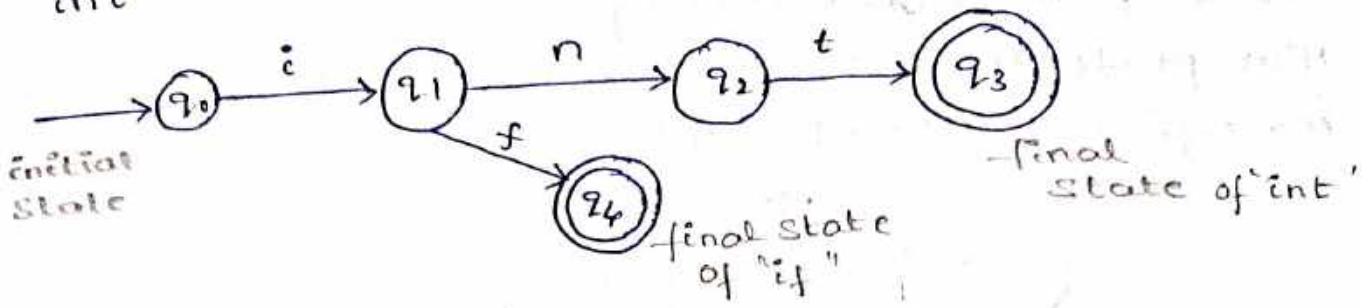
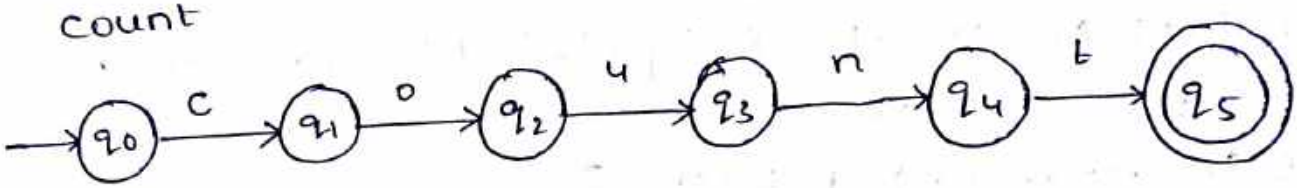
Unit-1

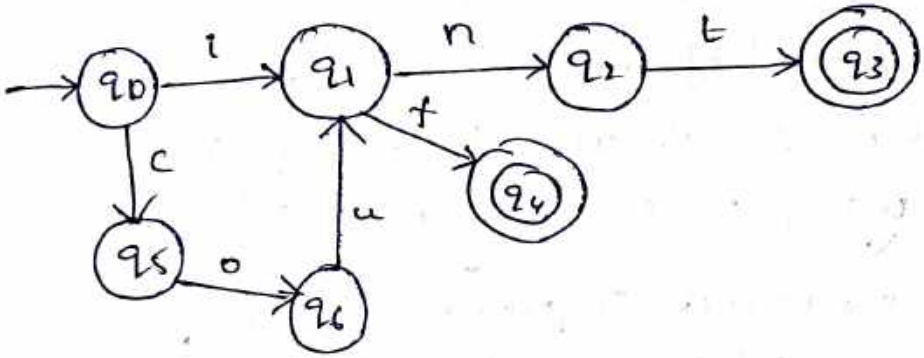
→ States:

int



count





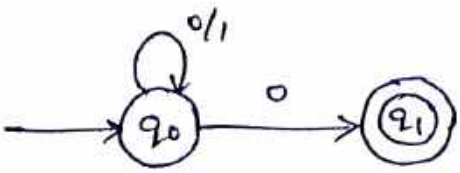
Q:- A string contains '0' at last position

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

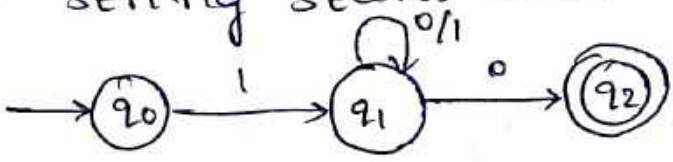
'0' can be one such string.

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'

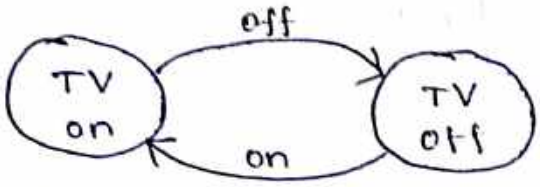


Automata:- It describes abstract model of emission

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 $\Sigma = \{0 \text{ to } 7\} = \{0,1,2,3,4,5,6,7\}$

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

Ex:- If $\Sigma = \{0,1\}$ $w = \{0, 1, 01, 10, 00, 11, 100 \ldots\}$

w_1 w_2 w_3 w_4

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.

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

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 $L = \{\epsilon, w1, w2, w3, \ldots\}$

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

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

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.

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

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

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

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

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

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

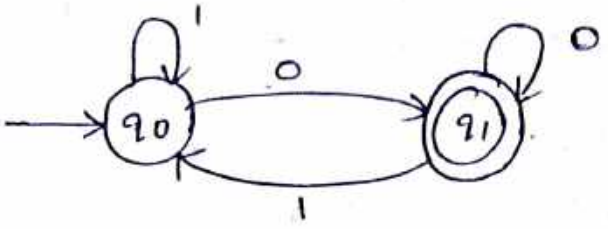
→ if ϵ is acceptable string the initial state will be final state

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

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

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

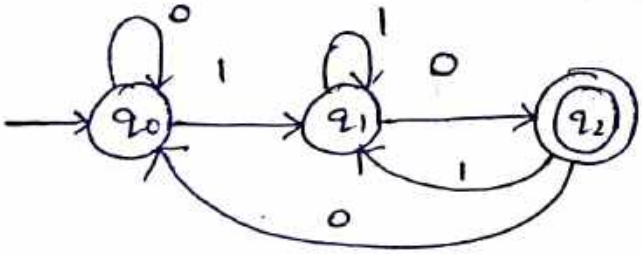
$L' = \{\epsilon, 1, 11, 01, 001, 011, 111, 101, \ldots\}$



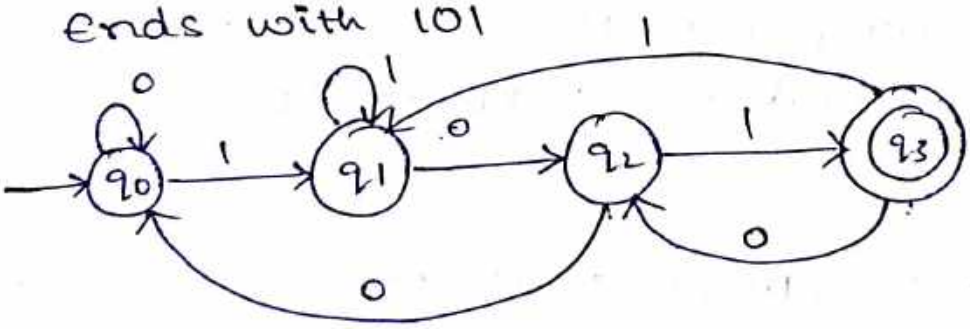
Ex:- Construct L for strings ends with 10

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

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



Ex:- Ends with 101



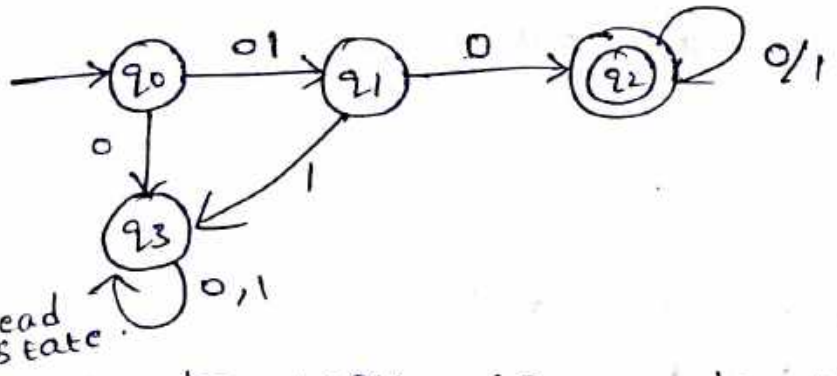
$L = \{101, 0101, 1101, 00101, 11101, 01101, \ldots\}$

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

Ex:- Starts with 10

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

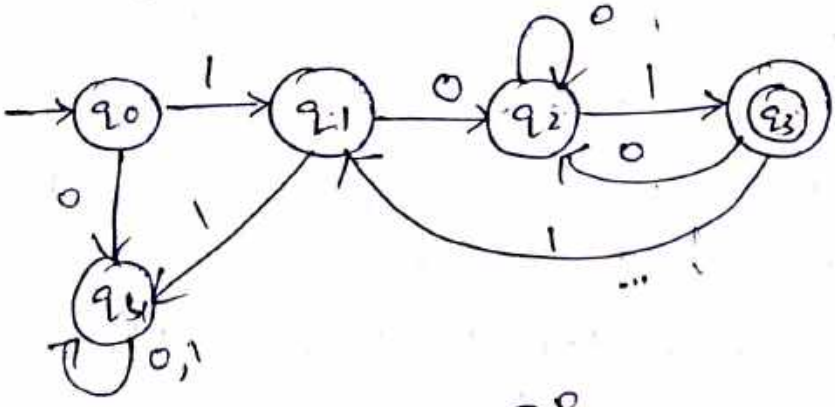
$L' = \{\epsilon, 0, 1, 01, 11, 00, 111, \ldots\}$



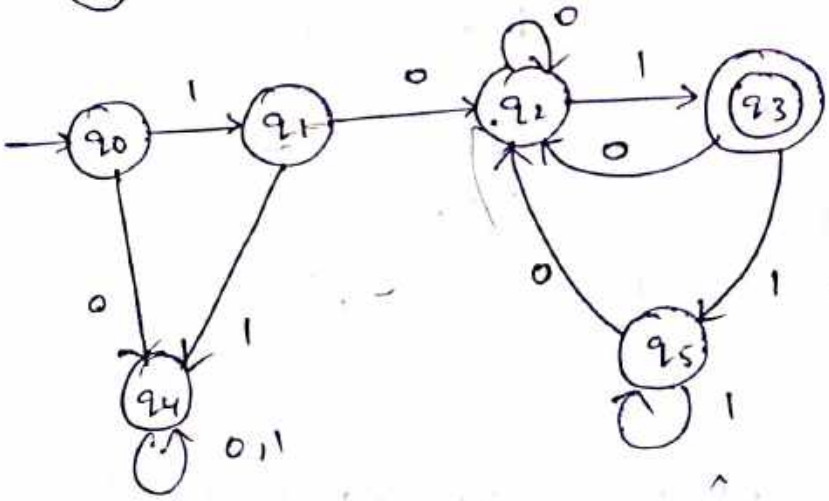
Ex:- Strats with 10 ends with 01

$L = \{101, 1001, 10101, 10001, \ldots\}$

$L' = \{\epsilon, 010, 0110, 01110, \ldots\}$



(wrong)



