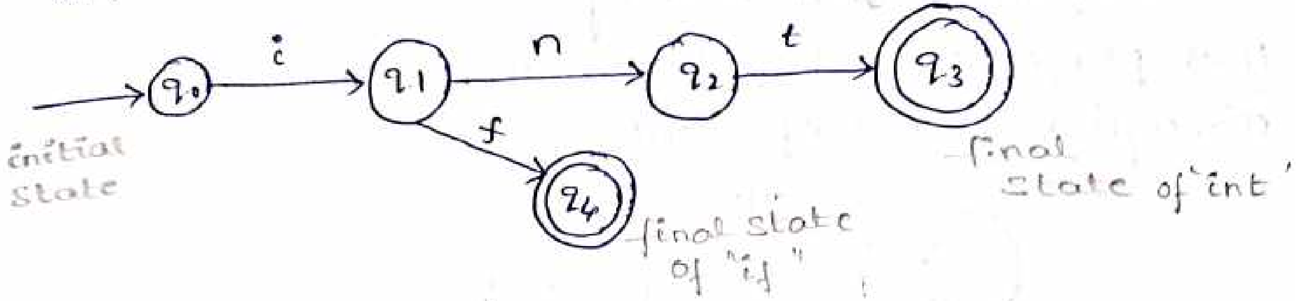
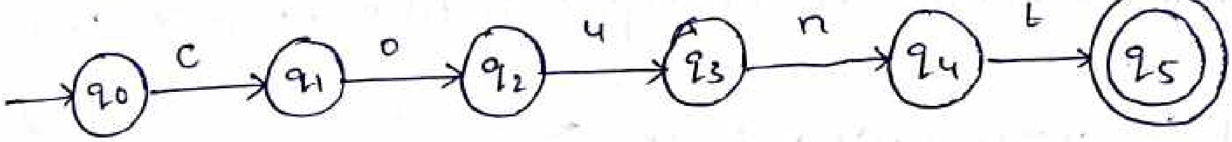
Unit-1

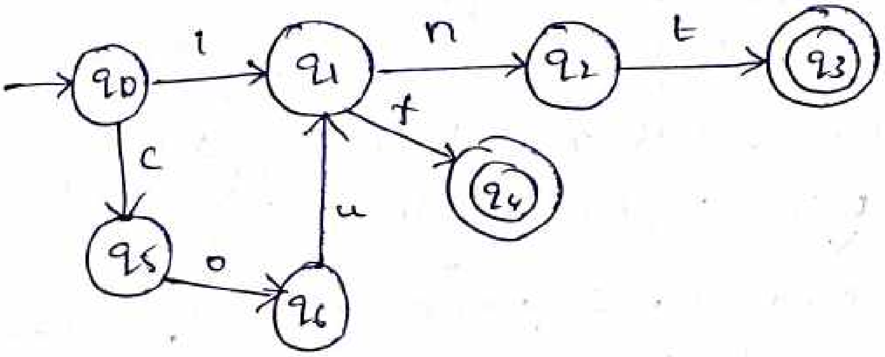
→ states:

int



count





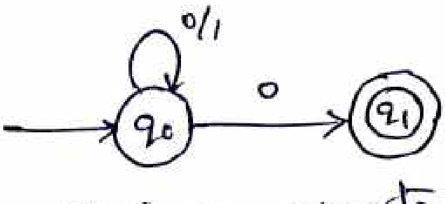
Q:- A string contains '0' at last position

If the question is given as above it means the alphabet set contains only {1,0}

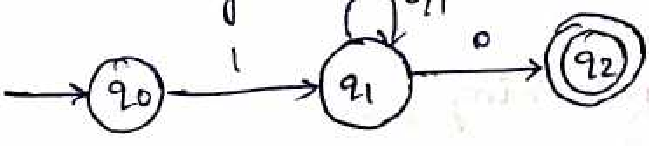
'0' can be one such string.

if we don't bother about the input character then there will be no change in state. It will be at self state.

{0, 00,10



Q:- String starts with '1' and ends with '0'

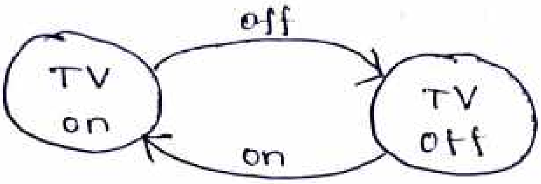


Automata:- It describes abstract model of emission

Ex: elevator, excelator etc--

Basics of automata theory:-

that performs computation on inputs by moving through a series of states.



Alphabet:- Finite set of symbols like english chars, digits, special characters etc-- (images, other languages

→ It is represented with Σ.

ex:- Binary alphabet $\Sigma = \{0,1\}$

for TV mentioned before $\Sigma = \{on, off\}$

If machine contains octal number supporting system $\Sigma = \{0 \underset{\times}{to} 7\} = \overset{\checkmark}{\{0,1,2,3,4,5,6,7\}}$

String:- It is an ordered sequence of characters from an alphabet. which can be represented by 'w'.

ex:- If $\Sigma = \{0,1\}$ $w = \{\underset{w_1}{0}, \underset{w_2}{1}, \underset{w_3}{01}, \underset{w_4}{10}, 00, 11, 100 ---\}$

The length of the string can be represented by $|w|$. $[\,|w_1| = 1 \quad |w_3| = 2 \quad |w_4| = 2\,]$

Language:- It is a set of strings of symbols from alphabet set, with condition based.

$L = \{w1, w2, w3, w4 ----\} = \{0, 1, 10, 01, -----\}$

Applications of Finite automata:-

1, Welding machines

2, Traffic lights

3, Video games

4, Text passing

5, Regular expression matching

6, protocol analysis

7, Natural language processing (NLP) etc--

→ The empty string with zero occurances of symbols is represented by "ϵ". (epsilon)

So $L = \{\epsilon, w1, w2, w3, ----\}$

→ Σ^* - It is a set of strings including empty string over the alphabet Σ.

$\Sigma^* = \{\epsilon, 0, 1, 10, 00, 01, 11, 100 ----\}$

also called as universal language.

⇒ difference b/w L & Σ^* is L takes string based on certain conditions but Σ^* has all possible strings including ϵ.

→ Σ^+ - It is a set of non-empty strings except empty string i.e; ϵ.

$\Sigma^+ = \{0, 1, 10, 00, 01, ----\}$

$\boxed{\Sigma^* = \Sigma^+ \cup \epsilon}$

Ex:- Set of strings over the $\Sigma = \{0, 1\}$ with equal no. of 0's and equal no. of 1's.

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

$\Sigma^* = \{\epsilon, 0, 1, 00, 11, 01, 10, 000, 001, 010, 011,$
$100, 101, 110, 111, 0000, 0001, 0010, 0011,$
$0100, 0101, 0110, 0111, 1000 ----\}$

$L = \{01, 10, 0011, 0101, 0110, 1001, 1010, \epsilon, --\}$

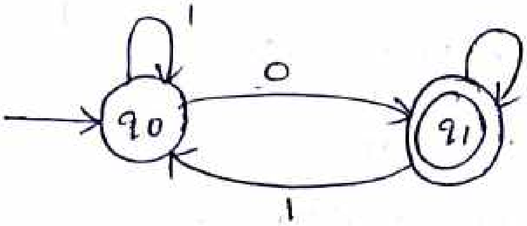
→ if ϵ is acceptable string the initial state will be final state

Ex:- Construct L with strings which should ends with '0'.

$L = \{0, 00, 10, 000, 110, 100, 010, \ldots\}$

$\Sigma^* = L \cup L'$

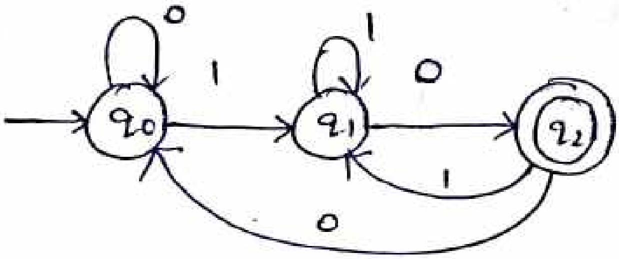
$L' = \{\epsilon, 1, 11, 01, 001, 011, 111, 101, \ldots\}$



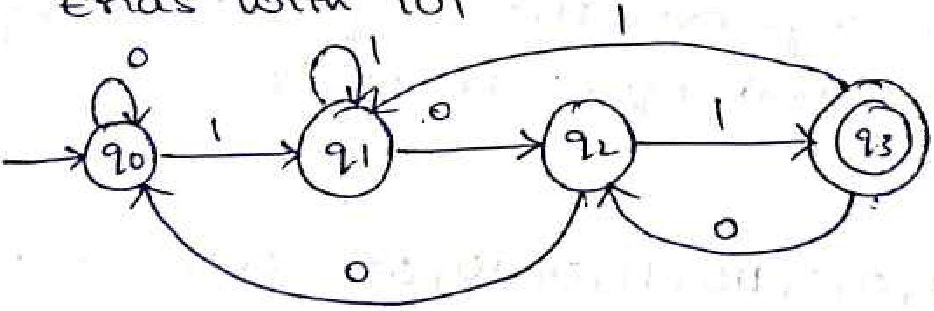
Ex:- Construct L for strings ends with 10

$L = \{10, 010, 110, 0010, 1110, 0110, 1010, \ldots$

$L' = \{\epsilon, 0, 1, 00, 11, 01, 001, 101, 111, 000, \ldots\}$



ex:- Ends with 101



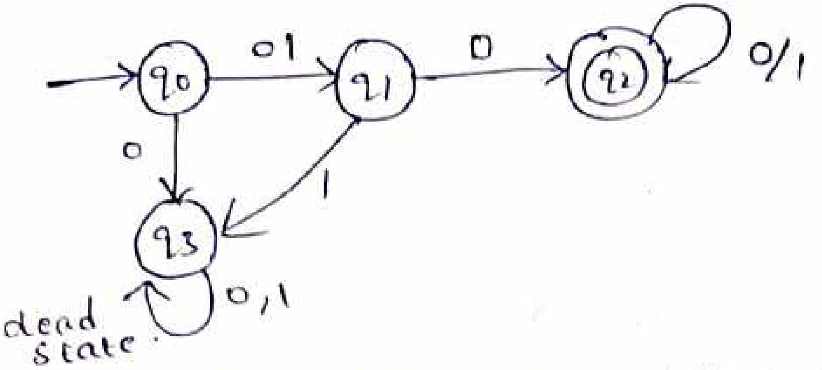
$L = \{101, 0101, 1101, 00101, 11101, 01101, \ldots$

