Prerequisite

- subset : A is subset of B, $A \subseteq B$

eg $A = \{4, 2, 3\}$ $B = \{1, 2, 3, 4, 5\}$

- Power set : set of all possible subsets of given set.

eg $A = \{1, 2, 3\}$ $P(A) = 2^A = \{\{\phi\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 3\}, \{1, 2, 3\}\}$

- Cartesian product :- of A & B

eg $A \times B = \{(a, b) \mid a \in A \ \& \ b \in B\}$

$A = \{1, 2, 3\}, B = \{a, b\}$

$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$

* Relation ~~of~~ and function

- A relation which contain two elements i.e. a relation which is defined over 2 set is called binary relation

eg

$A = \{1, 2, 3, 4, 5\}$. $A \rightarrow A$

$R = \{a, b \mid a < b\}$ $\boxed{R \subseteq A \times B}$

$R = \{(1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5)\} \rightarrow |A \times B| = 25$

$R = \{(a, b) \mid aRb \ \& \ a \in A \ \& \ b \in B\}$

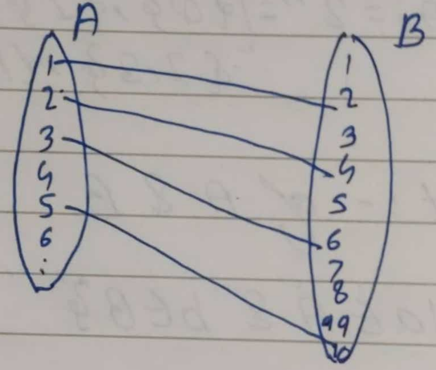
- Function is a relation. maps from A ~~&~~ to B for all elements of set A.

A function f from A ~~&~~ to B, denoted by $f: A \to B$

$2x \quad 2x = y$

$x \to \boxed{f(x)} \to y \in B$

$\in$
A

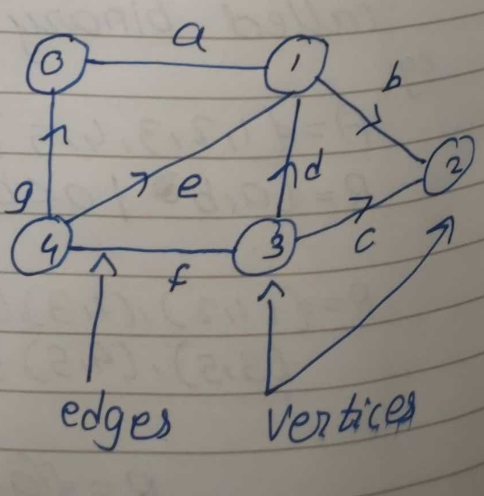


* Graph:-

A graph $G = (V, E)$ is a pair consisting of a set V, whose elements are called vertices, & a set E, where each element of E is a pair of ≠ distint vertices. The elements of E are called edges.

$V = \{0, 1, 2, 3, 4\}$
$E = \{a, b, c, d, e, f\}$.

$E = \{(0,1), (1,2), (0,4),$
$(4,8), (1,4), (2,3),$
$(3,2) \ldots \}$



classmate
Date
Page

Unit I:- Formal language theory & finite Automata.

* Alphabet:-

An alphabet is a finite set of symbols it is denoted by Σ

eg
$\Sigma = \{a, b, c, d \dots z\}$
$\Sigma = \{a, b\}$
$\Sigma = \{0, 1, 2\}$

* String (or word):-

A string or word over an alphabet Σ is a finite sequence of concatenated symbols of Σ.

eg 0110, 11, 110, 001 are few strings over the binary alphabets $\{0, 1\}$

* $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \dots \cup \Sigma^\infty$

$\Sigma^0 = \{\phi\}$
$\Sigma^1 = \{a, b, c \dots z\}$
$\Sigma^2 = \{aa, ab, ac\}$
$\Sigma^3 = \{aaa, aab, aac \dots zzz\}$.

* Language:-

A language is defined as a set of strings contains symbols from alphabet.

eg. $\Sigma = \{0, 1\}$ $\Sigma^* \{0, 1, 00, 01, 10, 11, 000 \dots 111\}$

L_1 is a language which contains even binary numbers $\therefore L_1 = \{10, 110, 101010 \dots\}$
$L_2 \rightarrow$ odd numbers $\rightarrow 1$
$L_3 \rightarrow$ starts with 1 & ends with 0
$L_4 \rightarrow$ ends with 00/11/110

* Deterministic finite Automata:-

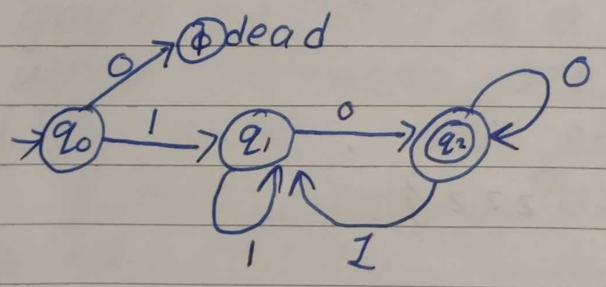
- ~~It~~ is the one in which each move is uniquely determined by the current configuration.
- Deterministic finite Automata (DFS) is defined by 5-tupples, $M = (Q, \Sigma, \delta, q_0, F)$ where

Q = Finite set of states
Σ = Finite set of input alphabet
δ = Transition function which maps from $Q \times \Sigma = Q$
q_0 = Initial state of FA, $q_0 = Q$
F = Finite set of final state

Ex1 Design Finite Automata for a language which accept words starting with 1 & end with 0 over $\{0,1\}$.