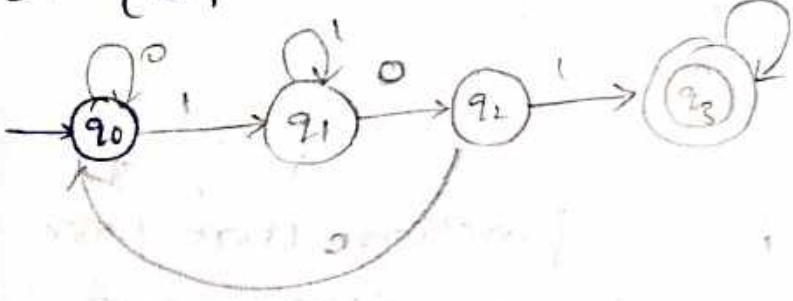
(wrong)

Ex: Substring is 101

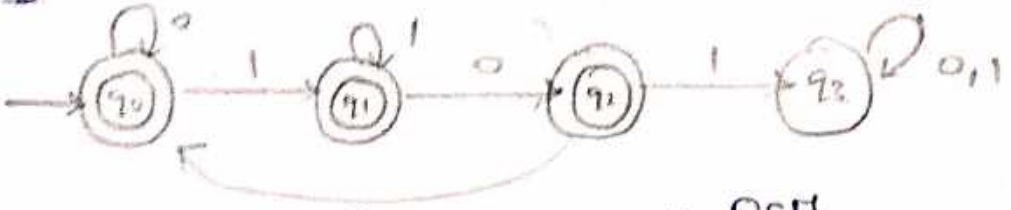
$L = \{101, 0101, 1101, 1010, 1011, \ldots\}$

$L' = \{\epsilon, 010, 001, 011, \ldots\}$

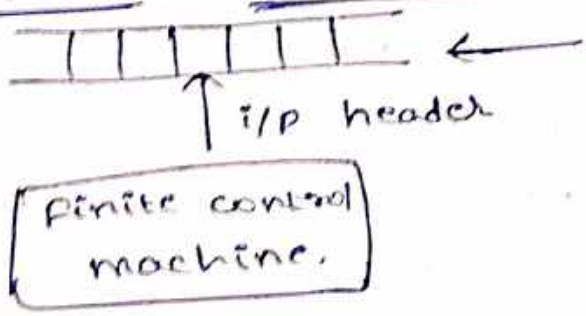
1101110



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

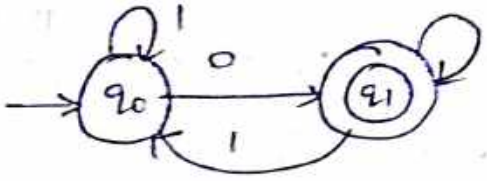
F - final state(s)

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/O / States	0	1
→ q_0	q_1	q_0
(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

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

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

$\vdash \delta(q_0, 0)$

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

check for 101

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

$\vdash \delta(q_1, 1)$

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

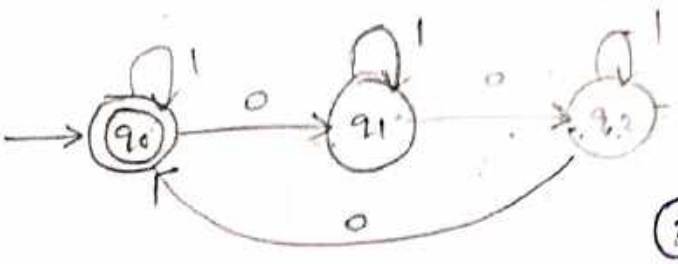
Steps to answer:-

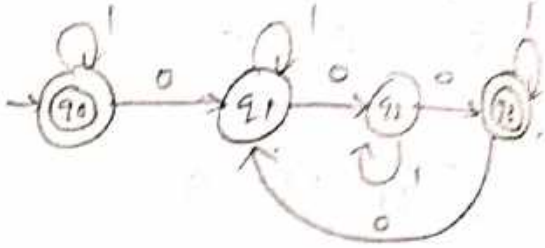
1. Construction of DFA
2. Write the tuples with transition functions
3. Write the transition table.
4. Acceptance string example.
5. Non-acceptance string example.

Ex:- Construct the PA 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}.

L = { ∈, 000, 1000, 10001, 0100, 0010, 0001, 111, 11, 1 }

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





① $M = (Q, \Sigma, \delta, q_0, F)$

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

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

$q_0 = q_0 \qquad F = q_0$

② $\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_1$

$\delta(q_2, 0) = q_0$

$\delta(q_2, 1) = q_2$

③

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

④ check for 10001

$\delta(q_0, \underline{1}0001) \vdash \delta(q_0, \underline{0}001)$

$\vdash \delta(q_1, \underline{0}01)$

$\vdash \delta(q_2, \underline{0}1)$

$\vdash \delta(q_0, 1)$

$\vdash q_0 = F$

(acceptable)

⑤ 110

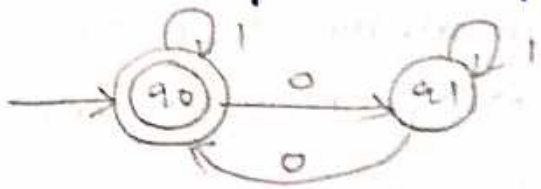
$\delta(q_0, \underline{1}10) \vdash \delta(q_0, 10)$

$\vdash \delta(q_0, 0)$

$\vdash q_1 \neq F$

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

$Q = \{q_0, q_1\}$

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

$q_0 = q_0 \quad F = q_0$

② $\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_0$

$\delta(q_1, 1) = q_1$

③

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

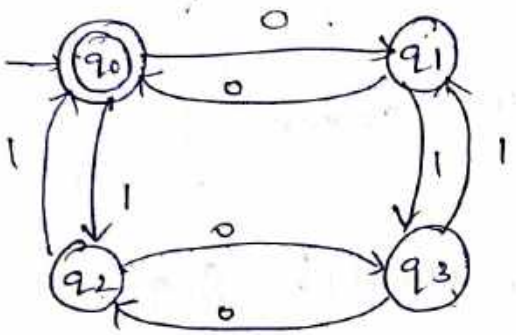
→ conditions for DFA

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.

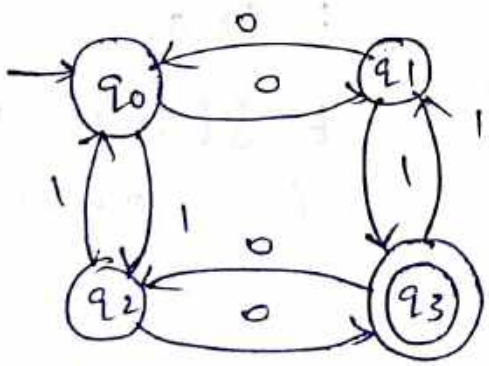
$L = \{\epsilon, 0011, 00, 11, 1100, \ldots\}$

$L' = \{10, 01, 1000, \ldots\}$

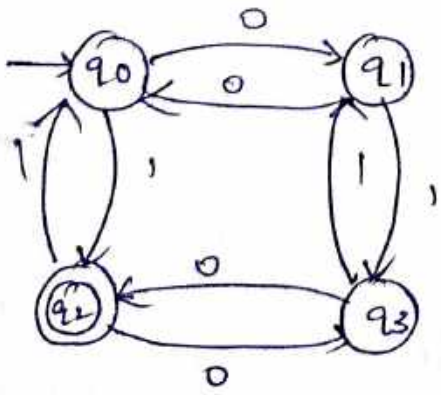


→ odd no. of 0's odd no. of 1's

$L = \{01, 10, 0111, 0001, \ldots\}$



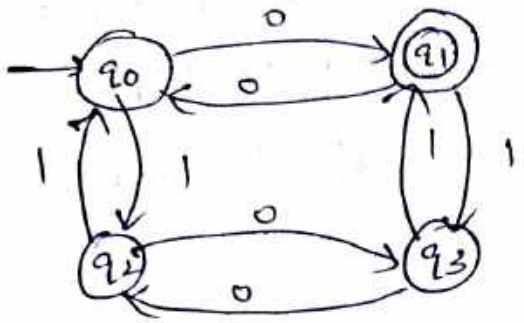
→ even no. of 0's odd no. of 1's.



$L = \{1, 001, 111, 010, \ldots\}$

→ 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

q_0 - 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

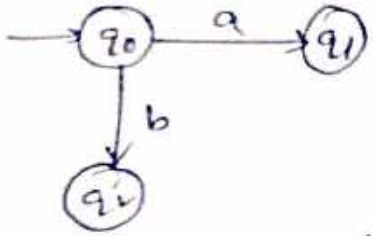
$$Q \times (\Sigma \cup \epsilon) \longrightarrow 2^Q$$

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

→ Conditions for NFA

1. Input Symbols can be used any no. of times
2. each & every alphabet over Σ need not be used. → ϵ can be used as input.

Ex:



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

$\delta(q_0, a) = q_0$

$\delta(q_0, a) = q_1$

$\delta(q_0, a) = q_2$

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

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

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

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

no. of states = 3 (Q)

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

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

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

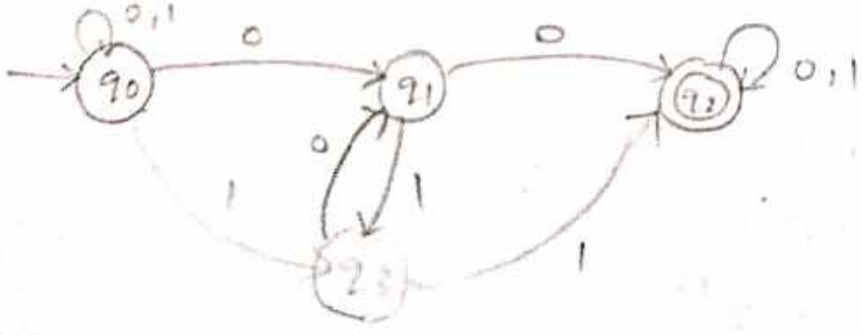
$\delta(q_0, a) = q_0$

$\delta(q_0, a) = q_1$

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

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

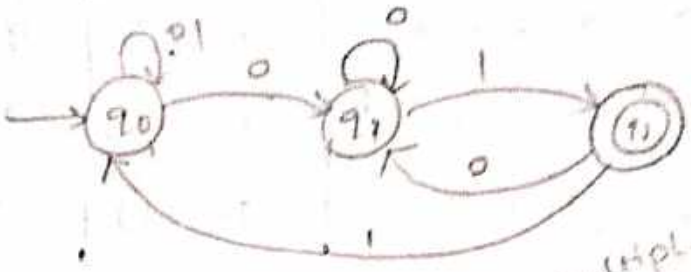
$L = \{00, 11, 100, 111, 000, 011 \cdots\}$ $L' = \{\epsilon, 101, 010 \cdots$



Ex:- ends with 01

NFA $L = \{01, 101, 001, 0001, 0101, 1001, 1101 \ldots\}$

$L' = \{\epsilon, 1, 0, 11, 00, 10, \ldots\}$

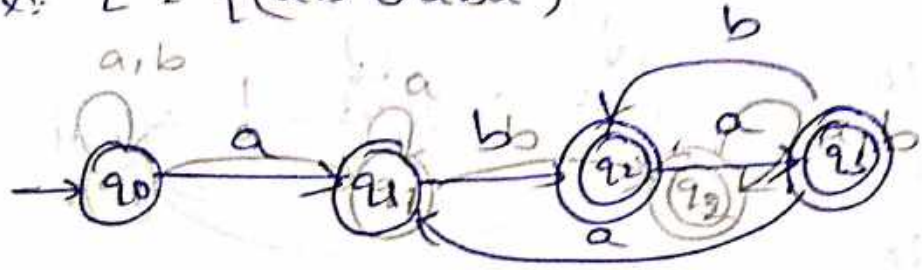


0*10*1

01101

1001

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

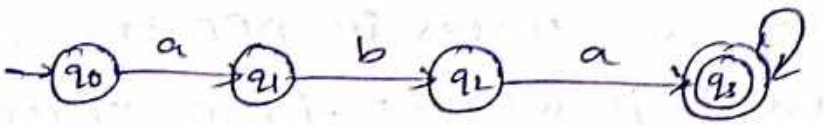


abababaab

acceptable

Ex:- Construct NFA for language

$L = \{ab^n ab^n \mid n \geq 0\}$ $L = \{aba, abab, ababb, \ldots\}$

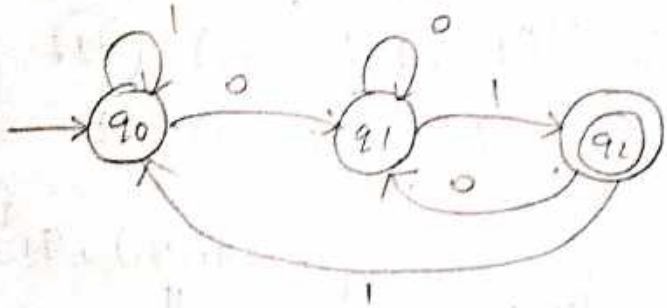


Ex:- design NFA & DFA accepting all strings ending with 01 over $\Sigma = \{0, 1\}$.