$L' = \{\epsilon, 010, 001, 111, 1010 \ldots\}$

Ex:- Starts with 10

$L = \{10, 101, 100, 1011, 1000, 1001, 1010, \ldots\}$

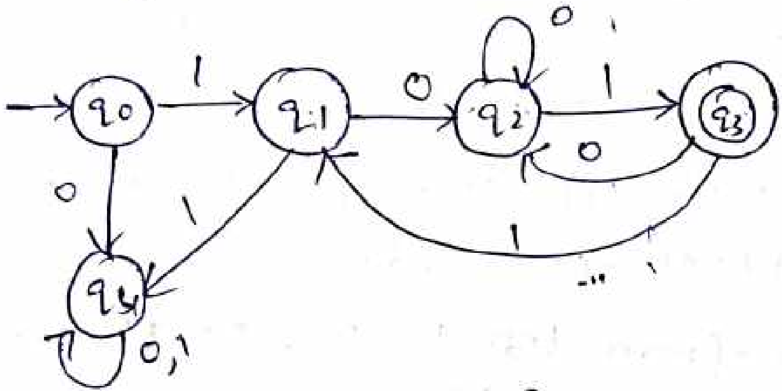
$L' = \{\epsilon, 0, 1, 01, 11, 00, 111, \ldots\}$



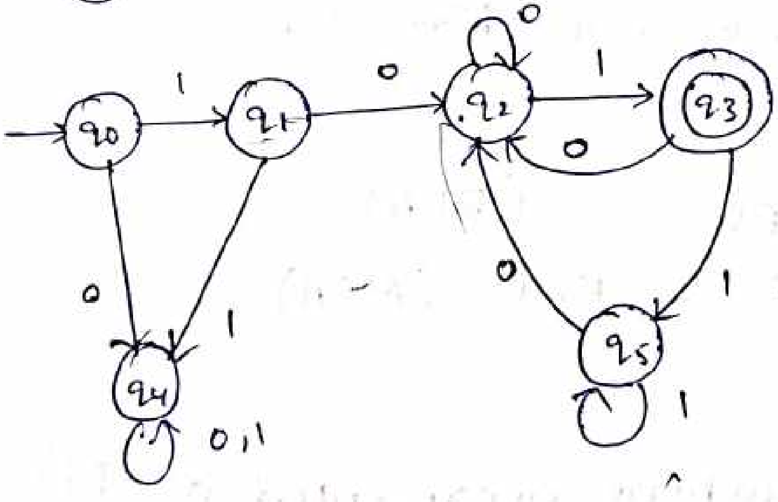
Ex:- Strats with 10 ends with 01

L = {101, 1001, 10101, 10001, ---}

L' = {ε, 010, 0110, 01110, -----}



~~(wrong)~~

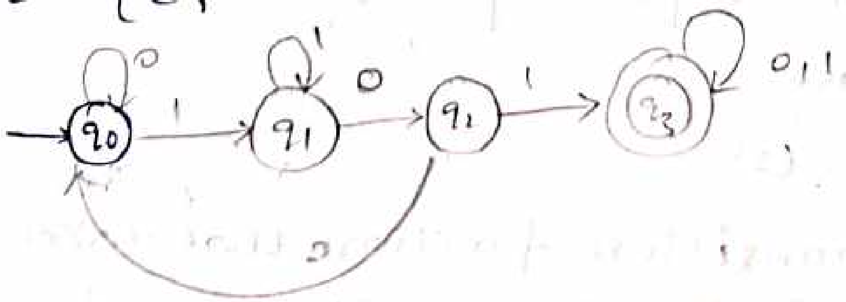


(wrong)

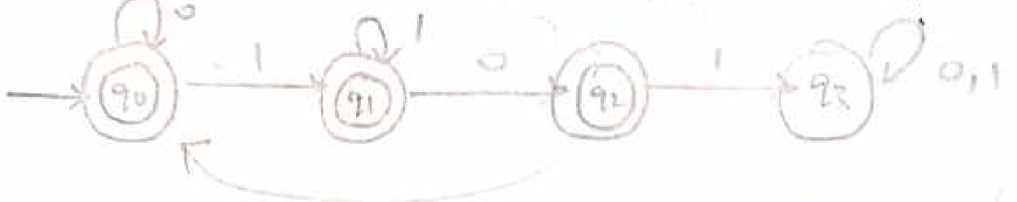
Ex: Substring is 101

L = {101, 0101, 1101, 1010, 1011, ----}

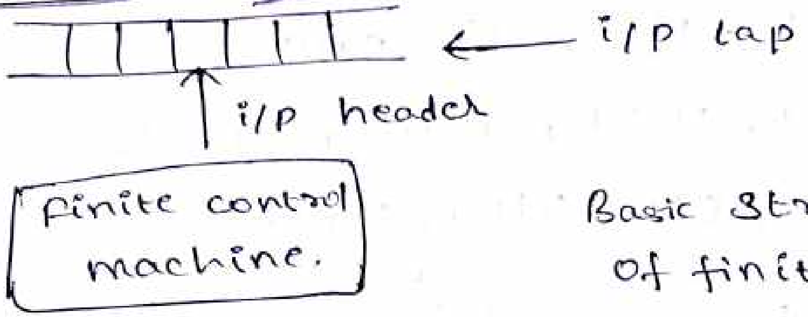
L' = {ε, 010, 001, 011, ----}   1101110



Ex:- not contains 101 as substring.



→ finite automata (or) FSM



Basic Structure of finite automata

finite control machine contains all control transitions. It takes i/p, changes state if required and moves forward.

⇒ Header moves from left to right. It doesn't move in backward direction.

* Types of FSM:-

① Deterministic FSM (DFA)

② Non-deterministic FSM (NFA)

① DFA :-

It is a 5-tuple machine represented as 'M'.

$M = (Q, \Sigma, \delta, q_0, F)$ [Here each & every i/p symbol should be used in DFA]

Q - finite set of states

Σ - finite set of input symbols

q_0 - initial state

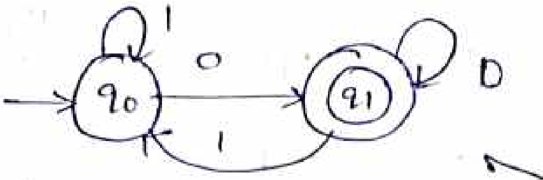
F - final state(s)

δ - it is a transition function that takes 2 arguments which are state and i/p symbol then returns a state as output. Ex:- $\delta(q_0, 0) = q_1$

→ Mapping function for DFA :-

$$Q \times \Sigma \longrightarrow Q$$

→ Transition table construction :-



ends with '0'.

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_1$

$\delta(q_1, 1) = q_0$

I/p / states	0	1
→ q_0	q_1	q_0
$\circled{q_1}$	q_1	q_0

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

$Q = \{q_0, q_1)$, $\Sigma = \{0, 1\}$, $q_0 = q_0$, $F = q_1$

check for 1010

$\delta(q_0, \underline{1}010) \vdash \delta(q_0, \underline{0}10)$

$\vdash \delta(q_1, \underline{1}0)$

$\vdash \delta(q_0, 0)$

$\vdash q_1$ ⟶ $F = q_1$. So it is acceptable i/p.

check for 101

$\delta(q_0, \underline{1}01) \vdash \delta(q_0, \underline{0}1)$

$\vdash \delta(q_1, 1)$

$\vdash q_0$ → $F \neq q_0$. So it is not acceptable string

Steps to answer:-

1, Construction of DFA

2, Write the tuples with transition functions

3, Write the transition table.

4, Acceptance string example.

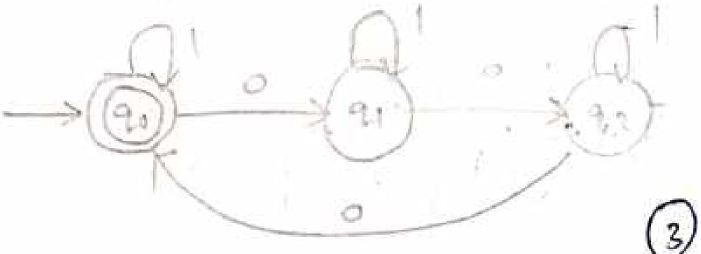
5, Non-acceptance string example.

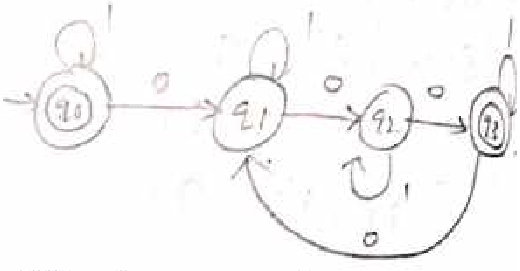
Ex:- Construct the FA which accepts set of strings where no. of 0's in every string is in multiples of 3. over the Σ alphabet set Σ = {0,1}.

11,1

L = { ε, 000, 1000, 10001, 0100, 0010, 0001, 111,

L' = { 1, 100, 001, . . - - - }





① M = (Q, Σ, δ, q0, F)

Q = {q0, q1, q2}

Σ = {0,1}

q0 = q0   F = q0.

③

I/P States	0	1
→ q0	q1	q0
q1	q2	q1
q2	q0	q2

② δ(q0,0) = q1

δ(q0,1) = q0

δ(q1,0) = q2

δ(q1,1) = q1

δ(q2,0) = q0

δ(q2,1) = q2

④ check for 10001

δ(q0, 10001) ⊢ δ(q0, 0001)

⊢ δ(q1, 001)

⊢ δ(q2, 01)

⊢ δ(q0, 1)

⊢ q0 = F

(acceptable)

⑤ 110

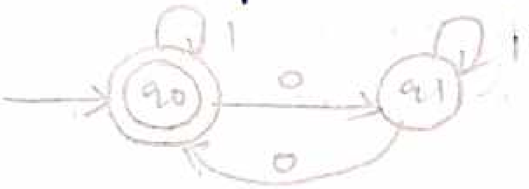
δ(q0, 110) ⊢ δ(q0, 10)

⊢ δ(q0, 0)

⊢ q1 ≠ F

(not acceptable.

ex:- String having even no's 0's.



