Terminologies:-

→ <u>Language</u> :- denoted by L, it is a subset or set of string over an alphabet.

→ <u>Alphabet</u> :- denoted by Σ ← sigma
is a finite, non-empty set of symbols
for e.g :- Σ {0, 01} ← it is a set of binary alphabets
Σ {A, B, ... Z} ← set of upper case letter

→ <u>String</u>.
{ } — empty ϵ ← Epsillon
having no symbol is known as empty string.
String length is 0.
String length :- defined as no. of elements or symbols present in a given string.
ex:- x = 12010 = |x| = 5. mode || is the symbol.

→ <u>Operation of langugе</u>
Union of 2 language :- L1 & L2 denoted by $L_1 \cup L_2 = L_1 \cup L_2 = \{00, 01, 00, bb, aba$

concatenation of 2 languge :- $L_1 L_2$
eg:- $L_1 \{00, 01\}$ {00, ab, 00, bb, 00, aba, 01, ab, 01, bb
$L_2 \{ab, bb, aba\}$ 01, aba}

Date:
Page No.:

Regular Expression (Regular Languages)

→ Method to Represent a language (L) $L = \{\varepsilon, a, aa, aaa, \ldots\}$

$L = a^*$

→ Let 'R' be a Regular Expression over Alphabet Σ if R is:

1) ~~Let~~ 'ε', is Regular Expression denoting the set $\{\varepsilon\}$
2) ϕ, is Regular Expression denoting the empty set $\{\}$
3) For each symbol $a \in \Sigma$, a is regular expression denoting set $\{a\}$
4) Union of two RE is also Regular.
5) ~~Union~~ Concatenation of two R.E is also Regular
6) Kleen closure of RE is also Regular.
7) If R is Regular, (R) is also Regular
8) Nothing else, Repeat 1 to 7 recursively

PAGE No.
DATE / /

→ Closure of language, denoted by L^* (asterisk)

e.g.:- $L = \{ba\} = L^1$ ← 1 closure
$L^0 = \{\epsilon\}$ = empty
$L^2 = \{L \cup L\} = \{ba\ ba\}$

$L^* = L^0 \cup L^1 \cup L^2$
$= \{\epsilon\ ba\ ba\}$

can be 0 more than 0

→ Positive closure of language

denoted by L^+ (positive)

doesn't take empty set
minimum L^1 hona chahiye.

e.g.:- $L^+ = L^1 \cup L^2 \cup L^3$
$L^+ = \{ba, baba\}$

* Regular Expression (RE) :- used to specify the string.

RE	RL ← Regular Language
(r)	$L(r)$
→ a	$L = \{a\}$
→ b	$\{b\}$
→ a/b	$\{a\} \cup \{b\} = \{a, b\}$
→ a+b	(a, b) ⇒ + is also used for OR symbol
	Ya toh a Ya toh b

→ $a.b$	$\{a\}.\{b\} = \{ab\}$
→ ab	$\{ab\}$
→ a^*	$\{\epsilon, a, aa, aaa, \ldots\}$
→ a^+	$\{a, aa, aaa \ldots\}$
→ $(ab)^*$	$\{\epsilon, ab, abab, ababab \ldots\}$
→ $(a+b)^*$	$\{\epsilon, a, b, aa, ab, ba, bb \ldots\}$

* Write a regular expression for the Language accepting all combination of a's over the set $\Sigma = \{a\}$

→ $RE = a^*$
$RL = \{\epsilon, a, aa \ldots\}$

* design the regular expression for the Language. accepting all combination without null of a's over the set $\Sigma = \{a\}$

→ $RE = a^+$

$RL = \{a, aa, aaa, \dots\}$

* design RE for the L containing all the string / any no. of a's and b's

→ $RE = (a+b)^*$

$RL = \{E, a, b, aa, ab, ba, bb \dots\}$

→ expe. accept the null :-

$RE = (a+b)^+$

$RL = \{a, b, aa, ab, ba, bb, \dots\}$

AT

Q. Construct regular expression for the L ~~exceptir~~ accepting all the strings which are ending with 00 over the set $\Sigma = \{0, 1\}$

⇒ $RE = (0+1)^* 00$

Q. Write a regular expression for the L accepting strings which are starting with 1 and ending with 0 over the set $\Sigma = \{0, 1\}$

⇒ $RE = 1(0+1)^* 0$

Q. The language starting ~~wi~~ and ending with A and having any combination of ~~these~~ b's in between. What is the ~~RE~~ regular Expression?

⇒ $a(b)^* a$

Q. ~~The string Stat~~ Write the RE which consists of Language having string which should have at least one 0 and at least one 1

$\Rightarrow$ $RE = (0+1)^* \, 0 \, (0+1)^* \, 1 \, (0+1)^*$

or//

$RE = (0+1)^* \, 1 \, (0+1)^* \, 0 \, (0+1)^*$

Q. write a regular expression which denotes a language L over the set $\Sigma = \{1\}$ having odd length of strings.

$\Rightarrow$ $RE = 1(11)^*$

Q. Construct RE which denotes the L over the set $\Sigma = \{0\}$. having even length of string.

$\Rightarrow$ RE = ~~0/(0)*~~ $(00)^*$

Q. Write the RE to denote a language L over ~~where~~ $\Sigma = \{a, b\}$. ~~write~~ such that ~~it includes~~ the third character from right end of string is always a.

$\Rightarrow$ RE

any no. of a's and b's	a	either a or b	either a or b

$R.E = (a+b)^* \, a \, (a+b)(a+b)$

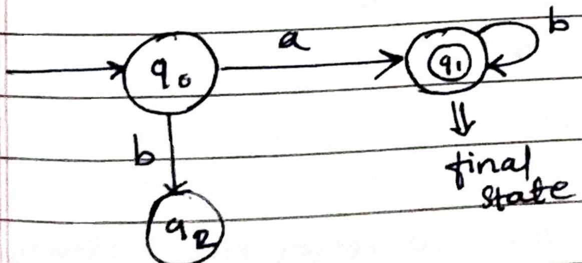
Finite Automata

Automata:- The term automata is derived from greek word 'Automata', which means self acting.
The word automata is for automation means a system where energy and info are transformed and used for performing some fn without direct involvement of humans is called automation.
An Automata with finite number of states is called finite automata (FA) or finite state machine (FSM).

