delta, q_0, F)$ where :→

→ Q is a finite set of states.

→ Σ is a finite set of symbols called the alphabets.

→ δ is the transition fun. where $\delta: Q \times \{\Sigma \cup \varepsilon\} \to 2^Q$

Here the power set of Q (2^Q) has been taken because in case of NDFA, from a state, transition can occur to any combination of Q states).

→ q_0 is the initial state where any input is processed ($q_0 \in Q$)

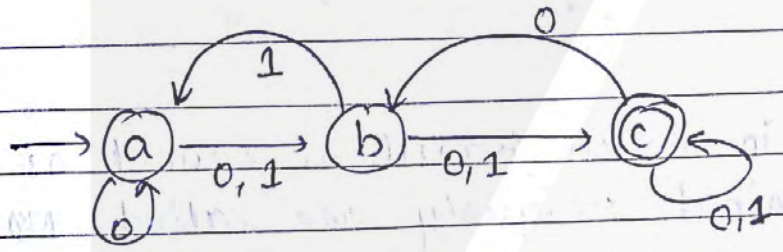
→ F is the final state / states of Q ($F \subseteq Q$).

Graphical representation :=

→ same as DFA

Example :→

$Q = \{a, b, c\}$
$\Sigma = \{0, 1\}$
$q_0 = \{a\}$
$F = \{c\}$



Present State	Input 0	Input 1
a	a, b	b
b	c	a, c
c	b, c	c

DFA	NDFA
·) Backtracking is allowed.	·) Backtracking is not allowed.
·) Requires more space.	·) Requires more space.
·) A string is accepted by DFA, if it transits to a final state.	·) A string is accepted by a NDFA, if at least one of the possible transition ends in a final state.

FA

DFA NDFA

DFA :→
whether a move is a single state.
whenever.

NFA :→
whenever move is not a single state.

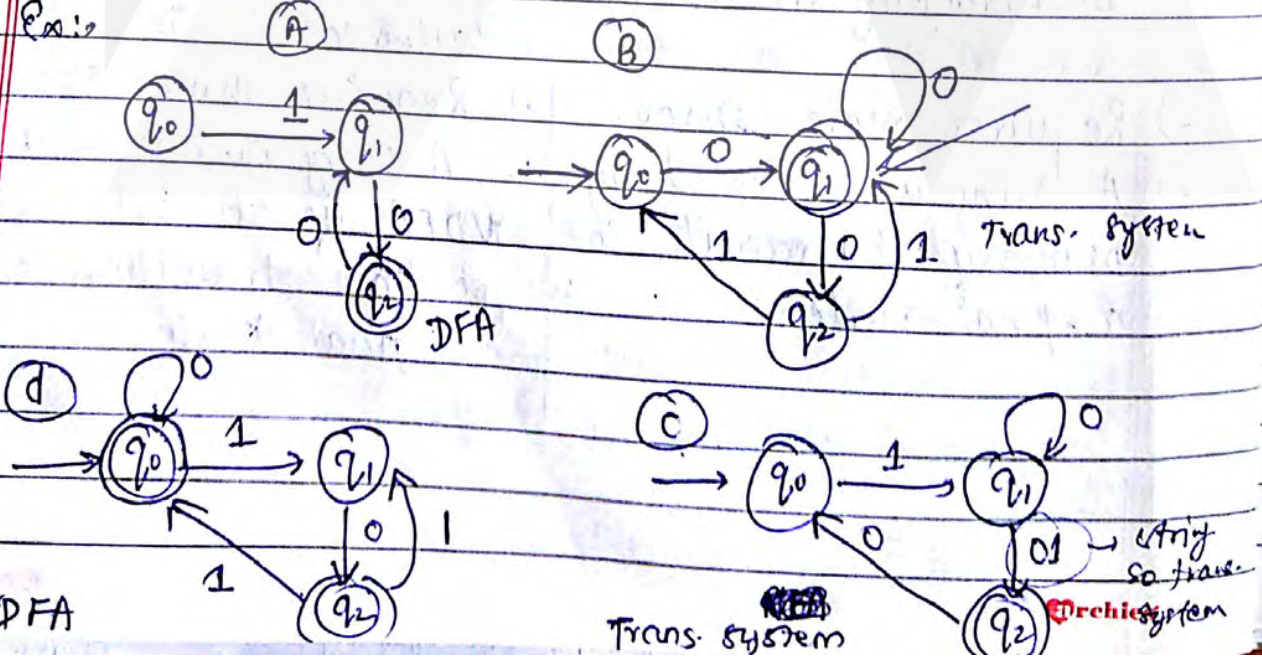
NDFA :→
The machines in which transitions cannot be defined determined uniquely are called NDFA.
Formally a non-deterministic FA can be defined as 5 tuples.

NFA = $\{Q, \delta, \Sigma, q_0, F)\}$

where
Q =
Σ =
δ = It is a transition fun. which maps

$$Q \times \Sigma \xrightarrow[\text{into}]{} 2^Q$$

Ex:→

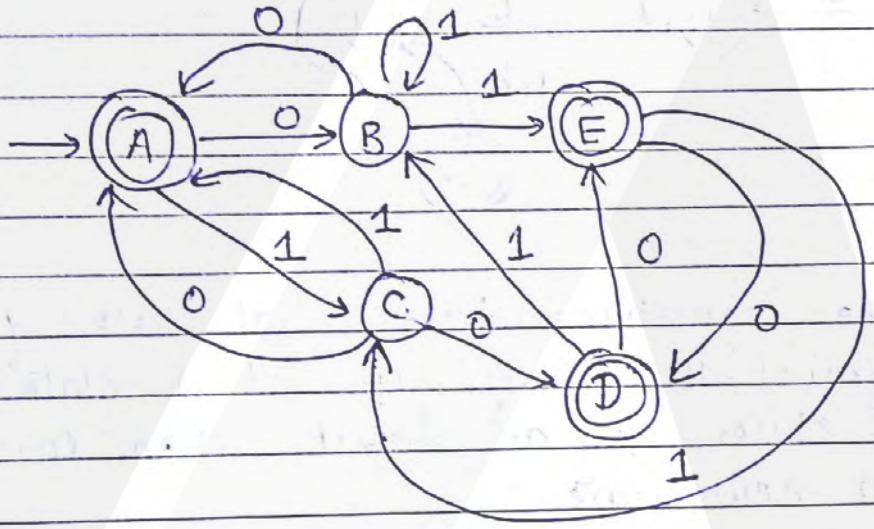


Date...../...../.....

Page.....11.....

Present state	Input 0	Input 1
→ * q_0	q_0, q_1	q_1
q_1	q_2	-
q_2	-	q_1, q_0

		0	1
→	* A	B	C
	B	A	{E, B}
	C	{D, A}	A
*	D	E	B
*	E	D	C



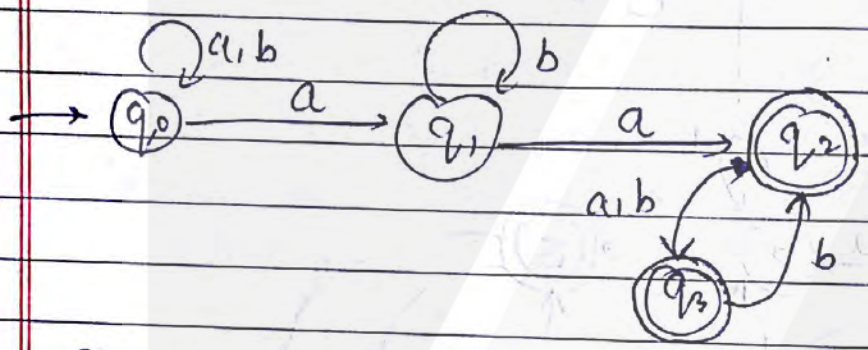
Transformation of NFA to DFA :⇒

1) For every NFA there exists a DFA which simulates the behaviour of NFA or if a language L is accepted by NFA then there exists a DFA which also accepts L. While converting NFA to DFA language acceptance power (LAP) of NFA should not be disturbed.

LAP means that if a string w is accepted by NFA than its corresponding DFA must accept w.

The method of converting NFA to DFA is also called subset construction because every state of a DFA will be represented by some sub set of states of NFA.

Convert the following NFA to DFA :→



Step 1 :→

1). See all the transitions from intial state q_0 for every symbol in Σ if you get a state or a set of states for an input then consider that as a new state.

2) In Step 1 if we get a new state than repeat Step 1 for every new state.

3). Repeat Step 2 till you are getting any new state.

All those states which contains atleast one accepting state of given NFA as a member state will be considered as final state of DFA.

Date...../...../.....

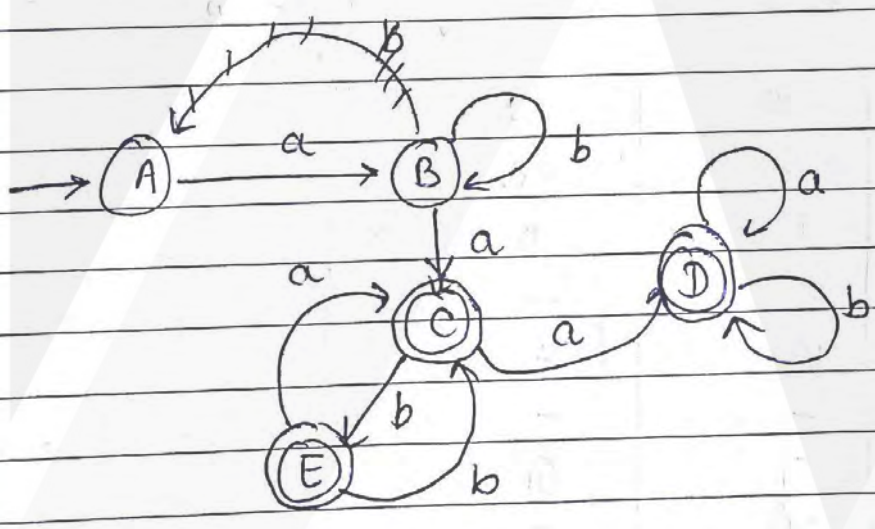
Page.....13.....

→ initial state

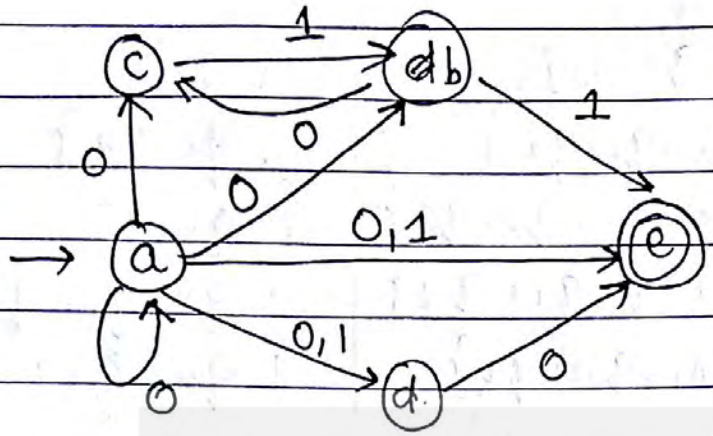
P.S.	a	b
→ {q_0} (A)	{q_0, q_1}	{q_0}
{q_0, q_1} (B)	{q_0, q_1, q_2}	{q_0, q_1}
* {q_0, q_1, q_2} (C)	{q_0, q_1, q_2, q_3}	{q_0, q_1, q_3}
* {q_0, q_1, q_2, q_3} (D)	{q_0, q_1, q_2, q_3}	{q_0, q_1, q_2, q_3}
* {q_0, q_1, q_3} (E)	{q_0, q_1, q_2}	{q_0, q_1, q_2}

↳ states inc. q_2 & q_3

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

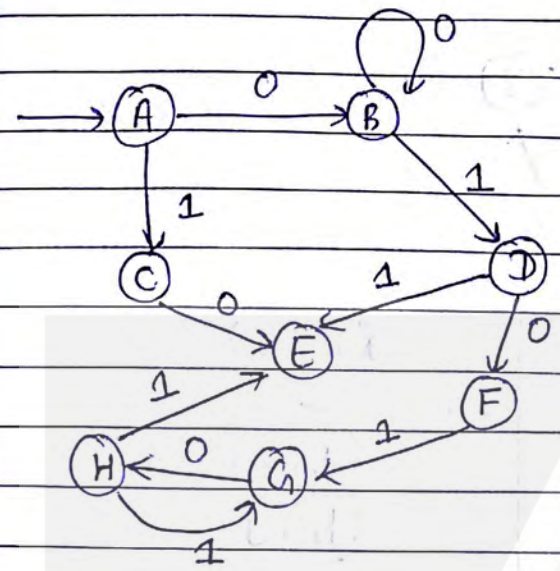


H.W :→ 5 conversion examples NFA to DFA.

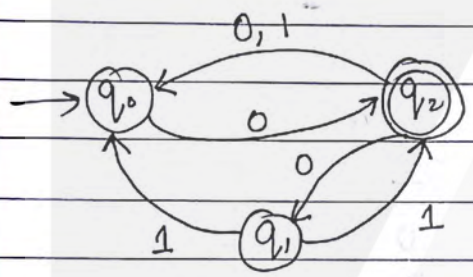


P.S.		$\delta(q,0)$	$\delta(q,1)$
a		$\{a,b,c,d,e\}$	$\{d,e\}$
$\{a,b,c,d,e\}$	(B)	$\{a,b,c,d,e\}$	$\{d,e,b,\}$
$\{d,e\}$	(C)	$\{e\}$	$\{-\}$
$\{b,d,e\}$	(D)	$\{c,e,\}$	$\{e\}$
$\{e\}$	(E)	–	–
$\{c,e\}$	(F)	–	$\{b\}$
b	(G)	c	e
c	(H)	–	b

P.S.	0	1
→ A	B	C
* B	B	D
* C	E	–
* D	F	E
E	–	–
* F	–	G
G	H	E
H	–	G

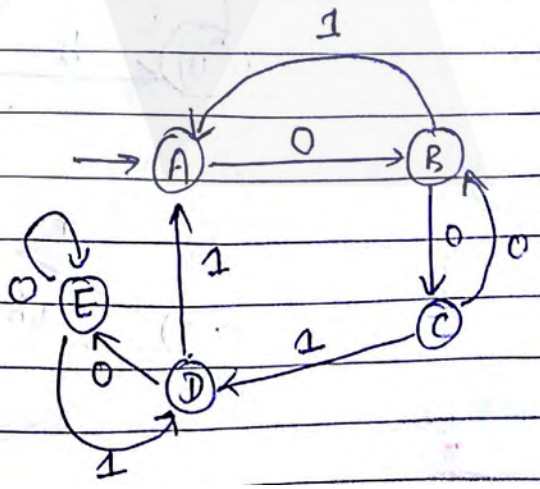


Q.

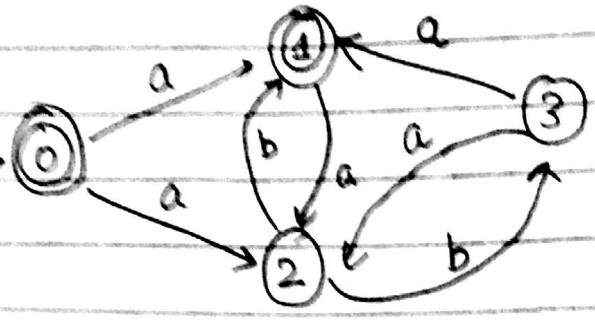


	P.S.	$\delta(q,0)$	$\delta(q,1)$
(A) →	q_0	q_2	—
(B) *	~~q_1~~ q_2	$\{q_0, q_1\}$	q_0,
(C)	~~q_2~~ $\{q_0 q_1\}$	q_2, q_0	$\{q_0, q_2\}$
(D) *	$\{q_0, q_2\}$	$\{q_2, q_1\}$	q_0
(E) *	$\{q_1, q_2\}$	$\{q_0, q_1\}$ q_2	$\{q_0, q_2\}$

	P.S.	$\delta(q,0)$	$\delta(q,1)$
→	A	B	—
*	B	C	A
	C	B	D
*	D	E	A
*	E	E	D.

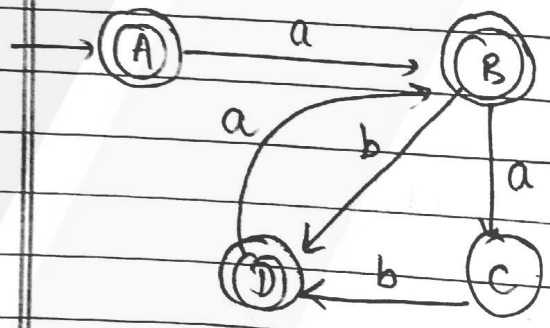


3.

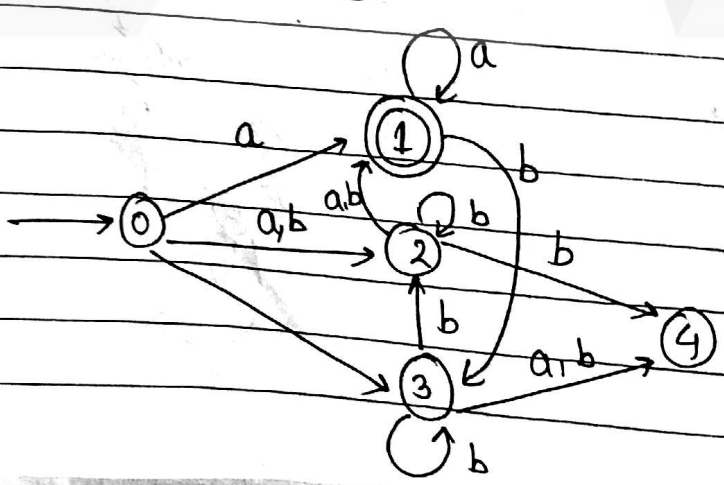


	P.S		$\delta(q,a)$	$\delta(q,b)$
→ *	0	(A)	{1,2}	–
*	{1,2}	(B)	Q	{1,3}
	2	(C)	–	{1,3}
*	{1,3}	(D)	{1,2}	{-

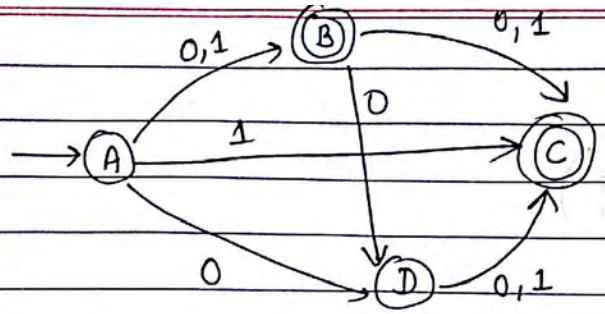
	P.S	a	b
→ *	A	B	–
*	B	C	D
	C	–	D
*	D	B	–



4.

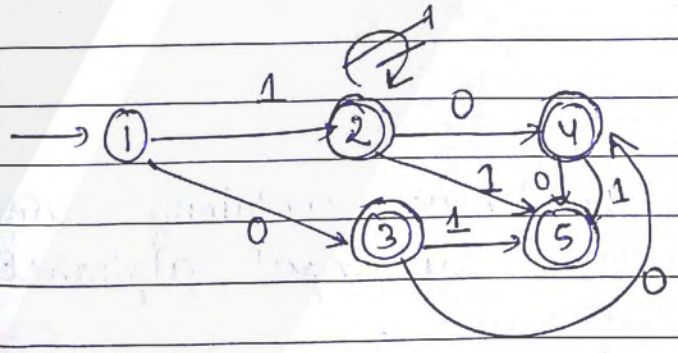


(4.)

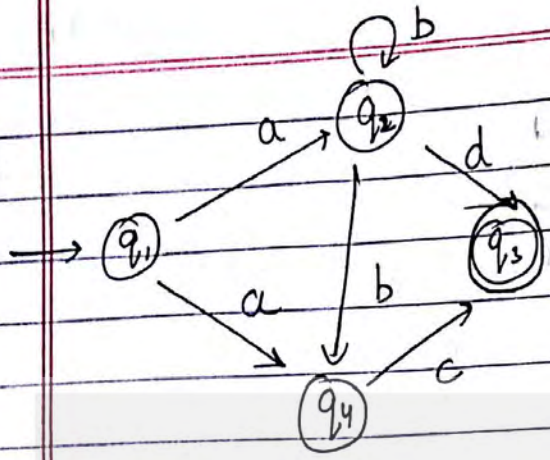


	P.S.	0	1
→	A (1)	{B, D}	{B, C}
*	{B, C} (2)	{C, D}	C,
*	{B, D} (3)	{C, D}	C
*	{C, D} (4)	C	C
*	{C} (5)	–	–

	P.S	0	1
→	1	3	2
*	2	4	5
*	3	4	5
*	4	5	5
*	5	–	–

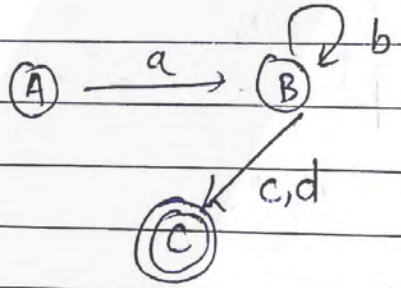


(3)



Q\∈	a	b	c	d
→ q_1 (A)	q_2, q_4	–	–	–
$\{q_2, q_4\}$ (B)	–	q_2, q_4	q_3	q_3
* q_3 (C)	–	–	–	–

Q\∈	a	b	c	d
→ A	B	–	–	–
B	–	B	C	C
* C	–	–	–	–



6|01.

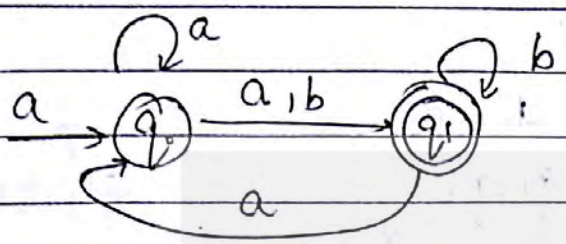
Design a DFA for the language containing strings starting with (001) over the input alphabet $\Sigma = \{0, a\}$

For the language containg strings ending with (01) over the $\Sigma = (0, 1)$

Date...../...../.....

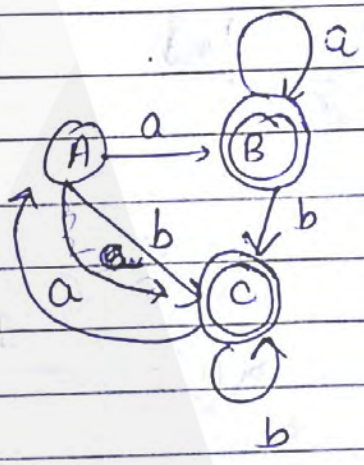
Page 19

Lang. containg string such that every string contain abb as a substrate. $\Sigma = (a, b)$

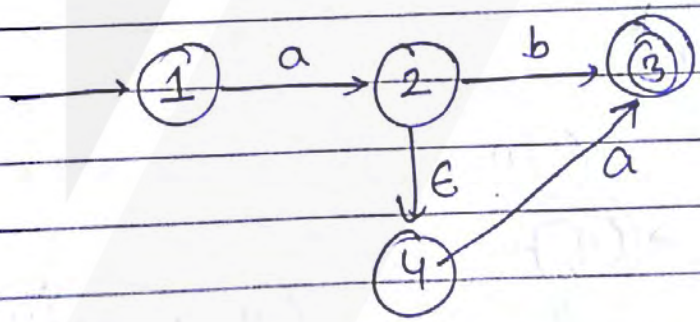


	a	b
q_0 (A)	$\{q_0, q_1\}$	q_1
$\{q_0, q_1\}$ (B)	$\{q_0, q_1\}$	q_1
$\{q_1\}$ (C)	q_0,	q_1

	a	b
A	B	C
B	B	C
C	A	C



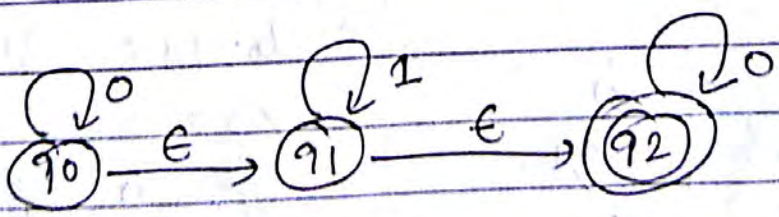
NFA with ϵ to NFA without ϵ (DFA)



$w\epsilon = \epsilon w = w$

1 with a → (4) (2)

$a\epsilon = \epsilon a = a$



ϵ closure (q_0) : $\{q_0, q_1, q_2\}$

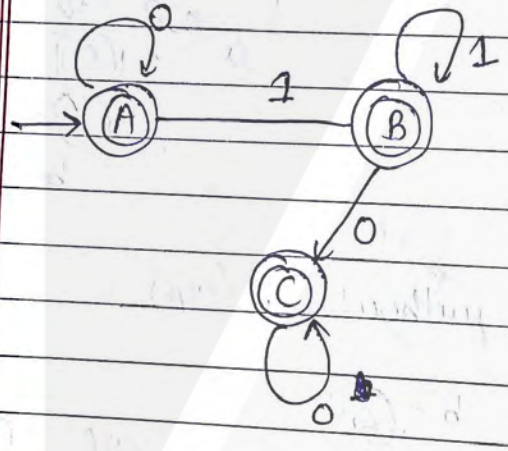
Never goby to be empty set

ϵ closure $(q_1) = \{q_1, q_2\}$

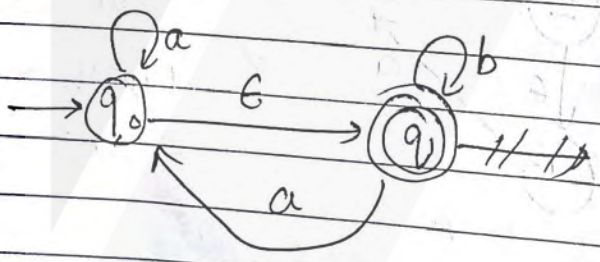
ϵ closure $(q_2) = \{q_2\}$

$\{\epsilon\text{clo}\, q_0 \cup \epsilon\text{clo}\, q_1\}$

		0	1
→ *	$\{q_0, q_1, q_2\}$ (A)	$q_0 q_1 q_2$ (q_0, q_1, q_2) A	$\{q_1, q_2\}$ B
*	$\{q_1, q_2\}$ (B)	q_2 C	$\{q_1, q_2\}$ B
*	$\{q_2\}$ (C)	q_2 C	–



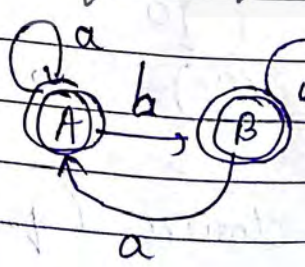
Q1.