=>

$\Sigma = \{0, 1\}$ $W = 1\_\_\_0$ $L = \{10, 1\_\_\_0\}$



State transition Graph.

$Q\downarrow$ \ Σ	0	1
q_0	Φ	q_1
q_1	q_2	q_1
q_2^*	q_2	q_1

State transition table

Transition Function:-

$\delta(q_0, 0) = \phi$
$\delta(q_0, 1) = q_1$
$\delta(q_1, 1) = q_1$
$\delta(q_1, 0) = q_2$
$\delta(q_2, 0) = q_2$
$\delta(q_2, 1) = q_1$

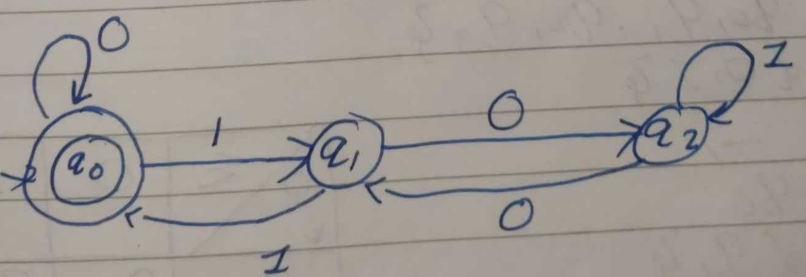
Ex 2 Design Finite Automata for a language which accept binary numbers multiple of 3.

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

Σ^*

$000 \rightarrow 0 \% 3 = 0 \rightarrow q_0$
$001 \rightarrow 1 \% 3 = 1 \rightarrow q_1$
$010 \rightarrow 2 \% 3 = 2 \rightarrow q_2$
$011 \rightarrow 3 \% 3 = 0 \rightarrow q_0$
$100 \rightarrow 4 \% 3 = 1 \rightarrow q_1$
$101 \rightarrow 5 \% 3 = 2 \rightarrow q_2$
$110 \rightarrow 6 \% 3 = 0 \rightarrow q_0$
$111 \rightarrow 7 \% 3 = 1 \rightarrow q_1$

$=0 \% 3 = 0 \rightarrow q_0$

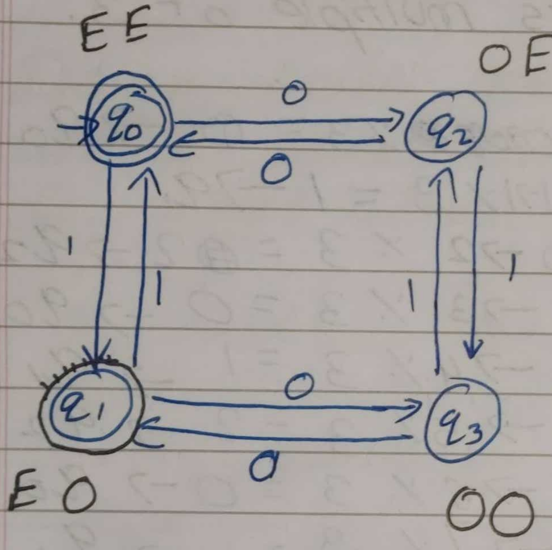


Q ↓ \ Σ	0	1
q_0*	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Ex4 Design finite automata for a language whi[ch] accept binary numbers having even 0's & od[d] 1's.

⇒

	0s	1s	
1)	Even	Even	→ q_0
2)	Even	Odd	→ q_1
3)	Odd	Even	→ q_2
4)	Odd	Odd	→ q_3



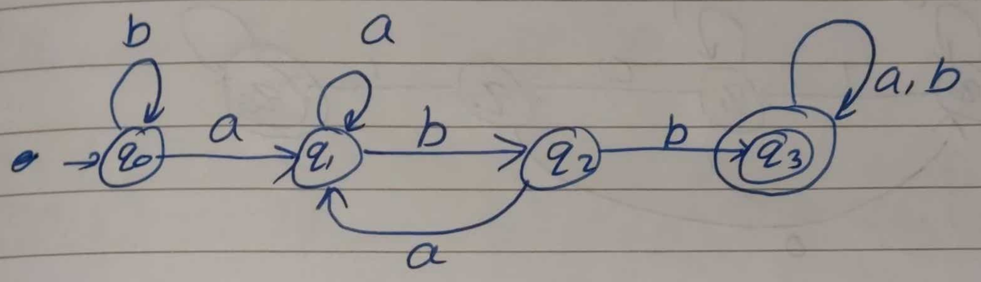
0 → O, E
1 → E, O
00 → E, E
01 → O, O
10 → O, O
11 → E, E
010 → E, O
100 → E, O
011 → O, E

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$\delta = \rightarrow$
$q_0 = q_0$
$F = \{q_1\}$

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

Ex5 Design Finite Automata for a language which accept substring "abb" over {a,b}

$L = \{abb, \_\_abb, abb\_\_, \_abba\_\}$

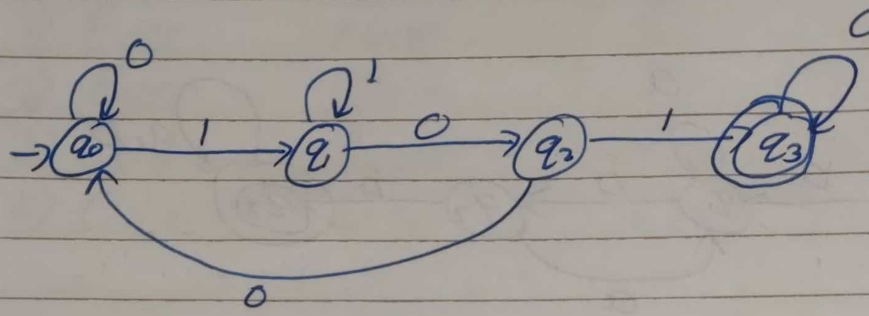


* Non-Deterministic Finite Automata [NFA]:-
NFA is defined by 5-tupples, $M = (Q, \Sigma, \delta', q_0, F)$
where,