Finite Automata :- An automata can be represented by five tuples $M = \{Q, \delta, q_0, \Sigma, F\}$

where Q is a finite set of states
Σ is a finite set of input symbols
δ is ~~mapping~~ Mapping function or transition function
q_0 is initial state $\boxed{q_0 \in Q}$
F is set of final state $\boxed{F \subseteq Q}$

State Transition diagram :-



Transition Table:-

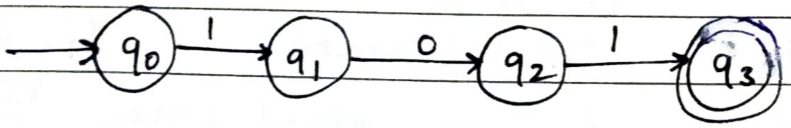
Current states	a	b
q_0	q_1	q_2
q_1*	–	q_1
q_2	–	–

Transition Function :

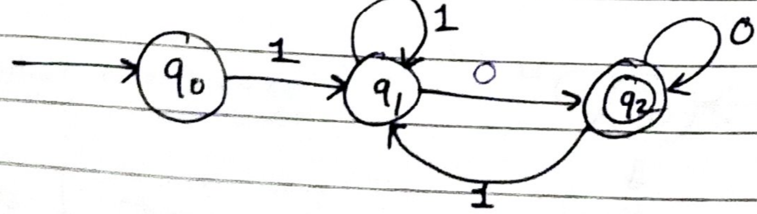
$q_1 = (q_0, a)$ means concurrent state q_0 with input a from, next state is q_1

Q. Deterministic Finite Automata (DFA)

Q. Design a DFA which accept the only input 101 over the input $\Sigma = \{0, 1\}$.

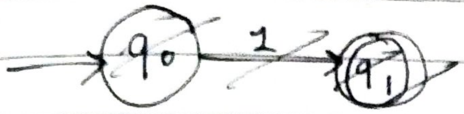


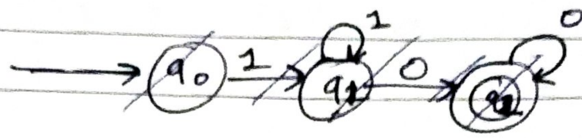
Q. Design a DFA which accept only those string which starts with 1 and ends with 0.

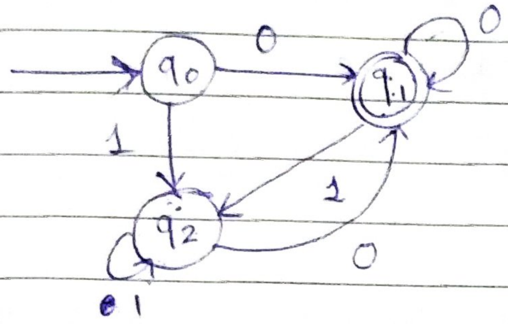


Q. Design a DFA which checks whether the binary number is even.

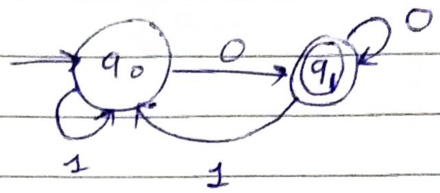
$\Sigma = \{0, 1\}$



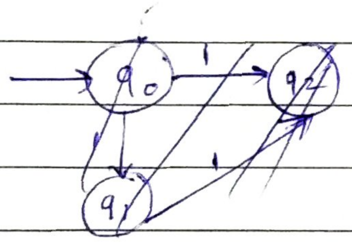


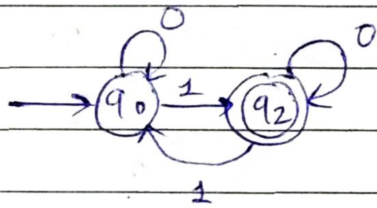


or

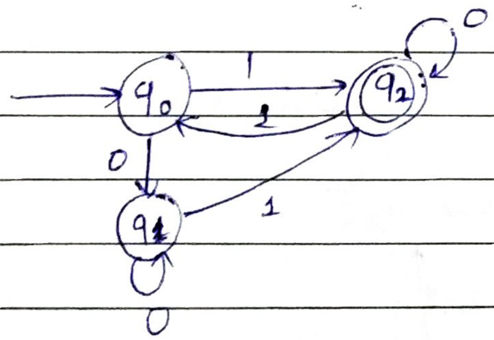


Q. Design a DFA which accepts odd no. of ~~zero~~ 1's and any no. of 0's.

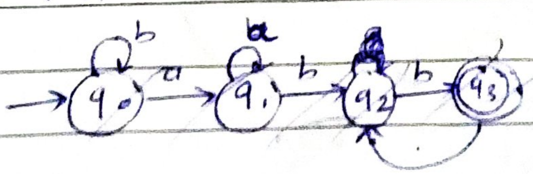


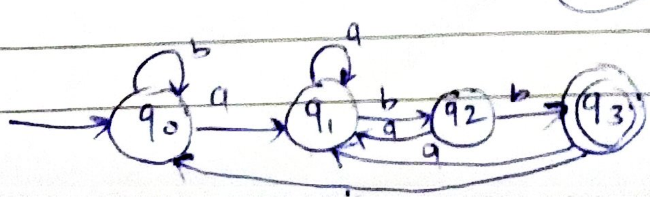


OR



Q. Design a DFA to accept the string of a's and b's which ends with abb

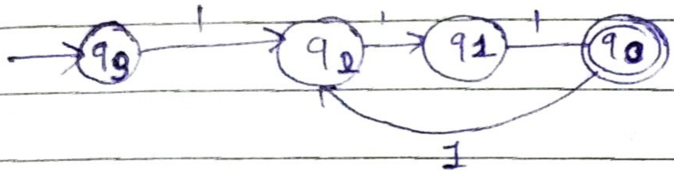




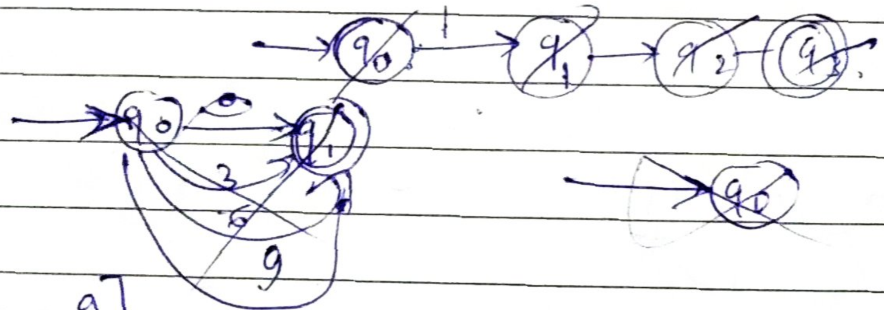
Q. Design a DFA which checks whether the given unary number is divisible by 3.
unary
1 or 1's