$\epsilon\text{clo}\, q_0 = \{q_0, q_1\}$

$\epsilon\text{clo}\, q_1 = q_1$

	a	b
→ * A $\{q_0, q_1\}$	$\{q_0, q_1\}$	q_1
* B $\{q_1\}$	$\{q_0, q_1\}$ A	q_1 B
	$\{q_0, q_1\}$ A	q_1 B

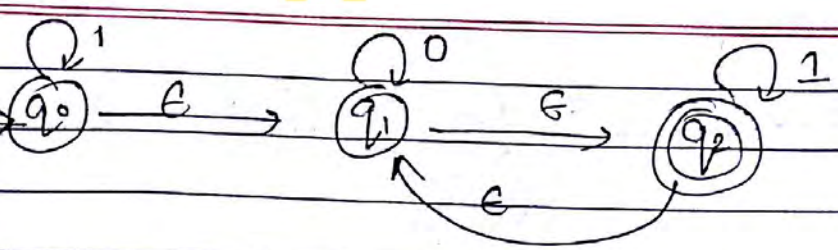


Date...../...../.....

Page.....

Q1.

(1)



ϵ clos q_0 = $\{q_0, q_1, q_2\}$

ϵ clos q_1 = $\{q_1, q_2\}$

ϵ clos q_2 = $\{q_2, q_1\}$

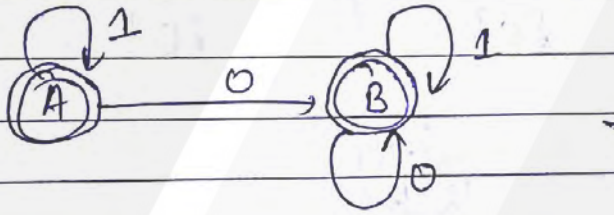
	0	1
→ * $\{q_0, q_1, q_2\}$ (A)	$\{q_1, q_2\}$ (B)	$\{q_0, q_1, q_2\}$ (A)
* $\{q_1, q_2\}$ (B)	$\{q_1, q_2\}$ (B)	$\{q_1, q_2\}$ (B)
~~$\{q_0, q_1, q_2\}$~~		

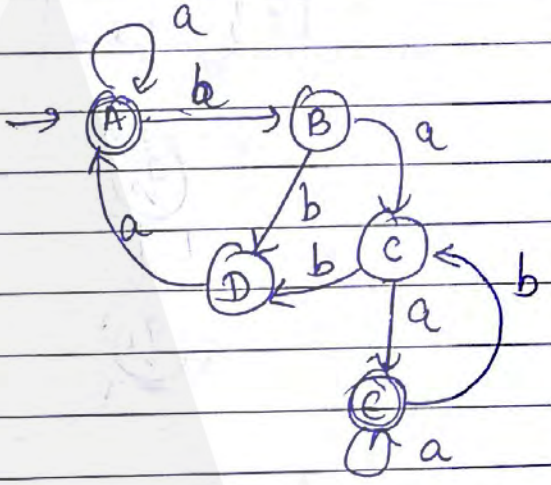
q_1

q_1

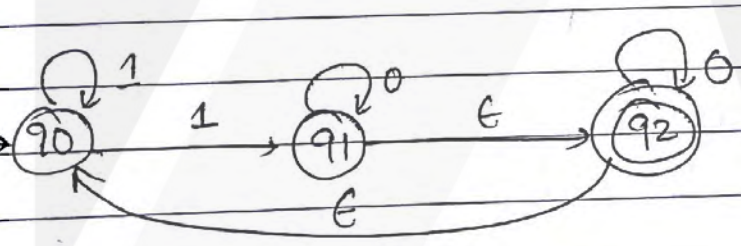
$\{q_0, q_2\}$

q_1





(2)



ϵ clos q_0 = $\{q_0\}$

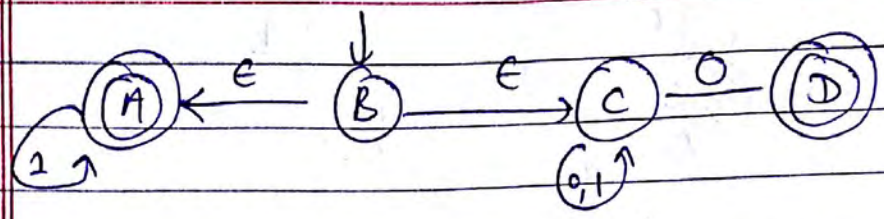
ϵ clos q_1 = $\{q_1, q_2, q_0\}$

ϵ clos q_2 = $\{q_2, q_0\}$

$\{q_0, q_1\}$

	0	1
q_0	-	$\{q_0, q_1, q_2\}$
$\{q_0, q_1, q_2\}$		

(2).



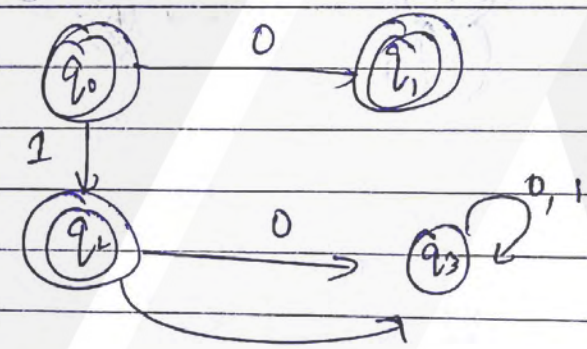
ϵ closure (B) = {B, C, A}

(A) = {A}

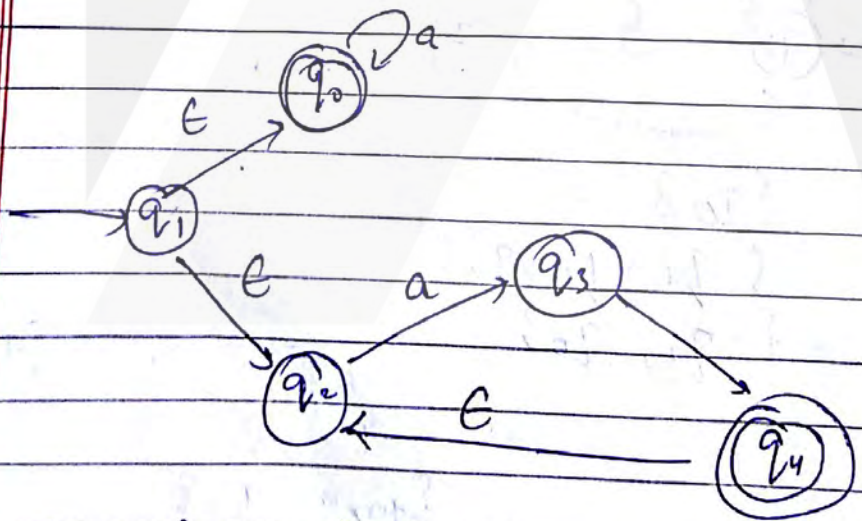
(C) = {C}

(D) = {D}

		0	1
→ *	{A, B, C} q_0	{D}	{A, C}
*	{D} q_1	–	–
*	{A, C} q_2	{C}	{A, C}
	{C} q_3	{C}	{C}



Q.3.



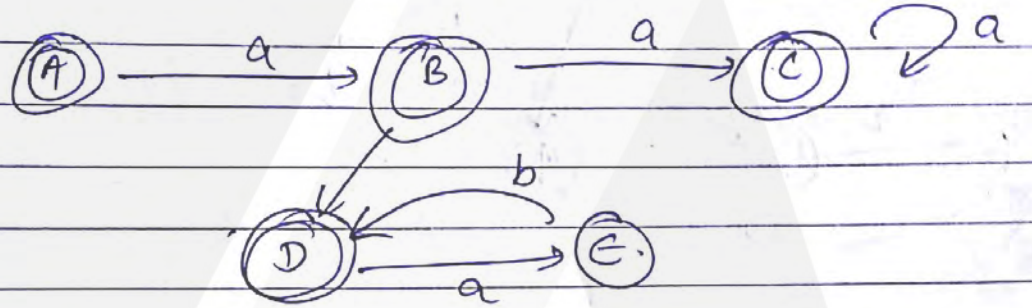
ϵ (q_0) : {q_0}

→ ϵ (q_1) = {q_1, q_0, q_2}

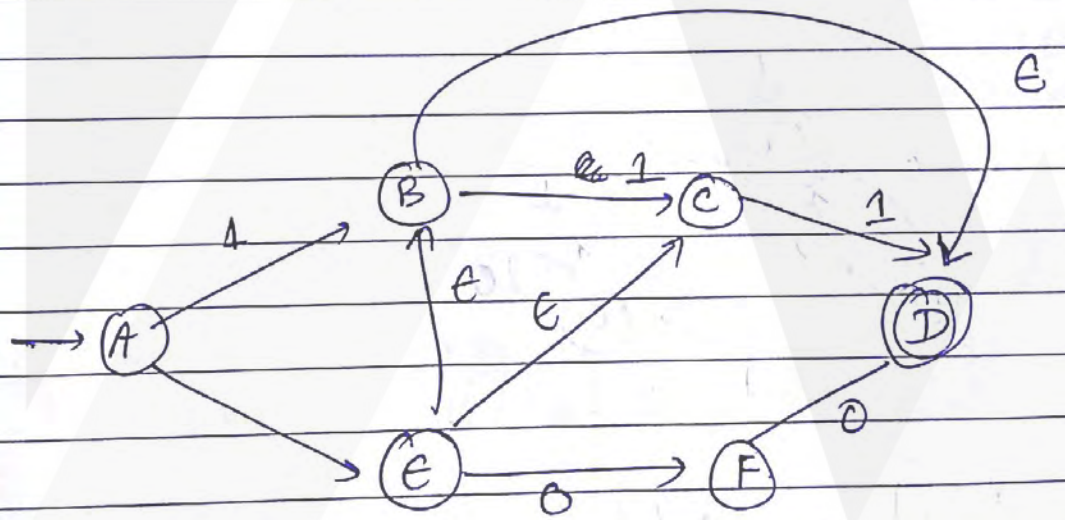
Page.....

$\epsilon(q_2) = \{q_2\}$
$\epsilon(q_3) = \{q_3\}$
$\epsilon(q_4) = \{q_4, q_2\}$

		a	b
→ *	A $\{q_1, q_0, q_2\}$	$\{q_0, q_3\}$	—
	B $\{q_0, q_3\}$	$\{q_3\}$	$\{q_2, q_4\}$
*	C $\{q_0\}$	$\{q_0\}$	—
*	D $\{q_2, q_4\}$	$\{q_3\}$	—
	E $\{q_3\}$	—	$\{q_4, q_2\}$.



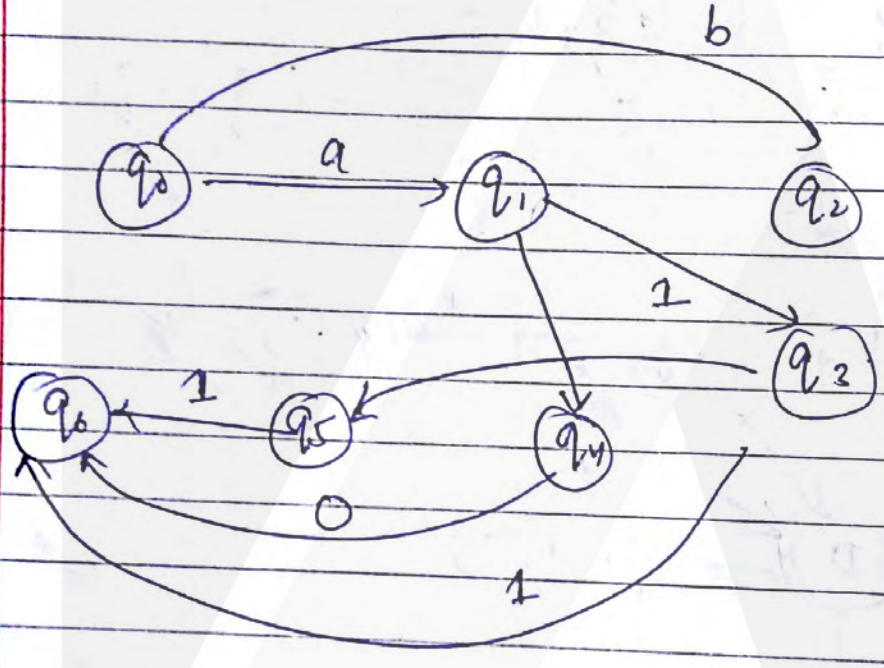
4)



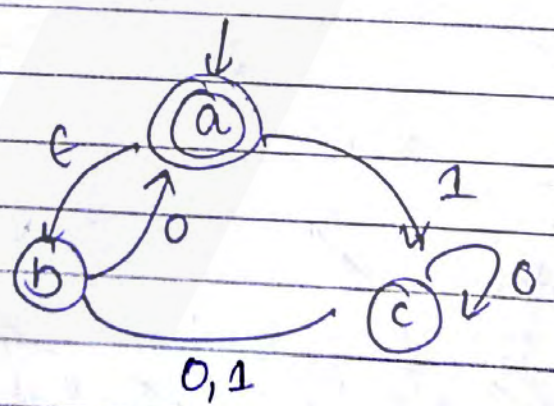
$\epsilon(A) = \{A\}$
$\epsilon(B) = \{B, D\}$
$\epsilon(C) = \{C\}$
$\epsilon(D) = \{D\}$

$\epsilon(E) = \{E, B, C\}$
$\epsilon(F) = \{F\}$

		0	1
{A}	q_0	{E, B, C}	{B, D}
{B, C, E}	q_1	{F}	{C, D}
{B, D}	q_2	-	{C}
{C, D}	q_3	-	{D}
{F}	q_4	{D}	-
{C}	q_5	-	{D}
{D}	q_6	-	-



5.



$\epsilon(a) = \{a, b\}$

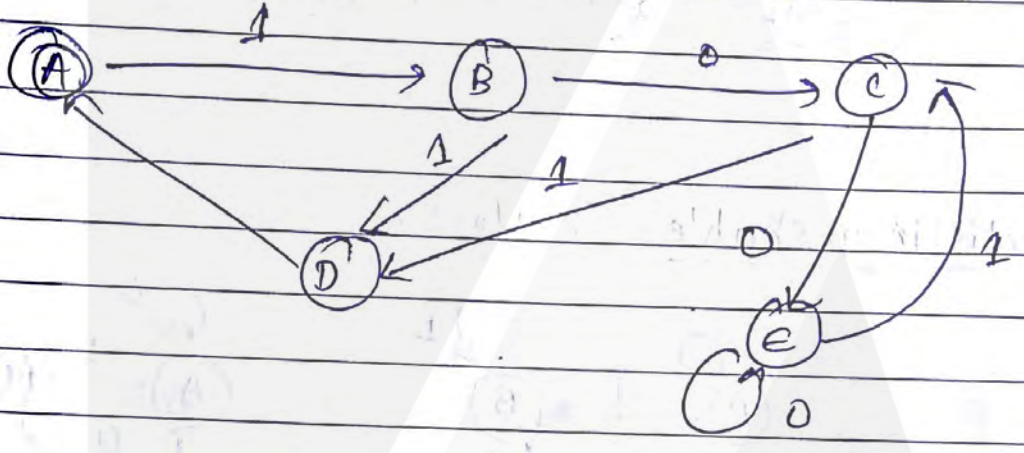
$\epsilon(b) = \{b\}$

$\epsilon(c) = \{c\}$

Date...../...../.....

Page.....

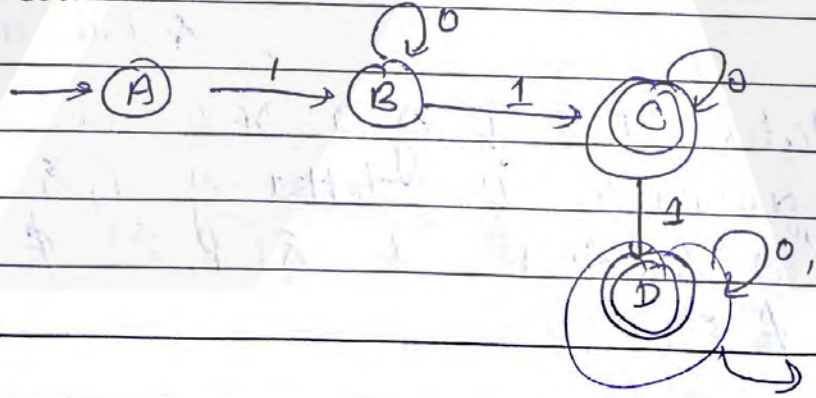
	0	1
→ {a, b} (A)	{a, b}	{c}
{c} (B)	{b, c}	{b}
{b, c} (C)	{a, b, c}	{b}
{b} (D)	{a, b}	-
{a, b, c} (E)	{a, b, c}	{b, c}



Minimization of DFA :→

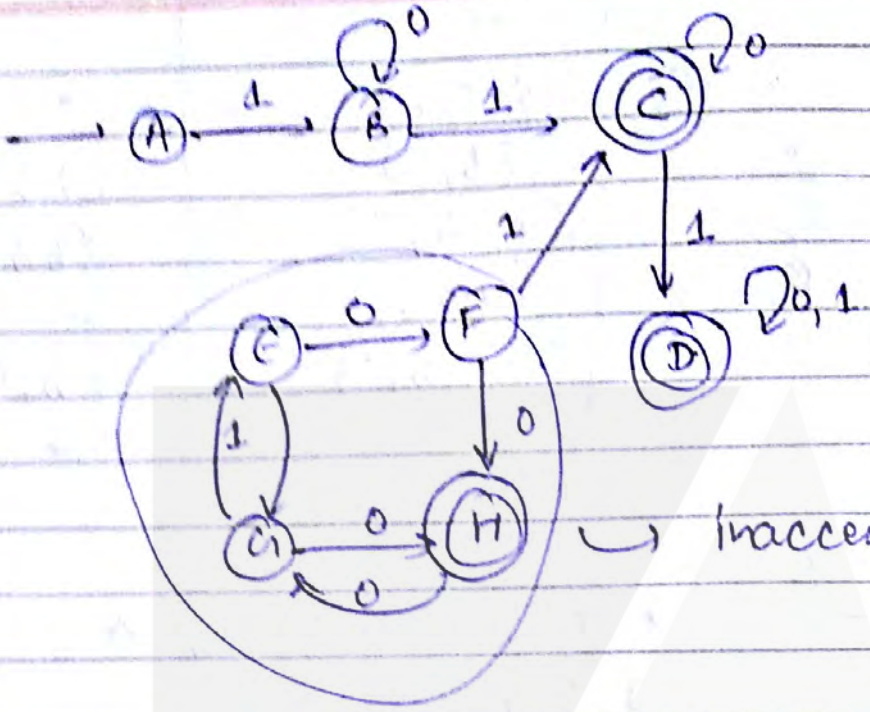
Dead State or sink State :→

All those non final states which transit to itself or transit to other dead states are called dead states.



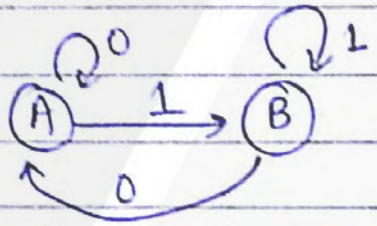
Inaccessible State :→

All those states which can never be reached from intial states are called inaccessible states.



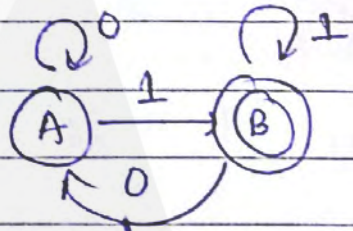
↳ Inaccessible State

Indistinguishable State :→



	0	1
→ A	A	B
B	A	B

Indistinguishable



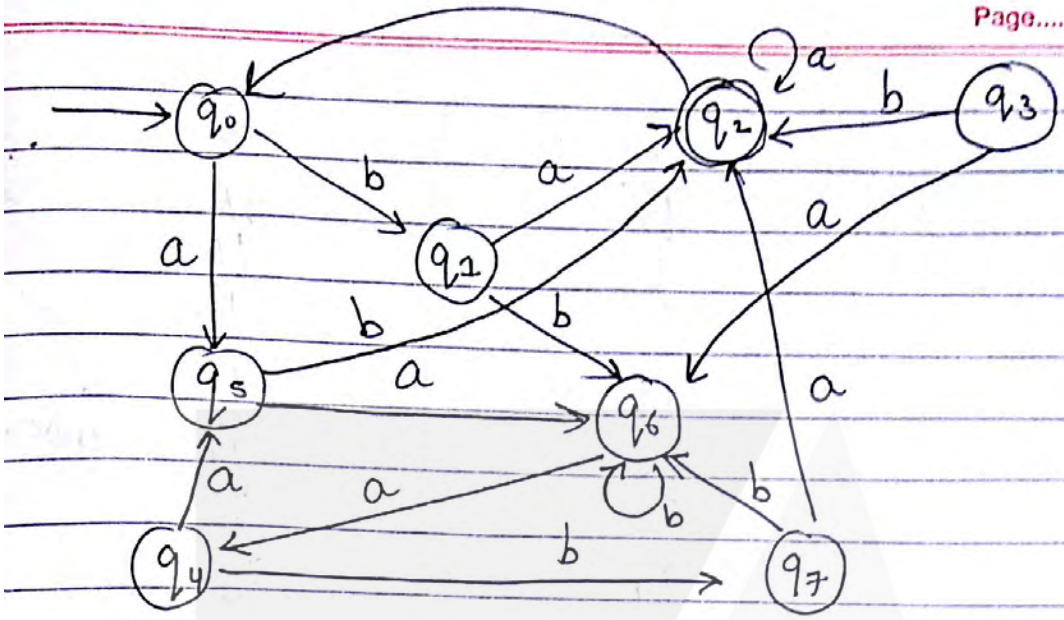
	0	1
→ A	A	B
* B	A	B

distinguishable one as it has final state & one normal state

Two states p & q of a DFA are called indistinguishable if delta of p, Σ belongs to F & δ(q, Σ) ∈ F & δ(p, Σ) ∉ F & δ(q, Σ) ∉ F.

Date...../...../.....

Page.....