$L = \{01, 101, 001, 1101, 0001, 1001, 0101 \ldots\}$

$L' = \{\epsilon, 0, 1, 10, 11, 00, 111, 000, \ldots\}$

NFA



DFA

* Converting from NFA to DFA:-

Transition table-NFA

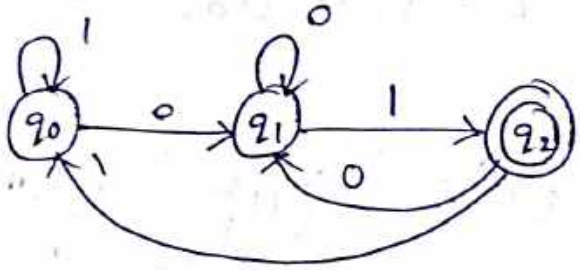
States \ i/p	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	$\emptyset$	q_2
(q_2)	$\emptyset$	$\emptyset$

Transition table for DFA

States \ i/p	0	1
→ 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

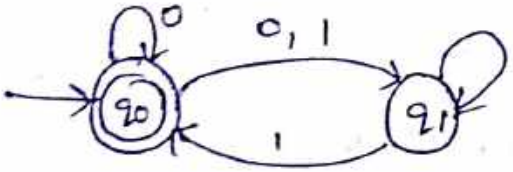
rename

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 (q_2) state in NFA. Then all states in DFA transition table which contain q_2 will be final states.

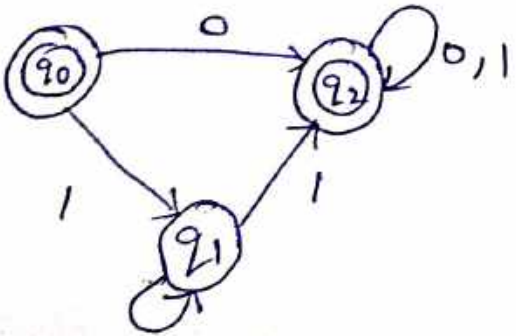
Ex:- Convert following NFA to DFA.



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_1)	(q_0, q_1)	(q_0, q_1)
q_1	q_1	(q_0, q_1)

(q_0) ←
(q_2)



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

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

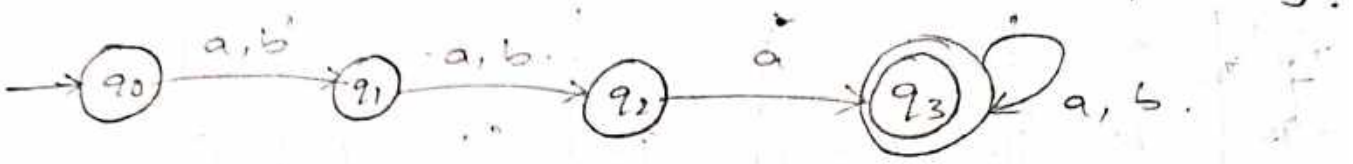
$f = \{q_0, (q_0, q_1)\}$

$= \{q_0, q_2\}$

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

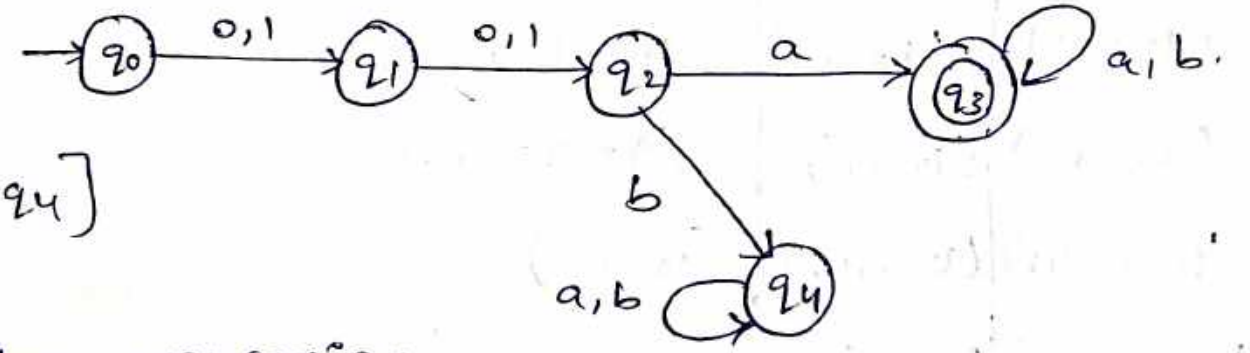
$\Sigma = \{a, b\}$

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

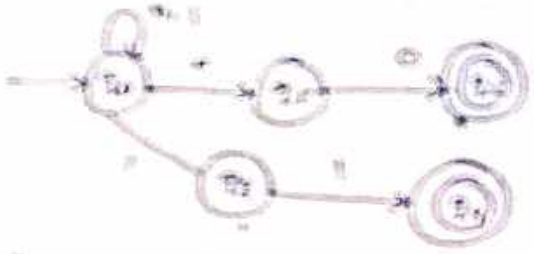


[∅ = q4]

Steps for conversion :-

1. Start with initial state.
2. After finding the transition of initial state 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.

Ex:-



Symbols / I/P	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_2, q_3\}$
q_1	q_2	ϕ
(*q_2)	ϕ	ϕ
q_3	ϕ	q_4
(*q_4)	ϕ	ϕ

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

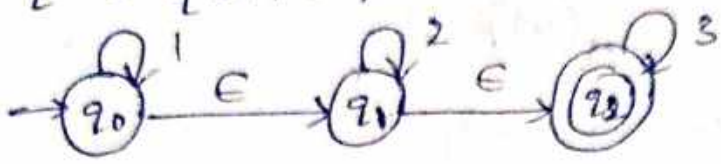
Symbols / I/P	0	1
→ q_0	(q_0, q_1)	(q_0, q_2)
* (q_0, q_1)	(q_0, q_1, q_2)	(q_0, q_2)
* (q_0, q_2)	(q_0, q_1)	(q_0, q_2, q_4)
* (q_0, q_1, q_2)	(q_0, q_1, q_2)	(q_0, q_2)
* (q_0, q_2, q_4)	(q_0, q_1)	(q_0, q_2, q_4)

* NFA with ε-moves:-

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

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

$L^1 = \{21, 31, 32 \ldots\}$

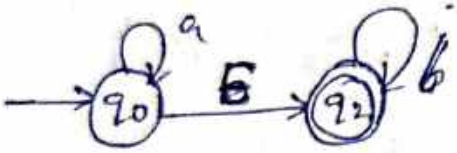


If 211 is i/p, it reaches q_2 by using ε, but it will not be accepted because the i/p string will not become empty.

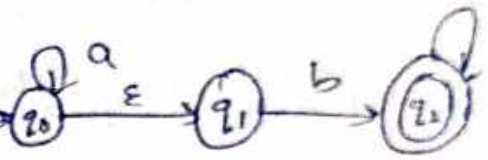
ex:- $L = \{a^* bb^* \ / \ \Sigma = \{a,b\}\}$

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

$L^1 = \{\epsilon, a, aaa \ldots\}$



(or)

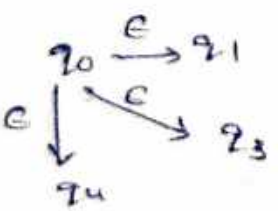


* ε-closure:- It is a set of all states which are reachable from state P on null transition (ε transition)

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

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

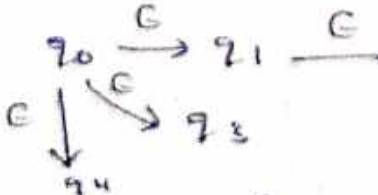
ex:-



ε-closure (q_0) = $\{q_1, q_3, q_4\}$

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

ex:-



ε-closure (q_0) = $\{q_1, q_2, q_3, q_4\}$

3. ε-closure (P) = $\{P, q, r\}$

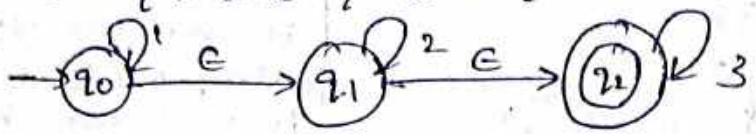
ε-closure (q_0) = $\{q_0, q_1, q_2, q_3, q_4\}$

For last diagram

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

ϵ-closure(q_3) = $\{q_3\}$.