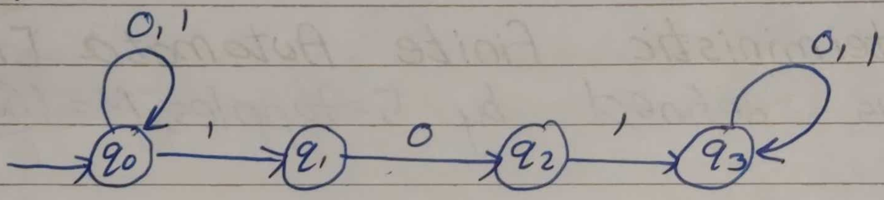
Q = Finite set of states
Σ = finite set of input alphabet
δ' = Transition function which maps from $Q \times \Sigma = 2^Q$
q_0 = Initial state of FA, $q_0 = Q$
F = Finite set of final state.

Ex1 Design NFA for a language accepting string having 101 as a substring over {0,1}

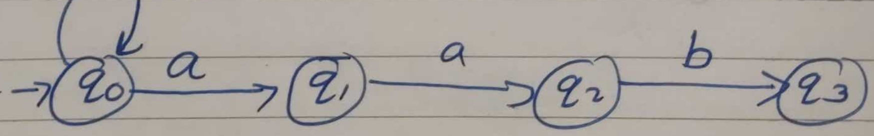
DFA :-



NFA :-



Ex2 Design NFA for a language accepting string ending in "aab" over {a,b}



NFA to DFA conversion.

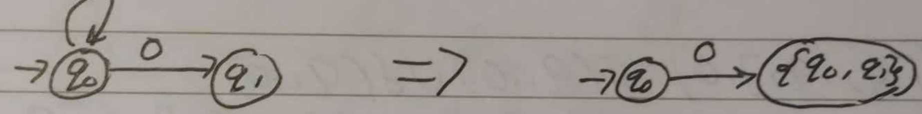
Equivalence:-

For every NFA, there exists a DFA which simulates the given NFA. It means that any language defined by NFA is also defined by DFA.

NFA to DFA conversion

Ex I:- $M = (\underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{0,1\}}_{\Sigma}, q_0, \underbrace{\{q_1\}}_{F})$ where $\delta(q_0,0)=\{q_0,q_1\}$, $\delta(q_0,1)=\{q_1\}$, $\delta(q_1,0)=\phi$, $\delta(q_1,1)=\{q_0,q_1\}$

=>

NFA 

Consider, ~~$[q_0, q_1]$~~:

$\delta([q_0,0]) = \{q_0,q_1\}$ so $\delta'([q_0],0) = [q_0,q_1]$

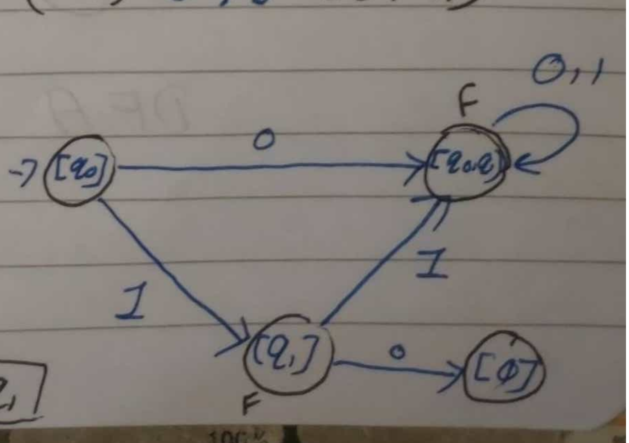
$\delta(q_0,1) = \{q_1\}$ so $\delta'([q_0],1) = [q_1]$

Consider $[q_0,q_1]$:

$\delta[[q_0,q_1],0] = \delta(q_0,0) \cup \delta(q,0)$
$= \{q_0,q_1\} \cup \phi$
$= [q_0 \, q_1]$

$Q \downarrow$ / $\Sigma \Rightarrow$	0	1
$\rightarrow q_0$	$\{q_0,q_1\}$	$\{q_1\}$
q_1^*	ϕ	$\{q_0,q_1\}$

$\delta'([q_0,q_1],1) = \delta(q_0,1) \cup \delta(q_1,1)$
$= \{q_1\} \cup \{q_0,q_1\}$ (NFA) $\delta \Rightarrow \delta'$ (DFA)
$= [q_0,q_1]$



Consider $[q_1]$

$\delta'([q_1],0) = \delta(q_1,0) = [\phi]$

$\delta'([q_1],1) = [q_0,q_1]$ $\boxed{F = q_1}$

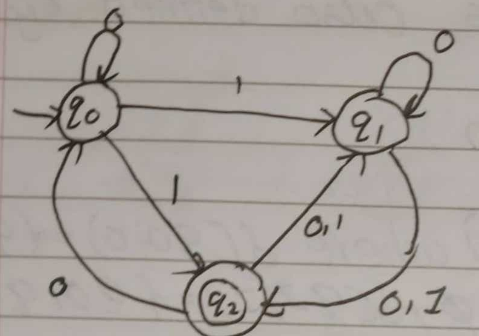
Ex 2:- Convert given NFA to it's equivalent DFA.