Step 1:→

Draw transition table.

	a	b
q_0	q_5	q_1
q_1		
q_2		
q_3		
q_4		
q_5		
q_6		
q_7		

	a	b
q_6	q_5	q_1
q_8	q_6	
q_1		

	a	b
→ q_0	q_5	q_1
q_1	q_2	q_6
q_2	q_2	q_0
q_3	q_6	q_2 → Inaccessible
q_4	q_5	q_7
q_5	q_6	q_2
q_6	q_4	q_6
q_7	q_2	q_6

Step 2 :→
Remove (eliminate) inaccessible state if any.

Note :→ Please do not eliminate dead state at that point.

	a	b
→ q_0	q_5	q_1
q_1	q_2	q_6
q_2	q_2	q_0
q_4	q_5	q_7
q_5	q_6	q_2
q_6	q_4	q_6
q_7	q_2	q_6

Step 3 :→
Divide the transition table into two diff. tables where table 1 contains set of non final states & table 2 contains set of final states.

Set ~~State~~ - I

Date...../...../.....

~~Stat~~ Page. Set - II

	a	b
q_0	q_5	q_1 q_3
q_1	q_2	q_6
q_4	q_5	q_7 q_1
q_5	q_6	q_2
q_6	q_4	q_6
~~q_7~~	~~q_2~~	q_6

	a	b
q_2	q_2	q_0

→ indistinguishable (so eliminate 1 row).

Now, they become indistinguishable (eliminate q_4).

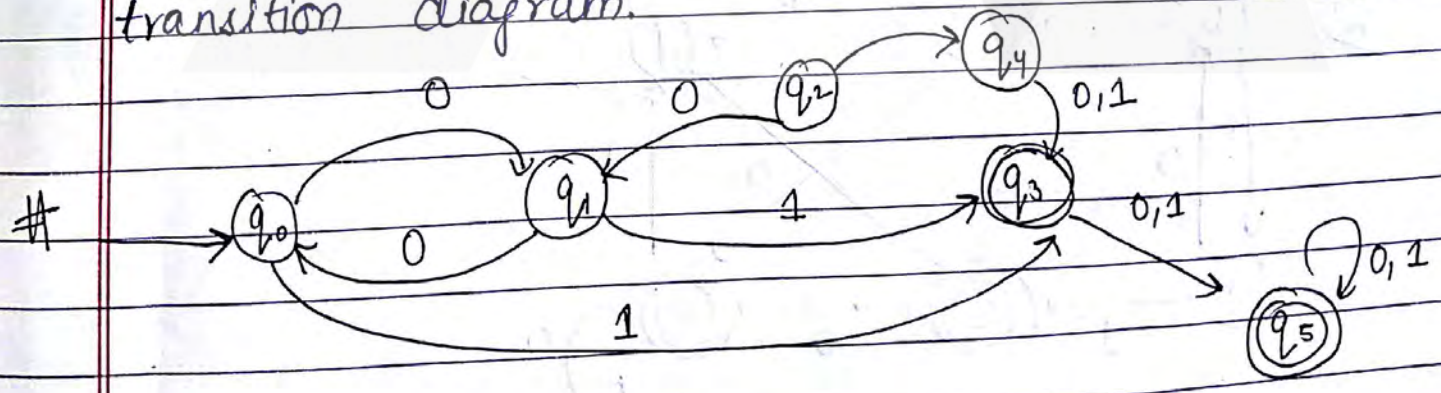
Step 4:- Merge both the tables & write final transition table.

	a	b
q_0	q_5	q_1
q_1	q_2	q_6
q_5	q_6	q_2
q_6	q_4	q_6
q_2	q_2	q_0

q_0	q_5	q_1
q_1	q_2	q_6
q_2	q_2	q_0
q_5	q_6	q_2
q_6	q_4	q_6

Step 5:- Now eliminate dead state if any.

⇒ This is the minimized DFA & draw that transition diagram.



q_2, q_4 → inaccessible

Date...../...../.....

Page.....

	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
q_3	q_5	q_5
q_5	q_5	q_5

Set - 1

	0	1
q_0	q_1	q_3
q_1	q_0	q_3

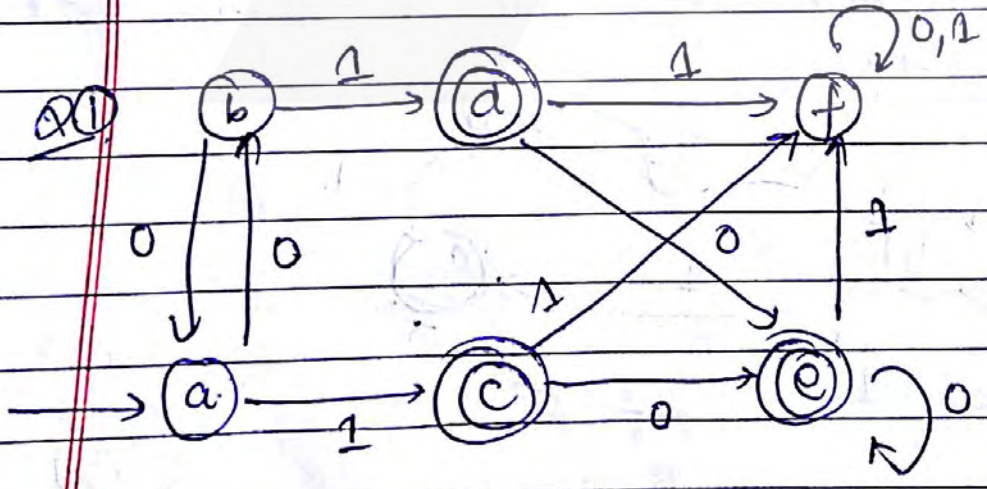
Set - 2

	0	1
q_3	q_5	q_5
q_5	q_5	q_5

→ Indistinguishable

	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
* q_3	q_3	q_3

Q1)



Date...../...../.....

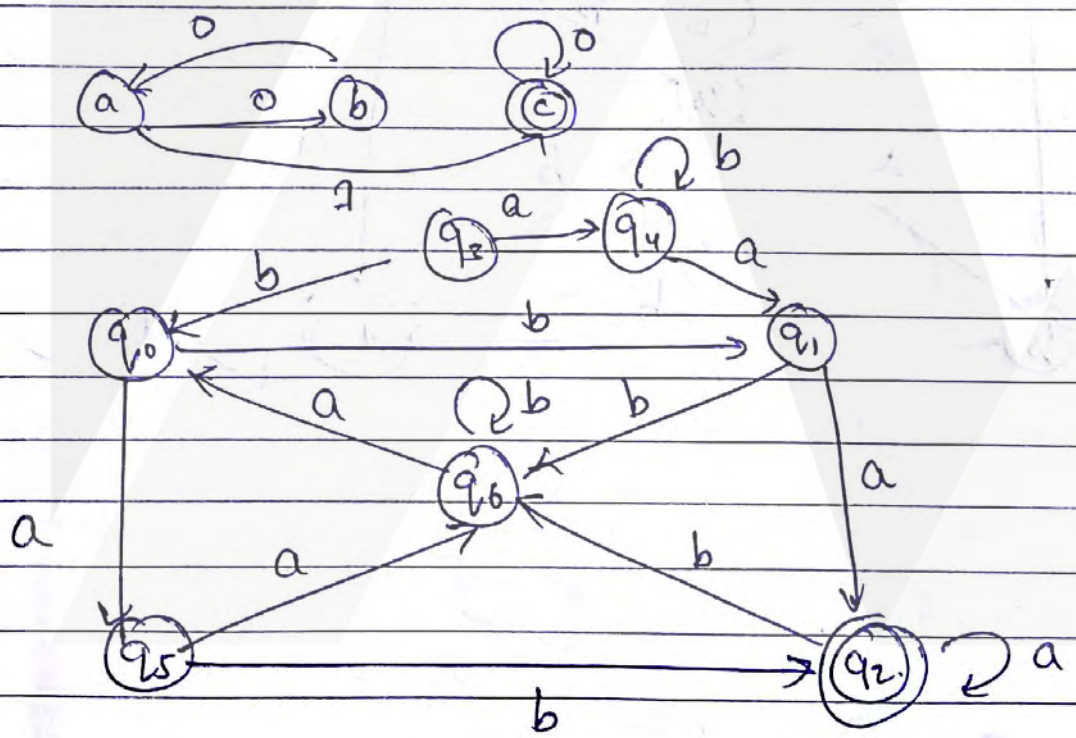
Page.....

Transition table	0	1	
→ a	b	c	
b	a	d	
* c	e	f	
* d	e	f	indistinguishable
* e	e	f	"
f	f	f	dead state.

final table

	0	1
→ a	b	c
b	a	–
* c	c	–

Q.2)



T.T.	a	b
q_0	q_5	q_1
q_1	q_2	q_6

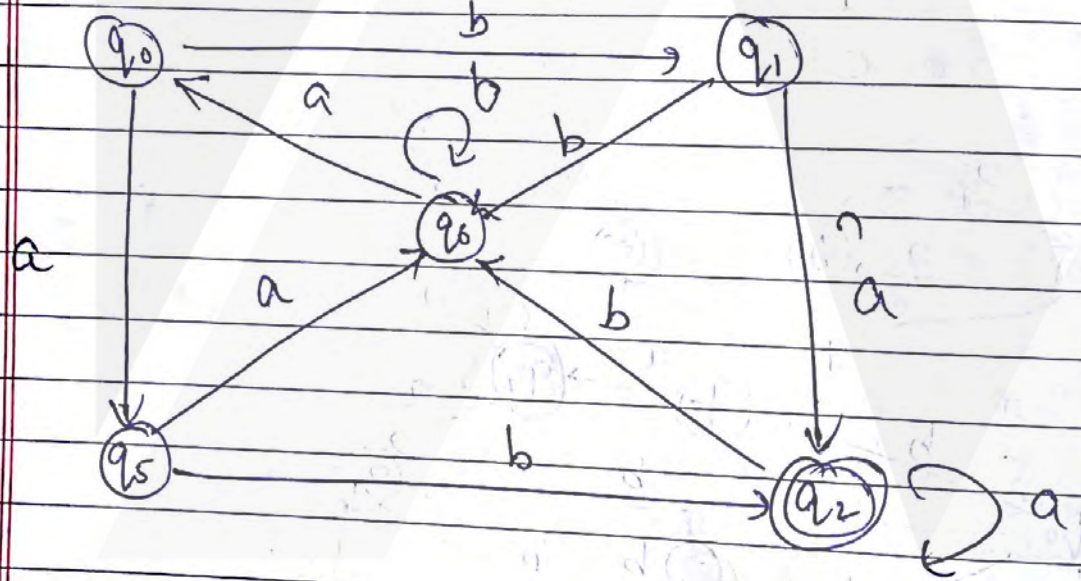
Date...../...../.....

Page.....

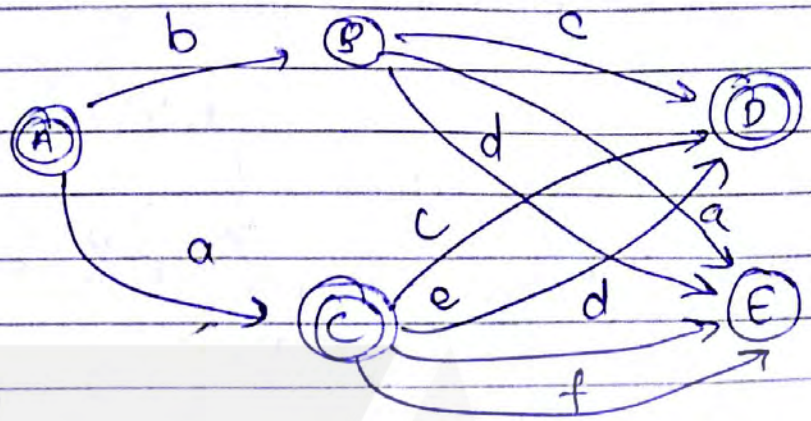
q_3	q_4	q_0
q_4	q_1	q_4
q_5	q_6	q_2
q_6	q_0	q_6

Table 2 :→

	a	b
q_0	q_5	q_1
q_1	q_2	q_6
q_5	q_6	q_2
q_6	q_6	q_6



Q.3).



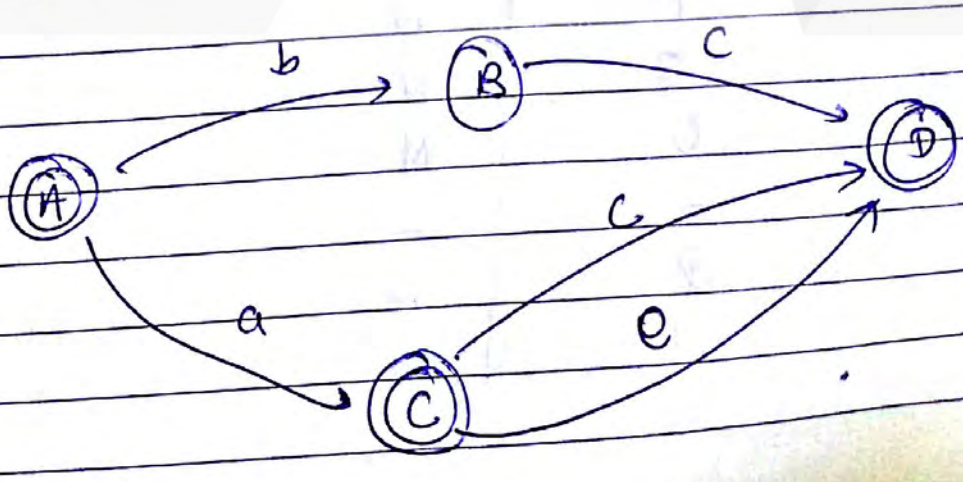
T.T.

		a	b	c	d	e	f
*	A	C	B	-	-	-	-
	B	E	-	D	E	-	-
*	C	-	-	D	E	D	E
*	D	-	-	-	-	-	-
	E	-	-	-	-	-	-

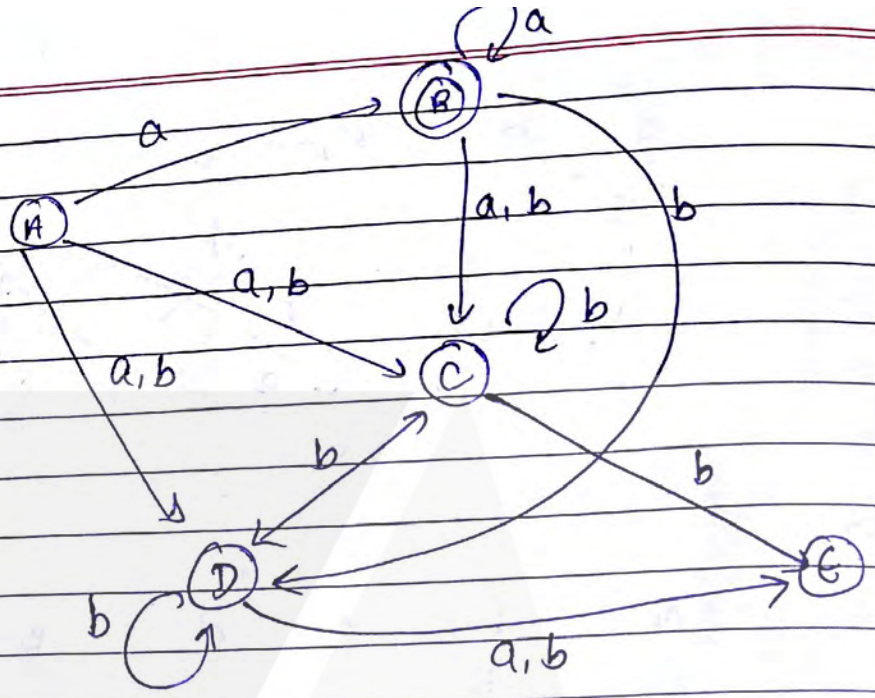
q Remove E as it is a "dead state".

Final table

		a	b	c	d	e	f
→ *	A	C	B	-	-	-	-
	B	-	-	D	-	-	-
*	C	-	-	D	-	D	-
*	D	-	-	-	-	-	-



Q.)



NFA → DFA → Min DFA → Result

		a	b
K →	A	{B, C, D}	{C, D}
L *	{B, C, D}	{B, C, E}	{C, D, E}
M	{C, D}	E	{C, D, E}
N	{C, D, E}	E	{C, D, E}
O	E	–	–
P *	{B, C, E}	{B, C}	{C, D, E}
Q *	{B, C}	{B, C}	{C, D, E}

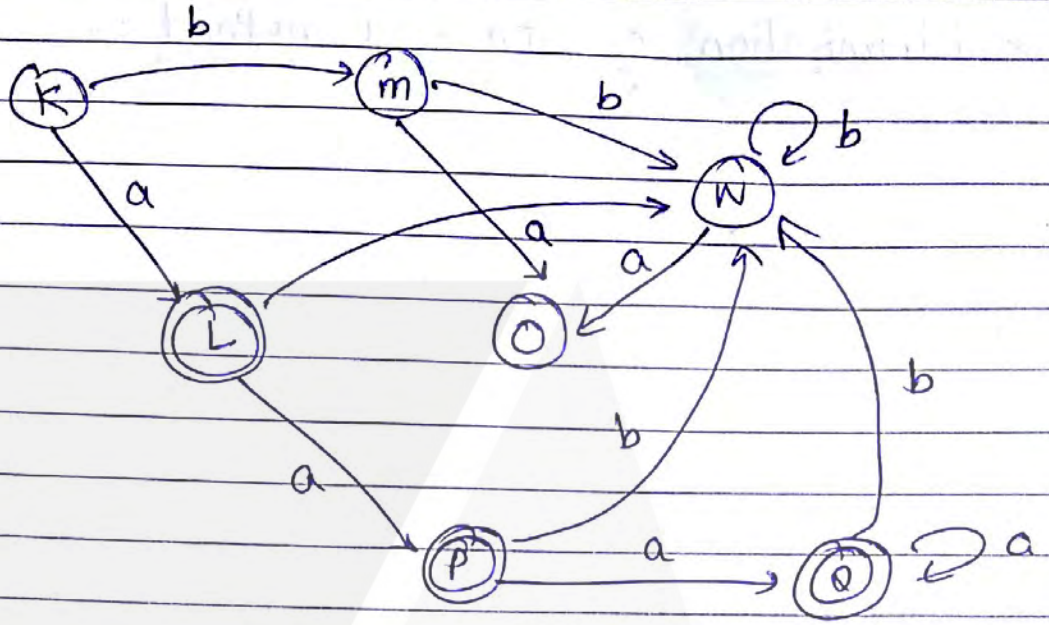
Table (DFA)

	a	b
→ K	L	M
* L	P	N
M	O	N
N	O	N
O	–	–
* P	Q	N
* Q	Q	N

Date...../...../.....

Page.....

DFA :-



Min. this DFA

T.T.

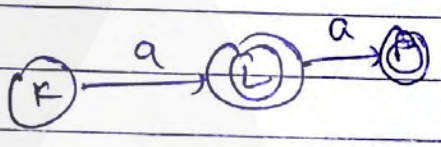
	a	b
→ K	L	M
* L	P	N
M	O	N
N	Q	N
O	—	—
* P	Q	N
* Q	Q	N

dead state ind.

dead state ind.

	a	b
→ K	L	M
* L	P	N
M	O	N
O	—	—
* P	Q	N

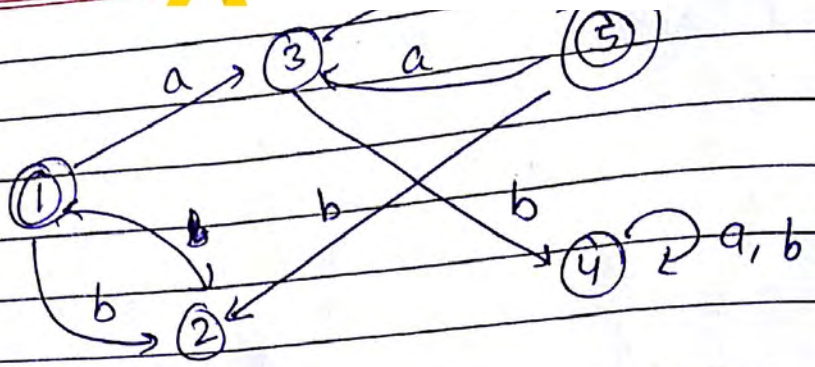
dead state.



	a	b
→ K	L	—
* L	P	—
* P	—	—

Minimization of DFA - Π method :→

12.



		a	b
→ *	1	3	2
	2	–	1
	3	5	4
	4	4	4
*	5	3	2

$\Pi_0 = \{\{1,5\}, \{2,3,4\}\}$

$\Pi_1 = \{\{1,5\}, \{2\}, \{3,4\}\}$

$\Pi_2 = \{\{1\}, \{5\}, \{2\}, \{3\}, \{4\}\}$

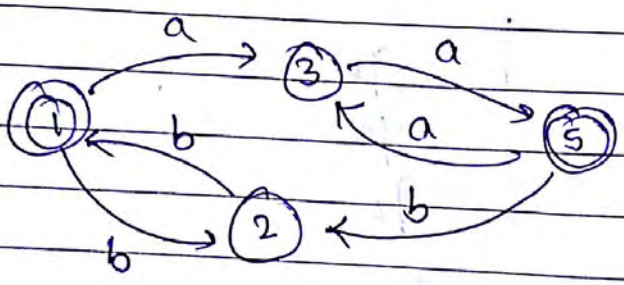
		a	b
→ *	1	3	2
	3	5	4
	2	–	1
*	5	3	2
	4	4	4

dead state

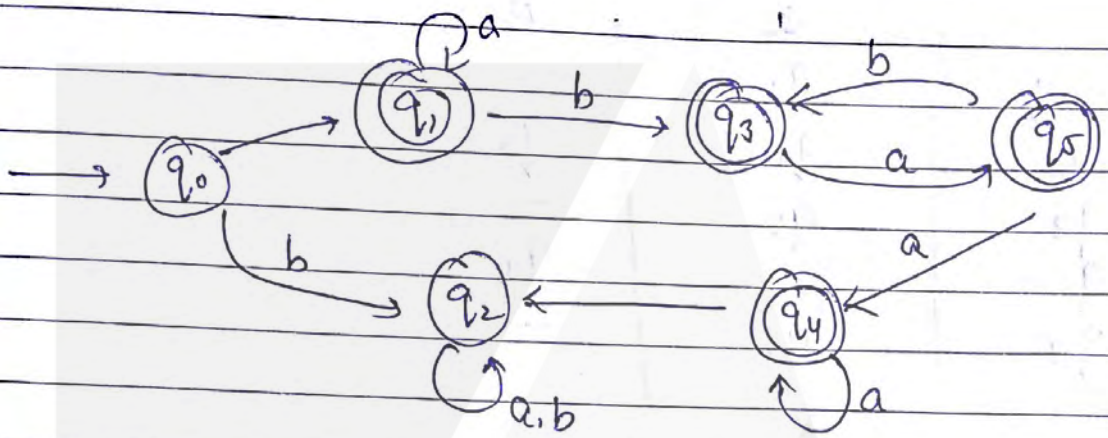
		a	b
→ *	1	3	2
	3	5	–
	2	–	1
*	5	3	2

Date...../...../.....

Page.....



(2)



	a	b
→ q_0	q_0	q_2
* q_1	q_1	q_3
Π q_2	q_2	q_2
* q_3	q_5	q_2
* q_4	q_4	q_2
* q_5	q_4	q_3

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

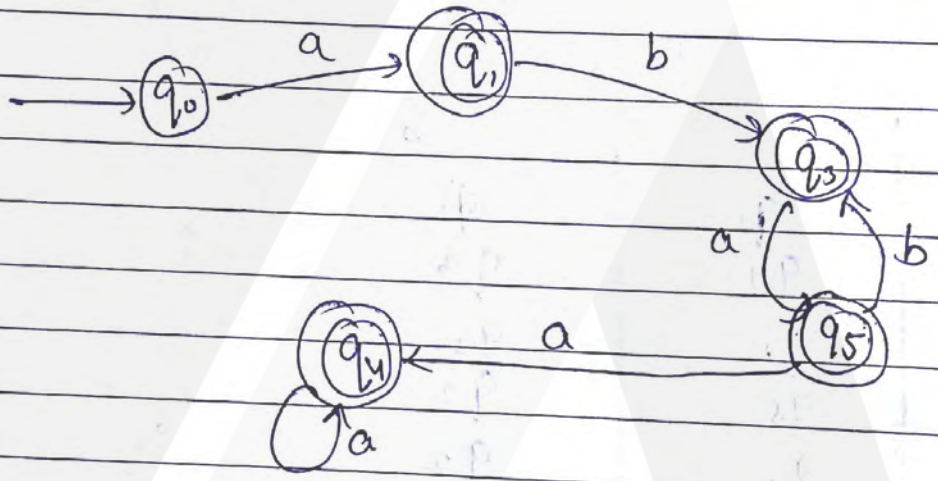
$\Pi_1 = \{\{q_0\}, \{q_2\}, \{q_1\ q_5\}, \{q_3, q_4\}\}$

$\Pi_2 = \{\{q_0\}, \{q_2\}, \{q_1\}, \{q_5\}, \{q_3\}\ \{q_4\}\}$.