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

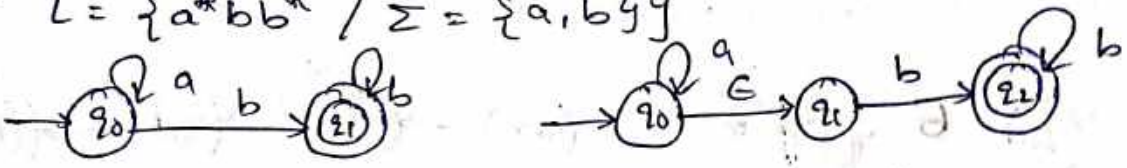


ϵ-closure(q_0) = $\{q_0, q_1, q_2\}$

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

ϵ-closure(q_2) = $\{q_2\}$

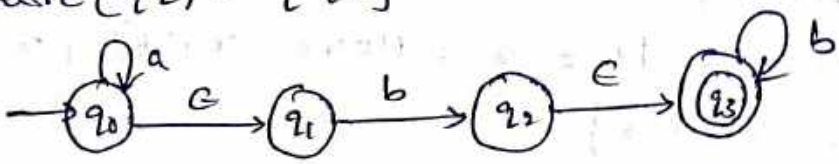
Ex:- $L = \{a^* b b^* / \Sigma = \{a, b\}\}$



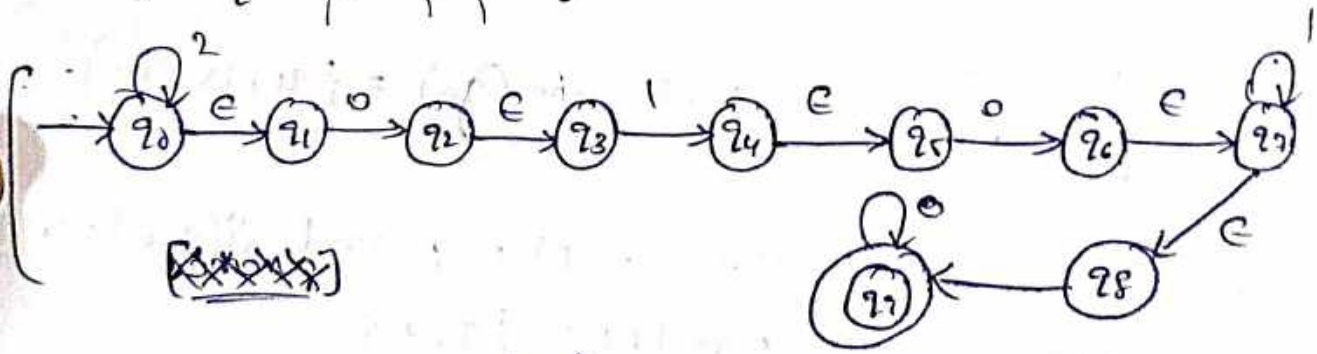
ϵ-closure(q_0) = $\{q_1\}$ = $\{q_1\}$ = $\{q_0, q_1\}$

ϵ-closure(q_1) = $\{q_1\}$

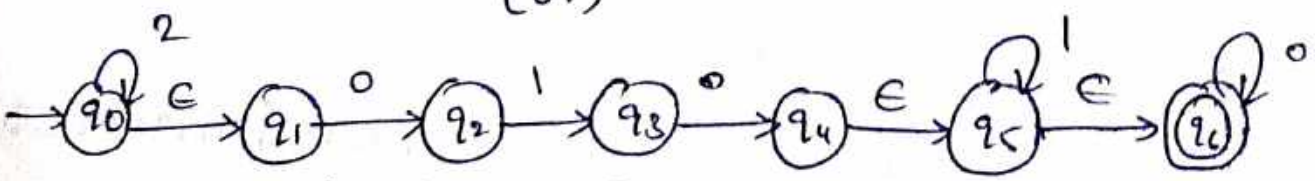
ϵ-closure(q_2) = $\{q_2\}$



Ex:- $L = \{2^* | 010 | 1^* | 0^*\}$



(or)



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

ϵ-closure(q_1) = $\{q_1\}$

ϵ-closure(q_2) = $\{q_2\}$

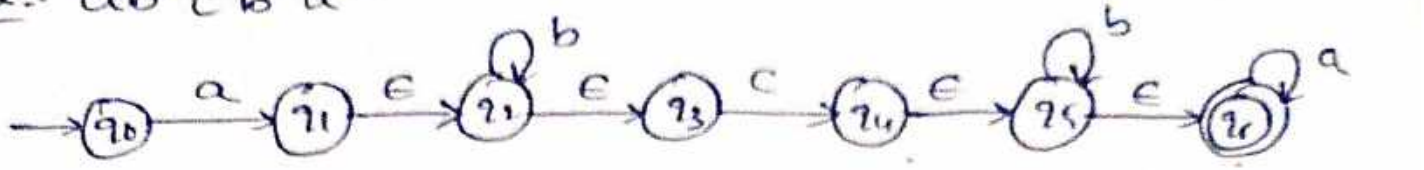
ϵ-closure(q_3) = $\{q_3\}$

ϵ-closure(q_4) = $\{q_4, q_5, q_6\}$

ϵ-Closure(q_5) = $\{q_5, q_6\}$

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

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

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

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

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

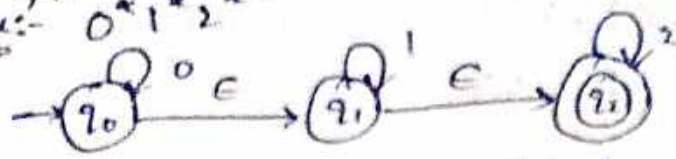
$= \delta'(\phi) = \phi$

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

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

$=$

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



[with ϵ-moves to without ϵ-moves]

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

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

ϵ-closure $(q_2) = \{q_2\}$.

$\delta'(q_0, 0) = \delta'(\delta(\delta'(q_0, \epsilon), 0))$

$= \delta'(\delta(\{q_0, q_1, q_2\}, 0))$

$= \delta'(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$= \delta'(q_0 \cup \phi \cup \phi)$

$= \delta'(q_0)$

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

$\delta'(q_0, 1) = \delta'(\delta(\delta'(q_0, \epsilon), 1))$

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

$= \delta'(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$

$= \delta'(\phi \cup q_1 \cup \phi)$

$= \delta'(q_1)$

$= \{q_1, q_2\}$

$\delta'(q_0, 2) = \delta'(\delta(\delta'(q_0, \epsilon), 2))$

$= \delta'(\delta(\{q_0, q_1, q_2\}, 2))$

$= \delta'(\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2))$

$= \delta'(\phi \cup \phi \cup q_2)$

$= \delta'(q_2)$

$= \{q_2\}$

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

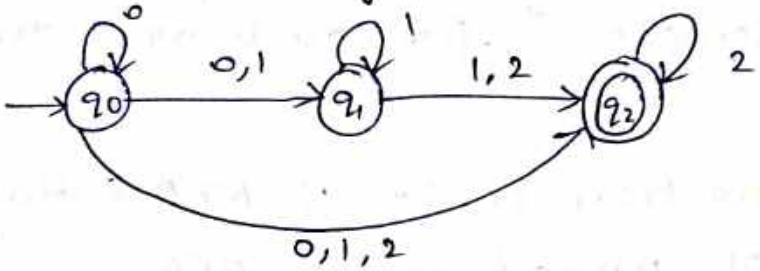
$$\delta'(q_1,1) = \delta'(\delta(\delta'(q_1,\epsilon),1))$$
$$= \delta'(\delta(\{q_1,q_2\},1))$$
$$= \delta'(\delta(q_1,1) \cup \delta(q_2,1))$$
$$= \delta'(q_1 \cup \phi)$$
$$= \delta'(q_1)$$
$$= \{q_1,q_2\}$$

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

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

$$\delta'(q_2,1) = \delta'(\delta(\delta'(q_2,\epsilon),1))$$
$$= \delta'(\delta(q_2,1))$$
$$= \delta'(\phi)$$
$$= \phi$$
$$\delta'(q_2,2) = \delta'(\delta(\delta'(q_2,\epsilon),2))$$
$$= \delta'(\delta(q_2,2))$$
$$= \delta'(q_2)$$
$$= \{q_2\}$$

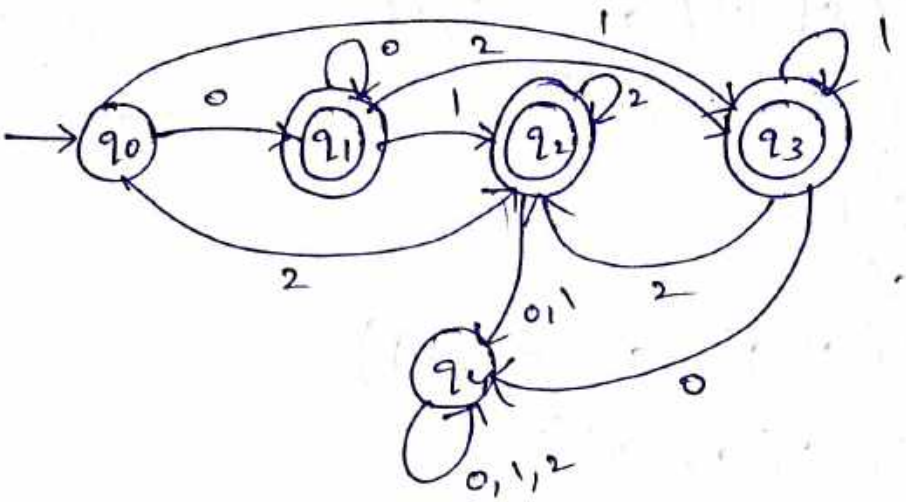
Transition diagram of NFA



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	$\{\phi\}$	$\{q_1,q_2\}$	$\{q_2\}$
(92)	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)	(91)
(q_1,q_2)	ϕ	(q_1,q_2)	q_2	(93)
q_2	ϕ	ϕ	q_2	(92)
ϕ	ϕ	ϕ	ϕ	94



→ 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

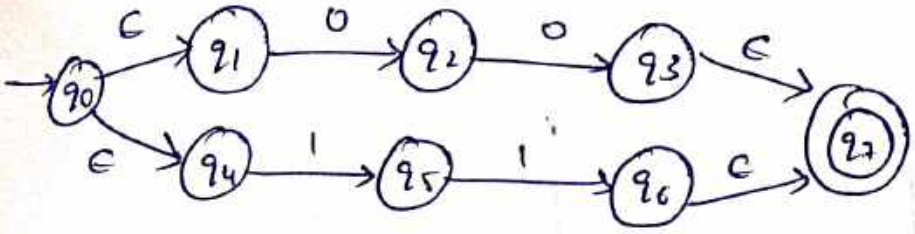
5, start conversion procedure of DFA.

6, Transition table of DFA

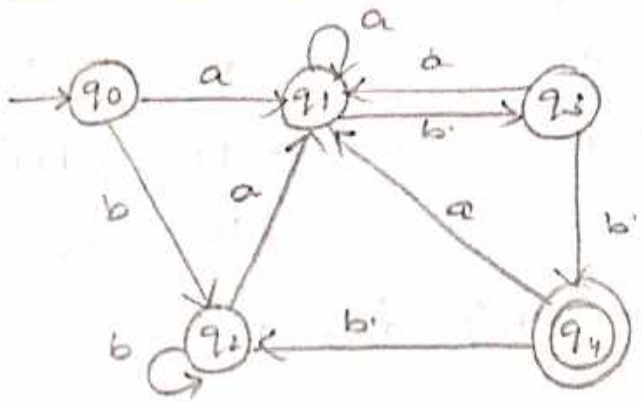
7, Transition diagram of DFA

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.