$\Sigma = \{1\}$

$\Rightarrow$

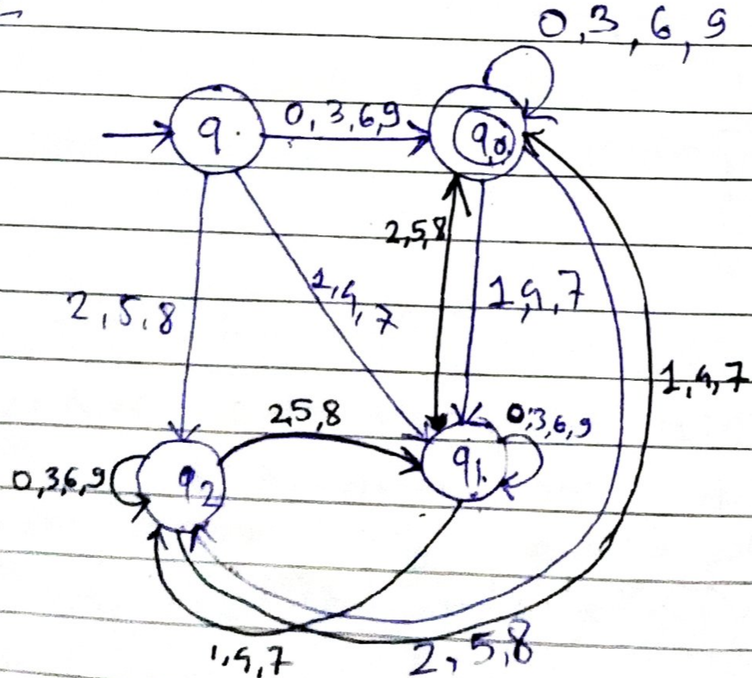


Q. Design a DFA which checks whether the given decimal number is divisible by 3.



[0, 1, 2 . . . 9]
divisible by 3
R = 0, 1, 2

0, 0.
0, 1.
0, 2
0, 3



1, 0
1, 1

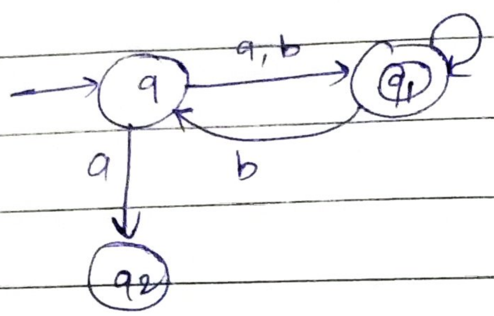
2/1/23

A.T

Non Deterministic Finite Automata (NFA)

The finite AT is DFA ~~It is called NFA~~ when there exist many path for specific input from current state to next state.

ex:



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

Defn (NFA) :- It is a collection of five tuples

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

F = final state

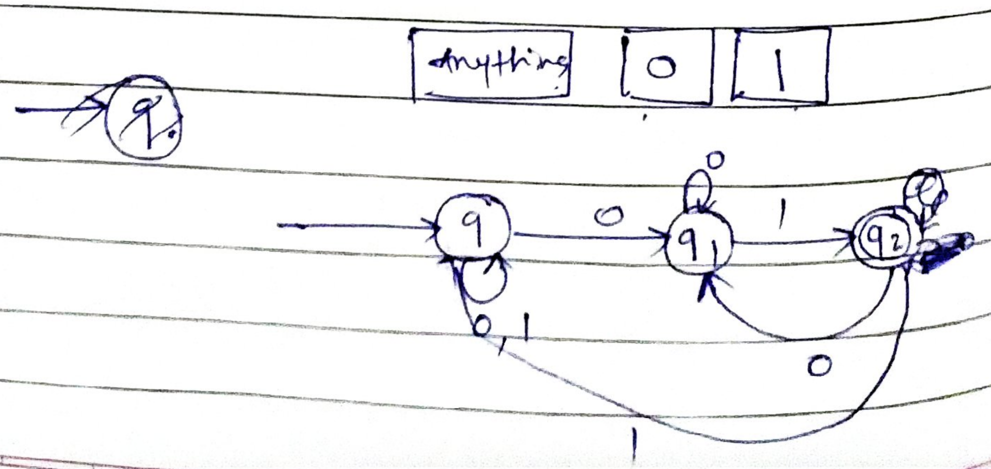
q_0 = initial state

Q = ~~set of~~ finite set of states

Σ = finite set of inputs symbols.

δ = ~~transiti~~ transition function.

Q. Design NFA accepting all strings ending with '01' over $\Sigma = \{0, 1\}$.

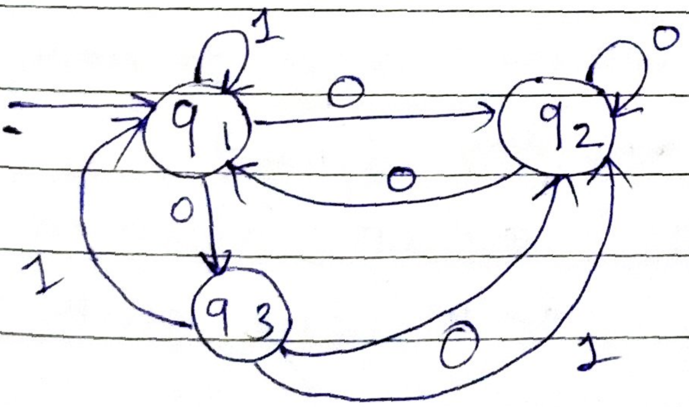


Q. Construct a transition diagram ~~from~~ for NFA
$M = (\{q_1, q_2, q_3\}, \delta, q_1, \{q_3\})$

where δ is given by $\delta(q_1, 0) = \{q_2, q_3\}$
$\delta(q_1, 1) = \{q_1\}$
$\delta(q_2, 0) = \{q_1, q_2\}$
$\delta(q_2, 1) = \phi$
$\delta(q_3, 0) = \{q_2\}$
$\delta(q_3, 1) = \{q_1, q_2\}$

State transition table

	0	1
$\rightarrow q_1$	$\{q_2, q_3\}$	$\{q_1\}$
q_2	$\{q_1, q_2\}$	ϕ
$q_3{*}$	$\{q_2\}$	$\{q_1, q_2\}$



DATE / /

Imp

Diff. betn NFA & DFA

DFA :- ① Every input string leads to the un.. state of finite Automata.