$L = \{\epsilon, 1, 11, 00, 010, 100, 001, \ldots\}$

$L' = \{000, 1000, 0100, \ldots\}$

① $M = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1\}$

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

$q_0 = q_0$ $F = q_0$.

③

I/P States	0	1
q_0	q_1	q_0
q_1	q_0	q_1

② $\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_0$

$\delta(q_1, 1) = q_1$

④ Check for

10011

$\delta(q_0, \underline{1}0011) \vdash \delta(q_0, \underline{0}011)$

$\vdash \delta(q_1, 011)$

$\vdash \delta(q_0, 11)$

$\vdash \delta(q_0, 1)$

$\vdash \delta(q_0) = F$

(acceptable)

⑤ check for

101100

$\delta(q_0, 101100) \vdash \delta(q_0, 01100)$

$\vdash \delta(q_1, 1100)$

$\vdash \delta(q_1, 100)$

$\vdash \delta(q_1, 00)$

$\vdash \delta(q_0, 0)$

$\vdash q_1 \neq F$

(unacceptable).

→ conditions for DFA

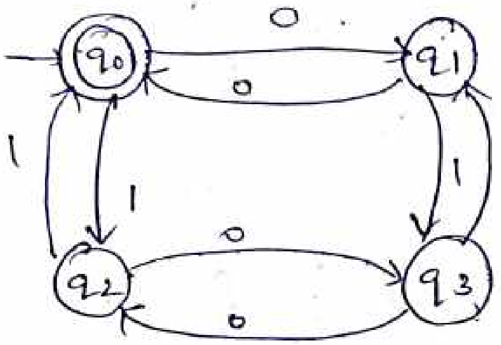
1, The each & every i/p alphabet over Σ should be used in transition functions

2, Used only once (i/p alphabet).

Ex:- Even no. of 0's and even no. of 1's.

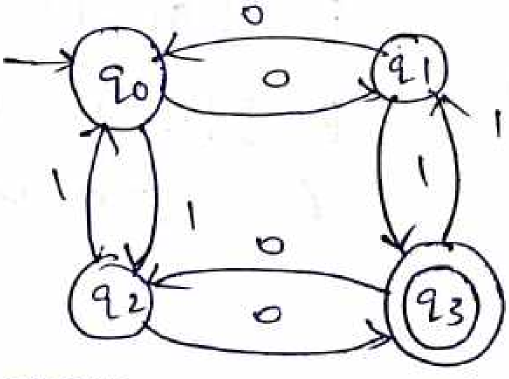
L = {ϵ, 0011, 00, 11, 1100, ----}

L' = {10, 01, 1000, ------}

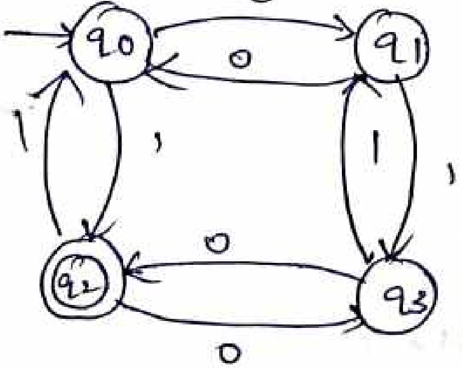


→ odd no. of 0's odd no. of 1's

L = {01, 10, 0111, 0001, ---}



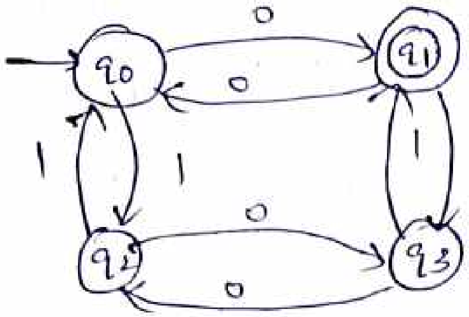
→ even no. of 0's odd no. of 1's.



L = {1, 001, 111, 010, --}

→ odd no. of 0's even no. of 1's

$L = \{0, 000, 11000, 110, 101, \ldots\}$



Assignment:-

① string ends with aab or aaba

② Construct language $L = \{(ab)^n / n \geq 0\}$ over $\Sigma = \{a, b\}$

$L = \{\epsilon, ab, abab, ababab, \ldots\}$

$L' = \{a, b, aa, bb, \ldots\}$

③ $L = \{w \in (0,1)^* /$ 3rd symbol is 0, 5th symbol is 1 $\}$ [* means ≥0]

$L = \{00\underline{0}0\underline{1}, 11\underline{0}1\underline{1}, 11\underline{0}0\underline{1}, \ldots\}$

② NFA :-

It is a 5 tuple machine represented as 'M'.

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

Q - finite set of states

Σ - alphabet set.

q0 - initial state

F - final set of states

Transition function that takes states as arguments and returns power set of Q is called as δ.

→ Mapping function

$Q \times (\Sigma \cup \epsilon) \longrightarrow 2^Q$

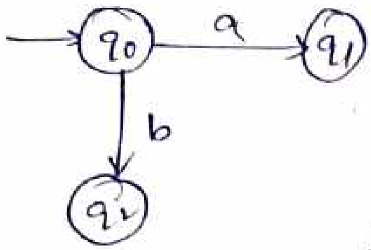
$Q \times \Sigma^* \longrightarrow 2^Q$

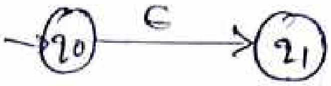
→ Conditions for NFA

1, Input Symbols can be used any no. of time

2, each & every alphabet over Σ need not be used. → ϵ can be used as input.

Ex:-





$\delta(q_0, a) = \epsilon$ $\qquad$ $\delta(q_0, a) = \{q_0, q_1\}$

$\delta(q_0, a) = q_0$ $\qquad$ $\delta(q_0, a) = \{q_0, q_2\}$

$\delta(q_0, a) = q_1$ $\qquad$ $\delta(q_0, a) = \{q_1, q_2\}$

$\delta(q_0, a) = q_2$ $\qquad$ $\delta(q_0, a) = \{q_1, q_2, q_3\}$

no. of states $= 3$ (Q)

So max. no. of combinations $= 2^Q = 2^3 = 8$.

ex:- $Q = \{q_0, q_1\}$.

$\delta(q_0, a) = \epsilon$

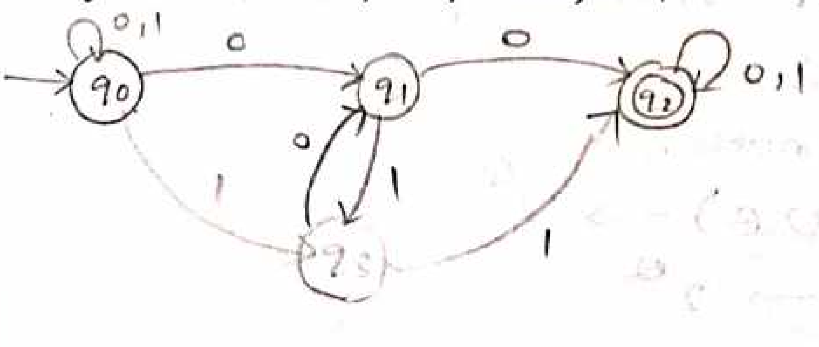
$\delta(q_0, a) = q_0$

$\delta(q_0, a) = q_1$

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

ex:- Design NFA for language L = { all the strings over $\Sigma = \{0,1\}$ has atleast 2 consecutive 0's or 1's }

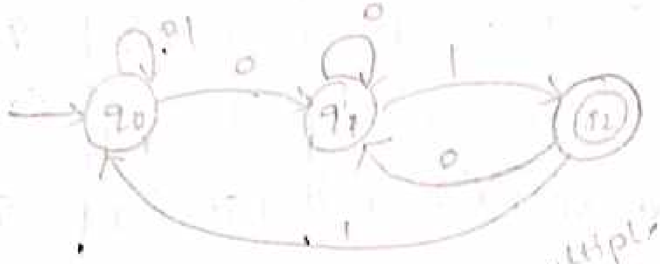
$L = \{00, 11, 100, 111, 000, 011 \cdots\}$ $\quad$ $L' = \{\epsilon, 101, 010, \cdots\}$



Ex:- ends with 01

L = {01, 101, 001, 0001, 0101, 1001, 1101 ...

L' = {ε, 1, 0, 11, 00, 10, ... }



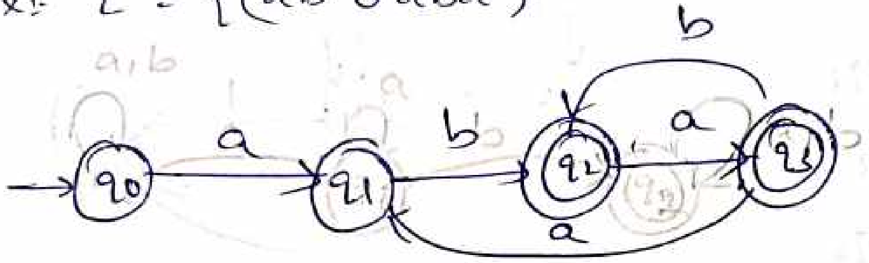
0101

011101

1001

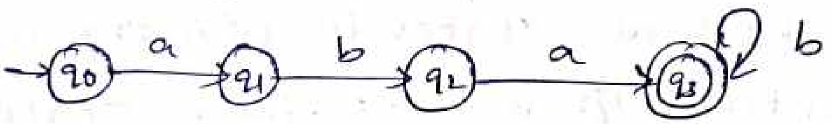
Ex:- L = {(ab $\overset{or}{\cup}$ aba)* } ← multiple times.