{q4} {q0, q1, q2, q3}

⇒ 1 - equivalence - π_1

{q4} {q0, q1, q2, q3}

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

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

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

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

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

{q1, q2} {q4}

So $q_0 \neq q_3$

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

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

$q_0 \equiv q_2$

{q4} {q3} {q0, q1, q2}

⇒ 2-equivalence - π_2

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

$\delta(q_0, a) - q_1$ $\delta(q_0, b) - q_3$ $\delta(q_0, a) - q_1$ $\delta(q_0, b) - q_2$

$\delta(q_1, a) - q_1$ $\delta(q_1, b) - q_2$ $\delta(q_3, a) - q_1$ $\delta(q_3, b) - q_4$

$q_0 \neq q_1$ $q_0 \equiv q_3$

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

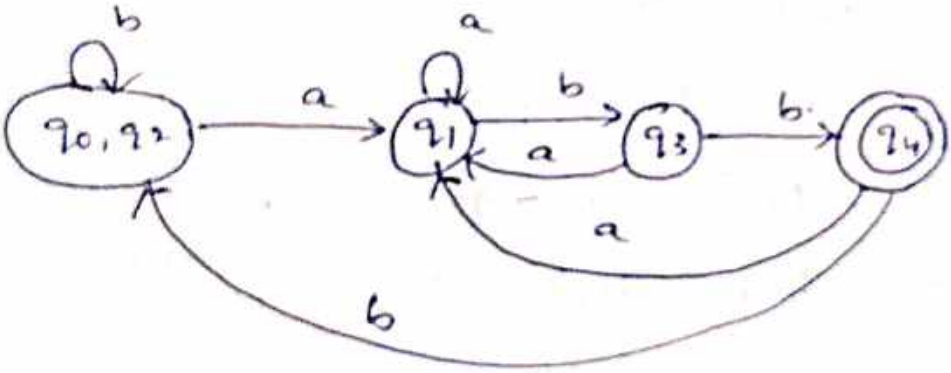
⇒ 3-equivalence - π_3

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

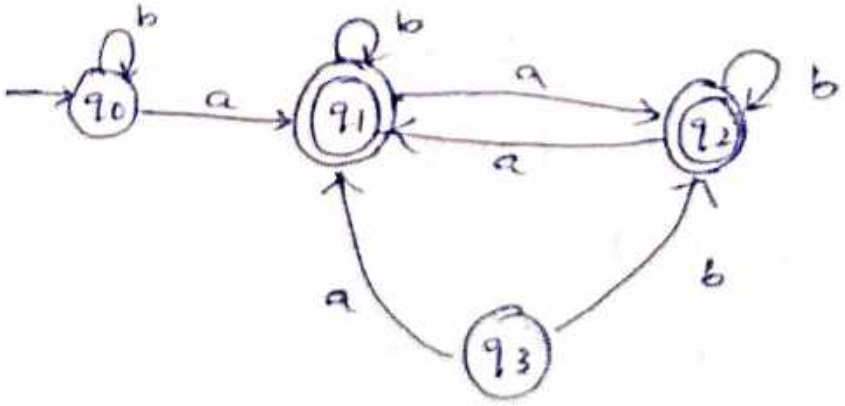
$\delta(q_0, a) - q_1$ $\delta(q_0, b) - q_2$

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

$q_0 \equiv q_3$



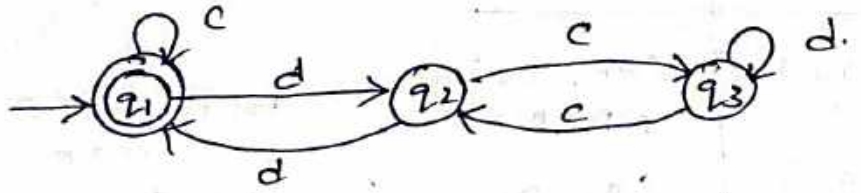
Ans



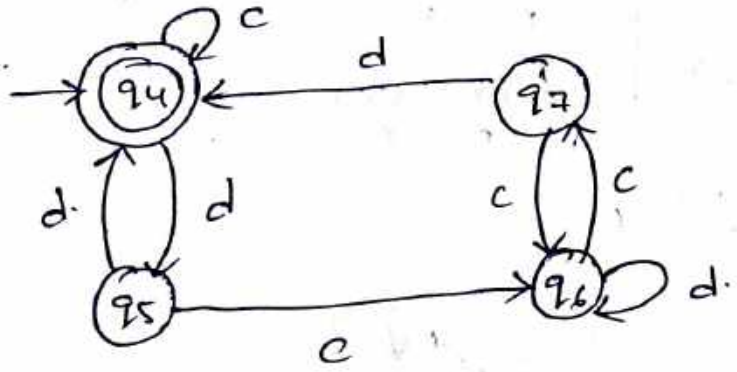
Equivalence of finite automata:

Ex:-

i)



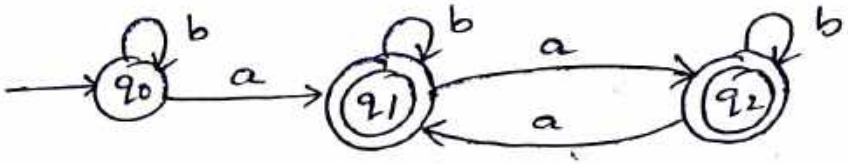
ii)



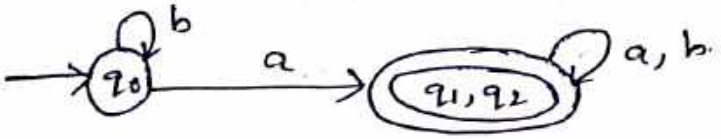
I/P / State	c	d.
{q1, q4}	(q1, q4) F F	(q2, q5) NF NF
{q2, q5}	(q3, q6) NF NF	(q1, q4) F F
(q3, q6)	(q2, q7) NF NF	(q3, q6) NF NF
(q2, q7)	(q3, q6) NF NF	(q1, q4) F F

Both DFA's are equivalent.

→ i)



ii)

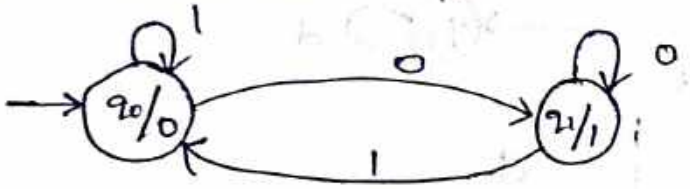


After renaming.

i/p / State	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).

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

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

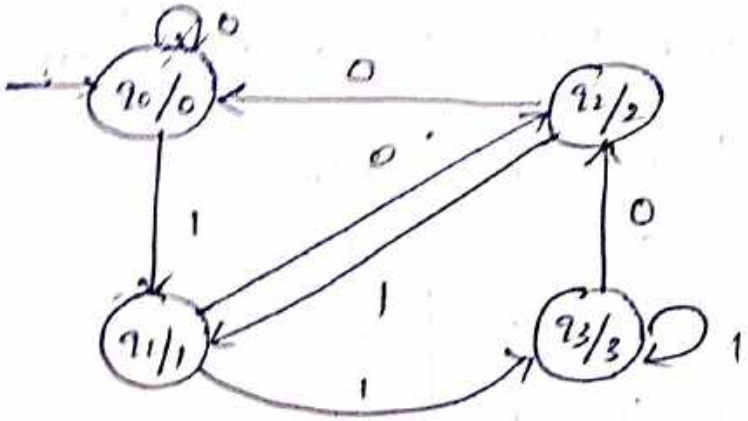
000 - 0
001 - 1
010 - 2
011 - 0
100 - 1
101 - 2

110 - 0.

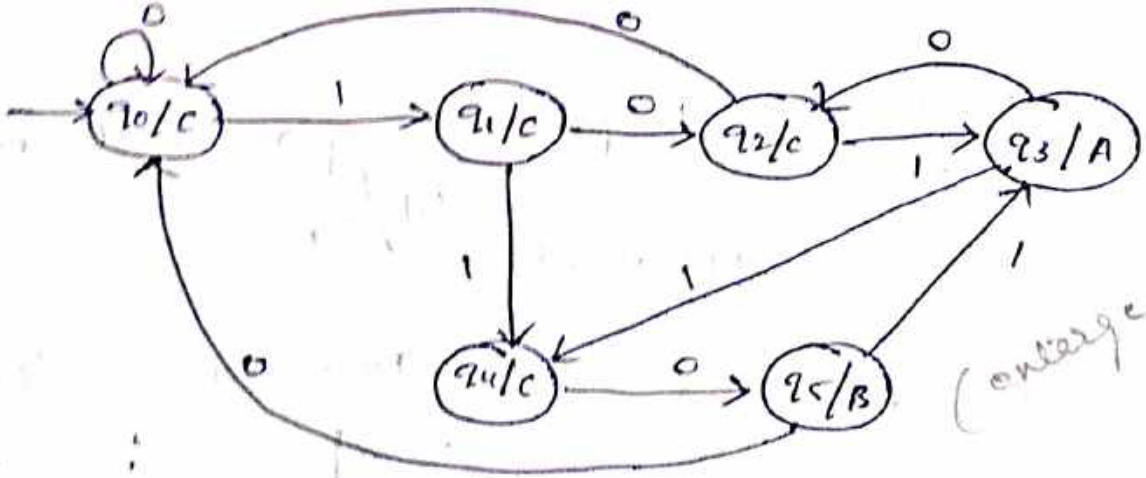
Ex: Modulo 4.

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

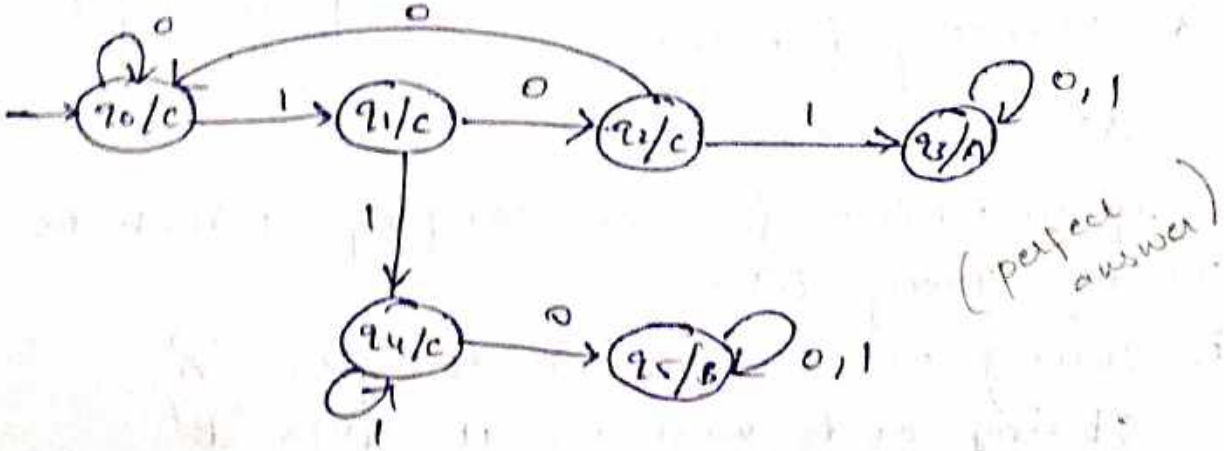
$\Delta = \{0, 1, 2, 3\}$



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'.

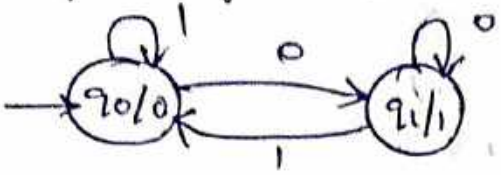


Ex: If the string has substring 101 o/p 'A' if 110 o/p 'B' otherwise C.



* Transition table of moore machines:-

Binary complement



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

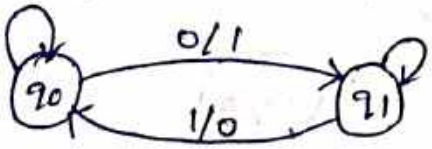
Transition table for above example.