⇒

① $\delta(q_0, 0) = \{q_0\}$ so $\delta'([q_0], 0) = [q_0]$

$\delta(q_0, 1) = \{q_1, q_2\}$ so $\delta'([q_0], 1) = [q_1, q_2]$

Q ⇒ Fig



From fig we get:-

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

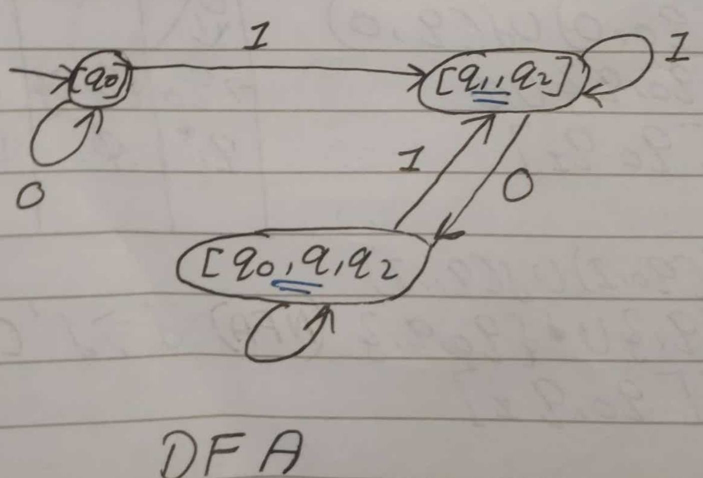
② $\delta'([q_1, q_2], 0) = \delta(q_1, 0) \cup \delta(q_2, 0)$

$= \{q_1, q_2\} \cup \{q_0, q_1\} = [q_0, q_1, q_2]$

$\delta'([q_1, q_2], 1) = \delta(q_1, 1) \cup \delta(q_2, 1) = \{q_2\} \cup \{q_1\} = [q_1, q_2]$

~~③ $\delta'([q_0, q_1, q_2], 0) = \delta(q_0, 0$~~

$F = q_2$ = final state



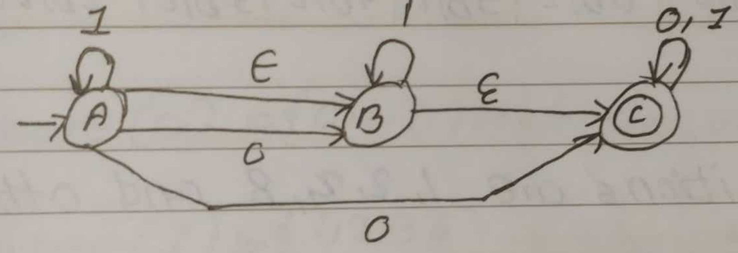
DFA

* NFA with ε moves :-

- Transition on empty input i.e., ε moves
- NFA-ε is defined as $(Q, \Sigma, \delta'', q_0, F)$ where.

Q = Set of States
Σ = Set of input alphabets
δ'' = Transition Function which maps from $Q \times \Sigma^* \rightarrow 2^Q$
q^0 = Initial state $\because \Sigma^* = \Sigma \cup \{\varepsilon\}$
F = Final state

eg



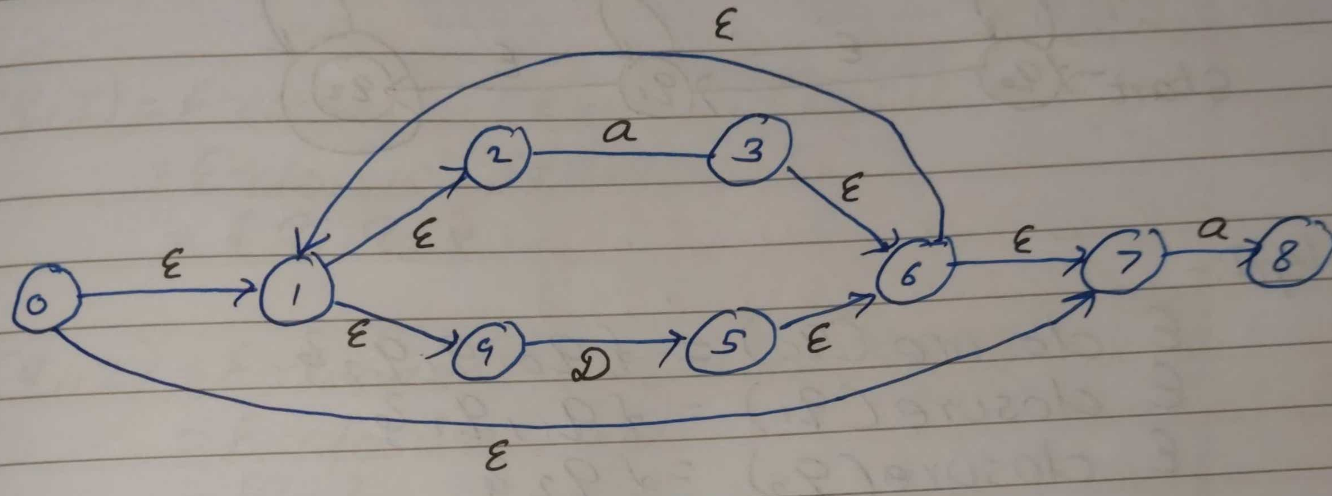
$\delta(A, \varepsilon) = B$
$\delta(A, 0) = \{BC\}$

	ε	0	1
→A	{B}	{B,C}	{A}
B	{C}	φ	{B}
C*	-	{C}	{C}

- NFA with ε moves
ε closure (q) : set of states transiting from a given state using ε moves.