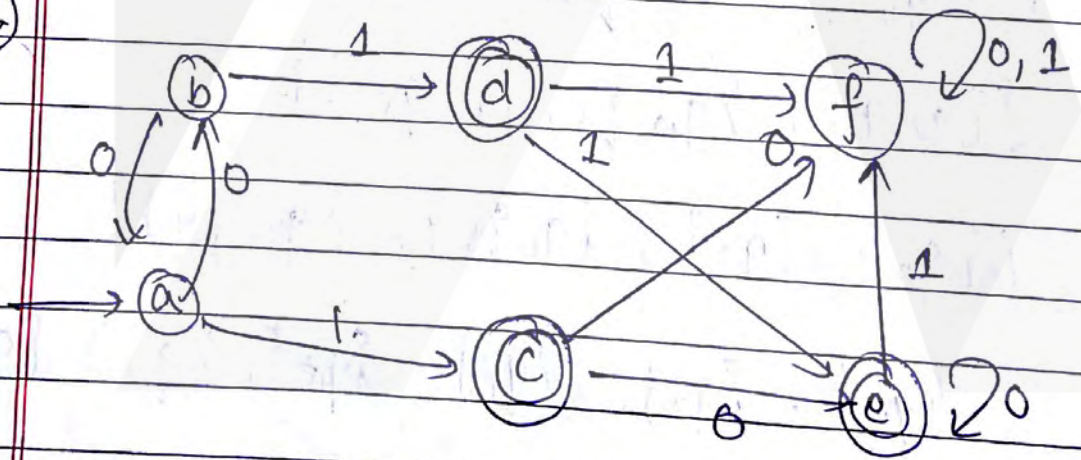
	a	b
→ q_0	q_1	q_2
* q_1	q_1	q_3
q_2	q_2	q_2

*	q_3	q_3	q_2
*	q_5	q_4	q_2
*	q_4	q_4	q_2

		a	b
→	q_0	q_1	–
*	q_1	q_1	q_3
*	q_3	q_5	–
*	q_5	q_4	q_3
*	q_4	q_4	–



(3)



		0	1
→	a	b	c
	b	a	d
*	c	e	f

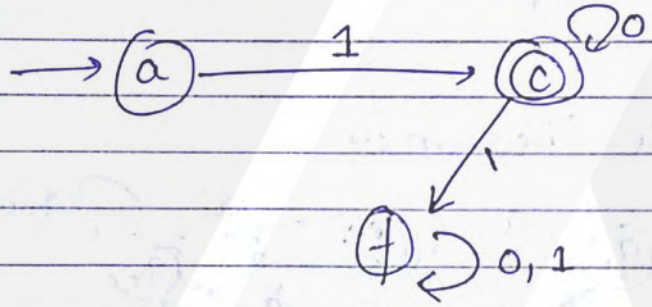
d	e	f
e	e	f
f	f	f

π_0 : $\{\{a, b, f\}, \{c, d, e\}\}$

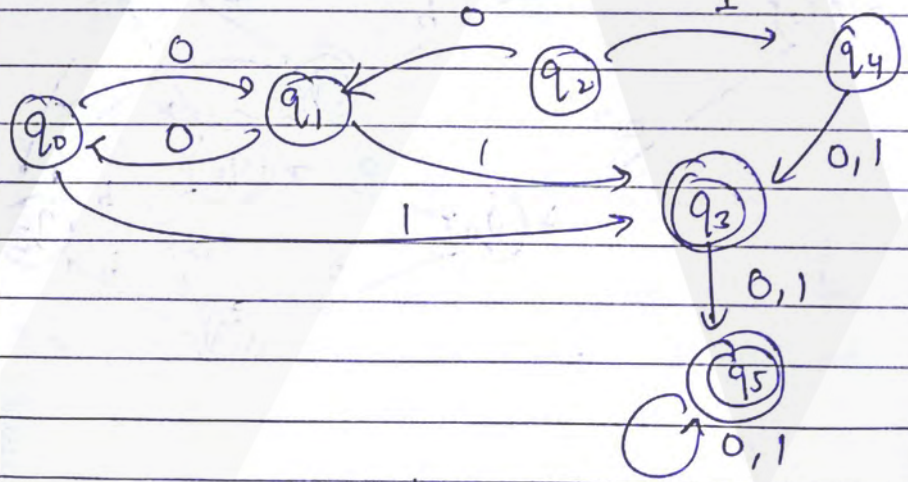
π_1 = $\{\{a, b\}, \{f\}, \{d, e\}\}$

π_2 = $\{\{a, b\}, \{f\}, \{c, d, e\}\}$

	0	1
→ a	a	c
* c	c	f
f	f	f



1.



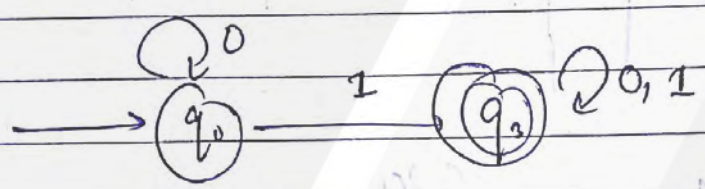
	0	1
→ q_0	q_1	q_2
q_1	q_0	q_3
q_2	q_1	q_4

* q_3	q_5	q_5
q_4	q_3	q_3
* q_5	q_5	q_5

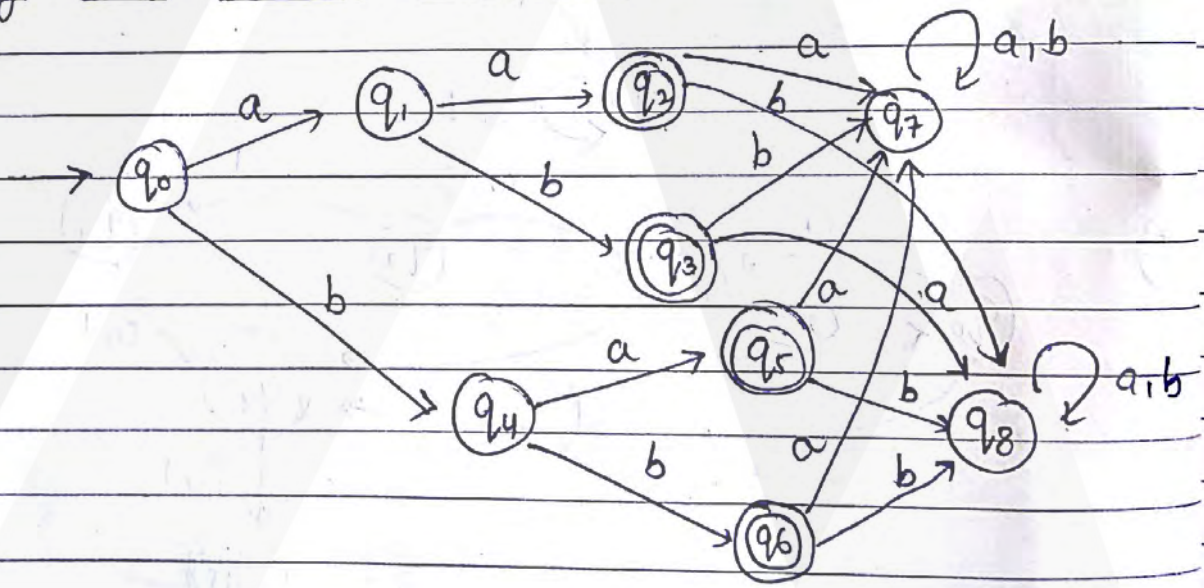
Π_0 : $\{\{q_0, q_1\}, \{q_3, q_5\}\}$

Π_1 : $\{q_0, q_1\}, \{q_3, q_5\}\}$.

	0	1
→ q_0	q_0	q_3
* q_3	q_3	q_3



* # My- Hill Nerode Theorem :→



Step 1 :-

Remove inaccessible state.

Note :- (Do not remove dead states).

There is no inaccessible state in the given DF

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$

$F = \{q_2, q_3, q_5, q_6\}$

$Q - F = \{q_0, q_1, q_4, q_7, q_8\}$

$F \times F = \{\{q_2, q_3\} \{q_2, q_5\} \{q_2, q_6\} \{q_3 q_5\} \{q_3, q_6\} \{q_5, q_6\}\}$

$Q-F \times Q-F$

$= \{\{q_0, q_1\} \{q_0, q_4\} \{q_0, q_7\} \{q_0, q_8\} \{q_1, q_4\} \{q_1, q_7\} \{q_1, q_8\} \{q_4, q_7\} \{q_4, q_8\} \{q_7, q_8\}\}$

$F \times Q-F = \{\{q_2 q_0\} \{q_2 q_1\} \{q_2, q_4\} \{q_2, q_7\} \{q_2, q_8\} \{q_3, q_0\} \{q_3, q_4\} \{q_3, q_7\} \{q_3, q_8\}$

$\{q_5, q_0\} \{q_5, q_1\} \{q_5, q_4\} \{q_5, q_7\} \{q_5, q_8\}$

$\{q_6, q_0\} \{q_6, q_1\} \{q_6, q_4\} \{q_6, q_7\} \{q_6, q_8\}\}$

Step 2 :→ Built a matrix with p rows and q columns for every state in Q. Put dashes along the diagonal and below the diagonal.

$(q_2 q_0)$ → same as (q_0, q_2)

Date...../...../.....

Page.....

P \ q	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
q_0	-	x	x	x	x	x	x	O(x)	O(x)
q_1	-	-	x	x	O	x	x	x	x
q_2	-	-	-	O	x	O	O	x	x
q_3	-	-	-	-	x	O	O	x	x
q_4	-	-	-	-	-	x	x	x	x
q_5	-	-	-	-	-	-	O	x	x
q_6	-	-	-	-	-	-	-	x	x
q_7	-	-	-	-	-	-	-	-	O
q_8	-	-	-	-	-	-	-	-	-

Step 3 :-

(i) Mark X in the upper part of the table for FX Q-F entries.

(ii) Mark x & O as follows:
Now, we will select p, q from Q-FX Q-F & find r, s as $r = \delta(p, a)$ and $s = \delta(q, a)$. If r, s is cross (X) or x then p, q will be small x. If r, s is neither (X) nor x then p, q will be O.

p, q

$\{q_0, q_1\}$

$r = \delta(q_0, a) \vdash q_1$

$s = \delta(q_1, a) \vdash q_2$

$\{q_1, q_2\}$ → is cross so $\{q_0 q_1\}$ →

{q4, q7}
r = q5 s = q8
(q5, q8} → x {q4, q8} → x

Date..... / ..... / .....
Page.....

{q7, q8}
r = q7 s = q8
{q7, q8} → — {q7, q8} → 0

{q0, q4}
r = δ[q0, a] ⊢ q1
s = δ[q4, a) ⊢ q5
(q1, q5) → x (q0, q4) → x

{q0, q7}
r = q1 s = q7
(q1, q7) → — x (q0, q7) → 0 (x)

{q0, q8)
r = q1 s = q8
{q1, q8} → — x {q0, q8} → 0 (x)

{q1, q4}
r = q2 s = q5
{q2, q5] → — {q1, q4) → 0

{q1, q7}
r = q2 s = q7
{q2, q7} → x {q1, q7} → x

{q1, q8}
r = q2 s = q8
{q2, q8} = x {q1, q8} → x

{q4, q7}
r = q5 s = q7
{q5, q7} → x {q4, q7} → x

Perform same on FXF

$\{q_2, q_3\}$

$r = q_7$ $s = q_8$

$\{q_7, q_8\} \rightarrow 0$ $\{q_2, q_3\} \rightarrow 0$

$\{q_2, q_5\}$

$r = q_7$ $s = q_7$

$\{q_7, q_7\} \rightarrow$ dash $\{q_2, q_5\} \rightarrow 0$

$\{q_2, q_6\}$

$r = q_7$ $s = q_7$

$\{q_7, q_7\} \rightarrow$ dash $\{q_2, q_6\} \rightarrow 0$

$\{q_3, q_5\}$

$r = q_8$ $q_5 \cdot q_7$

$\{q_8, q_7\} \rightarrow \neq 0$ $\{q_3, q_5\} \rightarrow 0$

$\{q_2, q_6\}$

$r = q_8$ $q_6 = q_7$

$\{q_8, q_7\} \rightarrow 0$ $\{q_3, q_6\} \rightarrow 0$

$\{q_5, q_6\}$

$r = q_7$ $q_6 = q_7$

$\{q_7, q_7\} \rightarrow 0$ $\{q_5, q_6\} \rightarrow 0$

Now, again repeat step 3. (and find errors)
Repeat step 3 again & again until the two iteration tables are same.

Step 4:→

Now, we have to write minimized state

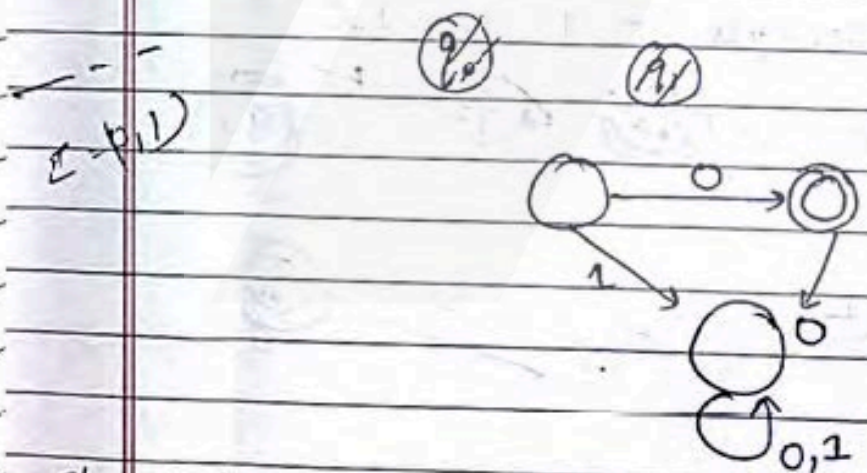
~~q_1, q_2~~

$\{q_1, q_4\}, \{q_2, q_5, q_6\}\ \{q_7, q_8\}$

	a	b
→ q_0	$\{q_1, q_4\}$ q_6	$\{q_1, q_4\}$
$\{q_1, q_4\}$	$\{q_2, q_3, q_5\}$	$\{q_2, q_3, q_7, q_6\}$
~~$\{q_2, q_3, q_5, q_6\}$~~		
~~$\{q_3, q_6\}$~~		
$\{q_2, q_3, q_5, q_6\}$	$\{q_7, q_8\}$	$\{q_7, q_8\}$
$\{q_7, q_8\}$	$\{q_7, q_8\}$	$\{q_7, q_8\}$ → dead state

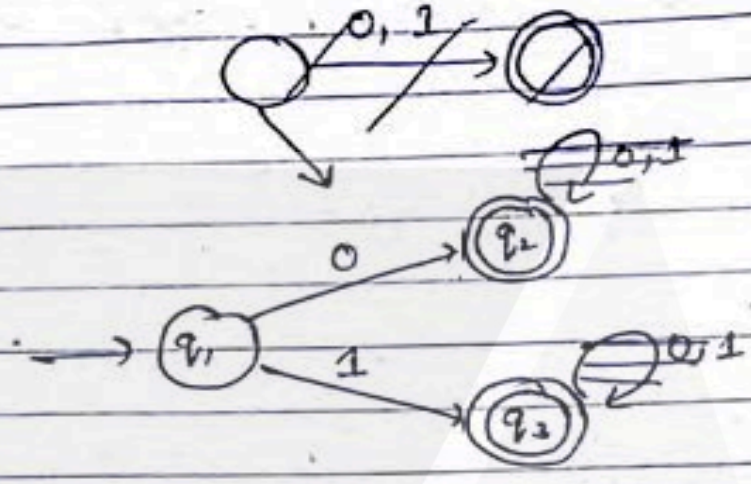
Now, eliminate the dead state.

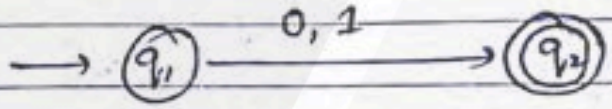
Q4. Draw a D.F.A. that can accept string 0 over the input alphabet $\Sigma = \{0, 1\}$



Q4. design a D.F.A. that accepts either 0, 1 over the input alphabet $\Sigma = \{0, 1\}$

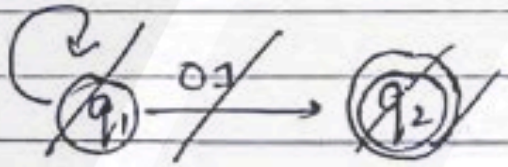
0, 1

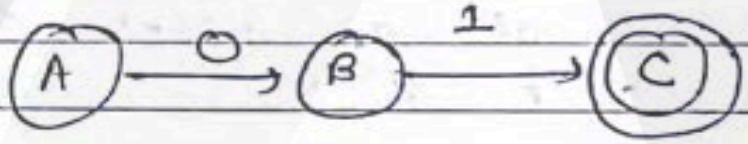




Q4. String {0, 1} =

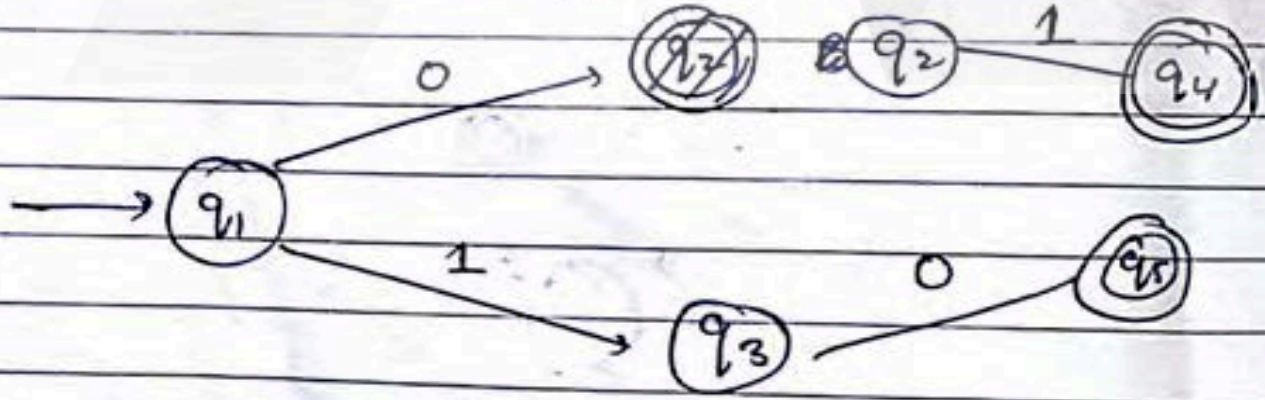
01



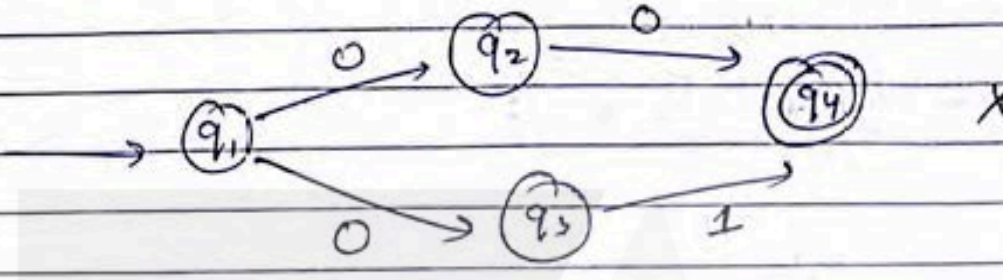


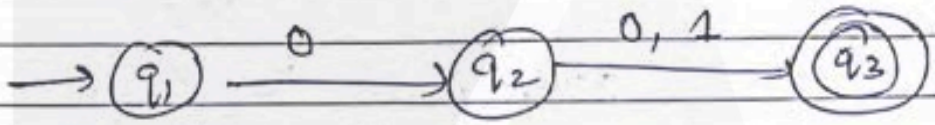
Q5. D.F.A that accepts 0,1 or 1,0.

=

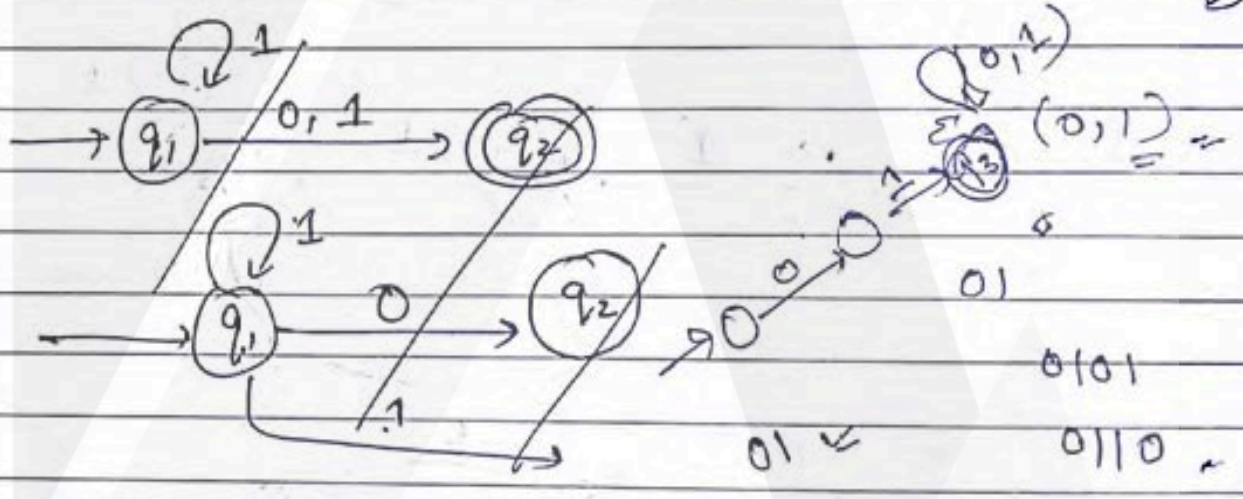


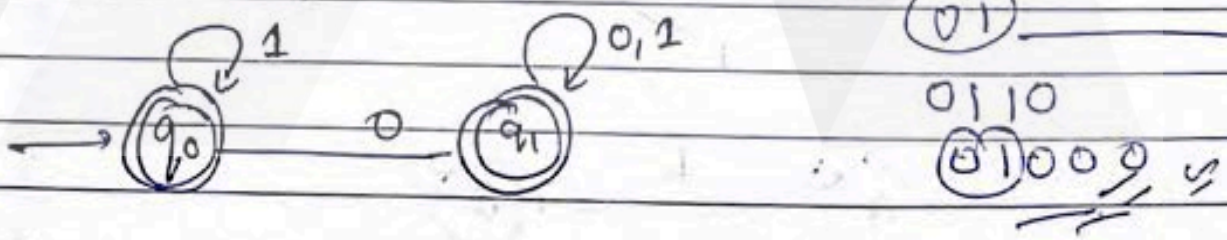
Q1. D.F.A. that accept either 0,0 or 0,1.



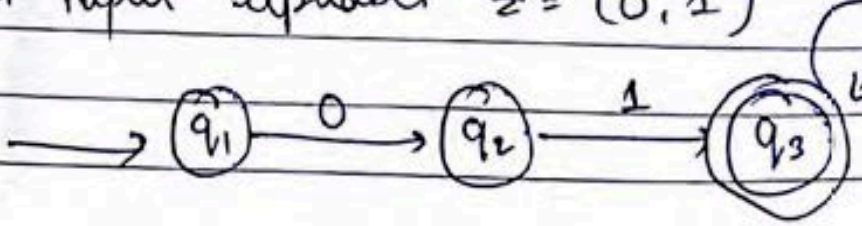


Q2. D.F.A. that accepts all the strings having with input alphabet $\Sigma = (0,1)$ any no. of 1





Q3. D.F.A. consisting of all strings starting with 01 with input alphabet $\Sigma = (0,1)$



Page.....