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 transition

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

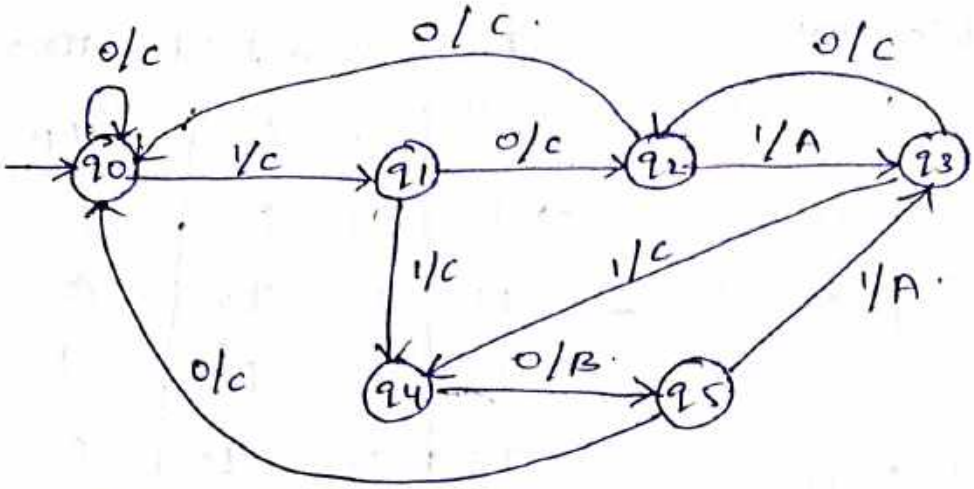
$M = \{Q, \Sigma, \delta, q_0, \Delta, \lambda\}$

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'.

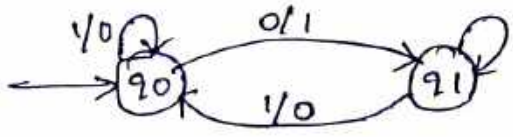


Transition table :-

i/p / state	0	o/p	1	o/p
q_0	q_0	C	q_1	C
q_1	q_2	C	q_4	C
q_2	q_0	C	q_3	A
q_3	q_2	C	q_4	C
q_4	q_5	B	q_4	C
q_5	q_0	C	q_3	A

* Melay to Moore Conversion :

Ex :-



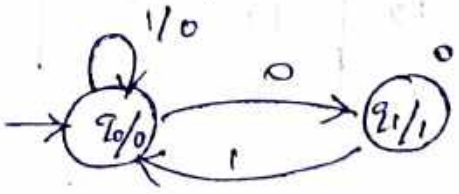
Transition table - Mealy

i/p / State	0	o/p	1	o/p
q_0	q_1	1	q_0	0
q_1	q_1	1	q_0	0

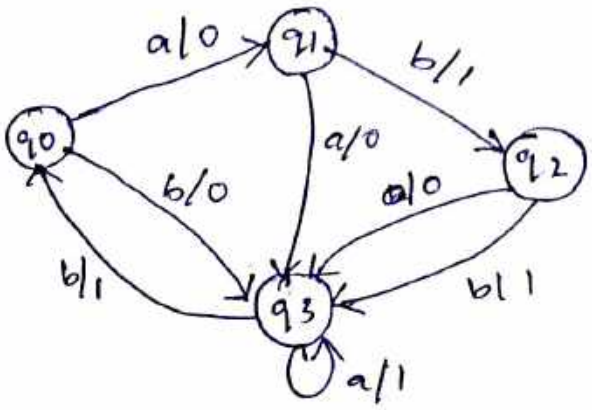
Transition table - moore

i/p / State	0	1	o/p
→ q_0	q_1	q_0	0
q_1	q_1	q_0	1

⇒



Ex :-

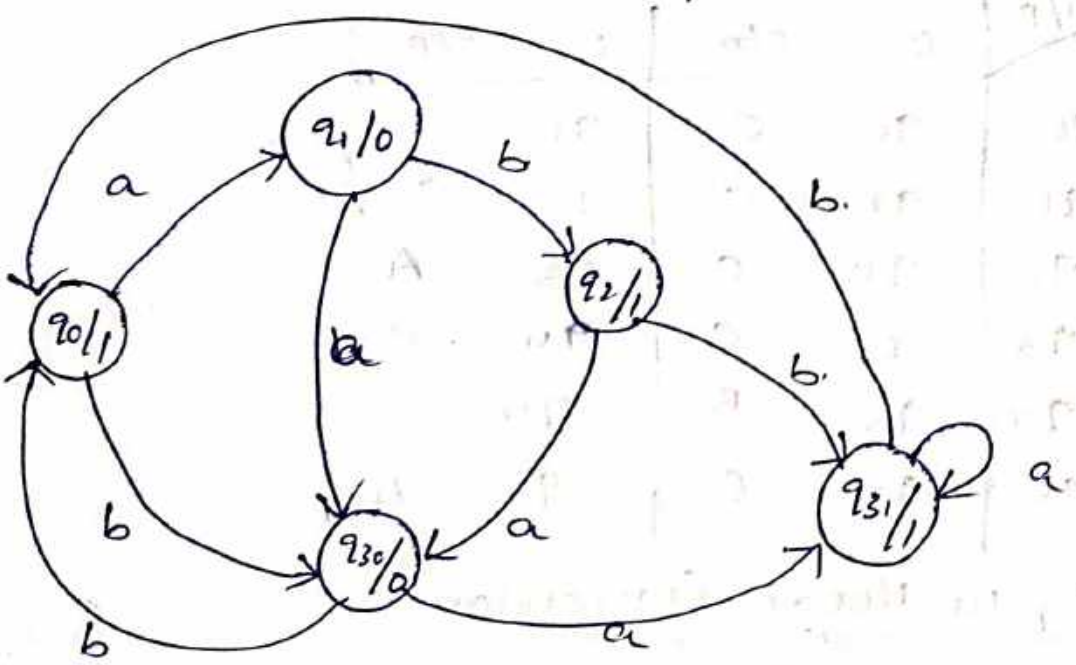


Transition table - Melay

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 - Moore

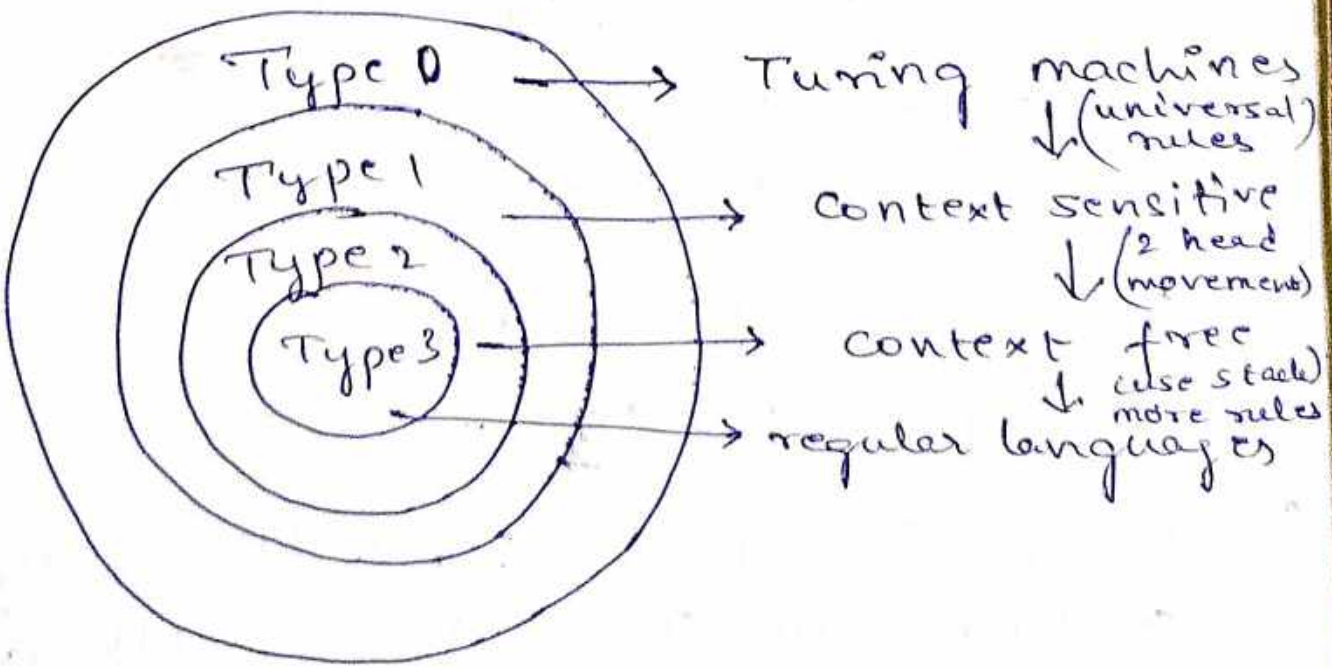
State \ i/p	a	b	o/p
→ q_0	q_1	q_{30}	1
q_1	q_{30}	q_2	0
q_2	q_{30}	q_{31}	1
q_{30}	q_{31}	q_0	0
q_{31}	q_{31}	q_0	1



Ex 1-

	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



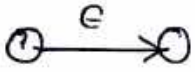
Unit-2

* **Regular Expression:-** The language accepted by finite automata is easily described by simple expression called regular expression.

* **Regular Language:-** Language accepted by regular expression.

The regular expression over Σ can be defined

1. $\emptyset$ is a regular expression which denotes empty set $\{\}$.
2. ϵ is a RE denotes the set $\{\epsilon\}$. It is called as null string.

   eg:- 

3. for each "a" in Σ just A is a regular expression if the language is $\{a\}$.
4. If "r" and "s" are RE's denoting languages L1 and L2 respectively then the regular expression is r+s ; L1 ∪ L2

   eg:- a+b · $\{a\} \cup \{b\}$

5. Concatenation: rs ; L1 L2

   eg:- ab $\{a\}\{b\}$

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

$a^+ = \{a, aa, aaa, ----\}$

Q:- Design a regular expression for the language contains all the strings with a and b combinations.

Ans:- Universal language for a and b is

RE = $(a+b)^*$.

Q:- Construct RE for lang accepting all string should end with 00 / $\Sigma = \{0,1\}$

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

Q:- Construct RE start with 1 ends with 0

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

Q:- Atleast 1 a followed by b

$RE = a^+ * b \cdot \quad a^+ b$

Q:- Substring 110

$RE = (0+1)^* 110 (0+1)^*$.

Q:- Having atleast 2 0's.

~~$RE = (0+1)^* (00)^+ (0+1)^*$~~ (wrong)

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

Q:- Exactly 2 0's.

$RE = 1^* 0 1^* 0 1^*$

Q:- atmost 2 0's.

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

* Identity Rules :-

I_1 : $\phi + r = r$

I_2 : $\phi r = r\phi$

I_3 : $\epsilon r = r\epsilon = r$

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

I_5 : $r + r = r$

I_6 : $r^* r^* = r^*$

I_7 : $rr^* = r^* r = r^+$

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

I_9 : $\epsilon + rr^* = r^*$

I_{10} : $(pq)^* p = p(qp)^*$

I_{11} : $(p+q)^* = (p^* q^*)^* = (p^* + q^*)^*$

I_{12}: $(p+q)r = pr + qr$ $\not\equiv$ $r(p+q) = rp + rq$

Q:- Prove that $(1 + 00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= 0^*1(0+10^*1)^*$

L.H.S = $(1 + 00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= P + PQ^*Q$

$= P(\epsilon + Q^*Q)$

$= P(\epsilon + Q^+)$

$= PQ^*$

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

$= (1+0^+1)(0+10^*1)^*$

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

$= 0^*1(0+10^*1)^*$

= R.H.S.