ε closure (A) = {A, B, C}
ε closure (B) = {B, C}
ε closure (C) = {C}

* NFA with ε moves example



ε-closure (0) = {0, 1, 7, 2, 4}

ε-closure (1) = {1, 2, 3}

ε-closure (2) = {2}

ε-closure (3) = {3, 6, 7, 1, 2, 4}

ε-closure (4) = {4}

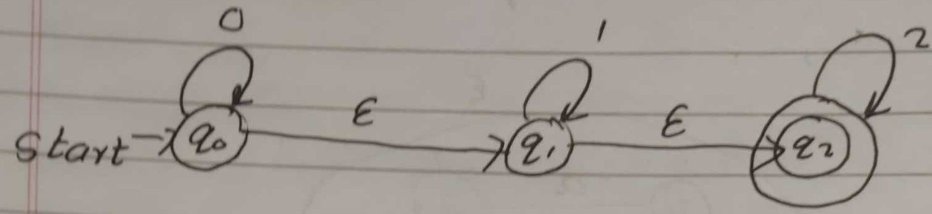
ε-closure (5) = {5, 6, 7, 1, 2, 4}

ε-closure (6) = {6, 7, 1, 2, 4}

ε-closure (7) = {7}

ε-closure (8) = {8}

* NFA-ε to NFA conversion



Step1:-

ε closure $(q_0) = \{q_0, q_1, q_2\}$
ε closure $(q_1) = \{q_1, q_2\}$
ε closure $(q_2) = \{q_2\}$

Step2:-

$\delta'(q0, 0) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q0), 0))$
$= \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), 0)$
$= \varepsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$
$= \varepsilon\text{-closure}(\{q_0\} \cup \phi \cup \phi)$
$= \{q_0, q_1, q_2\}$

~~Step3:-~~ $\delta'(q_0, 1) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q0), 1))$
$= \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), 1)$
$= \varepsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2 \cup 1)\}$
$= \varepsilon\text{-closure}(\phi \cup q_1 \cup \phi\})$
$= \varepsilon\text{-closure}(q_1)$
$= \{q_1, q_2\}$

$\delta'(q0, 2) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q0), 2))$
$= \varepsilon\text{-closure}(\delta\{q_0, q_1, q_2\}, 2)$
$= \varepsilon\text{-closure}(q_2)$
$= \{q_2\}$

Step 3:- $\delta'(q_1, 0) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1), 0))$
$= \varepsilon\text{-closure}(\phi)$
$= \{\phi\}$

$\delta'(q_1, 1) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1), 1))$
$= \varepsilon\text{-closure}(\delta\{q_1, q_2\}, 1)$
$= \{q_1, q_2\}$

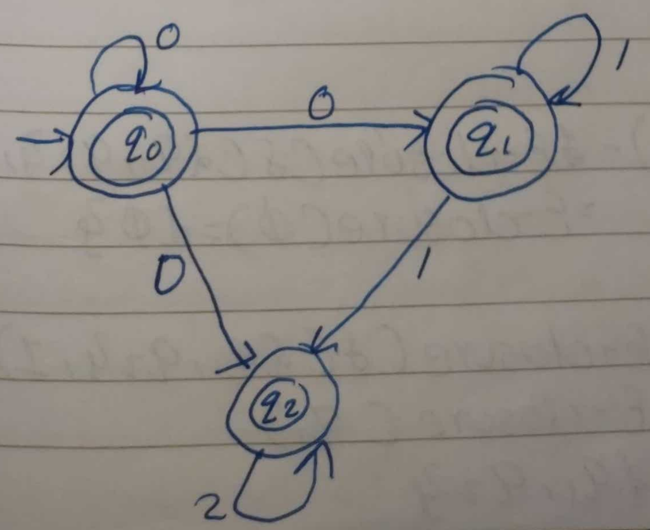
$\delta'(q_1, 2) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1), 2))$
$= \varepsilon\text{-closure}(q_2)$
$= \{q_2\}$

Step 4:- $\delta'(q_2, 0) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_2), 0))$
$= \varepsilon\text{-closure}(\phi)$
$= \{\phi\}$

$\delta'(q_2, 1) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_2), 1))$
$= \varepsilon\text{-closure}(\phi)$
$= \{\phi\}$

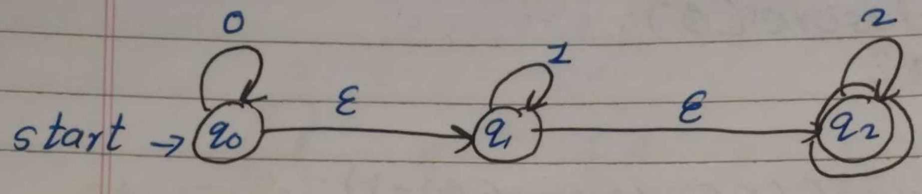
$\delta'(q_2, 2) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_2), 2))$
$= \varepsilon\text{-closure}(q_2)$
$= \{q_2\}$.

Step 5:- Draw state transition Table.



[Here all states are final state because in Q. q_2 is final state & ε closure (q_0) contains all q_0, q_1, & even q_2]

* NFA-δ to DFA conversion:-



Step1:- $\varepsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$
$\varepsilon\text{-closure}(q_1) = \{q_1, q_2\}$
$\varepsilon\text{-closure}(q_2) = \{q_2\}$

Step2:- Consider initial state when convert to DFA & go ahead with new states ahead.....

Consider ε-closure of initial state q_0 to start.