NFA :- ① For the same input there can be more than one next step.

DFA :- ② Difficult to design as transitions are deter..istic.

NFA :- ② Easy to design as compared to DFA

DFA :- ③ DFA is as powerful as NFA

NFA :- ③ NFA is as powerful as DFA

DFA :- ④ Conversion ~~to~~ of RE to DFA is complex

NFA : ④ R.E can be easily converted to NFA.

~~DFA~~ : ⑤ In DFA it is not possible to move to next ~~step~~ state without reading any symbol.

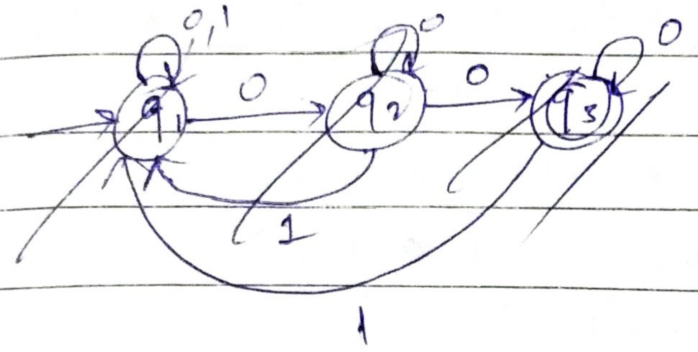
NFA : ⑤ In NFA we can move to next state without ~~re~~ any symbol.

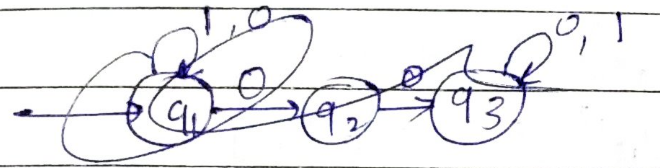
DFA : ⑥ ε transition are not allowed in DFA to move from one state to next state

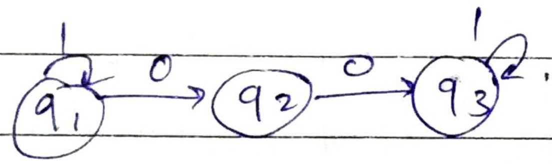
NFA : ⑥ ε transition are allowed in NFA to move from one state to next sta..

DATE / /

q. construct NFA accepting binary strings with two consecutive ~~zo~~ ~~00~~ zeroes.







~~100100~~

Conversion of NFA to DFA

Let $M = [\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\}]$

be a NFA where

$\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_0, 1) = \{q_1\}$

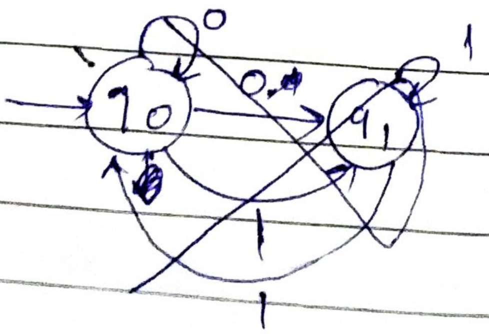
$\delta(q_1, 0) = \phi$

$\delta(q_1, 1) = \{q_0, q_1\}$

S.T.T

Step (1)

	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1 *	ϕ	$\{q_0, q_1\}$

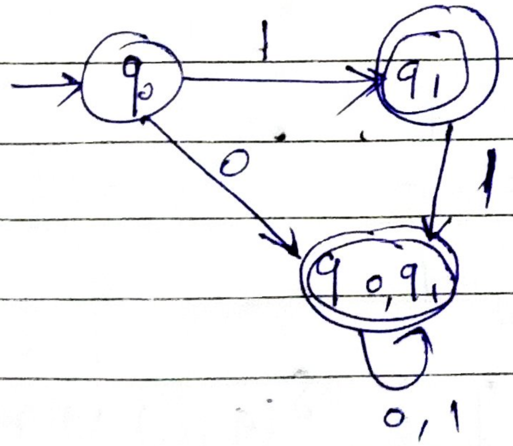


Step ②
$(q_0, 0) = \{q_0, q_1\} \rightarrow$ New state
$(q_0, 1) = \{q_1\}$
$(q_1, 0) = \phi$
$(q_1, 1) = \{q_0, q_1\}$

Step ③
$[\{q_0 q_1\}, 0] = \delta(q_0, 0) \cup \delta(q_1, 0)$
$= q_0 q_1 \cup \phi = \{q_0, q_1\}$ //
$[\{q_0, q_1\}, 1] = \delta(q_0, 1) \cup (q_1, 1)$
$= q_1 \cup (q_0, q_1)$
$= q_0 q_1$ //

∴ so jaha bhi q_1 hoga wo final state hogi kyuki NFA me last state q_1 hai or new state me bhi q_1 hai

Step ④



DATE / /

Q. Construct DFA ~~what~~ from NFA where
$M = [\{p, q, r, s\}, \{0,1\}, \delta, p, \{q, s\}]$

where δ	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
q^*	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
s^*	–	$\{p\}$

Sol

$(p, 0) = \{q, s\}$ ✓ N.S

$(p, 1) = \{q\}$ ✓

$(q^*, 0) = r$ ✓

$(q, 1) = \{q, r\}$ ✓ N.S

$(r, 0) = s$ ✓

$(r, 1) = p$ ✓

$(s, 0) = -$

$(s, 1) = p$

$[(q, s), 0] = \delta(q, 0) \cup \delta(s, 0)$
$= \cancel{\{q, s\}}\ r \cup \phi = r$