$\underbrace{abababaab}_{\text{acceptable}}$



Ex:- Construct NFA for language

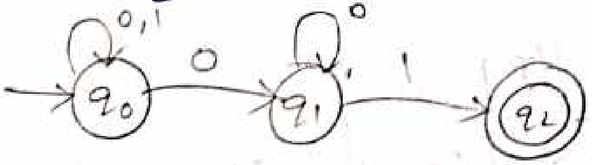
L = { abab^n / n ≥ 0 }   L = { aba, abab, ababb, ..}



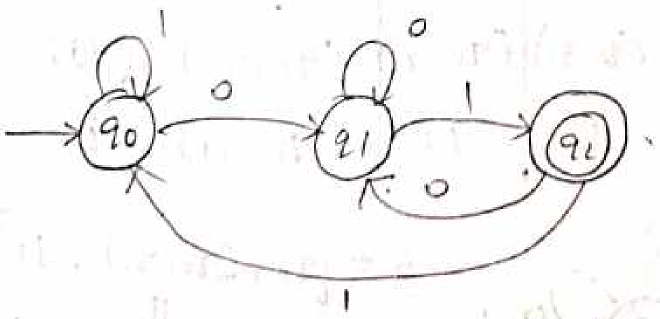
Ex:- desing NFA & DFA accepting all strings ending with 01 over Σ = {0, 1}.

L = { 01, 101, 001, 1101, 0001, 1001, 0101 ... }

L' = { ε, 0, 1, 10, 11, 00, 111, 000, ... }



NFA



DFA.

* Converting from NFA to DFA:-

Transition table-NFA

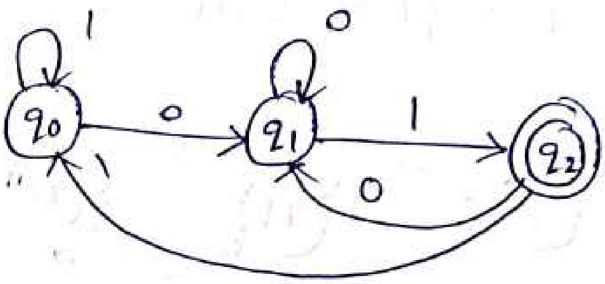
States \ i/p	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	∅	q_2
q_2	∅	∅

Transition table for DF

	States \ i/p	0	1
q_0 →	q_0	(q_0, q_1)	q_0
q_1	(q_0, q_1)	(q_0, q_1)	(q_0, q_2)
q_2	(q_0, q_2)	(q_0, q_1)	q_0.

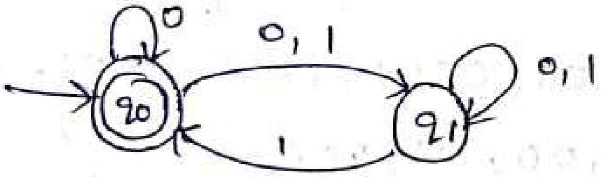
rename

States \ i/p	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0



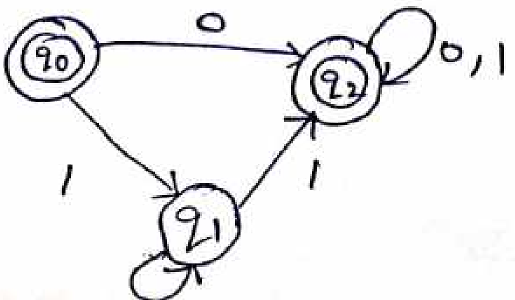
To determine final state, first consider final (q2) state in NFA. Then all states in DFA transition table which contain q_2 will be final states.

Ex:- Convert following NFA to DFA.



States \ i/p	0	1
→ q_0	$\{q_0, q_1\}$	q_1
q_1	q_1	$\{q_0, q_1\}$

States \ i/p	0	1	
→ q_0	(q_0, q_1)	q_1	q_0 ←
(q_0, q_1)	(q_0, q_1)	(q_0, q_1)	q_2
q_1	q_1	(q_0, q_1)	q_1



$Q = \{q_0, (q_0, q_1), q_1\}$

$\parallel$

$Q' = \{q_0, q_2, q_1\}$

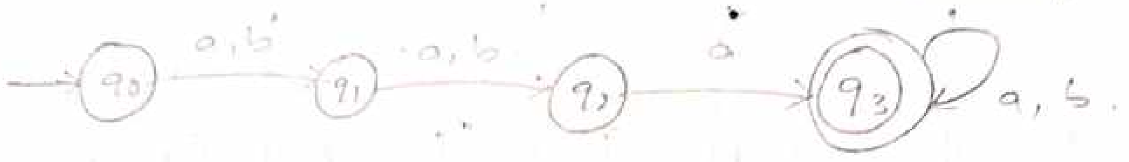
$F = \{q_0, (q_0, q_1)\}$

$= \{q_0, q_2\}$

Ex:- NFA - 3rd charcter should be 'a'

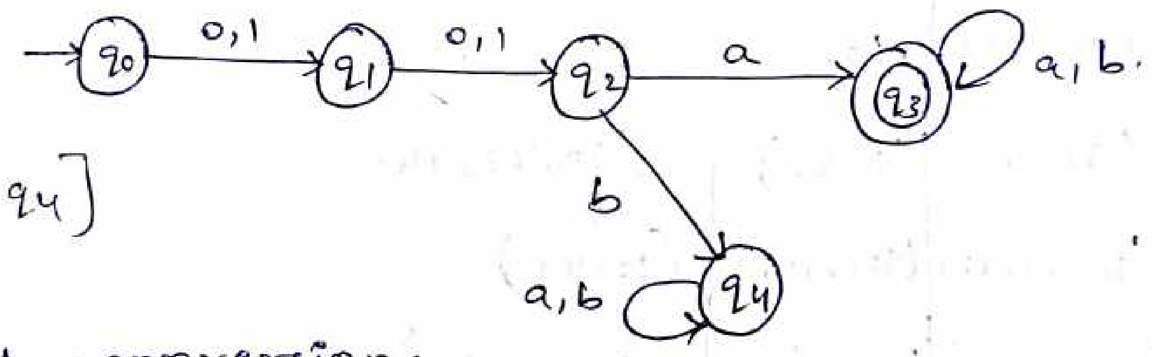
$\Sigma = \{a, b\}$

L= {aaa, bba, aba, baa, aaaa, bbab, abab, baab, baaa, - - -}.



i/p States	a	b.
→q0	q1	q1
q1	q2	q2
q2	q3	∅
q3	q3	q3

i/p states	a	b
→q0	q1	q1
q1	q2	q2
q2	q3	∅
(q4) ∅	∅	∅
q3	q3	q3



[∅ = q4]

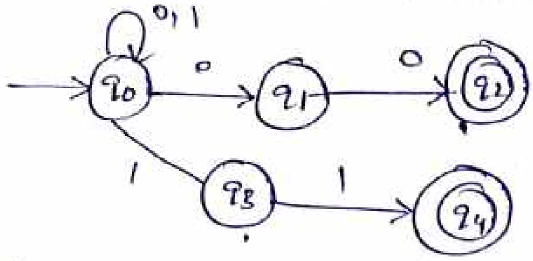
*Steps for conversion:-

1, start with initial state.

2, After finding the transition of initial stat only the resultant states into the list until no new state is added to the list

3, declare the states as final if it has atleast one final state of NFA.

Exc-



i/p States	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_1	q_2	ϕ
$\boxed{q_2}$	ϕ	ϕ
q_3	~~q_4~~ ϕ	~~ϕ~~ q_4
$\boxed{q_4}$	ϕ	ϕ

	0	1
~~q_0~~	~~(q_0, q_1)~~	~~(q_0, q_3)~~
~~(q_0, q_1)~~	~~(q_0, q_1, q_2)~~	~~(q_0, q_3, ϕ)~~
~~(q_0, q_3)~~	~~(q_0, q_1, ϕ)~~	~~(q_0, q_3, q_4)~~
~~(q_0, q_1, q_2)~~	~~(q_0, q_1, q_2, ϕ)~~	

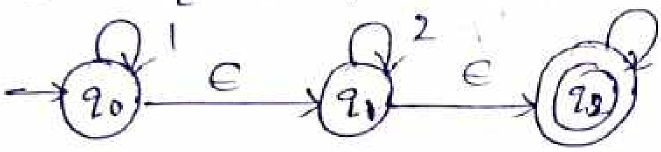
i/p States	0	1
→ q_0	(q_0, q_1)	(q_0, q_3)
q_1 (q_0, q_1)	(q_0, q_1, q_2)	(q_0, q_3)
q_2 (q_0, q_3)	(q_0, q_1)	(q_0, q_3, q_4)
q_3 (q_0, q_1, q_2)	(q_0, q_1, q_2)	(q_0, q_3)
q_4 (q_0, q_3, q_4)	(q_0, q_1)	(q_0, q_3, q_4)

* NFA with ε-moves:-

ex:- $L = \{1^* 2^* 3^* \,/\, \Sigma = \{1,2,3\}\}$

$L = \{\epsilon, 1, 2, 3, 12, 13, 23, 112 \dots\}$

$L' = \{21, 31, 32 \dots\}$

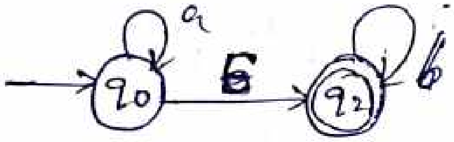


[if 211 is i/p, it reaches q2 by using ε, but it will not be accepted because the i/p string will not become empty.