Step3:- $\delta'(\{q_0, q_1, q_2\}, 0) = \varepsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 0\}$
$= \varepsilon\text{-closure}\{q_0\}$
$= \{q_0, q_1, q_2\}$

$\delta'(\{q_0, q_1, q_2\}, 1) = \varepsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 1)$
$= \varepsilon\text{-closure}(q_1)$
$= \{q_1, q_2\}$

$\delta'(\{q_0, q_1, q_2\}, 2) = \varepsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 2\})$
$= \varepsilon\text{-closure}(q_2)$
$= \{q_2\}$

Step4:-
$\delta'(\{q_1, q_2\}, 0) = \varepsilon\text{-closure}(\delta(\{q_1, q_2\}, 0)$
$= \varepsilon\text{-closure}(\phi) = \{\phi\}$

$\delta'(\{q_1, q_2\}, 1) = \varepsilon\text{-closure}(\delta(\{q_1, q_2\}, 1)$
$= \varepsilon\text{-closure}(q_1)$
$= \{q_1, q_2\}$

$\delta'(\{q_1, q_2\}, 2) = \varepsilon\text{-closure}(\delta\{q_1, q_2\}, 2)$
$= \varepsilon\text{-closure}(q_2) = \{q_2\}$

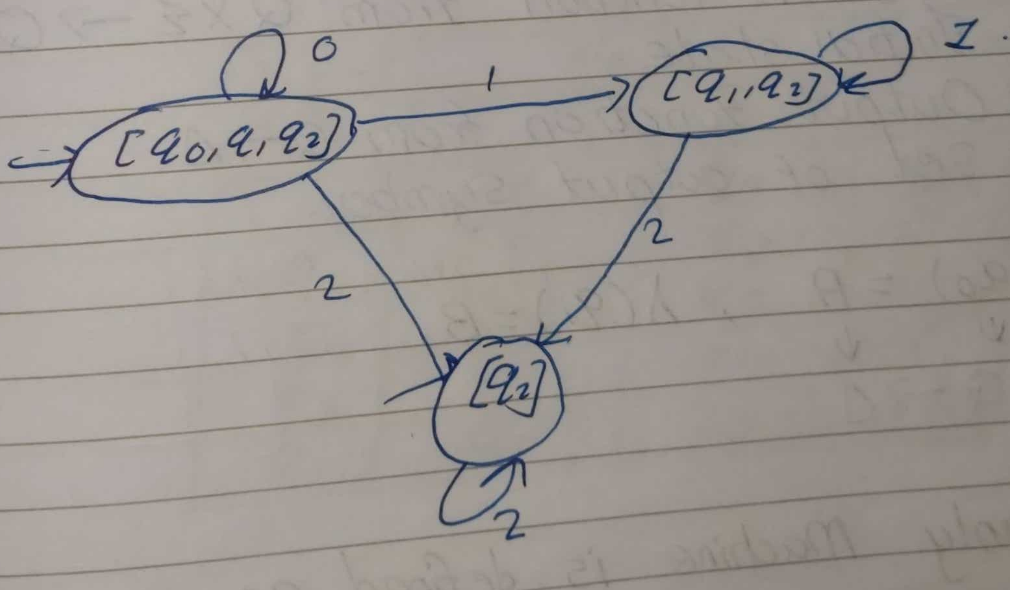
Step 5:-

$\delta'(\{q_2\}, 0) = \varepsilon\text{-closure}(\delta(\{q_2\}, 0))$
$= \varepsilon\text{-closure}(\phi)$
$= \{\phi\}$

$\delta'(\{q_2\}, 1) = \cancel{\{}\varepsilon\text{-closure}(\delta(\{q_2\}, 1))$
$= \{\phi\}$

$\delta'(\{q_2\}, 2) = \varepsilon\text{-closure}(\delta(\{q_2\}, 2)$
$= \{q_2\}$

Step 6:-



* Finite automata with output
  - FA is generating output on reading input string
  - Output can be associated with state [Moore machine] ...... $\rightarrow (q_0/A) \xrightarrow{0} (q_1/B)$
  - Output can be associated with transistion [Mealy Machine] ..... $\rightarrow (q_0) \xrightarrow{0/A} (q_1)$

* Moore machine is defined as

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

Q = ~~Sta~~ Set of states
Σ = Set of input alphabet.
δ = Transition function From $Q \times \Sigma \rightarrow Q$
q_0 = Initial state.
λ = Output Function from $Q \rightarrow \Delta$
Δ = Set of output symbol.

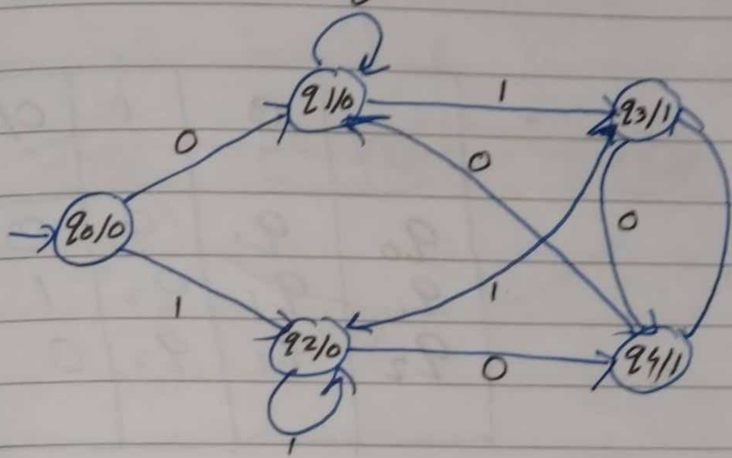
$$\underset{\downarrow}{\lambda(q_0)} = \underset{\downarrow}{A} \quad , \lambda(q_1) = B$$
$$\lambda : Q \rightarrow \Delta$$

* Meanly Machine is defined as.