$[(q, r), 0] = \delta(q, 0) \cup \delta(r, 0)$
$= \cancel{\{q, r\}}$
$= r \cup s$
$= rs \longrightarrow$ N.S

$[\{q,r\},1] = (q,1) \cup (r,1)$

~~$[\{q,s\},1]$~~

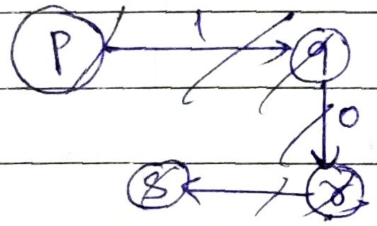
$= \{q,r\} \cup p$

$= pqr \rightarrow N.S$

$[\{q,s\},1] = (q,1) \cup (s,1)$

$= \{q,r\} \cup p$

$= pqr \rightarrow N.S$



Now,

rs and pqr are new states, Again

$\delta[(r,s),0] = s$

$\delta[(r,s),1] = p$

$\delta[(pqr),0] = qrs \rightarrow N.S.$

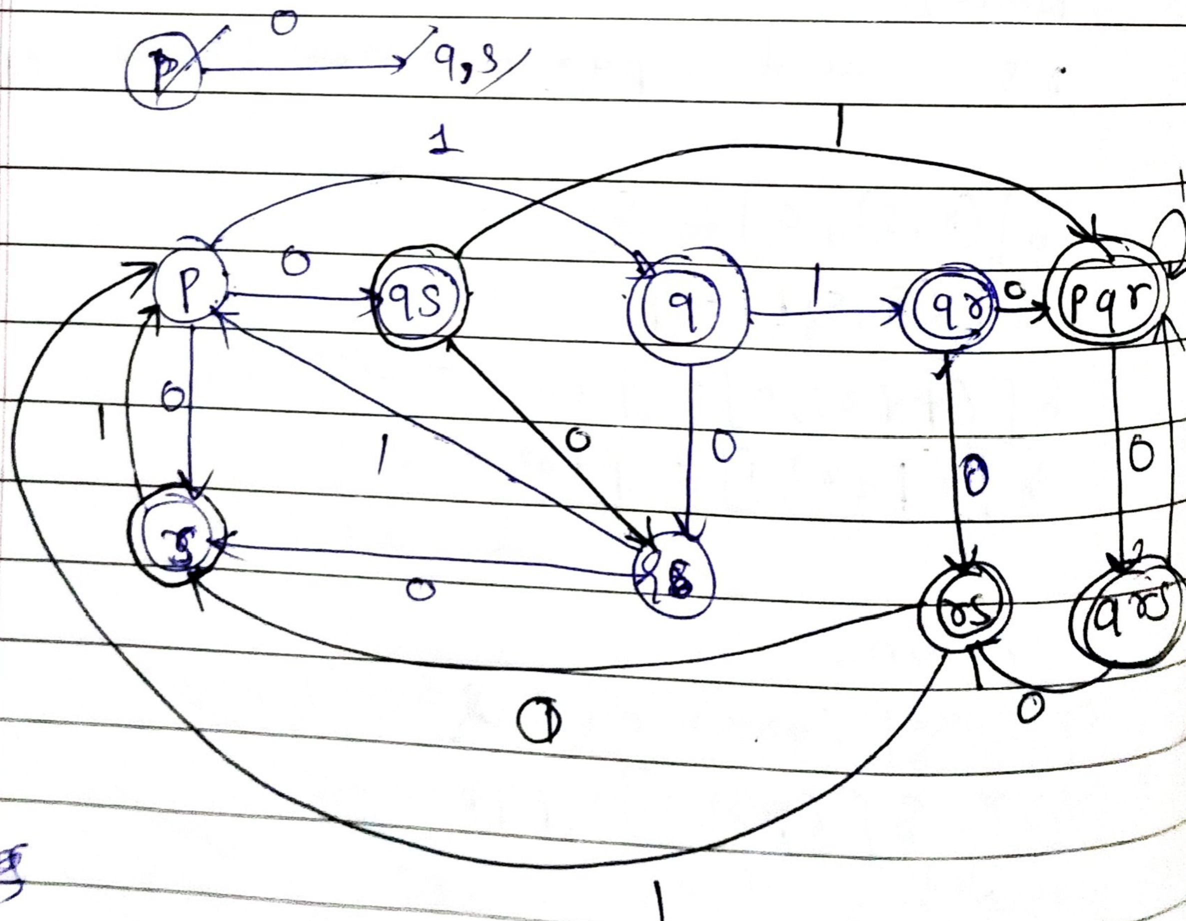
$\delta[(pqr),1] = pqr.$

Again,

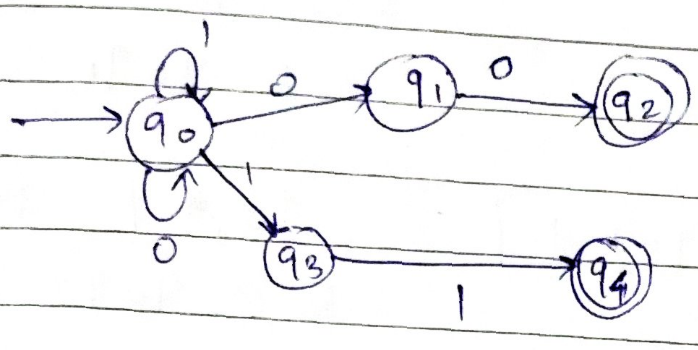
$\delta[(qrs),0] = rs$

$\delta[(qrs),1] = pqr$

where δ	0	1
p	{q,s}	q
q*	{r}	{q, r}
r	{s}	p
s*	—	p
qs*	r	pqr
qr*	rs	pqr
pqr*	qrs	pqr
rs	s	p
qrs*	~~r~~s	pqr