Automata with output :→

- Moore
- Mealy

Moore machine :→

It is a finite automaton in which output is associated with each state. formally it is defined by six tuples $m = \{Q, \Sigma, O', \delta, \lambda', Q_0\}$

$Q \rightarrow$ finite non-empty set of states
$\Sigma \rightarrow$ finite non-empty set of input alphabets
$Q_0 \rightarrow$ intial state $Q_0 \in Q$
$\delta \rightarrow Q \times \Sigma$ into Q
$O' \rightarrow$ it is the finite non-empty set of output alphabet.
$\lambda' \rightarrow$ it is an output fun. which maps Q into O'.

Mealy machine :→

is a finite automaton in which output is associated with both state & input.
It is defined by six tuples.
$m = \{Q, \Sigma, O', \delta, Q_0, \lambda'\}$
$\lambda' \rightarrow$ it is an output function. which maps $Q \times \Sigma$ into O'.
mealy machine can also be represented as following fun.

$$X(t) = \lambda(q(t), x(t))$$

where $q(t)$ is the present state.
$x(t)$ is the present input.

Date...../...../.....

Page.....

$z(t)$ = is the output function.

Similarly moove machine can be represented as

$$z(t) = \lambda(q(t))$$

where $q(t)$ is present state

$z'(t)$ is output function.

Transition table representation of Moore machine & Mealy machine :-

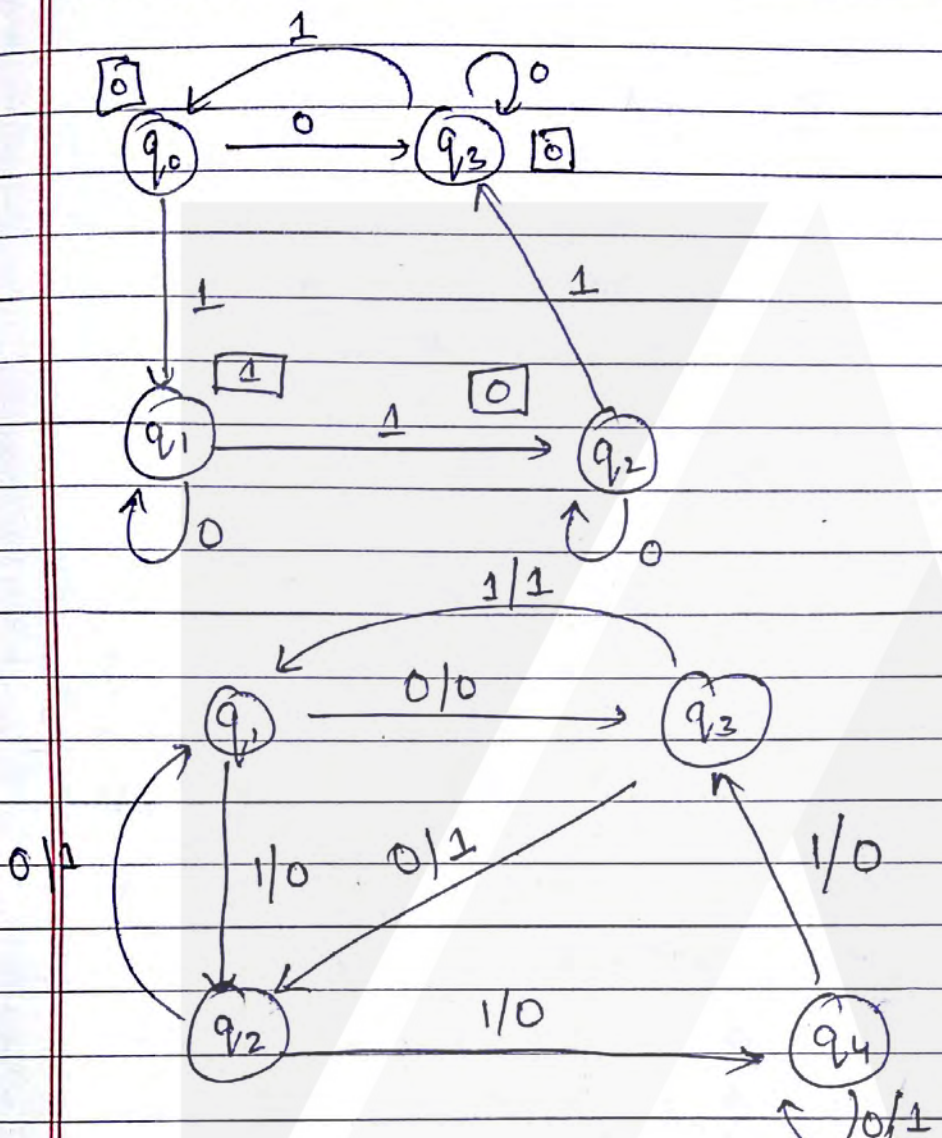
P.S.	Next State a=0	Next State a=1	O/P
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Moorey

P.S.	Next State a=0 State	Next State a=0 O/P	Next State a=1 State	Next State a=1 O/P
→ q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Mealy

Transition diag. Representation of Mealy & Moore Machines.



q_3 0

q_3 1

Transformation of Mealy machine to Moorey machine and vica versa.

F.A. to Moore :→

A finite automaton can be converted into a Moorey machine by introducing output alphabet $O' = 0, 1$ & defining the output function $\lambda(q)$ as follows

$$\lambda(q) = \begin{cases} 1 & \text{if } q \in F \\ 0 & \text{if } q \notin F \end{cases}$$

	a	b	O/P
→ q_0	q_2	q_1	0
* q_1	q_3	q_0	1
q_2	q_0	q_3	0
* q_3	q_1	q_3	1

	a	b	O/P
q_0	q_2	q_1	0
q_1	q_3	q_0	1
q_2	q_0	q_3	0
q_3	q_1	q_3	1

Remove the stars.

Q1. Convert the given F.A. to equivalent moorey machine

	a	b	O/P
→ * q_0	q_2	q_1	1
q_1	q_3	q_0	0
q_2	q_0	q_3	0
q_3	q_1	q_3	0

Remove star.

Page.....

Moovey to Healy machine :→

	a		b	
	State	O/P	State	O/P
→ q_0	q_2	0	q_1	0
q_1	q_3	0	q_0	1
q_2	q_0	1	q_3	0
q_3	q_1	0	q_3	0

Convert FA to Healy machine :→

	a	b
→ *) q_1	q_4	q_2
*) q_2	q_2	q_3
q_3	q_3	q_4
q_4	q_4	q_1

final state → 1
other states → 0

	a	b	O/P
→ q_1	q_4	q_2	1
q_2	q_2	q_3	1
q_3	q_3	q_4	0
q_4	q_4	q_1	0

Moovey

check from table

check from moovey machine

	a		b	
	State	O/P	State	O/P
→ q_1	q_4	0	q_2	1
q_2	q_2	1	q_3	1
q_3	q_3	0	q_4	0
q_4	q_4	0	q_1	1

Healy

Mealy to Moorey

Date...../...../.....
Page.....

System Appn Utility.

	a		b	
	State	O/P	State	O/P
→ q_1	q_3	1	q_2	0
q_2	q_1	1	q_4	1
q_3	q_2	0	q_1	1
q_4	q_4	1	q_3	0

(0,1)

q_4^0 1
q_3^1 q_4.

	a	b	O/P
→ q_1	q_3	q_2	1
q_2	q_1	q_4	0
q_3	q_2	q_1	0, 1
q_4	q_4	q_3	1

divide into two state
$q_3 \to 0$
$q_{31} \to 1$

	a	b	O/P
→ q_1	q_{31}	q_2	1
q_2	q_1	q_4	0
q_{30}	q_2	q_1	0
q_{31}	q_2	q_1	1
q_4	q_4	q_{30}	1

Moorey to F.A.

	a	b
→ * q_1	q_{31}	q_2
q_2	q_1	q_4
q_{30}	q_2	q_1
* q_{31}	q_2	q_1
* q_4	q_4	q_{30}

Mealy to Moorey

Date...../...../.....
Page.....

System Appn Utility

	a		b	
	state	o/p	state	o/p
→ q_1	q_3	1	q_2	0
q_2	q_1	1	q_4	1
q_3	q_2	0	q_1	1
q_4	q_4	1	q_3	0

(0,1)

q_4^0 , q_3^1 q_4.

	a	b	o/p
→ q_1	q_3	q_2	1
q_2	q_1	q_4	0
q_3	q_2	q_1	0, 1
q_4	q_4	q_3	1

divide into two state
$q_3 \to 0$
$q_{31} \to 1$

$q_{30} \to$
$q_{31} \to 1$

	a	b	o/p.
→ q_1	q_{31}	q_2	1
q_2	q_1	q_4	0
q_{30}	q_2	q_1	0
q_{31}	q_2	q_1	1
q_4	q_4	q_{30}	1

Moorey to F.A.

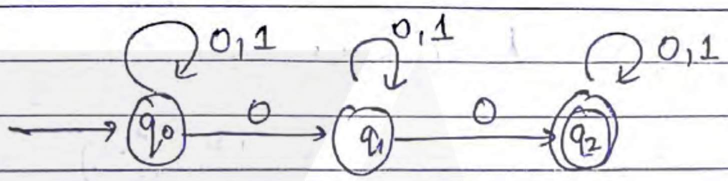
	a	b
→ * q_1	q_{31}	q_2
q_2	q_1	q_4
q_{30}	q_2	q_1
* q_{31}	q_2	q_1
* q_4	q_4	q_{30}

{0,1} {0101}0 Date ..../..../......

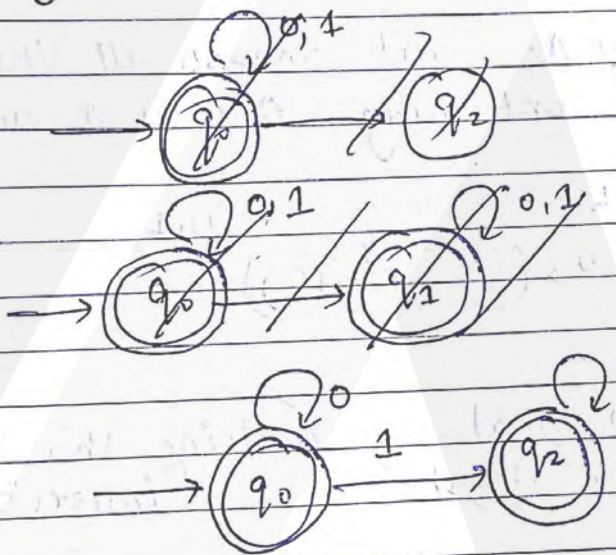
Page..........

TUT - 2.

Q1. Design a F.A. that recognize all the strings containing atleast 2 zero's over input alphabet $\Sigma = (0,1)$



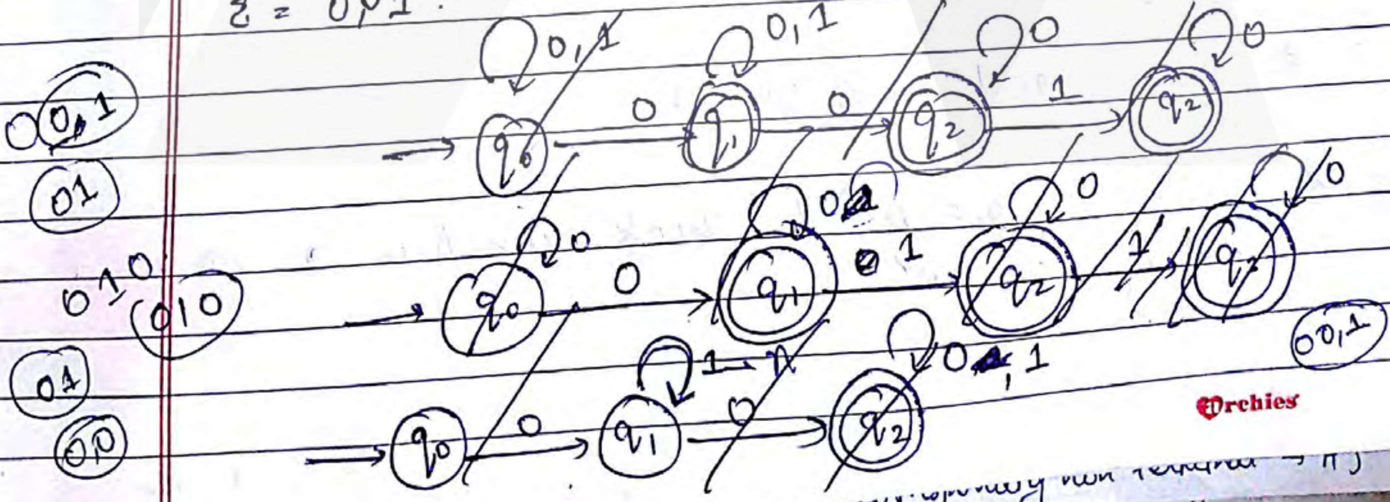
Q2. design a D.F.A that accepts all the strings containing atmost one 1 over the $\Sigma = 0,1$



0, 1, 1, 1

1, 1, 0

Q3. design a D.F.A. that acceps all the strings containing atleast 2 zeroe's & atmost 1, 1 over $\Sigma = 0, 1$.

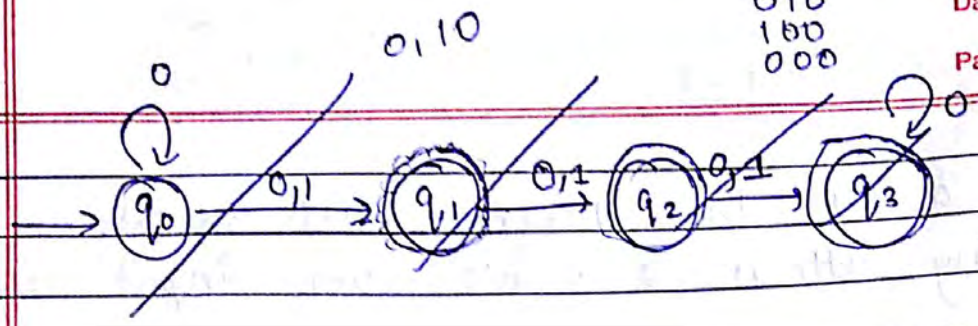


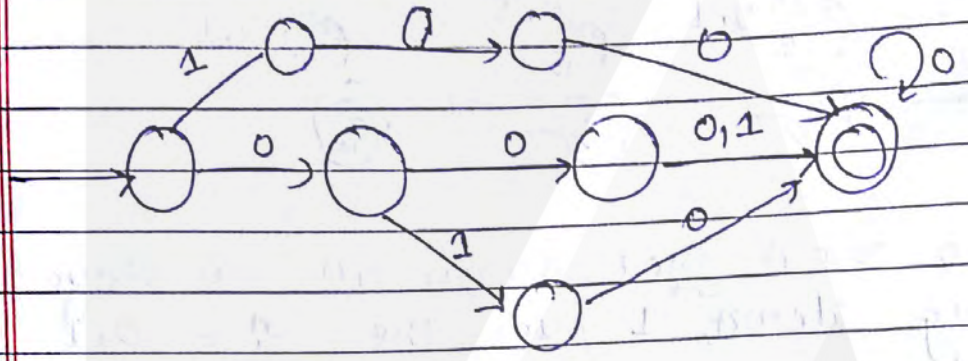
Orchies

001
010
100
000

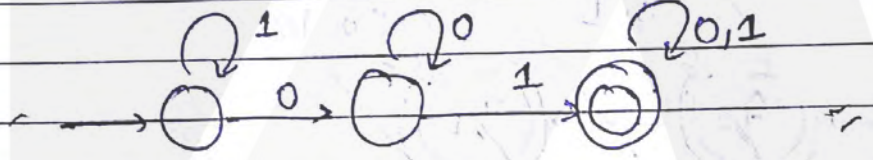
Date...../...../.....

Page.....

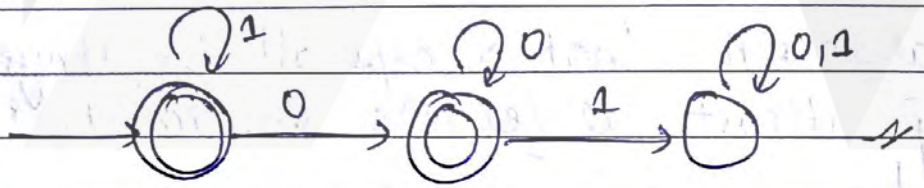




Q4. design a D.F.A. that accepts all the strings not containing 0, 1 as a substring.

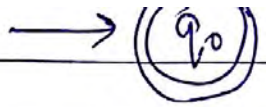


final → non final
non-final → final
(doing this ka inuernion)

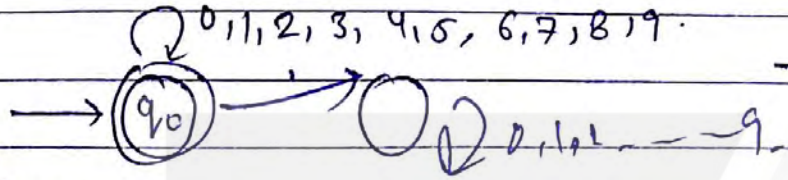


even & odd no. of zeroes

Q. design a D.F.A. to check whether a no. is binary or not.



to check whether a no. is decimal or not.



always

1). even no. of 1's & even no of 0's.

2). Design a D.F.A. which accept all the strings whose binary is divisible by 3.

0	0000
1	0001
2	0010
3	0011 ✗
4	0100
5	0101
6	0110 ✓
7	0111
8	1000
9	1001 ✓
10	1010
11	1011
12	1100 ✓

Date...../...../.....

Page.....

Q)02) 7.

Regular Expression is an algebric description of a language. It is defined as a declerative way to express the strings. The languages defined by regular expression are called regular language.

1). Any terminal symbol i.e. an element of Σ can be a regular expression.
2). λ, ϕ, ϵ are also regular expression.
3). Union of two regular expression R_1 and R_2 denoted by $R_1 + R_2$ is also a regular expression.
4). Concatenation of two regular expression R_1 & R_2 is denoted by $R_1 \cdot R_2$ is also a regular expression.
5). Iteration or closure of a regular expression is also a regular expression.
6). Regular expression is inclosed in {} is also regular exp. where {} influence order of evaluation. In absence of {} the order will be as follows
(i) Iteration
(ii) Concatenation
(iii) Union.

Regular Expression	Regular set
design a regular expression recognising 101	
recognize 101	{101}
abba	{abba}
either ϵ or ab	$\epsilon + ab$ {ϵ, ab}

10 or 01	$10+01$	$\{10, 01\}$
abb or b or a or baa	$abb+b+a+baa$	$\{abb, b, a, baa\}$
any combi of 0	0^*	$\{\epsilon, 0, 00, 000 \cdots\}$
Starting with 1 followed by any no of 0	10^*	$1\epsilon \to 1$ $\{1, 10, 100, 1000, \cdots\}$
Strings of 1's of even length	$(11)^*$	$\{\epsilon, 11, 1111, 111111 \cdots\}$
Any's containing any no of ab	$(ab)^*$	$\{\epsilon ab, abab, ababab \cdots\}$

Identities for regular expression :⇒

I_1 : $\phi + R = R$

I_2 : $\phi R = R\phi = \phi$

I_3 : $\lambda R = R\lambda = R$

I_4 : $\lambda^* = \lambda$ and $\phi^* = \lambda$

I_5 : $R + R = R$

I_6 : $R^* R^* = R^*$

I_7 : $RR^* = R^* R$

I_8 : $(R^*)^* = R^*$

I_9 : $\lambda + RR^* = R^* = \lambda + R^* R$

16-12-17 Fri

Date...../...../.....

Page.....

I_{10} :- $(PQ)^* P = P(QP)^*$

I_{11} : $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

I_{12} : $(P+Q)R = PR + QR$

and $R(P+Q) = RP + RQ$

Arden's Theorem :→

if $R = Q + R \cdot P$

then, $R = QP^*$

Q) Prove that :→

$(1 + 00^*1) + (1+00^*1)(0+10^*1)^*(\varnothing + 10^*1) = 0^*1(0+10^*1)^*$

⇒ $P + PQ^*\varnothing$

⇒ $P(\lambda + Q^*Q)$ By using I_9

⇒ PQ^*

⇒ $(1+00^*1)(0+10^*1)^*$

⇒ $((\lambda + 00^*)1)(0+10^*1)^*$

⇒ $0^*1(0+10^*1)^*$ = R.H.S.

R.E. → D.F.A & vice versa.

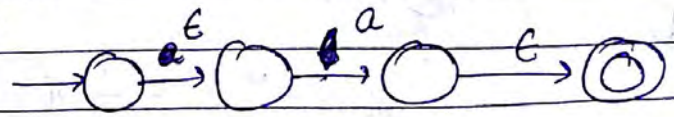
I. RE → FA wingt transitions

F.A.

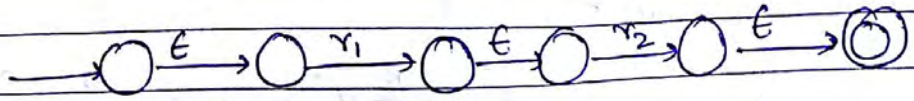
R.E. ϕ ⟶ ○ =

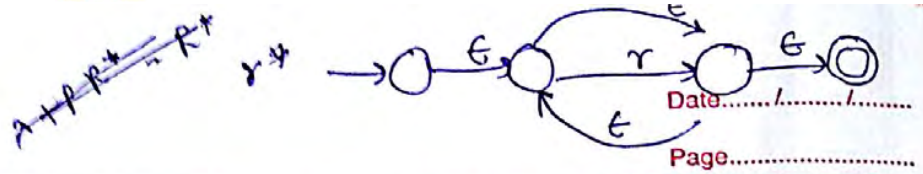
ϵ →◎

$= a$



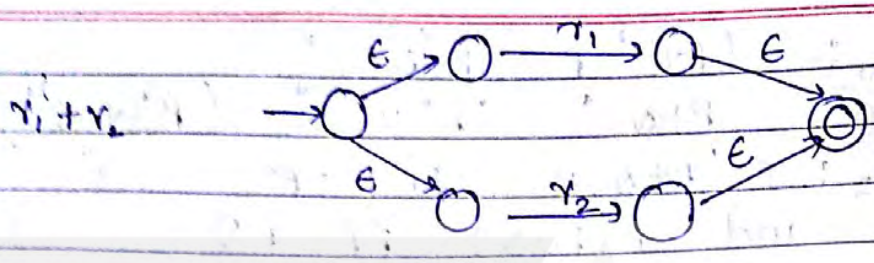
$r_1 r_2$





Date...../...../.....

Page.....



Q. design a F.A. with ϵ- transition for R.E as :-

① baa

② ba + ab

③ ba*

④ (ba + ab)*

⑤ a* baa + ba* b

⑥ a* (a+b)* d* + bd*

$(P^* \cdot Q^*)^*$

Q. $\lambda + 1^*(011)^* (1^*(011)^*)^* = (1 + 011)^*$

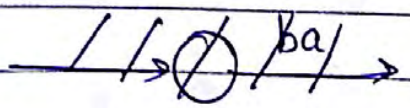
$\lambda + R \cdot R^* = R^*$

$(1/1+$

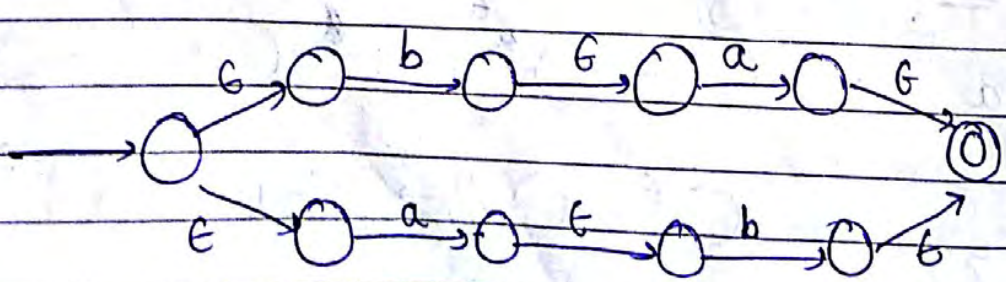
$\lambda/+ \ 1^*/+ \ (1^*(011)^*)^*$ I9

$(1+011)^*$ I11

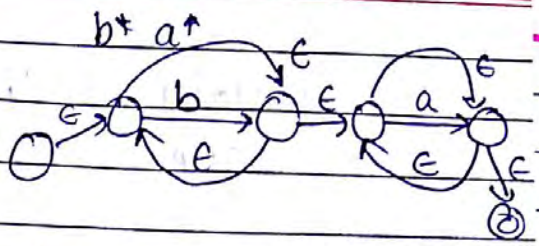
I.