$M = (Q, \Sigma, \delta, q_0, \lambda, \Delta)$ where.
Q = Set of states
Σ = Set of input alphabets.
δ = Transistion function $Q \times \Sigma \rightarrow Q$
λ = Output Function From $Q \times \Sigma \rightarrow \Delta$
q_0 = Initial state.
Δ = Set of output symbol
$\lambda(q_0, 0) = A \qquad \delta(q_0, 0) = q_1$

* Moore Machine.

λ = Output Function from $Q \to \Delta$

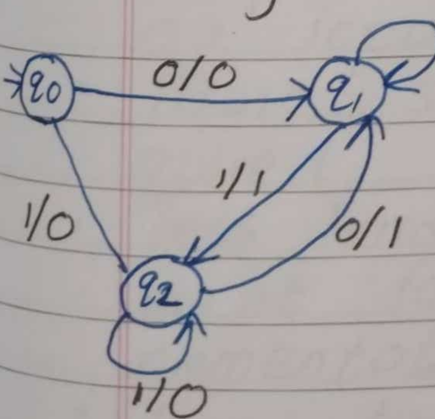


$\lambda(q_0) = 0$
$\lambda(q_1) = 0$
$\lambda(q_2) = 0$
$\lambda(q_3) = 1$
$\lambda(q_4) = 1$

Here, n input n+1 output.
State transition table:-

$Q \downarrow$ \ $\Sigma \to$	Input 0	Input 1	O/P
q_0	q_1	q_2	0
q_1	q_1	q_3	0
q_2	q_4	q_2	0
q_3	q_4	q_2	1
q_4	q_1	q_3	1

* Mealy Machine.



λ = Output Function From $Q \times \Sigma \to \Delta$

$\lambda(q_0, 0) = 0$
$\lambda(q_0, 0) = 0$

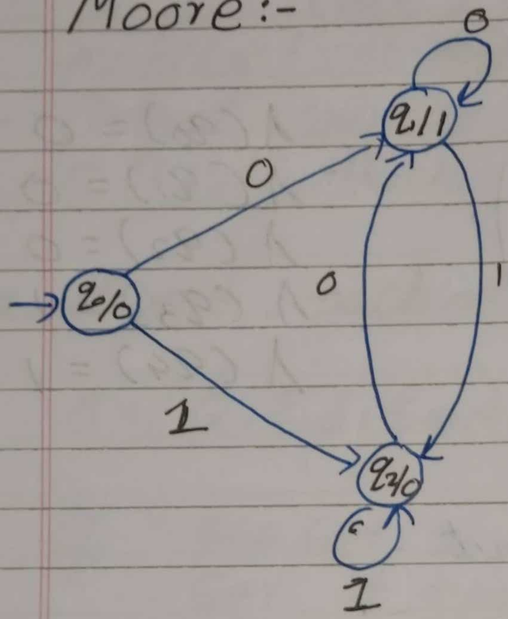
$Q \downarrow$	0: NS	0: O/P	1: NS	1: O/P
q_0	q_1	0	q_2	0
q_1	q_1	0	q_2	1
q_2	q_1	1	q_2	0

Here, n input n output

Ex1:- Design Moore and Mealy machine to gener
e 1's complement.

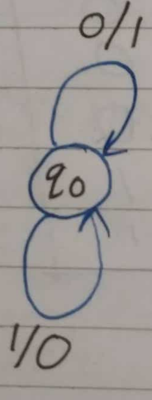
⇒

Moore:-



	0	1	O/P
q_0	q_1	q_2	0
q_1	q_1	q_2	1
q_2	q_1	q_2	0

Mealy:-

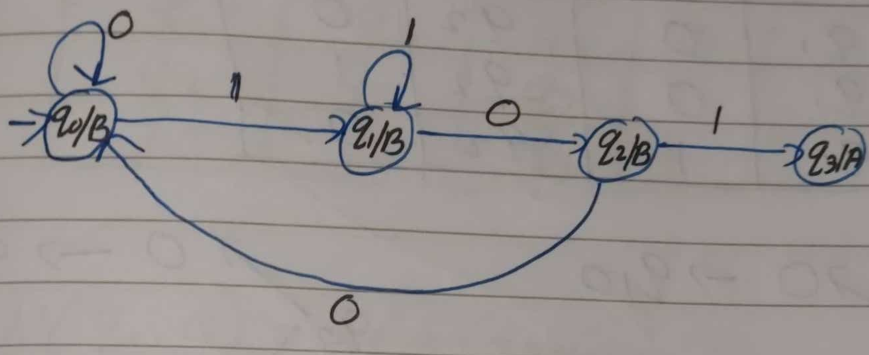


	0		1	
	N/S	O/P	N/S	O/P
q_0	q_0	1	q_0	0

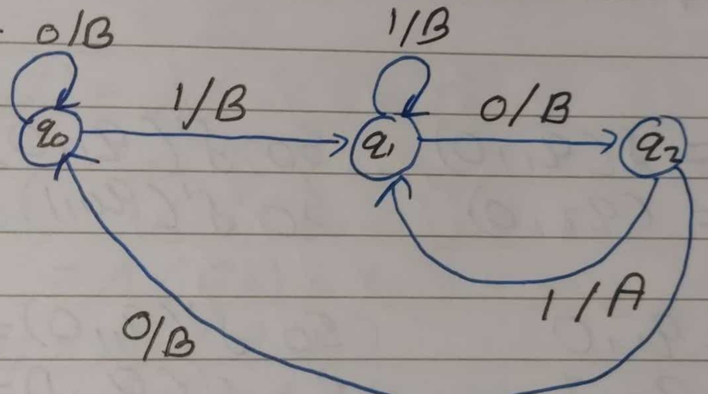
Ex2 Design Moore and Mealy machine which outputs A if 101 is recognized otherwise B

=>

Moore :-



Mealy :-



Moore machine	Mealy Machine.
• Output depends on present state	• Output depends on transition
• Output not change on input change	• Output changes on input change.
• More state are required	• less states are required
• less hardware requirement for circuit implementation	• More hardware required for circuit implementation.
• Easy to design	• Difficult to design
• n input n+1 output	• n input n output

* Conversion from Mealy to Moore.