Q. Convert given NFA to its equivalent DFA



δ	0	1
q_0	$q_0 q_1$	$q_0 q_3$
q_1	q_2*	–
q_2*	–	–
q_3	–	q_4
q_4	–	–

$(q_0, 0) = q_0 q_1 \rightarrow N.S$

$(q_0, 1) = q_0 q_3 \rightarrow N.S$

$(q_1, 0) = q_2$

$(q_3, 1) = q_4$

$= [q_0 q_1, 0]$

$\cancel{\delta(q_0 q_1) \cup} = \delta(q_0, 0) \cup \delta(q_1, 0)$

$= q_0 q_1 \cup \cancel{q_0 q_2}\ q_2$

$= q_0 q_1 q_2 \ // \rightarrow N.S$

$= [q_0 q_1, 1]$

$= \delta(q_0, 1) \cup \delta(q_1, 1)$

$= q_0 q_3 \cup \cancel{q_0 q_3}\ \emptyset$

$= q_0 q_3 \ //$

DATE / /

$$[(q_0 q_3); 0] = q_{0,0} \cup q_{3,0}$$
$$= q_0 q_1 \cup \phi$$
$$= q_0 q_1$$

$$[(q_0 q_3), 1] = q_{0,1} \cup q_{3,1}$$
$$= q_0 q_3 \cup q_4$$
$$= q_0 q_3 q_4$$

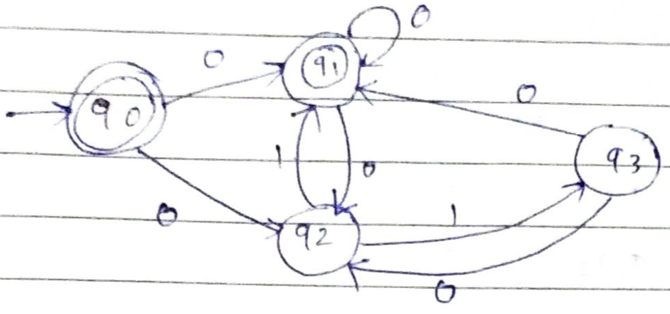
→ N.S

$$[(q_0 q_1 q_2), 0] = (q_0, 0) \cup (q_1, 0) \cup (q_2, 0)$$

$$[(q_0 q_1 q_2), 1]$$

=

Q. Convert the following ~~DFA~~ NFA to ~~D~~ it's equivalent DFA.



State	0	1
* q_0	q_1, q_2	-
* q_1	q_1, q_2	-
q_2	-	q_1, q_3
q_3	q_2, q_1	-

$[q_1 q_2, 0] = \delta(q_1, 0) \cup \delta(q_2, 0)$
$= q_1 q_2 \cup -$
$= q_1 q_2$ → N.S

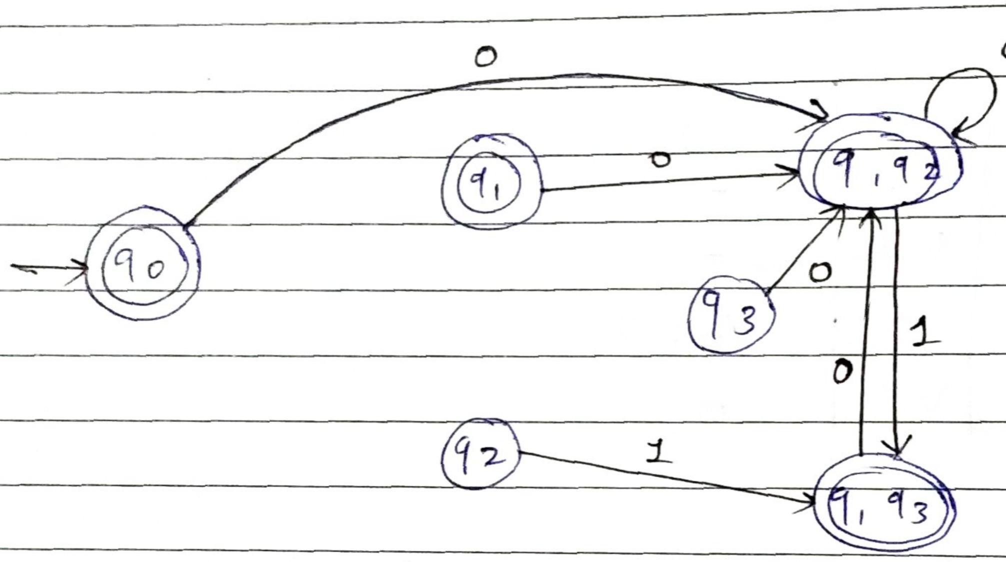
$[q_1 q_2, 1] = \delta(q_1, 1) \cup \delta(q_2, 1)$
$= - \cup q_1 q_3$
$= q_1 q_3$ → N.S

$[q_1 q_3, 0] = \delta(q_1, 0) \cup \delta(q_3, 0)$
$= q_1 q_2 \cup q_2 q_1$
$= q_1 q_2$

$[q_1 q_3, 1] = \delta(q_1, 1) \cup \delta(q_3, 1)$
$= - \cup -$
$= \phi$

New T.T.

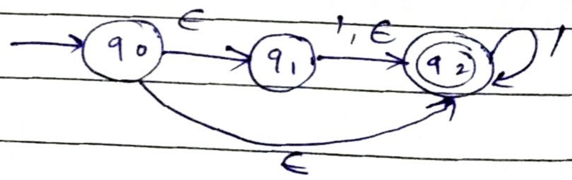
	0	1
q_0 *	$q_1 q_2$ (N.S)	–
q_1 *	$q_1 q_2$	
q_2	–	$q_1 q_3$
q_3	$q_1 q_2$ *	–
$q_1 q_2$ *	$q_1 q_2$	$q_1 q_3$
$q_1 q_3$ *	$q_1 q_2$	–



NFA with Epsillon (ϵ)

ϵ - Transition ⇒

ϵ - closure



ϵ closure(q_0) = $q_0 q_1 q_2$

ϵ closure (q_1) = $q_1 q_2$

ϵ closure (q_2) = q_2

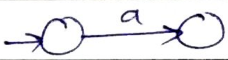
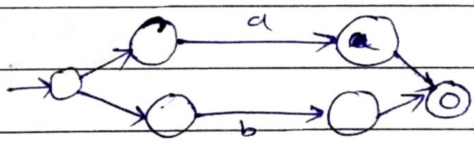
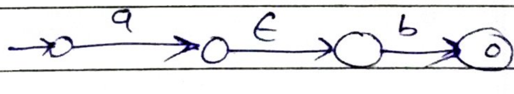
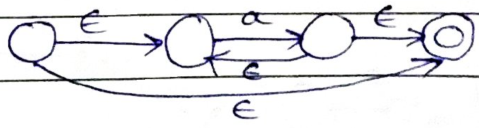
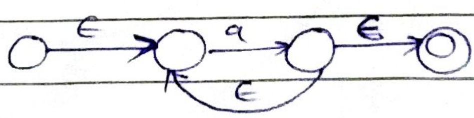
It implies that machine can make a transition without any input, we can change the state from q_0 to q_1 without any input symbol.