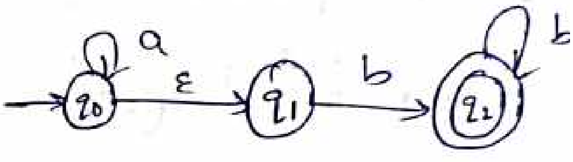
Ex:- $L = \{a^* b b^* \,/\, \Sigma = \{a, b\}\}$

$L = \{b, ab, abb, bb, aab, aabbb \dots\}$

$L' = \{\epsilon, a, aaa \dots\}$



(or)

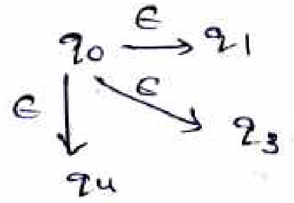


* ε-closure:- It is a set of all states which are reachable from state P on null transition (ε transition)

1. ε-closure of P = x , Here x ∈ Q [x is a set of states]

   ε-closure (P) = x , x ∈ Q [x is a set of states]

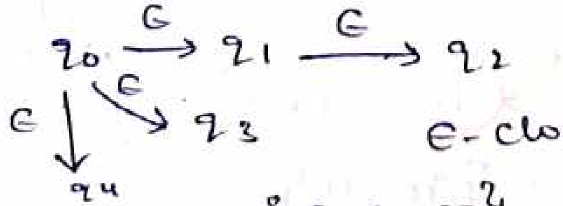
ex:-



ε-closure (q0) = {q1, q3, q4}

2. If there exists ε-closure (P) = q and $\delta(q, \epsilon) = r$ and then ε-closure (P) = {q, r}

ex:-



ε-closure (q0) = {q1, q2, q3, q4}

3. ε-closure (P) = {P, q, r}

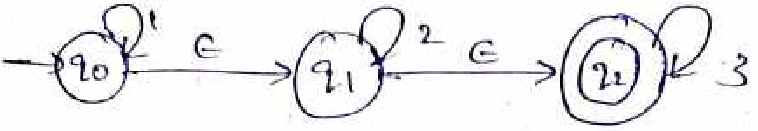
   ε-closure (q0) = {q0, q1, q2, q3, q4}

For last diagram

ϵ-closure(q_1) = {q_2} = {q_2} = {q_1, q_2}

ϵ-closure(q_3) = {q_3}.

Ex:- $L = \{1^* 2^* 3^* \,/\, \Sigma = \{1,2,3\}\}$

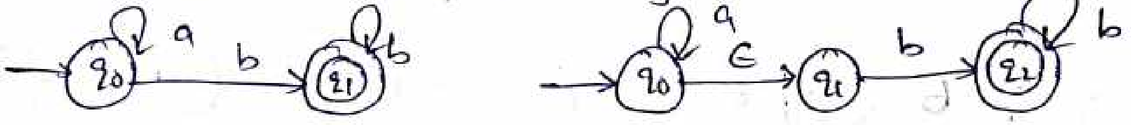


ϵ-Closure(q_0) = {q_0, q_1, q_2}

ϵ-closure(q_1) = {q_2} = {q_2} = {q_1, q_2}

ϵ-Closure(q_2) = {q_2}

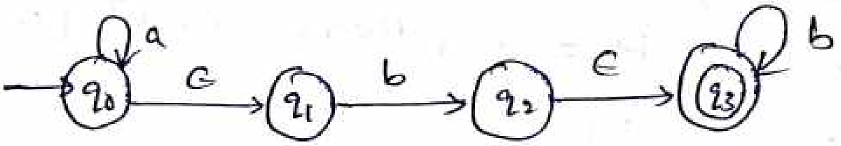
Ex:- $L = \{a^* b b^* \,/\, \Sigma = \{a,b\}\}$



ϵ-closure(q_0) = {q_1} = {q_1} = {q_0, q_1}

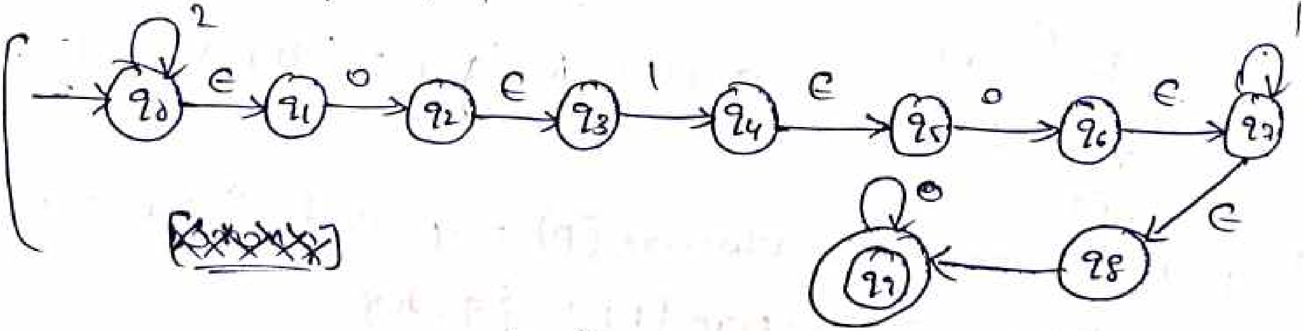
ϵ-closure(q_1) = {q_1}

ϵ-closure(q_2) = {q_2}

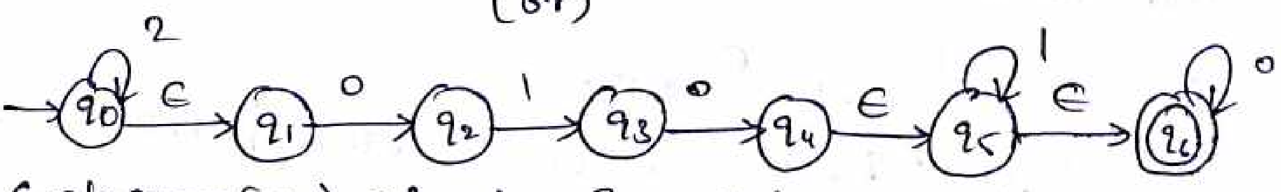


Ex:-

$L = \{2^* 0 1 0 1^* 0^*\}$



(or)



ϵ-closure(q_0) = {q_1} = {q_0, q_1}

ϵ-closure(q_1) = {q_1}

ϵ-Closure(q_2) = {q_2}

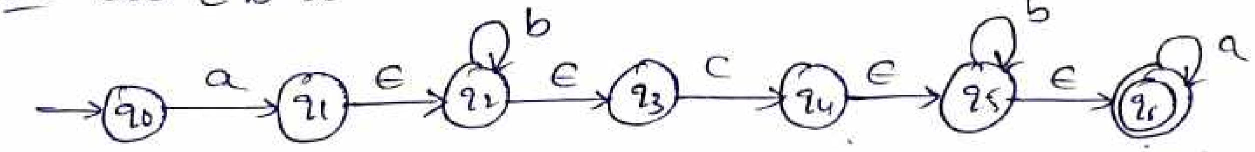
ϵ-Closure(q_3) = {q_3}

ϵ-Closure(q_4) = {q_4, q_5, q_6}

ϵ-Closure(q_5) = {q_5, q_6}

ϵclosure(q_6) = {q_6}

Ex:- $ab^*cb^*a^*$.



ε-closure(q_0) = {q_0}

ε-closure(q_1) = {q_1, q_2, q_3}

ε-closure(q_2) = {q_2, q_3}

ε-closure(q_3) = {q_3}

ε-closure(q_4) = {q_4, q_5, q_6}

ε-closure(q_5) = {q_5, q_6}

ε-closure(q_6) = {q_6}

* Conversion of NFA with ε-moves to without ε-moves:-

→ ε-closure denoted by δ'

$$\delta'(q,a) = \epsilon\text{-closure}(\delta(\delta'(q), a))$$

Ex:- $\delta'(q_0, a) = \delta'(\delta(\delta'(q_0), a))$

$= \delta'(\delta(q_0, a))$

$= \delta'(q_1) = \{q_1, q_2, q_3\}$

$\delta'(q_0, b) = \delta'(\delta(\delta'(q_0), b))$

$= \delta'(\delta(q_0, b))$

$= \delta'(\phi) = \phi$

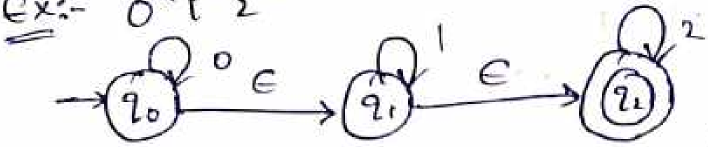
$\delta'(q_1, a) = \delta'(\delta(\delta'(q_1), a))$

$= \delta'(\delta(\{q_1, q_2, q_3\}, a))$

$=$

Ex:-** $0^*1^*2^*$

[with ε-moves to without ε-moves



$\epsilon\text{-closure}(q_0) = \{q_0\} = \{q_1, q_2\} = \{q_0, q_1, q_2\}$

$\epsilon\text{-closure}(q_1) = \{q_2\} = \{q_1, q_2\}$

$\epsilon\text{-closure}(q_2) = \{q_2\}$.

$$\begin{aligned}
\delta'(q_0,0) &= \delta'(\delta(\delta'(q_0,\epsilon),0))\\
&= \delta'(\delta(\{q_0,q_1,q_2\},0))\\
&= \delta'(\delta(q_0,0)\cup\delta(q_1,0)\cup\delta(q_2,0))\\
&= \delta'(q_0\cup\phi\cup\phi)\\
&= \delta'(q_0)\\
&= \{q_0,q_1,q_2\}
\end{aligned}$$

$$\begin{aligned}
\delta'(q_0,1) &= \delta'(\delta(\delta'(q_0,\epsilon),1))\\
&= \delta'(\delta(\{q_0,q_1,q_2\},1))\\
&= \delta'(\delta(q_0,1)\cup\delta(q_1,1)\cup\delta(q_2,1))\\
&= \delta'(\phi\cup q_1\cup\phi)\\
&= \delta'(q_1)\\
&= \{q_1,q_2\}
\end{aligned}$$

$$\begin{aligned}
\delta'(q_0,2) &= \delta'(\delta(\delta'(q_0,\epsilon),2))\\
&= \delta'(\delta(\{q_0,q_1,q_2\},2))\\
&= \delta'(\delta(q_0,2)\cup\delta(q_1,2)\cup\delta(q_2,2))\\
&= \delta'(\phi\cup\phi\cup q_2)\\
&= \delta'(q_2)\\
&= \{q_2\}
\end{aligned}$$

$$\delta'(q_1,0) = \delta'(\delta(\delta'(q_1,\epsilon),0))$$
$$= \delta'(\delta(\{q_1,q_2\},0))$$
$$= \delta'(\delta(q_1,0) \cup \delta(q_2,0))$$
$$= \delta'(\phi \cup \phi)$$
$$= \delta'(\phi)$$
$$= \phi$$

$$\delta'(q_1,1) = \delta'(\delta(\delta'(q_1,\epsilon),1))$$
$$= \delta'(\delta(\{q_1,q_2\},1))$$
$$= \delta'(\delta(q_1,1) \cup \delta(q_2,1))$$
$$= \delta'(q_1 \cup \phi)$$
$$= \delta'(q_1)$$
$$= \{q_1,q_2\}$$