Q.

	Input = 0		Input = 1	
P.S	N.S	O/P	N.S	O/P
q_0	q_1	0	q_2	0
q_1	q_1	0	q_2	1
q_2	q_1	1	q_2	0

⇒

$q_1 \to 0 \to q_{10}$
$q_1 \to 1 \to q_{11}$

$q_2 \to 0 \to q_{20}$
$q_2 \to 1 \to q_{21}$

$\delta(q_0,0) = \delta(q_1,0)$ So $\delta'(q_1,0) = q_{10}$
$\delta(q_0,1) = (q_2,0)$ So $\delta'(q_0,1) = q_{20}$

$\delta(q_1,0) = q_1,0$ So $\delta'(q_1,0) = q_{10}$
$\delta(q_1,1) = q_2,1$ So $\delta'(q_1,1) = q_{21}$
$\delta(q_{11},0) = q_1,0$ So $\delta'(q_{11},0) = q_{10}$
$\delta(q_{11},1) = q_2,1$ So $\delta'(q_{11},1) = q_{21}$

$\delta(q_{20},0) = q_1,1$ So $\delta'(q_{20},0) = q_{11}$
$\delta(q_{20},1) = q_2,0$ So $\delta'(q_{20},1) = q_{20}$
$\delta(q_{21},0) = q_1,1$ So $\delta'(q_{21},0) = q_{11}$
$\delta(q_{21},1) = q_2,0$ So $\delta'(q_{21},1) = q_{20}$

$Q \downarrow$ $\Sigma \to$	0	1	O/P
q_0	q_{10}	q_{20}	
q_{10}	q_{10}	q_{21}	0
q_{11}	q_{10}	q_{21}	1
q_{20}	q_{11}	q_{20}	0
q_{21}	q_{11}	q_{20}	1

* Conversion from Moore to Mealy

Q

states	0	1	O/P
q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_2	1

=>

① $\lambda'(q_0, 0) = \lambda(\delta(q_0, 0))$
$= \lambda(q_1)$
$= 0$

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1))$
$= \lambda(q_2) = 1.$

② $\lambda'(q_1, 0) = \lambda(\delta(q_1, 0))$
$= \lambda(q_3) = 1$

$\lambda'(q_1, 1) = \lambda(\delta(q_1, 1))$
$= \lambda(q_2) = 1$

③ $\lambda'(q_2, 0) = \lambda(\delta(q_2, 0))$.
$= \lambda(q_2) = 1$

$\lambda'(q_2, 1) = \lambda(\delta(q_2, 1))$
$= \lambda(q_1) = 0$

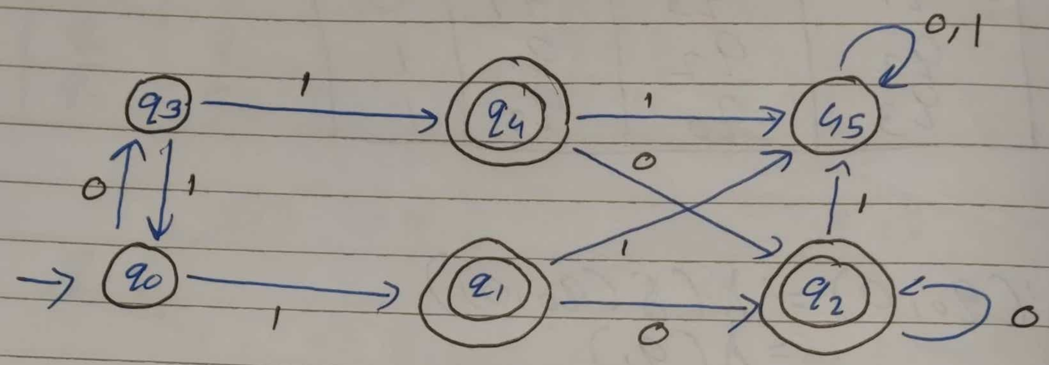
④ $\lambda'(q_3, 0) = \lambda(\delta(q_3, 0))$
$= \delta(q_0) = 1$

$\lambda'(q_3, 1) = \lambda(\delta(q_3, 1))$
$= \delta(q_2) = 1.$

Q ↓	0		1	
	N.S	O/P	N.S	OP.
q_0	q_1	0	q_2	1
q_1	q_3	1	q_2	1
q_2	q_2	1	q_1	0
q_3	q_0	1	q_2	1

* Minimization of DFA

DFA minimization stands for converting a given DFA to it's equivalent DFA with minimum number of states



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

① $P_0 = \{\{q_1, q_2, q_4\}, \{q_0, q_3, q_5\}\}$ .... final & non-final partition.

② $P_1 = \{\{q_1, q_2, q_4\}, \{q_0, q_3\}, \{q_5\}\}$

③ $P_2 = \{\{q_1, q_2, q_4\}, \{q_0, q_3\}, \{q_5\}\}$