ε Transition → It is a move from one state to another without any input symbol.

ε closure → It is a set of all states which are reachable from current state with epsillon transition and self state.

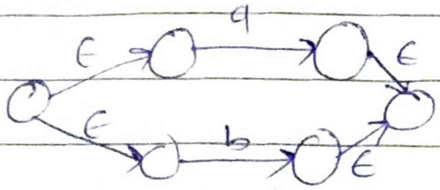
~~from above diagram..~~

Rules:

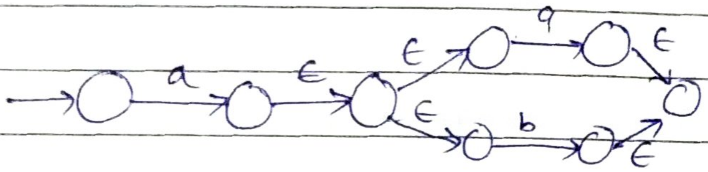
R.E	NFA with ε moves
(1) ε	
(2) a	
(3) a + b	
(4) ab	
(5) a *	
(6) a +	

Q. Draw NFA with E moves with R.E = $a(a+b).b^*$

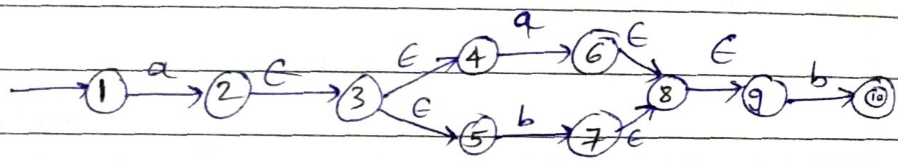
step ①



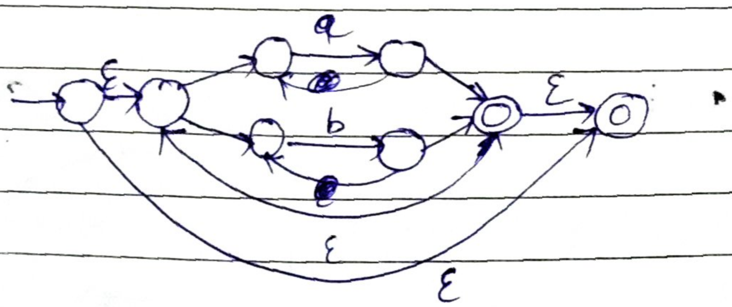
step ②

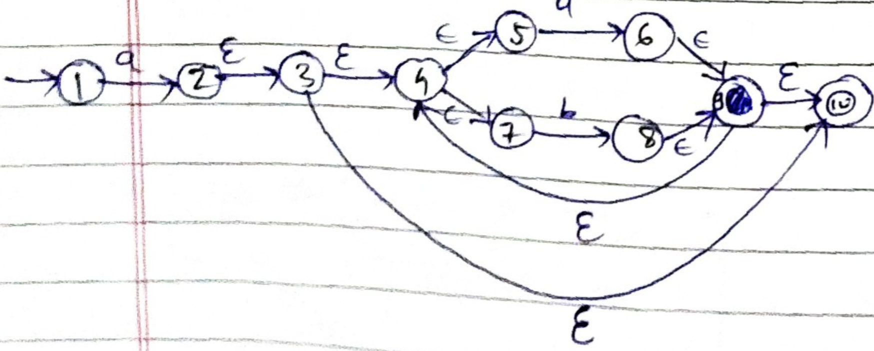


step ③



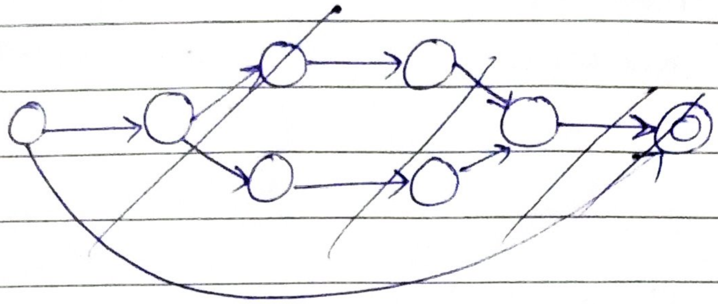
Q. $a(a+b)^*$

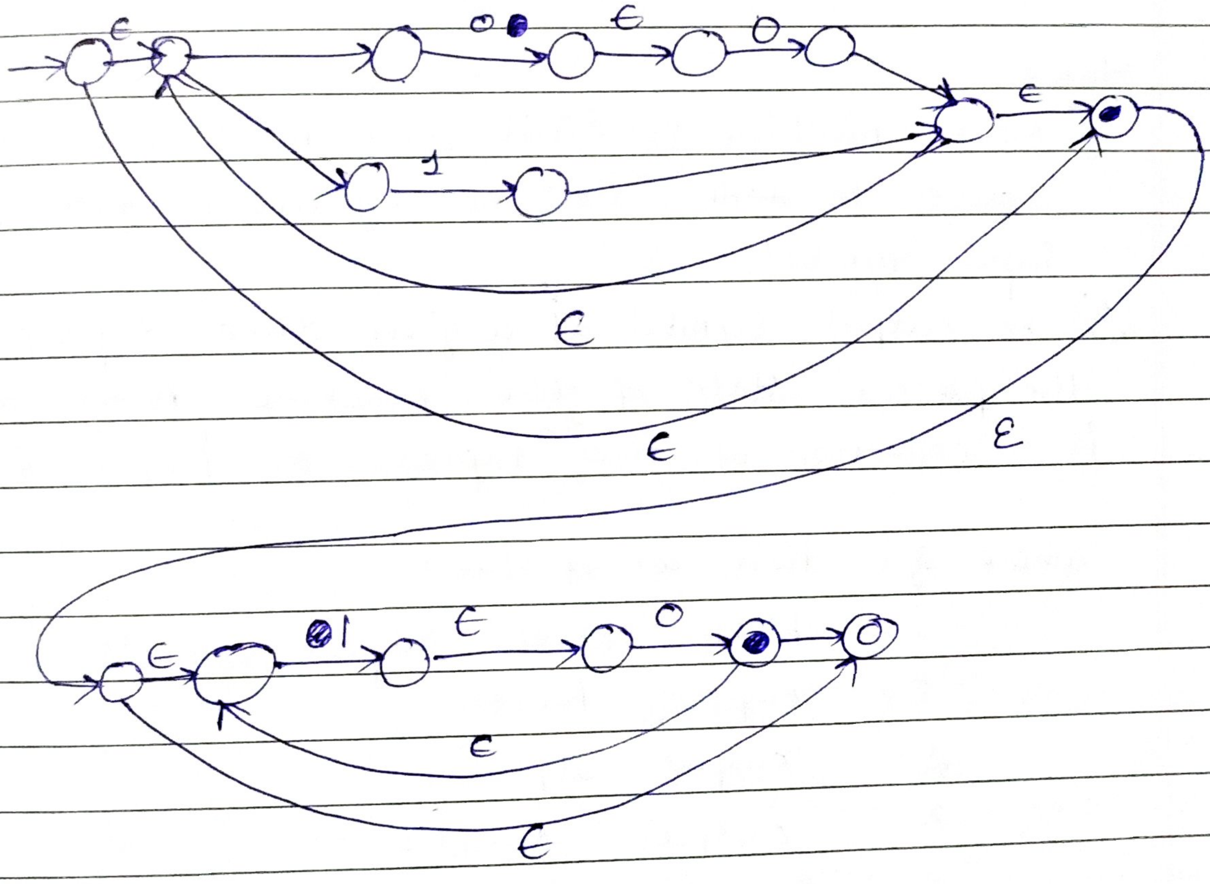




Date
Page No.

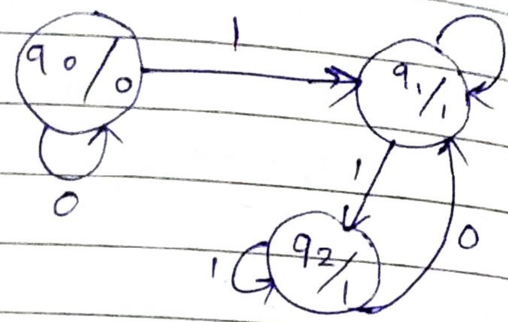
Q. RE = $(00+1)^* \cdot (10)^*$





Finite Automata (FA) with output

Types: (1) Moore machine
(2) mealy Machine



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

Diagram of Moore Machine

~~There~~