Q:- $0^*(10^*)^*$

a) $(1^*0^*)1^*$ b) $0+(0+10)^*$

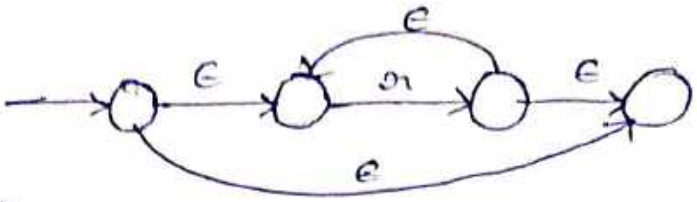
c) $(0+1)^*10(0+1)^*$ d) $(0+10)^*$

$(0^*1^*)10(0^*1^*)$

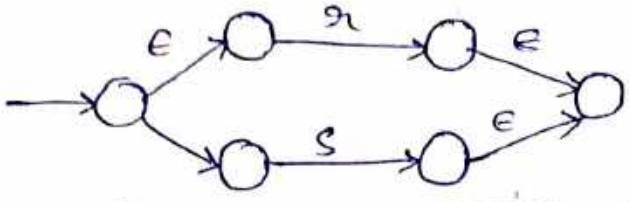
$0^*1^+0^+1^*$

* Conversion from expression to finite automata

→ r^*.

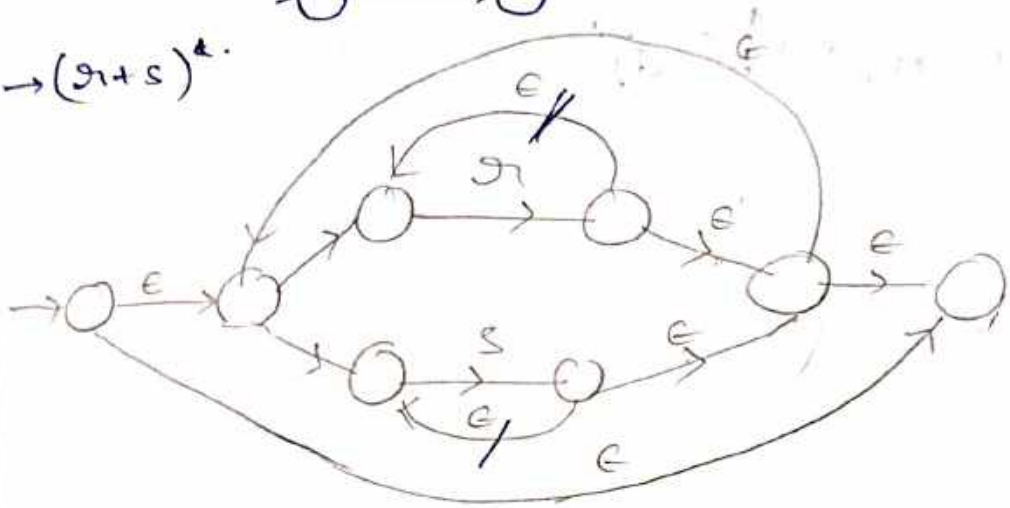


→ $r + s$



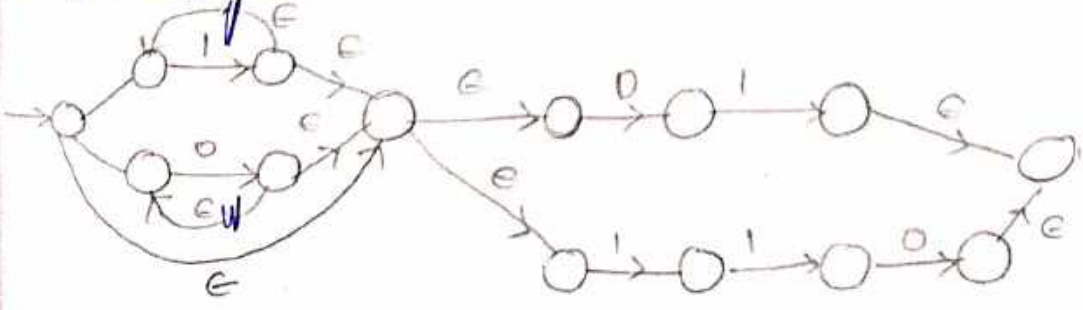
→ rs

→ $(r+s)^*$.

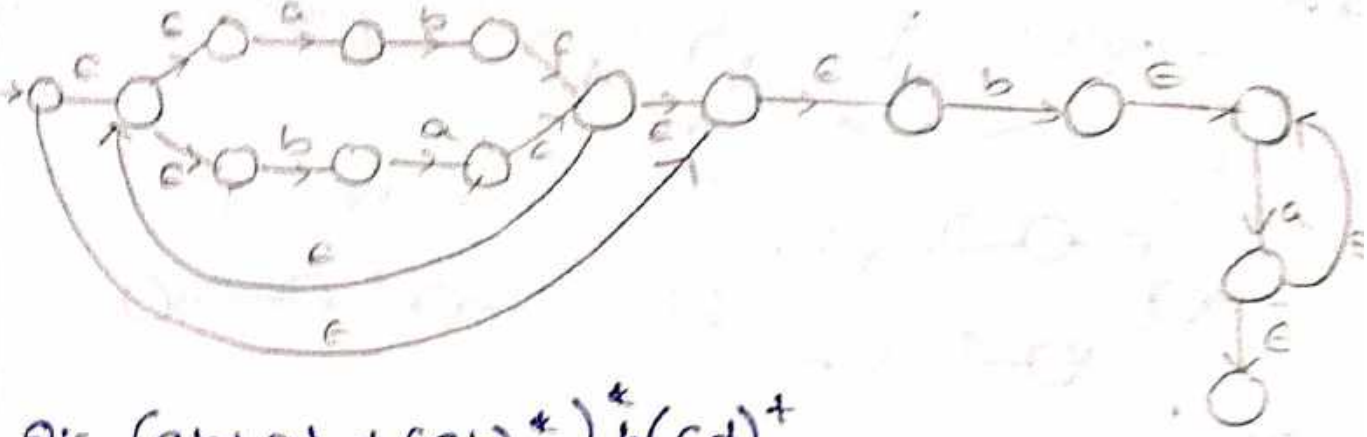


→ Construct finite automata for $b^*(aa)^*b^*$.

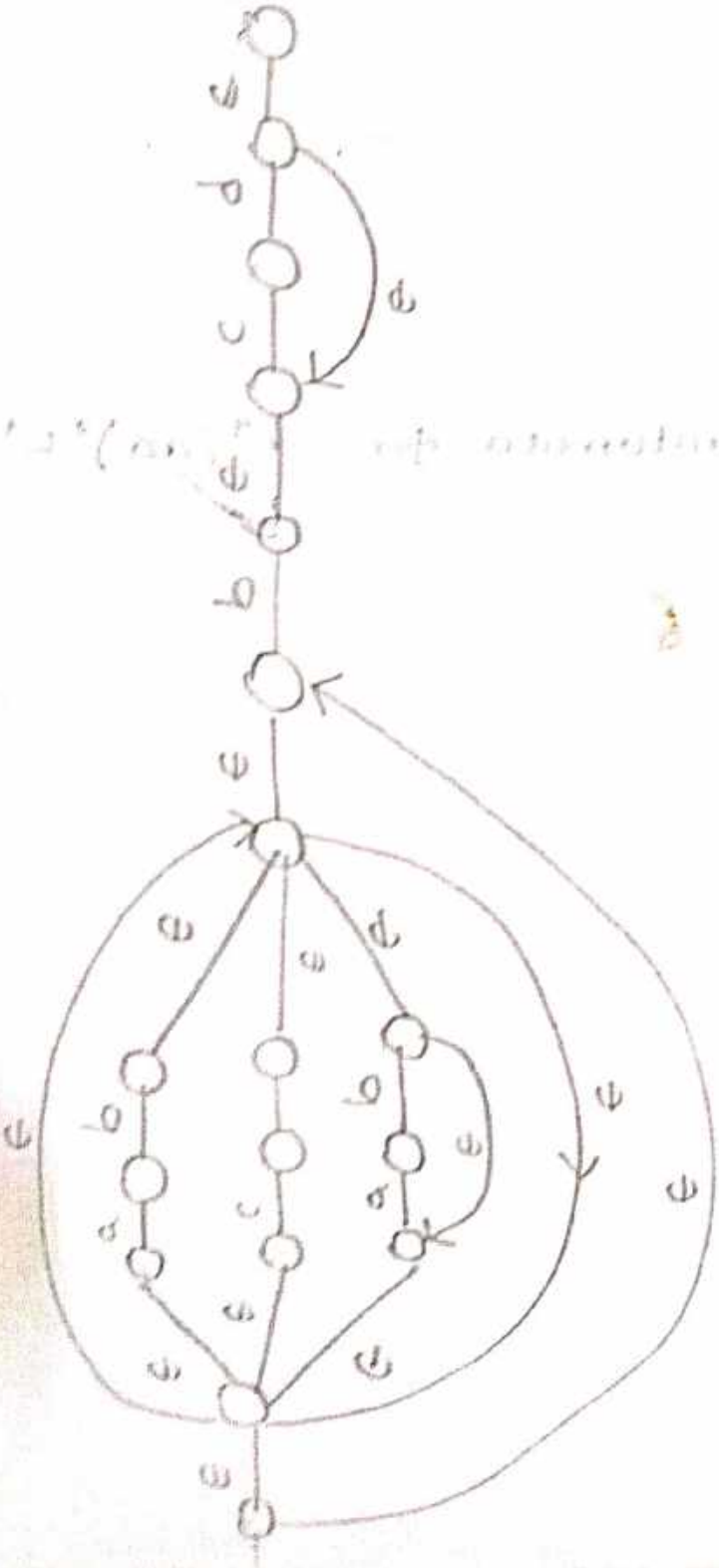
→ $(0+1)^*(01+110)$



Ex:- $(ab+ba)^* bb^+$



Q:- $(ab+cd+(ab)^*)^* b(cd)^+$



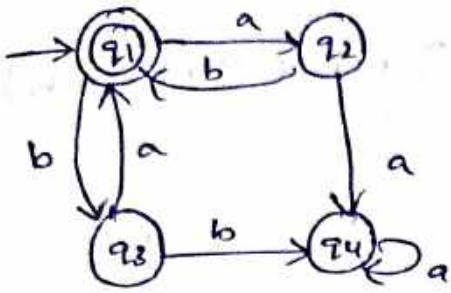
* Finite automata to Regular expression by using Ardens theorm :-