(a)



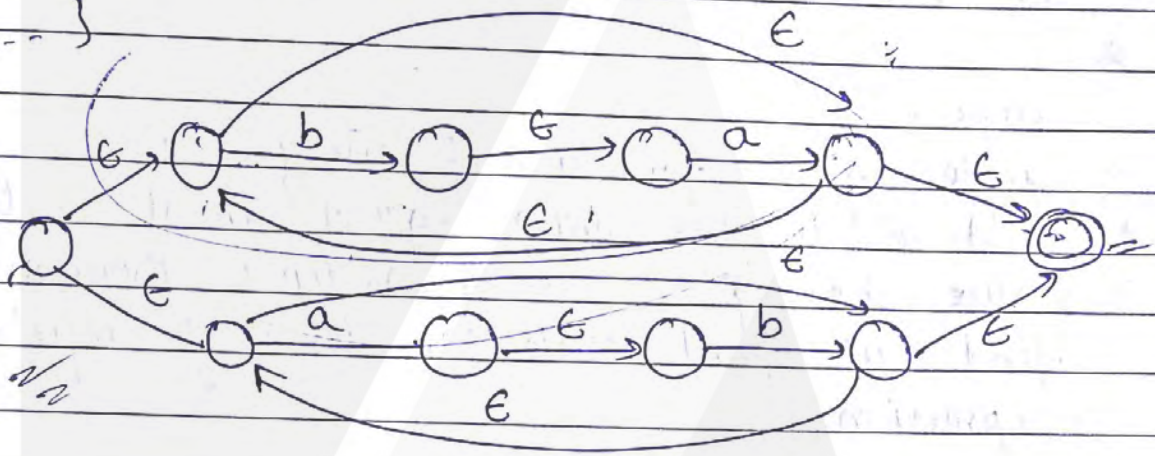
(3.)



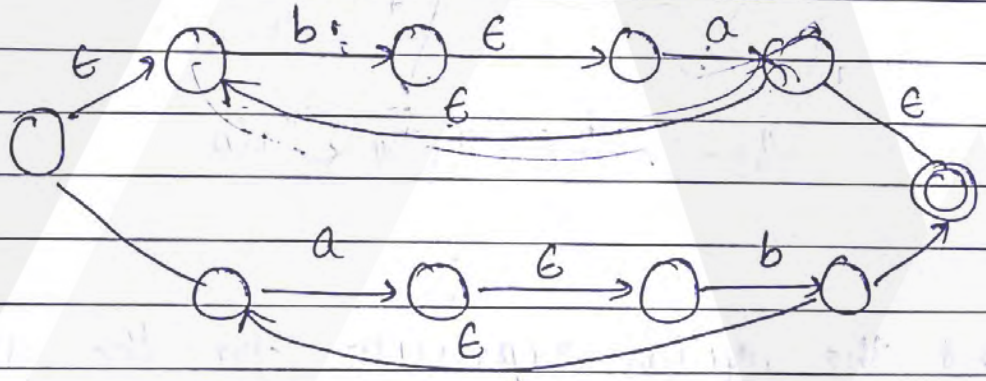
$a^* = \{\epsilon, a, aa, aaa \ldots\}$

(4) $(ba)^+ + (ab)^*$ $\quad a^+ = \{a, aa, aaa \ldots\}$

$\{ba, baba \ldots\}$



$(ba)^{*+} + (ab)^+$



D.F.A. to R.E.

for that we use Arden's Theorem :→

Arden's Theorem : Let P & Q be two regular expression over Σ. If P does not

Date...../...../.....

Page.....

contain ε then equation $R = Q + RP$ has a unique soln given by $R = QP^*$

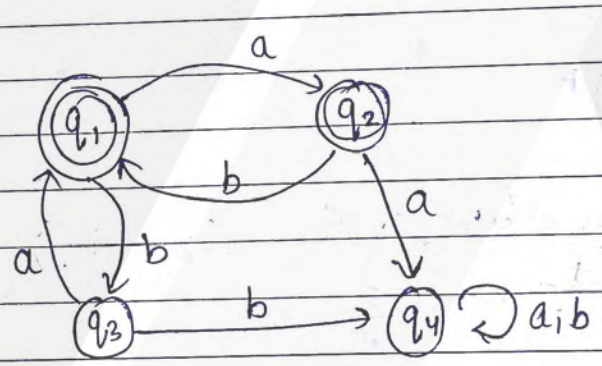
Prerequisites :⇒

1. The given transition diagram should strictly be a complete D.F.A.

Steps :⇒

1. Write eqn's for each & every state.
2. Add ε to the ~~initio~~ eqn of initial state.
3. Solve the eqn's using arden's theorem to find out final state in terms of regular expression.

Q4.



Find the regular expression for the given D.F.A

	a	b
q_1	$q_2 b$	$+ q_3 a$
q_2	q_4	q_1
q_3	q_1	q_4
q_4	q_4	q_4

← first state then input

$q_1 = q_2 b + q_3 a + \varepsilon$ — ①

$q_2 = q_1 a$ — ②

Jisme state na ho → regular expressions

Date...../...../.....

Page.....

$q_3 = q_1 b$ — (3)

$q_4 = q_2 a + q_3 b + q_4 b$ — (4)

$q_1 = q_0 \epsilon + q_1 ab + q_1 ba + \epsilon$

$\Rightarrow q_1 (ab + ba) + \epsilon$

$q_1 = \epsilon + q_1 (ab + ba)$

$Q \quad R \quad P$

$R = Q + R \cdot P$

$R = QP^*$

$= \epsilon (ab + ba)^*$

$q_1 = \epsilon (ab + ba)^*$

$q_1 = (ab + ba)^*$

$R = Q + RP$

$R = QP^*$

q_1

$QP^* \quad q_1 = \epsilon (ab + ba)^*$

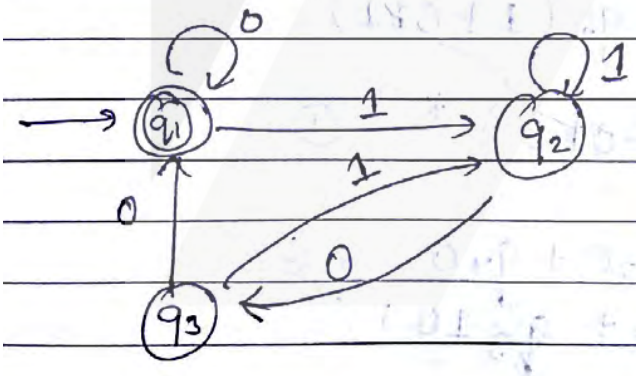
$q_1 + q_2 =$

$q_2 = q_1 (a)$

$\Rightarrow (ab + ba)^* a$

$q_1 + q_2 = (ab + ba)^* + (ab + ba)^* a$

because two final states so $q_1 \neq q_2$ if 3 final states, then $q_1 + q_2 + q_3$



$q_1 = q_1 0 + q_2 (0)$

$q_2 =$

$q_1 = q_1 0 + q_3 0 + \epsilon$

$q_2 = q_1 1 + q_2 1 + q_3 1$

$q_3 = q_2 0$

$q_2 = q_1 1 + q_2 1 + q_2 0$

$q_3 = q_2$

$q_1 = q_1 0 + q_2 00$

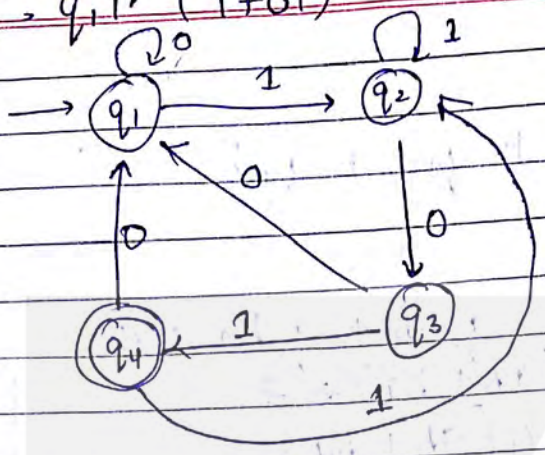
$q_2 = (q_1 0 + q_2 00) 1 + q_2 1 + q_2 0$

$= q_1 01 + q_2 001 + q_2 1 + q_2 0$

$q_2 = q_1 1 (1+01)$

$q_3 = q_1 1 (1+01)^*$

Q4.



$q_1 = q_1 0 + q_3 0 + q_4 0.$

q_2

q_3

q_4

$q_1 = q_3 0 + q_4 0 + q_1 0 + \epsilon$

$q_2 = q_1(1) + q_2(1) + q_4(1)$

$q_3 = q_2(0)^*$

$q_4 = q_3 1$

$q_1 = q_2 00 + q_4 0 + q_1 0$

$q_2 = q_1 1 + q_2 1 + q_3 11$

$q_2 = q_1 1 + q_2 1 + q_2 011$

$q_2 = q_1 1 + q_2(1 + 011)$

$q_2 = q_1 1 (1+011)^* \quad -①$

$q_1 = q_1 0 + q_3 0 + q_4 0 + \epsilon$

$q_1 = q_1 0 + q_2 00 + q_3 10 + \epsilon$

$q_1 = q_1 0 + q_1 1 (1+011)^* 00 + q_2 010 + \epsilon$

$q_1 = q_1 0 + q_1 1 (1+011)^* 00 + q_1 1 (1+011)^* 010 + \epsilon$

$q_1 = q_1 (0 + 1(1+011)^* 00 + 1(1+011)^* 010) + \epsilon$

$Q, \delta, \Sigma, q_0, F$

0.1 (000 + 1010)

Mod - 1 (for minor)

** 001 + 101

01(0+1) ✗

(0+1)01 (✓)

001 + 010

0(01 + 10) ✓

001 + 011

0(0+1)1

final state ki terms mai

$q_1 + q_4 \Rightarrow$

$q_4 = q_3 1$

$= q_2 0 1$

$= q_1 1 (1+011)^* 01$

$= (0+1(1+011)^* 00 + 1(1+011)^* 010) 1 (1+011)^* 01$

Pumping lema for regular sets :→

Pumping lema is used to prove that certain sets are not regular. It is called so because it gives a method of pumping or generating many input strings from a given expression.

Theorem :-

Let $m = Q, \delta, \Sigma, q_0, F$ be a finite automaton with n states. Let l be the regular set exp accepted by m. Let there is a string w belongs to L and length of w is greater than or equal to some no a if $a \geq n$ then there exists x, y, z such that $w = xyz$, $y \neq \lambda$ and $xy^iz \in L$. For each $i \geq 0$ to be regular.

length of string ≥ no. of states

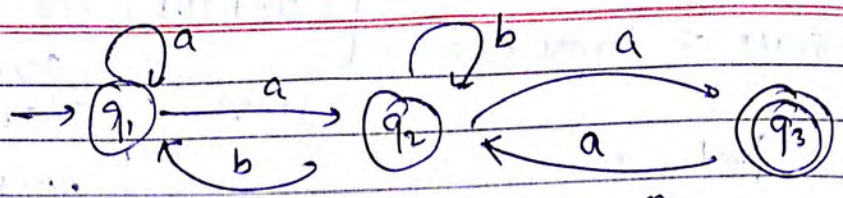
How to apply pumping lema :→

Step 1 :→ Assume that L is regular. Let n be the no. of states in the corresponding F.A.

Step 2 :→ Choose a string w such that $|w| \geq n$.

Using pumping lema to write $w = xyz$ such that $|xy| \leq n$

(1)



$q_1 = q_1a + q_2b + \epsilon$ — (1)

$q_2 = q_2b + q_1a + q_3a$ — (2)

$q_3 = q_2a$ — (3)

$q_2 = q_2b + q_2a + q_1a$

$q_2 = q_2(b+a) + q_1a$

R R P Q

$q_2 = q_1a(b+a)^*$

$q_3 = q_1a(b+a)^*a$

$q_1 = q_1a + q_1a(b+a)^*b + \epsilon$

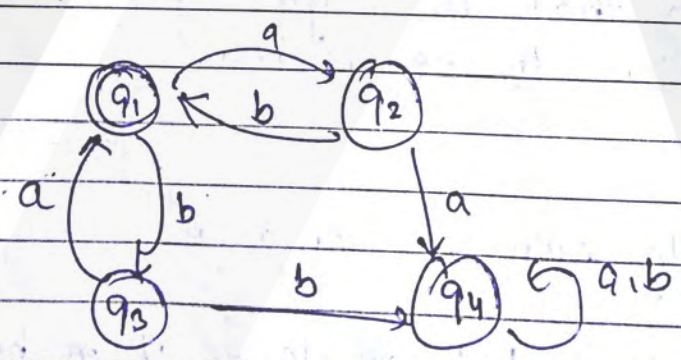
$q_1 = q_1(a + a(b+a)^*b) + \epsilon$

$q_1 = \epsilon(a + a(b+a)^*b)^*$

$q_2 = q_1a(b+a)^*b$

$q_3 = (a + a(b+a)^*b)^*(a(b+a)^*)$

(2)



$q_1 = q_2b + q_3a + \epsilon$

$q_2 = q_1a$

$q_3 = q_1b$

$q_4 = q_4a + q_4b + q_2a + q_3b$

(3.)

and

Step 3:- Fin
This
regul

Q.Y. Pro

Step 1:- A
state

Step 2:-

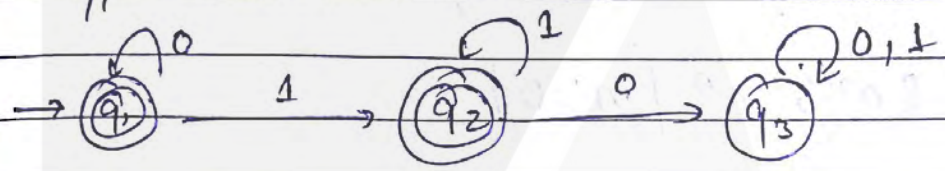
$q_1 = (q_1 a)b + (q_1 b)a + \epsilon$

$q_1 = q_1(ab + ba) + \epsilon$

$R = RA$

$q_1 = \epsilon(ab + ba)^*$

(3.)



$q_1 = q_1 0 + \epsilon$

$q_2 = q_2 1 + q_1 1$

$q_3 = q_3 0 + q_2 1 + q_3 0$

$q_1 = q_1 0 + \epsilon$

$q_1 = \epsilon 0^*$

$q_2 = q_2 1 + 0^* 1$

$R = (0^* 1)(1)^*$

and $|y| > 0$.

Step 3:→ Find suitable integer i such that $xy^i z \notin L$. This contradicts our assumptions. Hence L is not regular.

Q. Prove that $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

Step 1:→ Assume L is regular. Let n be the no. of states in the corresponding F.A. accepting L.

Step 2:→ $w = a^n b^n$

$x \quad y \quad z$

$a^{n-1} \quad a \quad b^n$

You can design a D.F.A → regular
cannot " → irregular

* Cond^n to be satisfied
* length of $|xy| \leq n$

x can be ϵ

$xy^i z$ for $i = 2$

$a^{n-1}\ a^2\ b^n = a^{n+1} b^n \notin L$

This contradicts our assumption

Q. $\{a^n b^n c^n \mid n \geq 0\}$

$a^{n-1} \cdot a^1 \cdot b^n c^n$

$x \quad y^i \quad z$

for $i = 2$.

$a^{n+1} b^n c^n$

Q. $\{a^n b a^n \mid n \geq 0\}$.

$a^n b a^n$ $\qquad$ $a^n / b / a^n$

$a^{n-1}\ a^i\ b a^n$

for $i = 2$

$a^{n+1} b a^n$

Q. $L = \{a^k \mid k$ is a prime no$\}$ is not regular.

$= a^{k-2}\ a\ a \qquad w = a^k$

$x / y / z / \qquad = a^{k-1} a \in$

$i = 2$

a^{k+1}

If k is a prime no than (k+1) can never be a prime no.

Date...../...../.....
Page.....(5)

$0 \geq (q-p)$ $(p-q) \geq 0$

Q. $\{L = 0^p 1^q \mid p \geq q$ is not regular $\}$

$w = 0^p 1^q 1^p$ (when $p \neq q$)

$= x \quad y \quad z$
$= 0^{p-i} \quad 0^i \quad 1^p$

(no. of 0 state = p) ~~$(0^q 1^q)$~~

~~for i~~ p = 2.
$0^{p+1} 1^p$

Q. $L = \{ww \mid w \in (a,b)^*\}$

let us suppose $w = a^n b^n$.
$w' = a^n b^n a^n b^n$ $a^n a^n$
$\neq a^n b^n$ $a^{n-1} a^i a^n$
$i = 2$
$a^{n+1} a^n$ not.

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

$w = a^n$
$= a^{n-1} \quad a^i \in$
$i = 2$
$= a^{n+1}$

It will be regular.

Q. design a DFA for A^*

String ↑ string followed by reverse ↑

Q1. $L = \{ ww^r \mid w \in \{a,b\}^* \}$

Q2. $L = \{ wcw \mid w \in \{a,b\}^* \}$

Q3. $L = \{ wcw^r \mid w \in \{a,b\}^* \}$

koi bhi variable

1). $a^n b^n b^n a^n$

$w = a^n b^n b^n a^n$ #

$a^{n-1} a^i b^n b^n a$

a

Context free grammar is

is defined as $G = V_n, V_T, P, S$

when

V_n → set of

P → set of prodn of the firm.

↳ $\alpha \to \beta$

α → ~~set of~~ belongs to V_n

β → belongs to $(V_n \cup V_T)^*$

$L(G) = \{ w \text{ in } V_t^* \mid w \in$

string can be generated by 0, 1, 2 ---

$S \xrightarrow[G]{*} w$

Sentential :-

derivation from the start symbol produces sub-strings, we call these sentential form. L(G) is in the sential form that all the characters in it belongs to V_t^*

Backus Normal form :→

It is a meta syntax used to represent C.F.G. i.e. a formal way to describe a language. A B.N.F. specification is a set of derivation rule written as

Symbol → expression

where symbol is a non-terminal & expression consists of any no. of terminals & non-terminals.

Parse tree :→ (derivation tree, syntax tree, generation tree, production tree).

Q. G

$S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid ss \mid ba$

$w = aabbaa$

$S \overset{*}{\underset{G}{\Rightarrow}} w$

$S \rightarrow aAS$	
$S \rightarrow ass$	$S \rightarrow ass$
$S \rightarrow aaAss$	$S \rightarrow aaSAsa$
$S \rightarrow aababss$	$S \rightarrow aabass$
$S \rightarrow aab$	$S \rightarrow$

$S \rightarrow aAS$

$S \rightarrow ass$ aabAs

$S \rightarrow aaAss$ aabbaS

$S \rightarrow aababs$ aabbaa

aAS / ass

$w = aabbaaaaa$

$S \rightarrow aAS$

$\rightarrow aSSS$

$\rightarrow aaAsss$

$\rightarrow aaSbAsss$

$\rightarrow aa$

$S \rightarrow aAS$

$\rightarrow asss$

$\rightarrow aaAsss$ aaSs aaAsss

$\rightarrow aabasss$

A

Q4. $S \rightarrow S+S \mid S*S \mid S-S \mid S\%S \mid (S) \mid (a)$

$w = a+a*a$

$S \rightarrow S+S$

$S+S*S$

$a+a*a$

Q4. $w = (a+a)*(a+a)$

$= S*S$

$= (S+S)*(S+S)$

$= (a+a)*(a+a)$

Strings generated by a C.F.G. can be represented by a hierarchical structure called tree. A parse tree for the C.F.G. has the following characteristics.

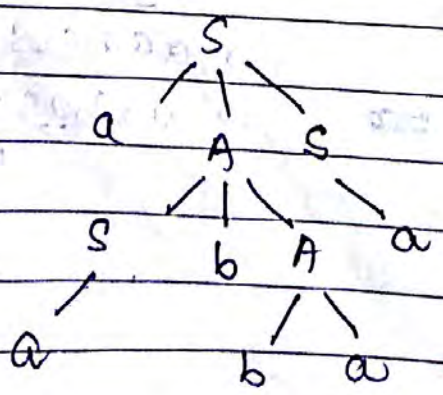
(i) Every vertex of a parse tree has the label which is either a terminal or non-terminal

(ii) Root always has a label S that is a start symbol.

(iii) Leaf node always has a label in Σ (i.e. terminal)

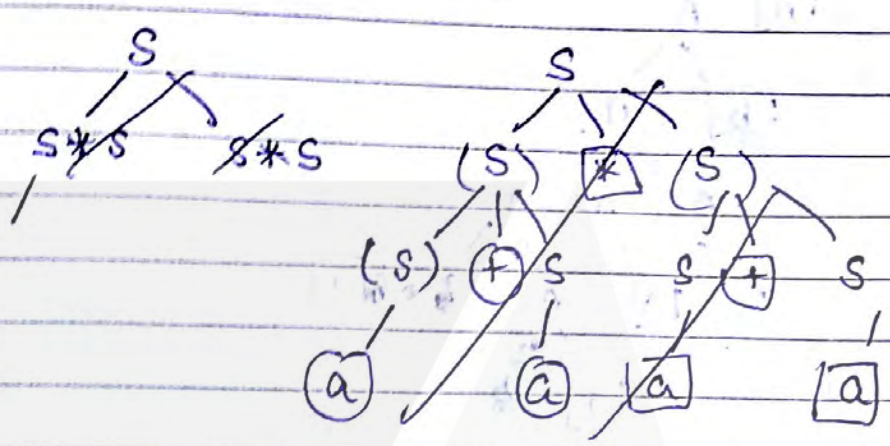
(iv) Internal node always have a label V_N

(v) If a vertex V has k children with labels $V_1, V_2, V_3 \dots V_k$ then $V \rightarrow V_1 V_2 V_3 V_4 \dots V_k$ will be a production in the C.F.G.



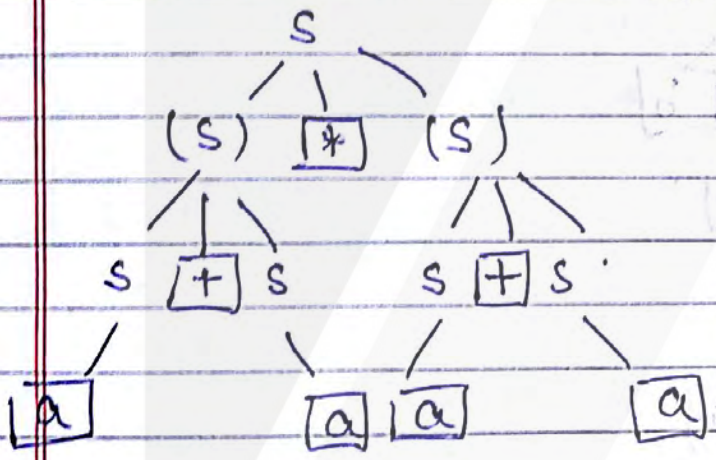
Q4. $S \to S+S \mid S*S \mid S-S \mid S/S \mid (S) \mid a$

$w = (a+a) * (a+a)$.



$(a+a) * (a+a)$

$a+$

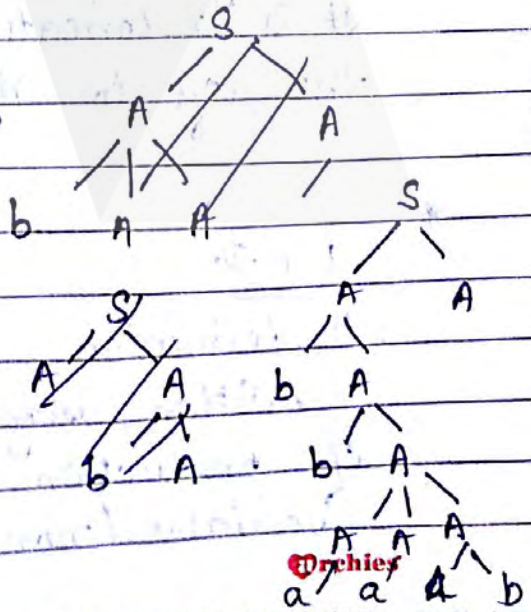


Q1.