(1) Moore machine is finite state machine in which next state is ~~defin~~ decided by current state and current input symbol.

(2) The output symbol at a given time depends only on the present state of the machine. Moore machine is a collection of ~~five~~ six tuples $M = \{Q, \Sigma, \delta, \Delta, \lambda, q_0\}$

where Q = finite set of states
Σ = finite set of input symbols
δ = Mapping function.
Δ = output alphabet
λ = output function.
q_0 = ~~Init~~ ~~final~~ Initial state

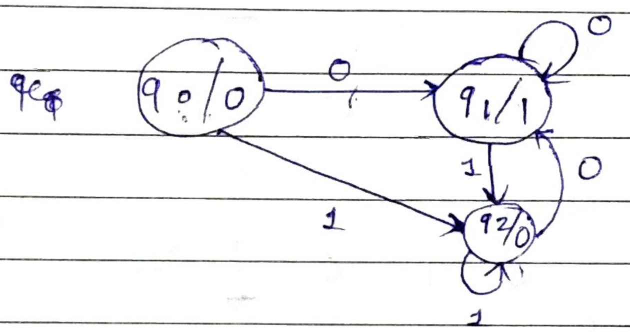
for diagram:

Q current State	δ Next state 1	0	λ o/p
q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	~~q_0~~ q_2	q_1	1

In moore machine, output is associated with each and every state.

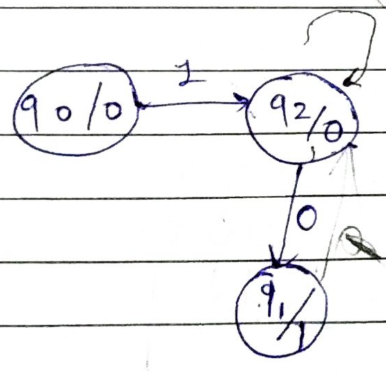
Q. Design a Moore machine to generate 1's complement of given binary number. $\Sigma = \{0, 1\}$

I/p 0 = O/p 1
I/p 1 = O/p 0

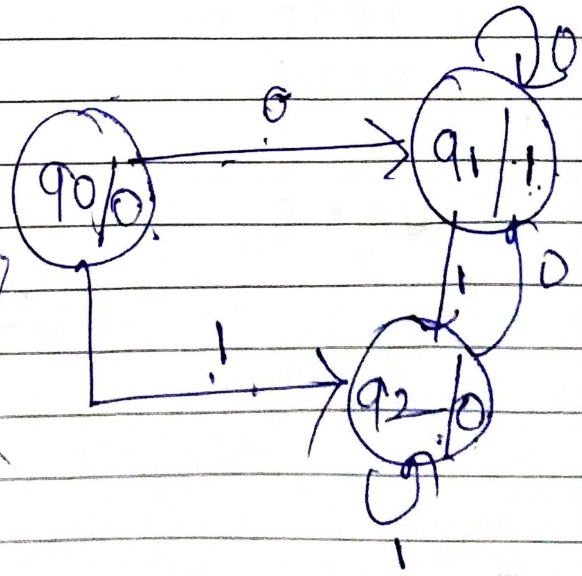


E.g. I/p → 1 0 1 1
states → q_0 q_2 q_1 q_2 q_2
O/p → 0 0 1 0 0

1 0 1 1 0
0 1 0 0 1



1's complement
→ 1 0
→ 0 +1
0 10



011101
100010