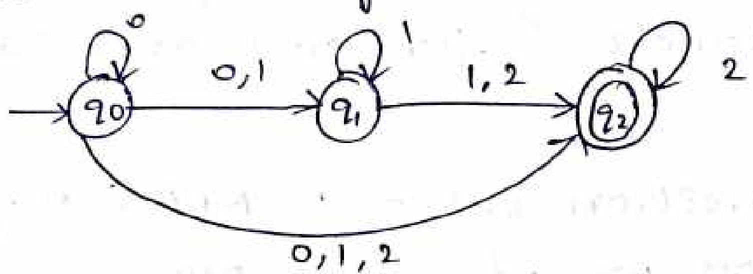
$$\delta'(q_1,2) = \delta'(\delta(\delta'(q_1,\epsilon),2))$$
$$= \delta'(\delta(\{q_1,q_2\},2))$$
$$= \delta'(\delta(q_1,2) \cup \delta(q_2,2))$$
$$= \delta'(\phi \cup q_2)$$
$$= \delta'(q_2)$$
$$= \{q_2\}$$

$$\delta'(q_2,0) = \delta'(\delta(\delta'(q_2,\epsilon),0))$$
$$= \delta'(\delta(\{q_2\},0))$$
$$= \delta'(\phi)$$
$$= \phi$$

$$\delta'(q_2,1) = \delta'(\delta(\delta'(q_2,\epsilon),1))$$
$$= \delta'(\delta(q_2,1))$$
$$= \delta'(\phi)$$
$$= \phi$$

$$\delta'(q_2,2) = \delta'(\delta(\delta'(q_2,\epsilon),2))$$
$$= \delta'(\delta(q_2,2))$$
$$= \delta'(q_2)$$
$$= \{q_2\}$$

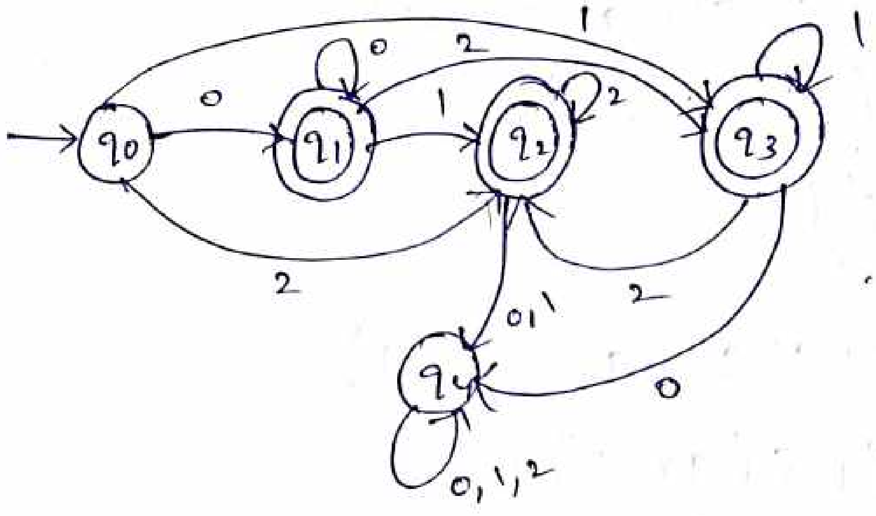
Transition diagram of NFA



Convertion to DFA.

States \ i/p	0	1	2
→ q_0	$\{q_0,q_1,q_2\}$	$\{q_1,q_2\}$	$\{q_2\}$
q_1	$\{\phi\}$	$\{q_1,q_2\}$	$\{q_2\}$.
$\boxed{q_2}$	ϕ	ϕ	$\{q_2\}$

States \ i/p	0	1	2	
q_0	(q_0,q_1,q_2)	(q_1,q_2)	(q_2)	q_0 ←
(q_0,q_1,q_2)	(q_0,q_1,q_2)	(q_1,q_2)	(q_2)	$\boxed{q_1}$
(q_1,q_2)	ϕ	(q_1,q_2)	q_2	$\boxed{q_3}$
q_2	ϕ	ϕ	q_2	$\boxed{q_2}$
ϕ	ϕ	ϕ	ϕ	q_4



→ Convert with ε-moves to DFA

1, Draw diagram for equivalent language

2, Calculate ε-closure of each & every state.

3, Compute or calculate δ' for each and every alphabet

4, Draw the transition table of NFA & diagram

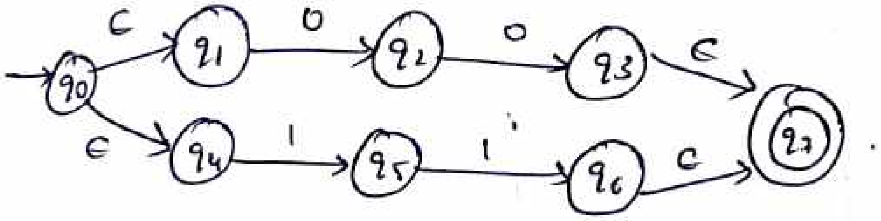
5, start conversion procedure of DFA.

6, Transition table of DFA

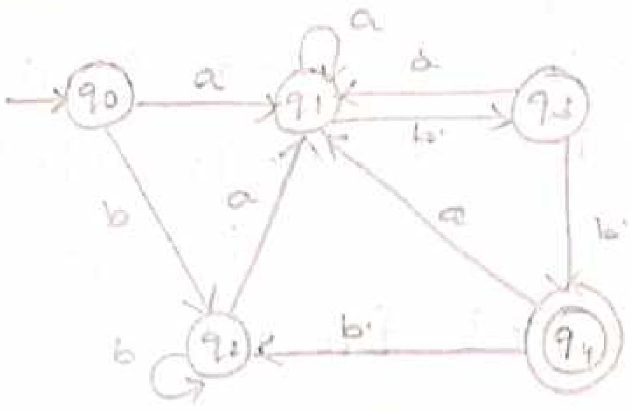
7, Transition diagram of DFA

8, Accepting string & non-accepting string procedure.

Ex:- (00+11)



* Minimisation of DFA :-



Equivalence method :-

⇒ 0 - equivalence - π_0 : Here we keep all final states in one set and non-final states in one set.

$\{q_4\}$ $\{q_0, q_1, q_2, q_3\}$

⇒ 1 - equivalence - π_1

$\{q_4\}$ $\{q_0, q_1, q_2, q_3\}$

$\delta(q_0, a) - q_1, \delta(q_0, b) - q_2$

$\delta(q_1, a) - q_1, \delta(q_1, b) - q_3.$

} → These transitions o/p belong to sam set. So $q_0 \equiv q_1$

ll4. $\delta(q_0, a) - q_1$ $\delta(q_0, b) - q_2$

$\delta(q_3, a) - q_1$ $\delta(q_3, b) - q_4$

$\{q_1, q_2\}$ $\{q_4\}$

so $q_0 \neq q_3$

ll4 $\delta(q_0, a) - q_1$ $\delta(q_0, b) - q_2$

$\delta(q_2, a) - q_1$ $\delta(q_2, b) - q_2$

$q_0 \equiv q_2$

$\{q_4\}$ $\{q_3\}$ $\{q_0, q_1, q_2\}$

⇒ 2-equivalence-π_2

$\{q_4\}$ $\{q_3\}$ $\{q_0, q_1, q_2\}$

$\delta(q_0,a)-q_1$ $\delta(q_0,b)-q_2$ $\delta(q_0,a)-q_1$ $\delta(q_0,b)-q_1$

$\delta(q_1,a)-q_1$ $\delta(q_1,b)-q_3$ $\delta(q_2,a)-q_1$ $\delta(q_2,b)-q$

$q_0 \neq q_1$ $q_0 \equiv q_2$

$\{q_4\}$ $\{q_3\}$ $\{q_1\}$ $\{q_0, q_2\}$

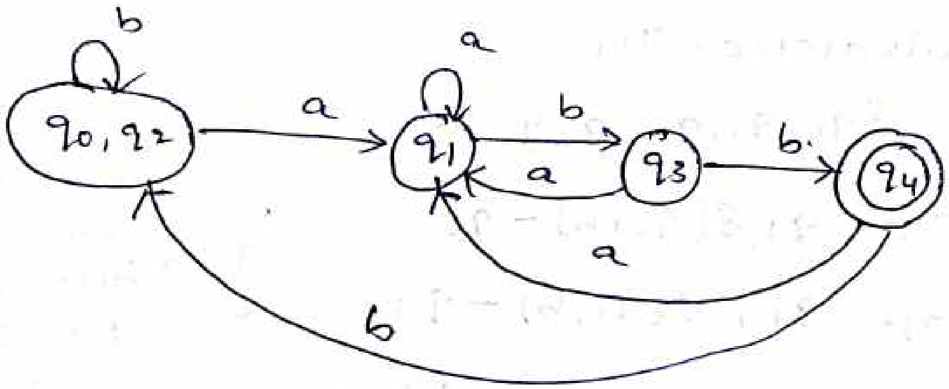
⇒ 3-equivalence - π_3

$\{q_4\}$ $\{q_3\}$ $\{q_1\}$ $\{q_0, q_2\}$

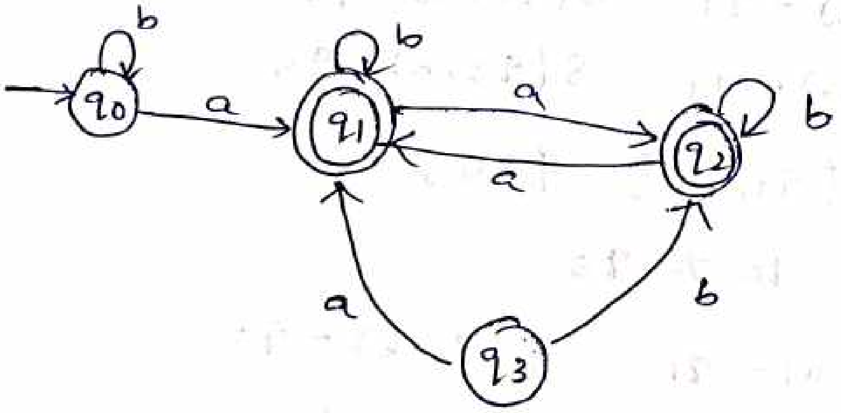
$\delta(q_0, a)- q_1$ $\delta(q_0, b) - q_2$