$S \to AA$

$A \to AAA \mid bA \mid Ab \mid a$

$w \to bbaaaab$



$S \to \underline{A}A$

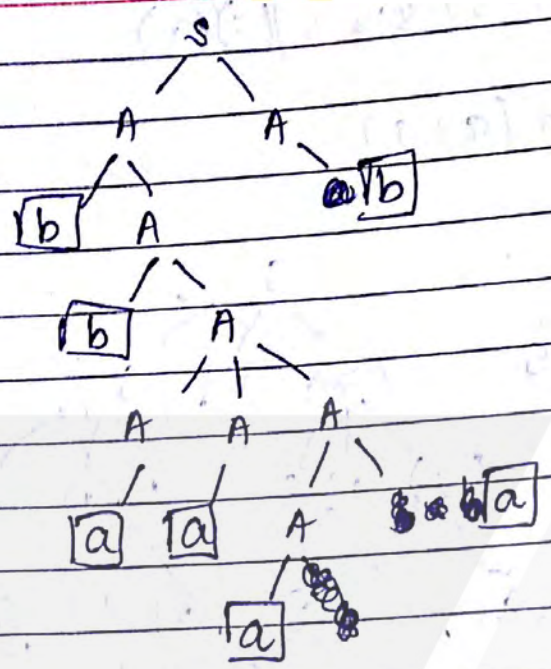
$S \to b\underline{A}A$

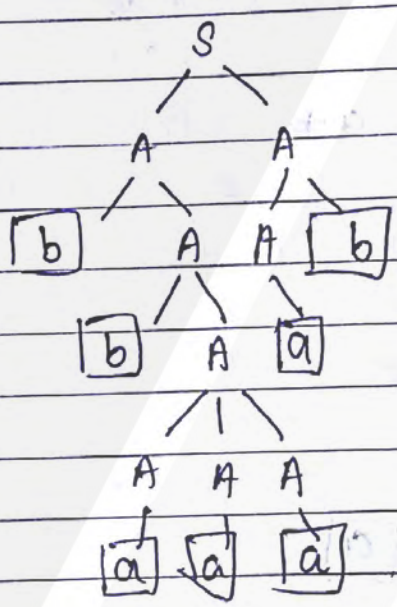
$S \to bb\underline{A}A$

$S \to bb\underline{A}\underline{A}\underline{A}A$

$S \to bbaaa\underline{A}b$

$S \to bbaaaab$





Yield of a parse tree:-
It is a concatination of labels of leaf node in the left to right ordering without repetition.

Q4.

* L.M.D.

A derivation S derives w after multiple substition using grammar G is called leftmost if production is applied to the left most variable (non-terminal) at every state.

R.M.D.

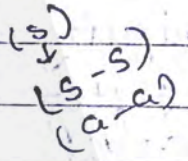
A derivation S derives w after multiple substitution using grammar G is called rightmost if the production is applied only to the right most non terminal at every step.

$S = S-S \mid S*S \mid a$ $\quad w = a-a*a$

L.H.D.	R.M.D.
$S \to S-S$	$S \to S-S$
$\to a-S$	$= S-S*S$
$\to a-S*S$	$= S-S*a$
$\to a-a*S$	$= S-a*a$
$\to a-a*a$	$= a-a*a$

$S \to S-S \mid S*S \mid a \mid (S)$

$w = (a-a)*a$



L.M.D.	R.M.D.
$S \to S*S$	$S \to S*S$
$\Rightarrow (S)*S$	$\Rightarrow (S)*S \quad S \to a$
$\Rightarrow (S-S)*S$	$\Rightarrow (S)*a$
$\Rightarrow (a-S)*S$	$\Rightarrow (S-S)*a$
$\Rightarrow (a-a)*S$	$\Rightarrow (a-S)*a$
$\Rightarrow (a-a)*a$	$\Rightarrow (a-a)*a$

Q1. $S \to AB \mid \epsilon$

$A \to aB$ $\quad w = aabbbb$

$B \to Sb$

$S \to AB$

$\to aBB$ $\quad aaSbSbbSb$

$\to aSbB$ $\quad \to aa\epsilon b\epsilon bb\epsilon b$

$\to aABbB$

$\Rightarrow aaBBbB$

$\Rightarrow$

B → ε

draw parse tree of both

L.M.D.	R.M.D.
S → AB	S → AB
→ aBB	S ⇒ → ASb
→ aSbB	→ AABb
→ aABbB	→ AASbb
→ aaBBbB	
→ aaSbBbB	
→ aaεbBbB	
→ aaεbSbbB	
→ aaεbεbbB	
→ aaεbεbbSb	
→ aaεbεbbεb	

Ambiguity :→ A grammar is called ambiguous if for atleast one word in the language that it generates, there are more than one possible derivation of the word that corresponds to diff. syntax trees. A non-ambiguous C.F.G. is called un-ambiguous.

S = aSb/SS/ε is ambiguous or not.

w = abab.

S → SS	S → SS
→ aSbS	→ SaSb
→ aεbS	→ SaεB
→ aεbaSb	→ asbaεb
→ aεbaεb	→ aεbaεb

Check aabb also.

Q. $S \to SbS/a$ is ambigious or not.

$w = ababa$ $\quad w = ababa$

$S \to \underline{S}bS$ → ambigious.

$\to a\underline{b}S$

$\to ab\underline{S}bS$

$\to ab\cancel{ab}\underline{S}bS$

$\to ab$

$\underline{S}bs$

$ab\underline{S}$

$ab\underline{S}bs$

$ababS$

ababa

Inherent Ambiguity :→

If L is a C.F.L for which there exist one ambigious grammar then L is said to be ambigious. If every grammar that generate L is ambigious then the language L is said to be inherently ambigious.

in the

that

(i) designing CFG for languages with linked substring like $u^n \# v^n$. A rule of the form $R \to uR_v \mid \#$ will be

$$L = \{a^n b^n \mid n \geq 0\}$$
$$S \to (asb \mid \epsilon)$$

$$\{a^n d b^n \mid n \geq 0\}$$

$$S \to asb \mid d$$

$$L = \{a^n b^n \mid n > 0\}$$

$$S \to asb \mid ab$$

(ii) If C.F.G. is a union of several C.F.G.'s rename non-terminals so that their disjoints and add new rule to the grammar

$$S \to S_1 \mid S_2 \mid S_3 \cdots S_n$$

$$L = \{\{0^n 1^m \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}$$

$$S_1 \to 0S_1 1 \mid \epsilon \qquad S_2 \to 1S_2 0 \mid \epsilon$$
$$S_1 \to 0S_1 1 \mid \epsilon$$
$$S_2 \to 1S_2 0 \mid \epsilon$$
$$S \to S_1 \mid S_2$$

(iii) To construct a C.F.G. for a regular language follow a D.F.A. as follows :→

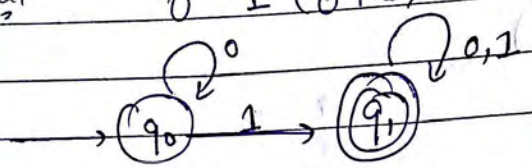
a) for initial state q_0 make R_0 the start symbol.
b) for state transition

$$\delta(q_i, a) \vdash q_j$$
$$R_i \to aR_j$$

for each final state q_f

$R_f \vdash \epsilon$

Q% $0^* 1 (0+1)^*$



q_i, a

$(q_0, 0) \vdash q_0^{q_i}$

$R_0 \rightarrow 0 R_0$

$R_0 \rightarrow 1 R_1$

$R_{0,1} \rightarrow 0 R_1$

$R_1 \rightarrow 1 R_1$

$R_1 \rightarrow \epsilon$

GNF
Greibach Normal form

** Chomsky Normal form :-

* A grammar is in C.N.F. if it has prodn of the form $V_N \rightarrow V_N V_N$

$V_N \rightarrow V_T$

$S \rightarrow C_a B$ ✓ (is in C.N.F.)

$C_a \rightarrow a$

$S \rightarrow A B C$ } (V) (will be in C.N.F)

$AB \rightarrow D_{AB}$

$S \rightarrow D_{AB} C$

$S \rightarrow A B C$ } X

$S \rightarrow a$ ✓

$S \rightarrow a B$ } X

Q%

Q%

Q%

Archies

$S \to \overbrace{A\ b}\ D$

$S \to C_{Ab}\ D$

$C_{Ab} \to Ab \to (C_b) \to (AC_b)$

~~$S \to A$~~ $\quad C_b \to b$

Q4. Convert grammer into C.N.F.

$S \to bA \mid aB$

$A \to bAA \mid aS \mid a$

$B \to aBB \mid bS \mid a$

$S \to C_bA \mid C_aB$

$A \to C_{bA}A \mid ~~aS \mid a~~\ C_aS \mid ~~C_a~~\ a$

$B \to C_aBB \mid C_bS \mid a$

$C_b \to b$

$C_a \to a$

$C_{bA} \to bA$

$C_{aB} \to aB$

bA

bbAA

$C_{bA} \to bA$

$C_{bA} \to bA$

$C_{aB} \to aB$

$C_b \to b$ ✓

$C_a \to a$ ✓

bAA

$C_{bA} \to bA$ ✓

$C_{aB} \to aB$ ✓

Q4. $\begin{cases} S \to abSb \mid a \mid aAb \\ A \to bS \mid aAAb \end{cases}$

$S \to C_{ab}C_{Sb} \mid C_aC_{Ab}$

$C_{ab} \to aC_b$

$C_b \to b$

$C_{Sb} \to SC_a$

$C_a \to a$

$C_{Ab} \to AC_b$

$A \to bS \mid aAAb$

$A \to C_bS \mid C_aC_{Ab}$

$C_b \to b$

$C_{aA} \to C_aA$

$C_a \to a$

$C_{Ab} \to AC_b$

TUT

Q4. CFG? $L = \{a^m b^m c^m d^n \mid n, m \geq 0\}$

$S_1 \to aS_1bc \mid dS_1 \mid a$

~~abcd~~

$S \to aSd \mid \epsilon$

$S \to aS \mid \epsilon$

$S \to asd \mid S_1 \mid \epsilon$

$S_1 \to bS_1c \mid \epsilon$

Q. $L = \{a^n b^n c^m d^m \mid n, m \geq 0\}$

$S \to aSbS \mid S_1 \mid \epsilon$

$S_1 \to cS_1d \mid \epsilon$

doubt = ?

$S \to$

Q. $L = \{a^n b^m \mid n < m\}$

$S \to aS \mid bS \mid \epsilon$

$asbs \mid a \mid \epsilon$

$S \to asbs \mid b \mid \epsilon$

$S \to asb \mid b \mid sb$ ✓

Q. $L = \{a^n b^m \mid n \leq m\}$

$L = \{a^n b^n \mid n < m\} \cup \{a^m b^m \mid n = m\}$

$S_1 \to aS_1b \mid b \mid Sb$

$S_2 \to aS_2b \mid S_2 \mid \epsilon$

$S \to S_1 \mid S_2$

GNF :→

A prodⁿ is in GNF if it is of the following :

$A \to a\alpha$

where, $A \to V_N$ & $a \in V_T$ & $\alpha \in (V_N)^*$

$S \to a$ ✓

$S \to A$

$S \to aA$

$S \to aABCD$ ✓

$S \to aAb$.

Lemma 1 :→

If a CFG $G = \{V_N, V_T, P, S\}$ where P contains

$A \to \beta Y$

$\beta \to \beta_1 | \beta_2 | \beta_3 | \ldots \beta_s$ then we define a new grammar G in which $P = \{P - \{A \to \beta Y\}\} \cup \{A \to \beta_i Y \mid 1 \le i \le s\}\}$

Q1. $S \to AB$

$A \to a | b | c$

$S \to aB | bB | cB$

$A \to a | b | c$

Q2. $A \to BC$

$B \to a | aAd | e$

$A \to aC | aAdC | eC$

$B \to a | aAd | e$

Lemma 2 :→

Let $G = \{V_N, V_T, P, S\}$ & the set of prodⁿ are

$A \to A\alpha_1 | A\alpha_2 | A\alpha_3 \ldots | A\alpha_n$

$A \to \beta_1 | \beta_2 | \beta_3 \ldots \beta_s$

then, we define a new non-terminal Z in the grammar with $V_N = \{V_N \cup Z\}$ & the set of prodⁿ are as follows :→

$A \to \beta_1 | \beta_2 | \beta_3 | \ldots | \beta_s$

$A \to \beta_1 x \mid \beta_2 x \mid \beta_3 x \mid \dots \mid \beta_s x$

$x \to \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \dots \mid \alpha_m$

$x \to \alpha_1 x \mid \alpha_2 x \mid \alpha_3 x \mid \dots \mid \alpha_m x$

Q4. Convert the following grammar into GNF.

$S \to AA \mid a$

$A \to SS \mid b$

Step 1 :→

Convert the given grammar into CNF.

$S \to AA \mid a$

$A \to SS \mid b$

Step 2 :→

Rename non-terminals.

$A_1 \to A_2 A_2 \mid a$

$A_2 \to A_1 A_1 \mid b$

choose higher to lower

$A_0 \to BY$ → applying lemma 1.

$A_2 \to A_1 A_1$

$A_2 \to b$

$A_2 \to bA_1$

$A_2 \to b$

$A_2 \to A_2 A_2 A_1 \mid aA_1$

$A_2 \to b$

→ applying lemma 2

A → Aα β1 β2

$A_2 \to A_2 A_2 A_1 \mid aA_1 \mid b$

α

$A_2 \rightarrow aA_1 \mid b \mid aA_1Z \mid bZ$ — ①

$Z \rightarrow A_2A_1 \mid A_2A_1Z$

↳ applying lemma 1.

$A \rightarrow \beta\gamma$

$Z \rightarrow aA_1A \mid bA_1 \mid aA_1ZA_1 \mid bZA_1 \mid aA_1A_1Z \mid bA_1Z \mid aA_1ZA_1Z \mid bZA_1Z$

↳ ②.

$A_1 \rightarrow A_2A_2 \mid a$

$A_1 \rightarrow aA_1A_2 \mid bA_2 \mid aA_1ZA_2 \mid bZA_2 \mid a$ //

Q. $S \rightarrow AB$ Convert grammar into GNF

$A \rightarrow BS \mid b$

$B \rightarrow SA \mid a$

$A_1 \rightarrow A_2A_3$
$A_2 \rightarrow A_3A_1 \mid b$
$A_3 \rightarrow A_1A_2 \mid a$
} all in C.N.F.

$A \rightarrow \beta\gamma$

$A_3 \rightarrow A_1A_2$ lemma 1

$A_3 \rightarrow a$

$A_3 \rightarrow A_1A_2 \mid a$

~~$A_3 \rightarrow A_3A_1A_2 \mid bA_2$~~ $A_3 \rightarrow A_2A_3A_2 \mid a$

$A_3 \rightarrow A_2A_3A_2 \mid a$

↳ lemma 1.

$A_2 \rightarrow A_3A_1 \mid b$ (β_1 β_2)

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

$A_3 \rightarrow A_2 A_1 A_3 A_2 \mid bA_3A_2 \mid a$

Lemma 2

$A \rightarrow A\alpha$

$A_2 \rightarrow bA_2A_2 \mid b \mid bA_3A_2Z \mid aZ$

$X \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z$

$A_2 \rightarrow A_3A_1 \mid b$

$A_2 \rightarrow bA_3A_2A_1 \mid aA_1 \mid bA_3A_2ZA_1 \mid aZA_1 \mid b$

$A_1 \rightarrow A_2A_3$

$A_1 \rightarrow bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2ZA_1A_3 \mid aZA_1A_3 \mid bA_3$

$X \rightarrow bA_3A_2A_1A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_2ZA_1A_3A_2 \mid aXA_1A_3A_3A_2 \mid bA_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2Z \mid aA_1A_3A_3A_2Z \mid bA_3A_2ZA_1A_3A_3A_2Z \mid aZA_1A_3A_3A_2Z \mid bA_3A_3A_2Z$

properties

Closure of CNF & ~~GNF~~ Regular language.

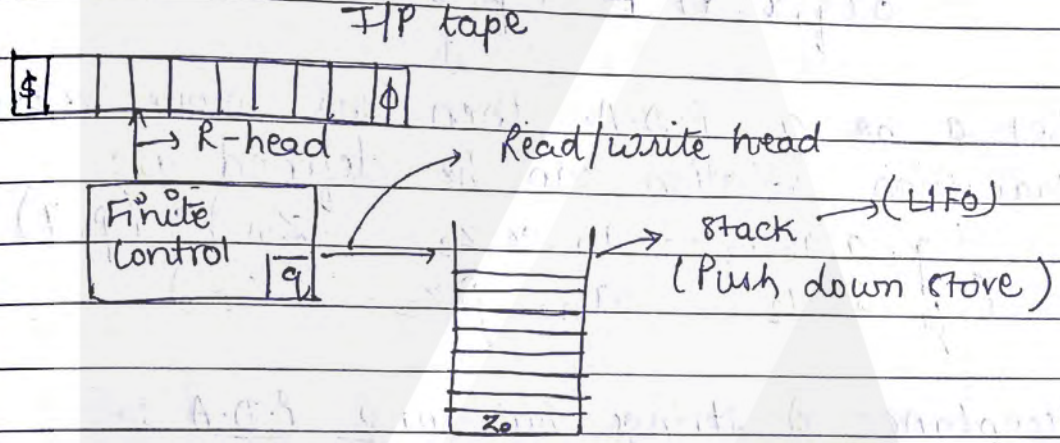
9/03/17

Date...../...../.....

Push down Automata

↳ C.F.L.

P.D.A. is essentially a finite automata with control of both input tape & a stack to store what it has read.



It is defined by 7 tuples

P.D.A = $\{ Q, \Sigma, \tau, \delta, q_0, z_0, f \}$ where

Q → non empty finite set of states

Σ → " " " " input.

q_0 → " " intial state ($q_0 \in Q$)

τ → is the set of push down symbols

z_0 → intial symbol on the stack

δ → transition function.

f → final state

$\delta(q_0, a)$ ✗

$Q \times \Sigma \times \tau$ into Q

$Q \times \{\Sigma \cup \lambda\} \times \tau \rightarrow Q \times \tau^*$

P.D.A

D.P.D.A.
Deterministic

N.P.D.A.
Non-deterministic.

$\epsilon \rightarrow pop$.

Representation of P.D.A. using I.D (Instaneous description).

It is represented as $\delta(q, x, a)$ where q represents the present state, x represents current input symbol and a represents top of the stack.

$$\delta(q, x, a) \vdash (p, \underset{y}{\gamma})$$

Let a be a P.D.A. then the move relation or transition relation can be defined as:

$$(q, a_1 a_2 a_3 \cdots a_n, z_1 z_2 \cdots z_n) \vdash (p, \gamma)$$
$$(q', a_2 a_3 \cdots a_n, \beta z_2 \cdots z_m)$$

Acceptance of strings by using P.D.A :-

The final acceptance of strings can be defined in terms of

(i) final state
(ii) Null store

(i) Acceptance by final state

Let $P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, f\}$ be a P.D.A. The set accepted by P.D.A. in terms of final state denoted by $T(P)$ is defined as

(* : after multiple transition; λ : empty input)

$$T(P) = \{\omega \in \Sigma^* \mid (q_0, w, z_0) \vdash^* (q_f, \lambda, z_0) \text{ for } q_f \in F\}$$

$$N(P) = \{\omega \in \Sigma^* \mid (q_0, \omega, z_0) \vdash^* (q_f, \lambda, \lambda) \text{ for } q_f \in F\}$$

CRC-ASET ATTC-AU

$a^n b^n$

$(q, a, z_0) \vdash (q, \epsilon)$

$(q, b, a) \vdash (q, \epsilon)$

Date...../...../.....

Page 27

input string

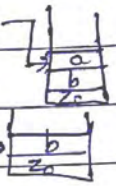
$(q, \epsilon, z_0) \vdash (q, z_0)$ null store

$(q, \epsilon, z_0) \vdash (q, \epsilon)$

Let $P = \{Q, \Sigma, \delta, \Gamma, q_0, z_0, f\}$ be a P.D.A. The set accepted by P.D.A in terms of null store or empty store is denoted by $N(P)$ & is defined as above.

$(q, a, b) \vdash (q, \epsilon)$ → (Pop)

$(q, a, b) \vdash (q, ab)$ → Push a on top of b

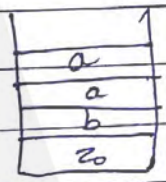


$(q, a, b) \vdash (q, a)$ → Pop b push a

$(q, a, b) \vdash (q, aa)$

II I

$(q, a, b) \vdash (q, aab)$



$(q, a, b) \vdash (q, b)$ → do nothing (pop then push)

diff. b/w
P.D.A & C.F.G. & vice versa :⇒

C.F.G. to P.D.A. :⇒

→ If a given C.F.G. is $G = \{P, S, V_N, V_T\}$ then its equivalent P.D.A. will be as follows ($z_0 \neq \Gamma$)

$$A = \{\{q\}, \{V_T\}, \{V_N \cup V_T\}, \delta, q, S, \phi\}$$
$$\Sigma \qquad \Gamma \text{ (stack symbols)}$$

where

δ can be defined as follows :-

(i) If the P.D.A. leads a non-terminal symbol say A on the top of the stack then it makes a λ move by placing right hand side of all A prodⁿ after erasing A.

(ii) If a P.D.A. reads a terminal symbol on the stack & if it matches with the current input then the P.D.A. erases A.

(iii) In other cases P.D.A. haults.

Q: $S \to OBB$
$B \to OS \mid 1S \mid 0$

Construct a P.D.A. equivalent to given grammer

$A = \{q\}\{0,1\}\{0,1,S,B\}\delta, q, S, q\}$

δ :-
1. $(q, \epsilon, \epsilon) \vdash (q, S)$
2. $(q, 1, 1) \vdash (q, \epsilon)$ — terminal
3. $(q, 0, 0) \vdash (q, \epsilon)$ — terminal
4. $(q, \epsilon, S) \vdash (q, OBB)$ — non-terminal
5. $(q, \epsilon, B) \vdash (q, OS) / (q, 1S) / (q, 0)$ — non-terminal

C

Date...../...../.....

Page 1 (21)

OSOS

Q. Equal no. of a's & equal no. of b's.

$S \rightarrow asa \mid bsb \mid \epsilon$

$A = \{ \{q\} \{a,b\} \{a,b,s\} \delta, q, s, q \}$.

1. $(q, \epsilon, \epsilon) \vdash (q, s)$
2. $(q, a, a) \vdash (q, \epsilon)$
3. $(q, b, b) \vdash (q, \epsilon)$
4. $(q, \epsilon, s) \vdash (q, asq) \mid (q, bsb) \mid (q, \epsilon)$

Q. P.D.A. for the language accepting

$L = \{ w c w^r \mid w \in (0,1)^* \}$

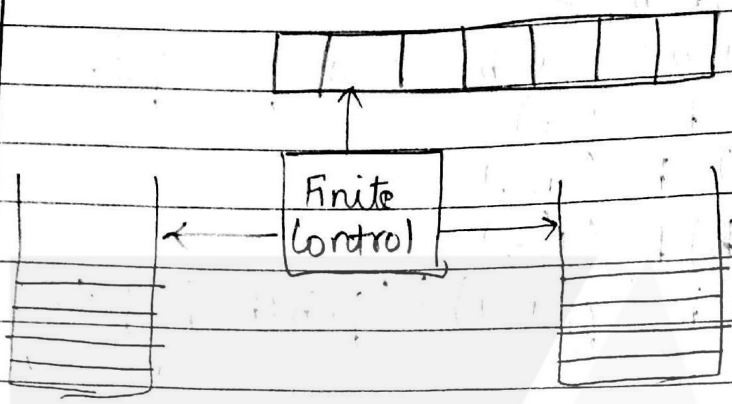
$S \rightarrow 0S0 \mid 1S1 \mid c$

$A = \{ \{q\} \{0,1\} \{0,1,c,s\} \delta, q, s, q \}$

1. $(q, \epsilon, \epsilon) \vdash (q, s)$
2. $(q, 0, 0) \vdash (q, \epsilon)$
3. $(q, 1, 1) \vdash (q, \epsilon)$
4. $(q, c, c) \vdash (q, \epsilon)$
5. $(q, \epsilon, s) \vdash (q, 0s0) \mid (q, 1s1) \mid (q, c)$

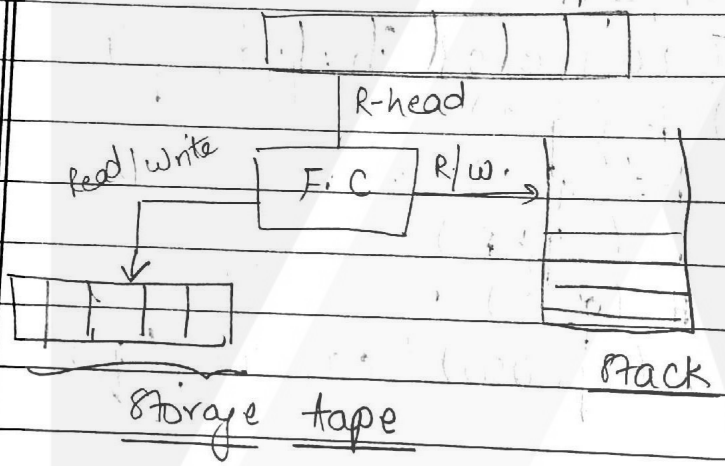
only definition

Two Stack P.D.A. :-



$Q \times \Sigma \times \Gamma \times \Gamma^* \rightarrow Q \times \Gamma \times \Gamma^*$ → ? (check)

Auxilary P.D.A. :-



(ω)

$Q \times \Sigma \times \Gamma \times \omega \rightarrow Q \times \Sigma \times \Gamma^* \times \omega\,(L, R)$

Pumping lemma for C.F.G.