$\delta(q_2, a)- q_1$ $\delta(q_2, b) - q_2$

$q_0 \equiv q_2$



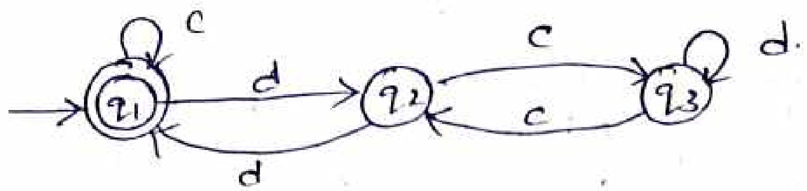
HW



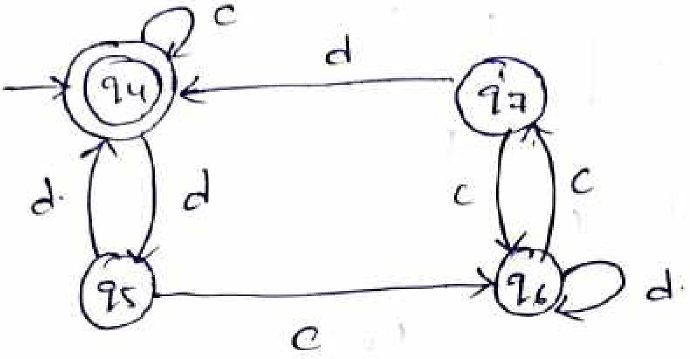
* Equivalence of finite automata:-

ex:-

(i)



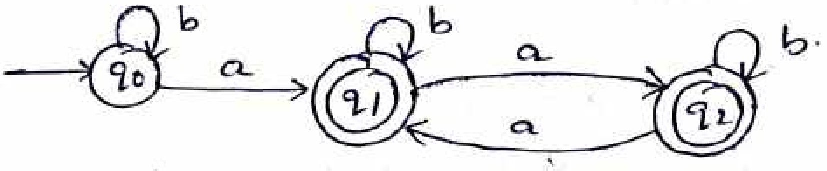
(ii)



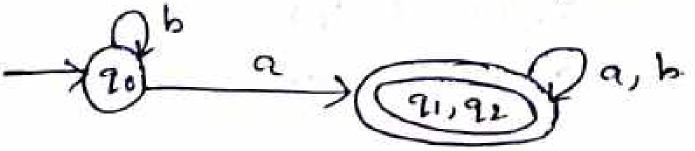
I/P State	c	d.
{q1, q4}	(q1, q4) F F	(q2, q5) NF NF
{q2, q5)	(q3, q6) NF NF	(q1, q4) F F
(q3, q6)	(q2, q7) NF NF	(q3, q6) NF NF
(q2, q7)	(q3, q6) NF NF	(q1, q4) F F

Both DFA's are equivalent

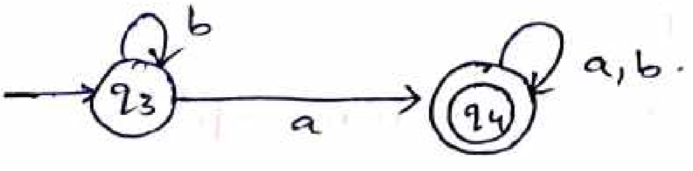
→ i)



ii)



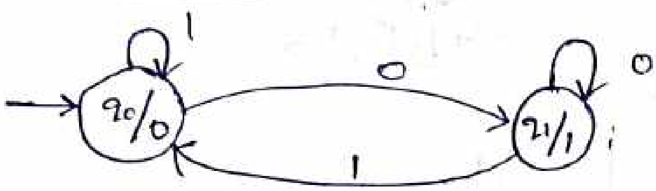
After renaming.



State \ i/p	a	b
(q_0, q_3)	(q_1, q_4) F F	(q_0, q_3) NF NF
(q_1, q_4)	(q_2, q_4) F F	(q_1, q_4) F F
(q_2, q_4)	(q_1, q_4) F F	(q_2, q_4) F F

Both DFA's are equivalent.

* Moore Machine:-



The o/p is associated with each state is called Moore machine.

Moore machine tuple

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

Q - finite set of states

Σ - finite set of i/p

δ - Transition function

q_0 - initial state

Δ - finite set of o/p's

λ - mapping function

$\lambda : Q \rightarrow \Delta$.

→ $Q = \{q_0, q_1\}$ $\Sigma = \{0, 1\}$

$\delta(q_0, 0) - q_1$ $\delta(q_1, 0) - q_1$

$\delta(q_0, 1) - q_0$ $\delta(q_1, 1) - q_0$

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

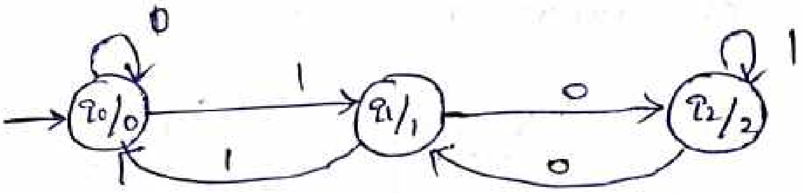
$\lambda : Q \rightarrow \Delta$.

Ex:- Construct Moore machine for %3 (modulo 3).

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

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

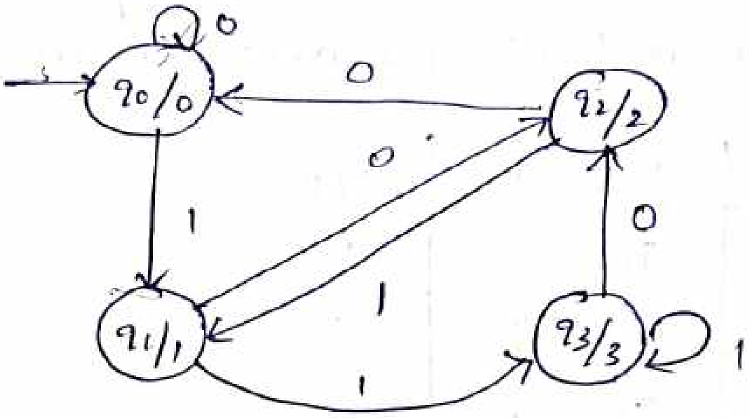
000	0
001	1
010	2
011	0
100	1
101	2
110	0.



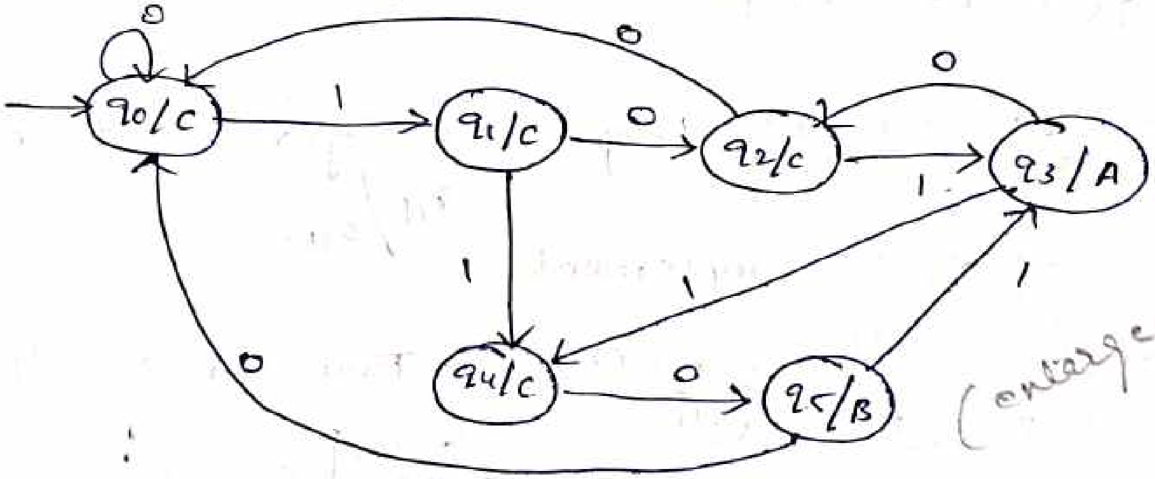
Ex:- Modulo4.

000	0
001	01
010	2
011	3
100	0
101	1
110	2
111	3

Δ = {0, 1, 2, 3}

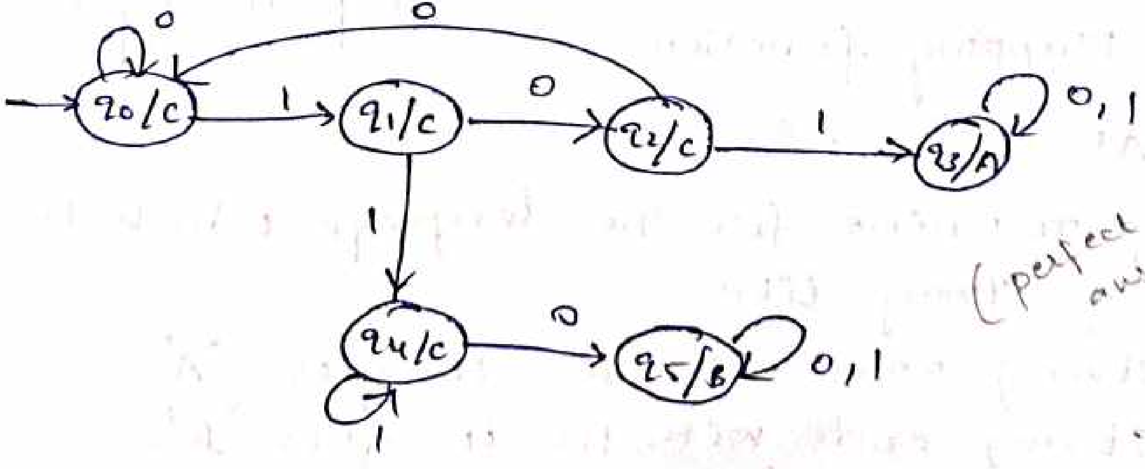


Ex:- If the substring ends with 101 which gives o/p 'A'. If the string ends with 110 which gives o/p 'B' otherwise 'C'.



(enlarged)

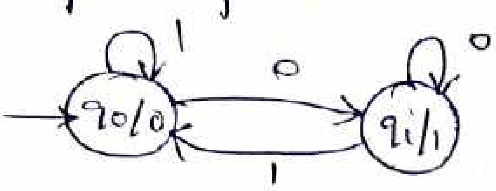
Ex:- If the string has substring 101 o/p 'A' if 110 o/p 'B' otherwise C.



(perfect answer)

* Transition table of moore machine:-

Binary complement



state \ i/p	0	1	o/p
→ q0	q1	q0	0
q1	q1	q0	1

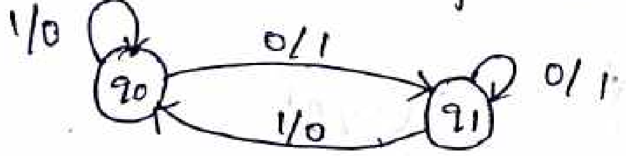
Transition table for above example.

state \ i/p	0	1	o/p
→ q0	q0	q1	C
q1	q2	q4	C
q2	q0	q3	C
q3	q3	q3	A
q4	q5	q4	C
q5	q5	q5	B

* Mealy machine:- Outputs are given to transitio

i/p / o/p.

Binary ~~equi~~. complement



Transition state/table

state \ i/p	0	o/p	1	o/p.
→ q0	q1	1	q0	0
q1	q1	1	q0	0

6 tuples

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

λ = Mapping function

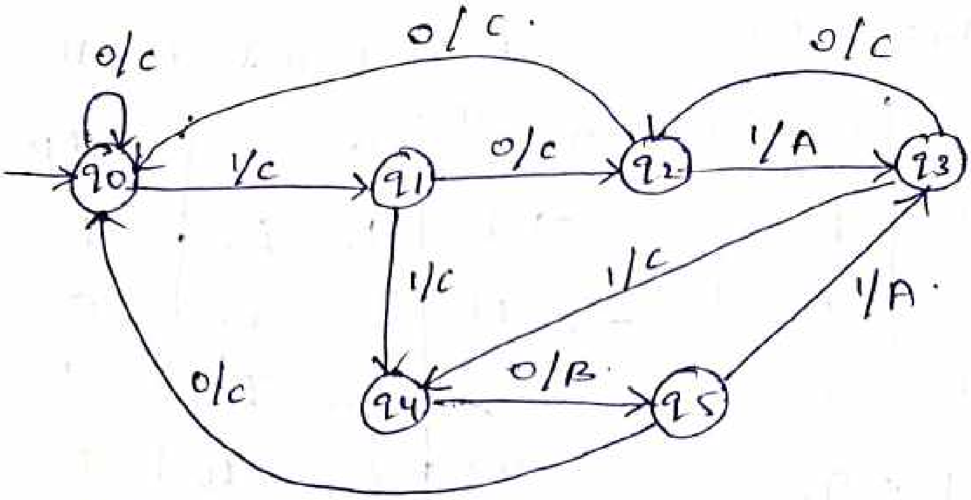
$\lambda : Q \times \Sigma \rightarrow \Delta$

Mealy machine for the language which is having strings like.

i, string ends with 101 it goes 'A'

string ends with 110 it goes 'B'.

otherwise 'C'.

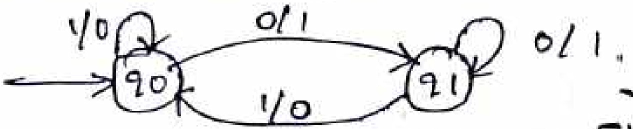


Transition table:-

i/p state	0	o/p	1	o/p
q0	q0	c	q1	c
q1	q2	c	q4	c
q2	q0	c	q3	A
q3	q2	c	q4	c
q4	q5	B	q4	c
q5	q0	c	q3	A

* Mealy to Moore Conversion:

Ex:-



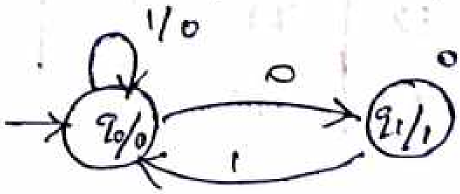
Transition table-Mealy

i/p State	0	o/p	1	o/p
q0	q1	1	q0	0
q1	q1	1	q0	0

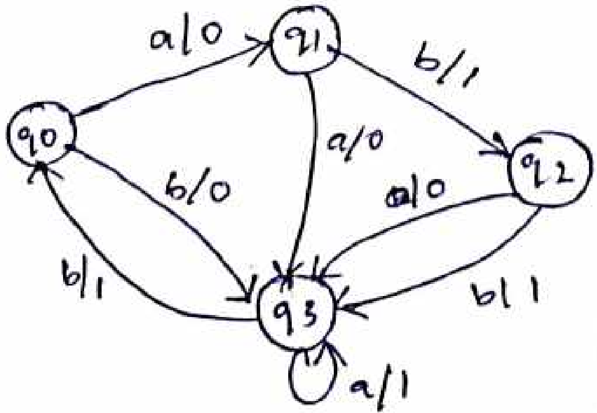
Transition table - moore.

i/p state	0	1	o/p
→ q0	q1	q0	0
q1	q1	q0	1

⇒



Ex:-



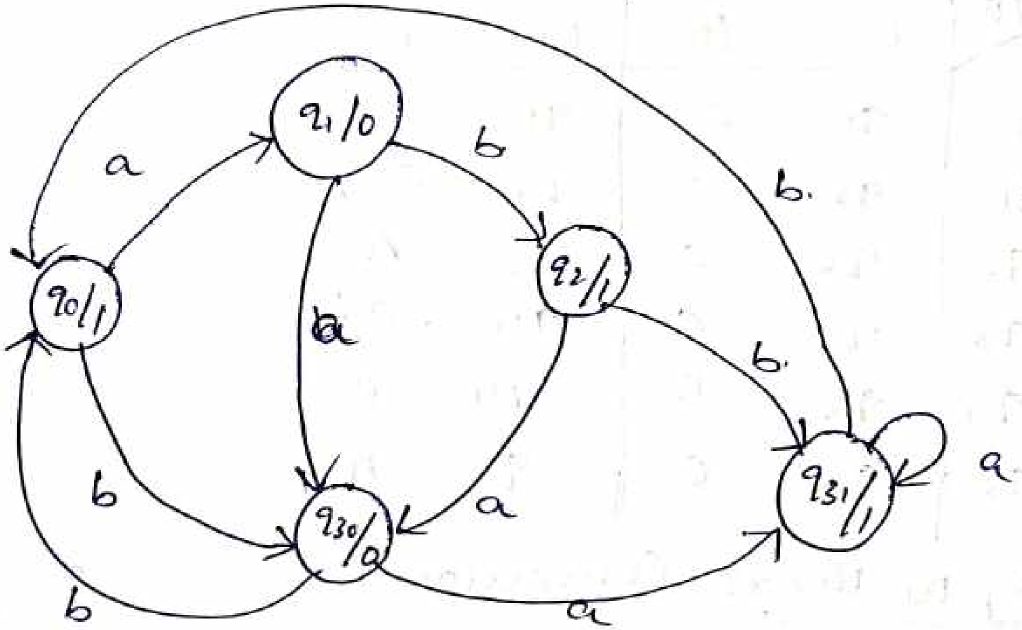
Transition table - Melay

State \ i/p	a	o/p	b	o/p
→ q_0	q_1	0	q_3	0
q_1	q_3	0	q_2	1
q_2	q_3	0	q_3	1
q_3	q_3	1	q_0	1

⇒

Transition table - Moor

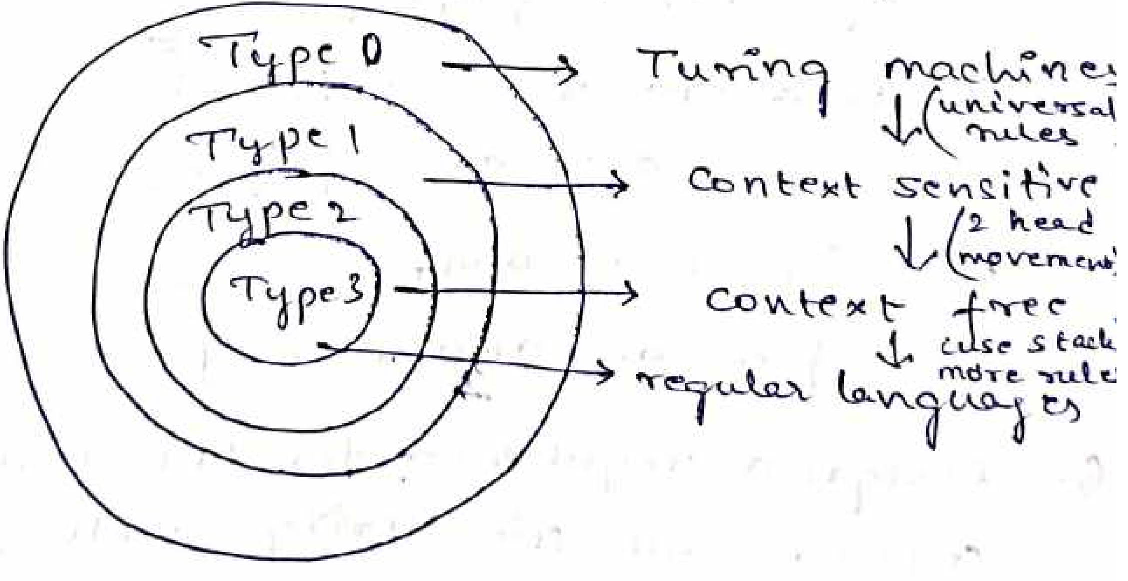
State \ i/p	a	b	o/p
→ q_0	q_1	q_{30}	1
q_1	q_{30}	q_2	0
q_2	q_{30}	q_{31}	1
q_{30}	q_{31}	q_0	0
q_{31}	q_{31}	q_0	1



Ex:-

	a	o/p	b	o/p
q_0	q_2	1	q_3	0
q_1	q_0	0	q_1	1
q_2	q_1	1	q_2	0
q_3	q_2	0.	q_0	1

* chomsky Hierarchy


Type 0
Type 1
Type 2
Type 3
Turing machines
universal rules
Context sensitive
2 head movement
Context free
use stack more rules
regular languages