Step 1 :-

Assume L is context free. Let n be a natural no. obtained by pumping lemma.

a string

(iii) Choose $z \in L$ such that $z \in L$ so that length of z is greater than equal to L ($|z| \geq L$).

Write $z = uvwxy$ using pumping lemma such that length of ($|vx| \geq 1$) & length of $|vwx| \leq n$.

Step 3 :-

Find a suitable integer k so that your $uv^k wx^k y \notin L$ This contradicts our assumption. Hence L is not context free.

Q) Prove that $\{L = a^n b^n c^n \mid n \geq 0\}$ is not a C.F.G.

Step 1 :- Assume L is context free. Let n be a natural no. obtained by pumping lemma.

2) let $z = uvwxy$ so that $a^n b^n c^n$, now we divide $z = uvwxy$ so that length of $|vx| \geq 1$ & length of $|vwx| \leq n$.

$z : a^{n-1}\ a\ b^{n-1}\ b\ c^n$

$a^{n-1}\ a\ b^{n-2}\ b\ bc^n$

for $a^{n-1}\ a\ b^{n-2}$ for $k = 2$

$a^{n-1} \cdot a^2 \cdot b^{n-2} \cdot b^2 \cdot bc^n$

$= a^{n+1} b^{n+1} c^n \notin L$

Q. 1) $L = a^p$ such that p is prime no. is not C.F.L

Q. 2) $L = a^{n^2}$ where n is natural no is not C.F.L.

Decision Algorithm :- (for CFGs)

I) Algorithm for deciding whether a CFL L is empty or not.

Step 1 :-
for each non terminal n that has some production of the form $V_N \to a$ we choose one of them & put in all the other productions having V_N on the right hand side.

Step 2 :-
Repeat step 1 until it eliminates S.

If S has been eliminated then CFG produces some words otherwise not.

Q) $V_N \to a$

$S \to XY$	$S \to XY$	$S \to acbb$
$X \to AX$	$X \to aX$	$X \to aaa$
$X \to AA$	$X \to aa$	$X \to aa$
$A \to a$	$A \to a$	$A \to a$
$Y \to BY$	$Y \to Bby$	$Y \to bbb$
$Y \to BB$	$Y \to bb$	$Y \to bb$
$B \to b$	$B \to b$	$B \to b$

Q)

$S \to XY$	$S \to XY$	$S \to (X)bb$ → empty
$X \to AX$	$X \to aX$	$X \to aX$
$Y \to BY$	$Y \to bY$	$Y \to bbb$
$A \to a$	$A \to a$	$A \to a$
$Y \to BB$	$Y \to bb$	$Y \to bb$
$B \to b$	$B \to b$	$B \to b$

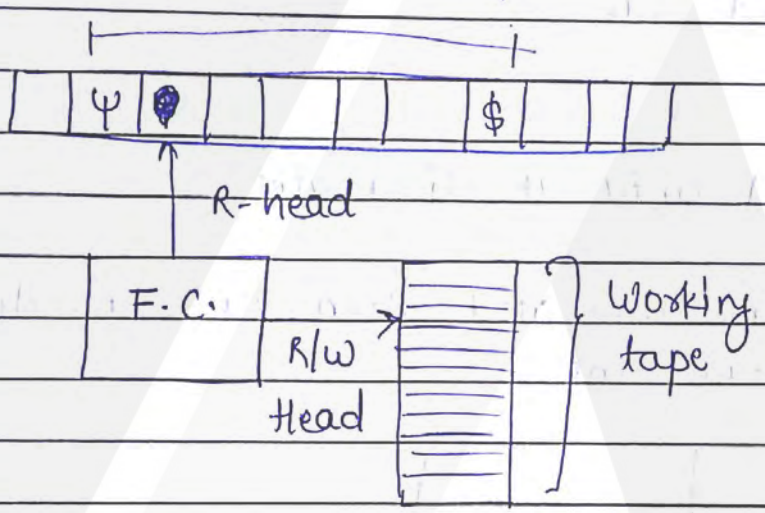
read

Date :
Page No. :

Linear Bounded Automata

It is used to recognize type 1 context sensitive language. In this case infinite storage is ~~preesit~~ restricted in size. It is called so because a linear fun is used to bound the length of a tape. L.B.A. is a non-deterministic turing machine which has a single tape whose length is infinite but bounded by linear fun. It is defined by using 9 tuples :-

$$\{Q, \Sigma, \tau, \delta, q_0, b, \psi, \$, F\}$$



b :→ is the blank symbol

δ :→

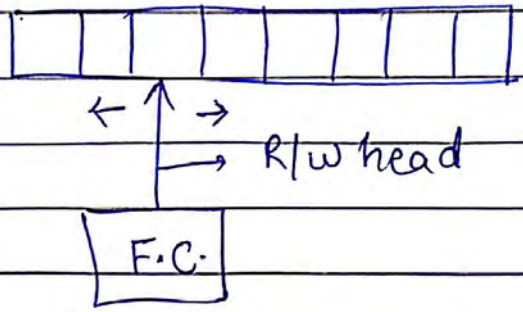
$$Q \times \Sigma \times \tau \longrightarrow Q \times \tau (L, R)$$

ψ → Start limiter

$ → end limiter.

Turing machine :-

It is a symbol mathematical model of a general purpose computer. Turing machine is capable of performing any calculation which can be performed by any computer machine.



tape is of infinite length.

It is defined by 7 tuples
$\{Q, \Sigma, \tau, \delta, q_0, b, F\}$

→ Here Σ is a subset of τ
→ τ set of all input tape symbols.
→ b → blank symbol $(b \in \tau)$ not $\in \Sigma$
→ δ δ

$$Q \times \tau \rightarrow Q \times \tau (L, R)$$

Representation of turing machine :→

1). ID (Instantaneous description)
2). Transition table.
3). Transition diagram.

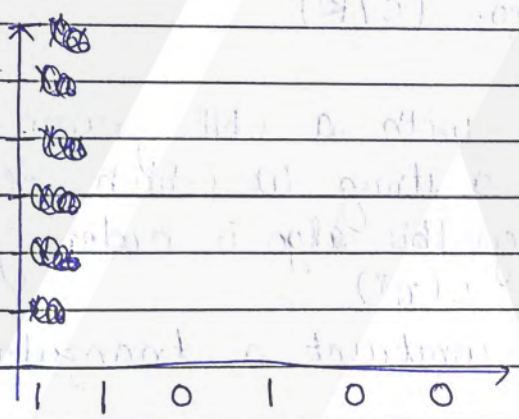
How to compute X_{ij}

$X_{ij} \neq X_{ii} / X_{i+1}$

$X_{ij} = X_{ii}$

$X_{ij} = (X_{11}, X_{25}), (X_{12}, X_{35}), (X_{13}, X_{45}), (X_{14}, X_{55})$

Qy.
$S \rightarrow AB \mid BC$
$A \rightarrow BB \mid 0$
$B \rightarrow BA \mid 1$
$C \rightarrow AC \mid AA \mid 0$
$w \rightarrow 110100$



①

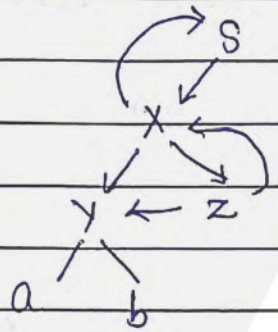
{S,B}	{A,C}				
{S,B}	{A,C}	{S}			
{A,C}	{A}	{S,A}	{S,B}		
{A}	{S,B}	{S}	{B,S}	{C}	
{B}	{B}	{A,C}	{B}	{A,C}	{A,C}
1	1	0	1	0	0

$X_{12} = (X_{11}, X_{22})$ Merge
B B $\Rightarrow$ BB (correspondiy non terminal $\rightarrow$ A)

*

II). Algorithm to define whether a C.F.L. is finite or not

Q1. $S \to XS/b$
$X \to YZ$
$Z \to XY$
$Y \to ab$



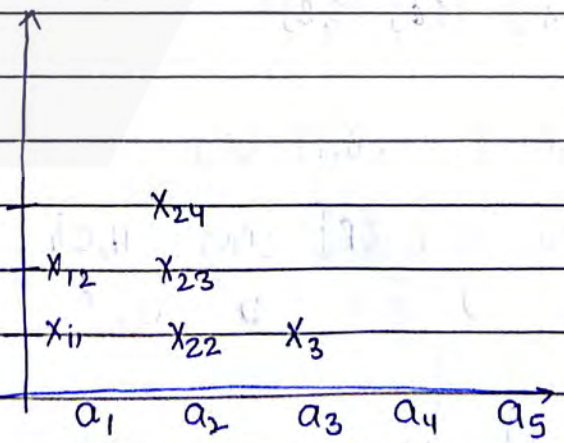
cycles (so infinite).

III). Membership algo. (CYK)

This algo starts with a CNF grammer. The input to CYK algo is a string w which belongs to Σ^*.
The complexity of this algo is order of n^3
$O(n^3)$

In this algo we construct a triangular table as follows:
Along the horizontal axis we write bits of strings.

$\Rightarrow \quad w = a_1 a_2 a_3 a_4 a_5$



X_{23} = X_{22}, X_{33}
(B {A,C}) → Merge
(BA, BC)
↳ {S, B}

X_{34} = X_{33} . X_{44}
({A,C}, {B}) → Merge.
(AB, CB)
↳ S.

X_{45} = X_{44}, X_{55}
= ({B}, {A,C}) → Merge.
(BA, BC)
↳ {B, S}

X_{56} = X_{55}, X_{66}
{A,C} {A,C}
({AA, AC, CA, CC})
↳ C

X_{13} = (X_{11}, X_{23}), (X_{12}, X_{33})
= (B, {S, B}) , (A, {A,C})
= (BS, BB) , (AA, AC)
(A, C)

X_{24} = (X_{22}, X_{34}) , (X_{23}, X_{44})
= (B, {S}) , ({S, B}, B)
= (BS), (SB, BB)
(A)

X_{35} = (X_{33}, X_{45}) , (X_{34}, X_{55})
= ({A,C}, {B,S}) , (S, {A,C})
= (AB, AS, CB, CS) , (SA, SC) ⇒ S

$X_{46} = (X_{44}, X_{56}), (X_{45}, X_{66})$

$= (B, C), (\{B,S\}, \{A,C\})$

$= (BC), (BA, BC, SA, SC)$

$S \qquad B, S$

$X_{14} = (X_{11}, X_{24}), (X_{12}, X_{34}), (X_{13}, X_{44})$

$= (B, A) \qquad (A, S) \qquad (B, \{A,C\}, B)$

$= B, S$

$X_{25} = (X_{22}, X_{35}), (X_{23}, X_{45}), (X_{24}, X_{55})$

$= \{B,S\}, (\{S,B\}, \{S,B\}), (A, \{A,C\})$

A, C

$X_{36} = (X_{33}, X_{46}), (X_{34}, X_{56}), (X_{35}, X_{66})$

$= (\{A,C\}, \{S,B\}), (S, C), (S, \{A,C\})$

S, B

$X_{15} = (X_{11}, X_{25}), (X_{12}, X_{35}), (X_{13}, X_{45}), (X_{14}, X_{55})$

$= (B, \{A,C\}), (A, S), (\{A,C\}, \{B,S\}), (\{B,S\}\{A,C\})$

$= (BA, BC), (A, S), (AB, AS, CB, CS), (BA, BC, SA, SC)$

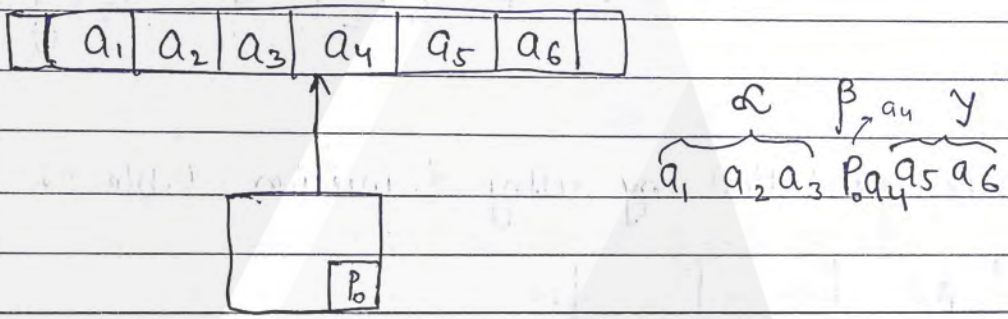
$(B, S) \qquad (S) \qquad (B, C)$

Date...../...../.....

Page..... (1)

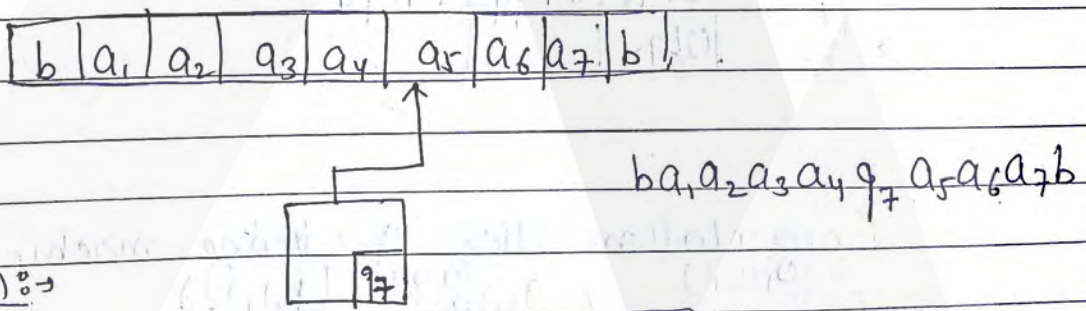
Representation of Turing machine :-

The ID (Instaneous description) of a turing machine is $\alpha\beta\gamma$ where β is the present state. The entire string is splitted as α & γ. The first symbol of γ say a is the current symbol under R/W head & γ has all the subsiquent symbol of the input string after a. The string α is the sub-string of input string formed of all the symbols left to a.

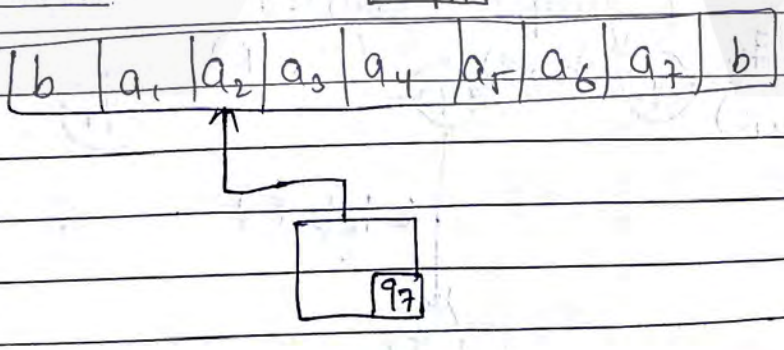


Transition of turing machine :-

Right transition:



left transition :-



Date...../...../.....

Page.....

$x_1, x_2, x_3, x_4 \dots x_i \dots x_n$

$(p, x_i) \vdash (q, x_i, R)$

→ $x_1 x_2 x_3 \dots x_{i-1}\ p\ x_i \dots x_n \vdash x_1 x_2 x_3 \dots x_{i-1} x_i\ q\ x_{i+1} \dots x_n$

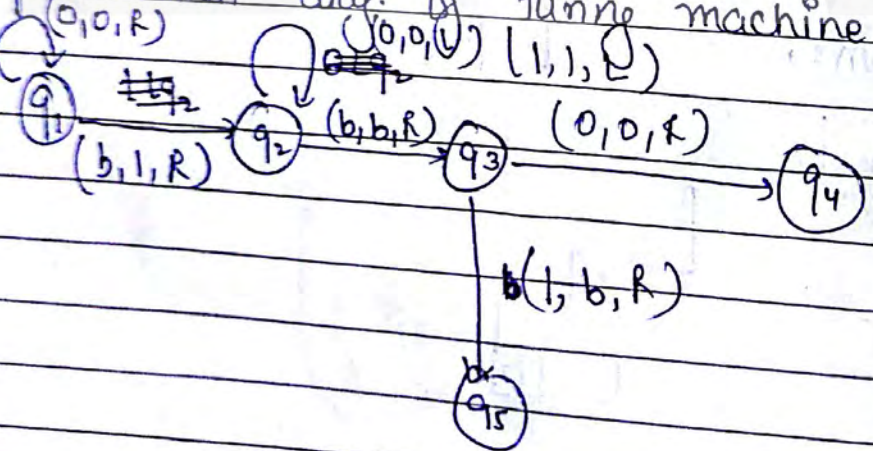
$(p, x_i) \vdash (q, y, L)$

$\vdash x_1 x_2 x_3 \dots x_{i-1}\ q\ x_{i-1} \dots x_n$.

Representation by using transition table :→

P.S	b	0	1
→ q_1	$1Lq_2$	$0Rq_1$	
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3		bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
* q_5	$0Lq_2$		

Representation diag. of turing machine :→



Date...../...../.....

Page.....

$0q_1 0b$

language acceptibility by turing machine:

Let $m = \{Q, \Sigma, \tau, \delta, q_0, b, F\}$ be a given turing machine. A string w in Σ^* is said to be accepted by m if $q_0 w \vdash^* \alpha_1 p \alpha_2$ for some $p \in F$ and $\alpha_1, \alpha_2 \in \tau^*$.

Q4 Draw the computation sequence or Check for the acceptance of $w = 00$ for the above given machine.

(can insert any no. of blank spaces to left or right)

$q_1 00b \vdash 0q_1 0b \vdash 00q_1 b \vdash 0q_2 01 \vdash bq_2 001$

$\vdash q_3 b001 \vdash bq_3 001 \vdash bbq_4 01 \vdash$

$bb0q_4 1 \vdash bb01q_4 b \vdash bb010q_5 b \vdash$

$bb01q_3 00 \vdash$

$bb0q_2 100 \vdash$

$bbq_2 0100 \vdash$

$bq_2 b0100 \vdash$

Designing T.M.:-

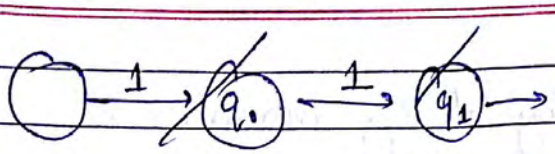
$Q \times \Sigma \times \tau \to Q \times \tau (L, R)$

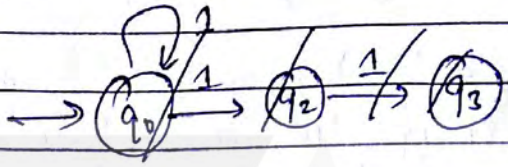
The state will be changed only when there is change in the written symbol or change in the movement of R/W head.

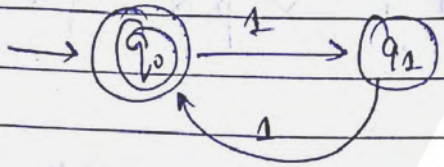
Q1. Design a turing machine to recognize all strings existing of even no. of 1's over the input alphabet $\Sigma = 1$.

Q.2). design a turing machine over 1, b which can compute concatenation fun over the input alphabet $\Sigma = 1$.

$G \to a$ $A \to a$

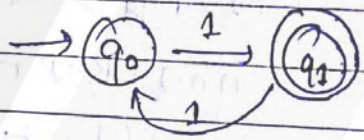






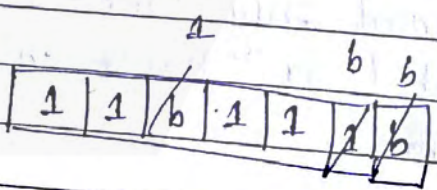
	1	b
→ * q_0	b R q_1	–
q_1	b R q_0	–

⇒ odd no. of 1.



	1	b
→ q_0	b R q_1	–
* q_1	b R q_0	–

2).



	1	b
→ q_0	1 R q_0	1 R q_1
q_1	1 R q_1	b L q_2
q_2	b R q_3	–
* q_3		

~~aabbbbcc~~ zRq3 1011b 1011b

Date / Page

design a turing machine for $1^n 2^n 3^n$

$a^n b^n c^n$

$a^n b^n$ ✓

equal no.'s of a's & b's & c's

palindrom ww^r

design a turing machine for $1^n 2^n 3^n$ such that $n \geq 0$.

b

1 1 1 | 2 2 2 2 | 3 3 3 3

x y z

bLq3

	1	2	3	b	x	y	z
→ q_0	xRq_1						
q_1	$1Rq_1$	yRq_2				yRq_2	
q_2		$2Rq_2$	zRq_3				zRq_2
q_3	$1Lq_3$	$2Lq_3$			xRq_{80}	yLq_3	zLq_3
q_4				bRq_5	xLq_4	yLq_4	xLq_4
*q_5							

Turing machine can act as a copy machine.

Universal turing machine.

A general purpose turing machine is called as

(i) the input data.
(ii) description of computation at i.e. an algorithm.

Multiple heads (R/w heads more than one).

K-dimensional

Haulting prob of turing machine:-
The haulting prob. of a turing machine over the input alphabet Σ is unsolvable i.e. the problem of determining whether or not an arbitary T.M. over Σ haults for an arbitary input x in Σ^* is unsolvable.

Unsolvable problem:- A class of problems with two outputs (Yes or No) is said to be solvable or decidable if there exist some definite algo which always terminate with two outputs i.e. Yes or No otherwise the class of prob. is unsolvable.

Church's Thesis
No computation procedure will be considered as an algo unless it can be represented as turing machine. This statement is called

non ← G→a A→a

Church's Thesis is given by Alonzo church in 1936.

(8 marks)

* PCP (Post correspondance problem) (MPCP)

Consider two list $x = x_1, x_2, x_3 \dots x_n$ and $y = y_1, y_2, \dots y_n$ of non empty strings over the same input alphabet Σ. The PCP is to determine whether or not there exist $i_1, i_2, i_3 \dots i_m$ where $1 \le i_j \le n$ such that $x_{i_1} x_{i_2} x_{i_3} \dots x_{i_n} = y_{i_1} y_{i_2} \dots y_{i_n}$

If There exists a sol^n to PCP then there exists infinitely many sol^n. Thus the PCP with the following list

$x = \{b, bab^2, aba\}$ and $y = \{b^3, ba, a\}$ has the sol^n or not.

for ex:-

	X	Y
x_1	11	01
x_2	10	1
x_3	11	011

	2	1	
X	bab^2	b	bab^3
Y	ba	b^3	

$x = \{b, bab^3, ba\}$ $y = \{b^3, ba, a\}$

2	1	1	3	
bab^3	b	b	ba	bab^6a
ba	b^3	b^3	a	bab^6a

$(2113)^n$

Q4.

x	y
ab	ba^2
aa	bba
bab	ab^2

No s/n because first bit for each x & y are different.

Q4

x	y
0 1	0110
10	1001
11	1001

length of the strings in y is lesser than the length of string x i.e. no s/n.

Q4

x	y
ab	aba
bab	bab^2
aa	aa
baa	ba^6

$\longrightarrow (3)^n$

Q4.

x	y
x_i	y_i
1	111
10111	10
10	0.

10111110 10111110

$(2113)^n$.

x	y
01	011
11	101^2
101^2	11
011	

4 3

No. of strings in both the list are diff.

Modified PCP :→

If the first substring used in PCP is always x_1 & y_1 then the pCP is known as MPCP. Let $x = x_1, x_2, \dots x_n$ and $y = y_1, y_2, \dots y_n$ then the MPCP has a solution of the form

$$x_1\, x_{i_1}\, x_{i_2} \dots x_n = y_1\, y_{i_1}\, y_{i_2} \dots y_n$$

Ex:→

$x = \{10, 011, 101\}$

$y = \{101, 11, 01\}$

→ (13)

10101

10101

[Type 0]

The language accepted by T.M. is known as Recursive and recursively enumerable language.