$R = Q + RP$
$= QP^*$

→ only 1 initial state

→ no null transitions

Ex:-



$q_1 = \epsilon + q_2 b + q_3 a$

$q_2 = q_1 a$

$q_3 = q_1 b$

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b.$

For final state

$q_1 = \epsilon + q_2 b + q_3 a$

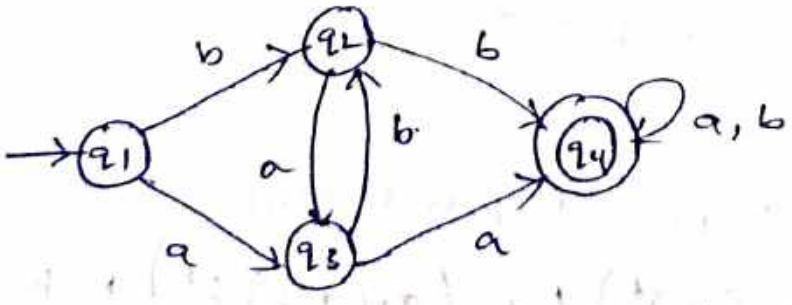
$= \epsilon + q_1 ab + q_1 ba$

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

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

$= (ab + ba)^*$

Ex:-



$q_1 = \epsilon.$

$q_2 = q_1 b + q_3 b \quad = (q_1 + q_3) b$

$q_3 = q_1 a + q_2 a$

$q_4 = q_2 b + q_3 a + q_4 a + q_4 b.$

For final state

$q_4 = q_2b + q_3a + q_4a + q_4b$

$= (q_1b + q_3b)b + (q_1a + q_2a)a + q_4(a+b)$

$= (q_1 + q_3)bb + (q_1 + q_2)aa + q_4(a+b)$

$\Rightarrow q_2 = q_1b + q_3b$

$= \epsilon b + (q_1a + q_2a)b$

$= q_1b + q_1ab + q_2ab$

$= q_1b(\epsilon + a) + q_2ab$

$= \underbrace{q_1ab}_{Q} + \underbrace{q_2ab}_{r\ P}$

$q_2 = q_1ab(ab)^*$

$\Rightarrow q_3 = q_1a + q_2a$

$= q_1a + q_1b + q_3b$

$= \underbrace{q_1(a+b)}_{Q} + \underbrace{q_3b}_{r\ P}$

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

$\Rightarrow q_4 = q_2b + q_3a + q_4a + q_4b$

$= ((q_1ab)(ab)^*)b + (q_1(a+b)b^*)a +$

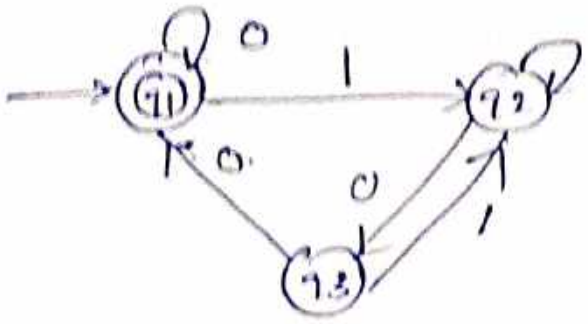
$q_4a + q_4b$

$= q_1(ab(ab)^*b + (a+b)b^*a)$

$+ q_4(a+b)$

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

Ans



$q_1 = q_1 0 + q_3 0$

$q_2 = q_1 1 | q_2 1 + q_3 1 = (q_1 + q_3)1 + q_2 1 = (q_1 + q_3)1^+$

$= (q_1 + q_3)1^+$

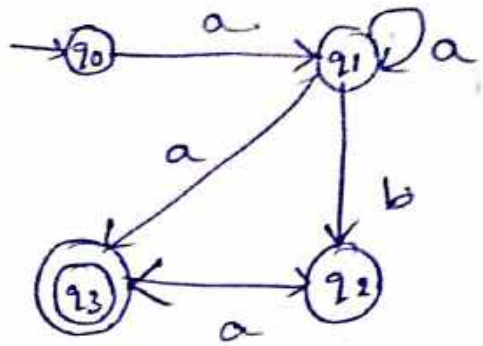
$q_3 = q_2 0$

$q_1 = q_1 0 + q_3 0$

$= q_1 0 + q_2 00$

$= q_1 0 +$

Ex:-



$q_0 = \epsilon$

$q_1 = q_0a + q_1a = q_0aa^* = q_0a^+ = \epsilon a^+ = a^+$

$q_2 = q_1b = a^+b$

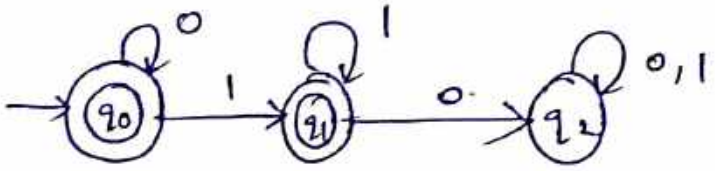
$q_3 = q_2a + q_1a = a^+ba + a^+a = a^+a(b+\epsilon)$

$= a^+ab.$

$q_3 = q_2a + q_1a$

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

Ex:-



$q_0 = q_00 + \epsilon = \epsilon 0^* = 0^*$

$q_1 = q_01 + q_11 = (q_01)^* = q_01^+ = 0^*1 + q_11$

$q_2 = q_10 + q_20 + q_21 = 0^*11^* = 0^*1^+$

$= q_10 + q_2(0+1)$

$= q_10(0+1)^*$

$= q_1001^+(0+1)^*$

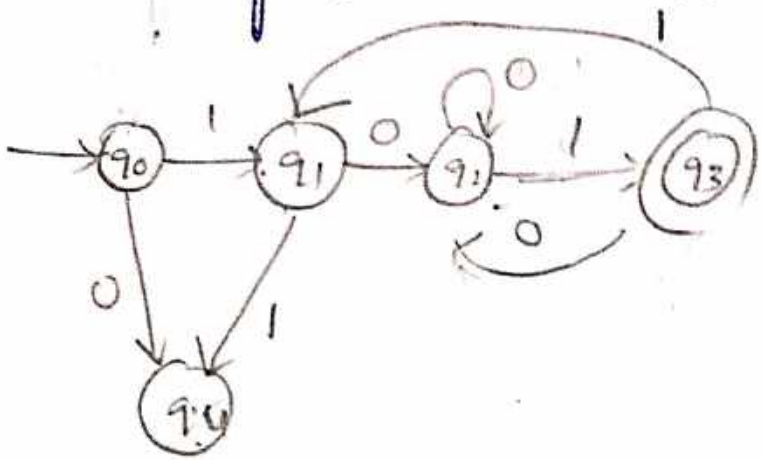
$= 0^*1^+0 + q_2(0+1)$

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

final state $= 0^* + 0^*1^+ = 0^*(\epsilon + 1^+)$

$= 0^*1^*$

Q:- string starts with 10 ends with 01



$q_0 = \epsilon$

$q_1 = q_01 + \epsilon 1 = 1 = q_01 + q_31 = 1 + q_31 = 1(1)^*$

$q_2 = q_10 + q_20 + q_30 = 1 + 0$

$q_4 = q_00 + q_11 = \epsilon 0 + 11 = 0 + 11$

~~$[q_5 = q_31 + q_5 1]$~~

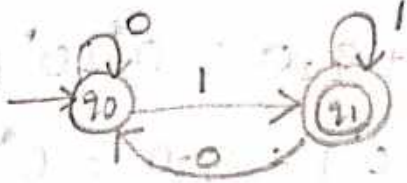
Q/S

1, Construct the expression for string ends with '1'.

2, never starts with 1 but ends with 1

3, atmost 3 1's in particular string.

①



$q_0 = q_00 + q_10 + \epsilon$

$= q_10 + q_00 + \epsilon$

$= Qp^*$

$= q_10\,0^*$

$= q_10^+$

$q_1 = q_01 + q_11$

$q_1 = q_100^*1 + q_11$

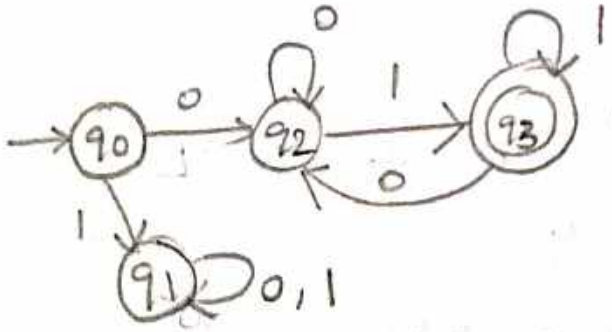
$= q_1(00^+1 + 1)$

$= q_1(0^+1 + 1)$

$= q_1(0^+ + \epsilon)1$

$= q_10^*1$

②



$q_0 = \epsilon$

$q_1 = q_01 + q_10 + q_11$

$q_2 = q_00 + q_20 + q_30$

$q_3 = q_21 + q_31$

$q_2 = \epsilon 0 + q_20 + q_30$

$= (0 + q_30) + q_20$

$= (0 + q_30)0^*$

$q_3 = q_21 + q_31$

$= q_211^* = q_21^+$

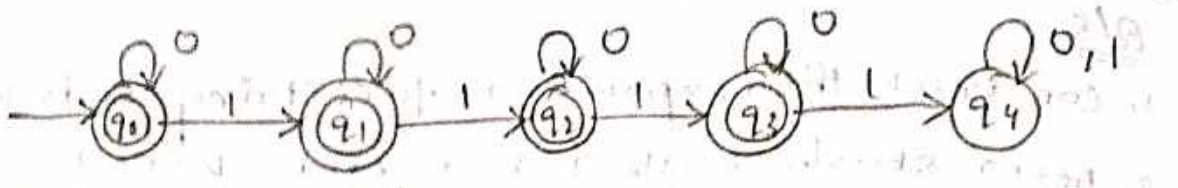
$= [(0 + q_30)0^*]1^+$

$q_3 = 0^+1^+ + q_30^+1^+$

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

$= (0^+1^+)^+$

(3)



$q_0 = q_00 + \epsilon = 0^*$

$q_1 = q_01 + q_10 = 0^*1 + q_10 = 0^*10^*$

$q_2 = q_11 + q_20 = 0^*10^*1 + q_20 = 0^*10^*10^*$

$q_3 = q_21 + q_30 = 0^*10^*10^*1 + q_30 = 0^*10^*10^*10^*$

$q_4 = q_31 + q_40 + q_41$

final states = $q_0 + q_1 + q_2 + q_3$

$= 0^* + (0^*10^*) + (0^*10^*10^*) + (0^*10^*10^*10^*)$

* Pumping Lemma

Let 'L' be a regular language there exits a constant 'n' for every string 'w' in 'L' such that $|w| \geq n$, now we can break 'w' into

$w = xyz$ such that

i, $y \neq \epsilon$

ii, $|xy| \leq n$

iii, $\forall\, k \geq 0$, xy^kz is also in L

ex:- $L = \{a^nb^n / n \geq 0\}$

$L = \{ab, aabb, aaabbb, ---\}$

$w = aabb$, (4) → const

$\underline{a}\,\underline{ab}\,\underline{b}$ (x y z)

$y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aababb \notin L$ [not satisfied]

$\underline{aa}\,\underline{bb}$ (x = ϵ, y z)

$x = \epsilon$ $y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aaaabb \notin L$

Ex:- $L = \{ a^n b a^n / n \geq 1 \}$

$L = \{ b, aba, aabaa, ---- \}$

$w = \underset{x}{aa}\underset{y}{b}\underset{z}{aa}$, 6

not regular expression

$\underset{x}{aa}\underset{y}{b}\underset{z}{aa}$ $\quad y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aabbaa \notin L$

$\underset{x}{a}\underset{y}{ab}\underset{z}{aa}$ $\quad y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aababaa \notin L$

Ex:- $L = \{ a^n b^m / m > n \}$

$L = \{ b, abb, aabbb, ---- \}$

$w = \underset{y}{aa}\underset{z}{bbb}$, 6. $\quad y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aaaabbb \notin L$

$\underset{x}{aa}\underset{y}{bb}\underset{z}{b}$. $\quad y \neq \epsilon$

$|xy| \leq n$

$xy^1z = aabbbbbb \in L$

[but the power of a is becoming const]

∴ not a regular language.

Ex:- $L = \{ a^n b^m / n, m \geq 0 \}$

$L = \{ a, b, ab, aab, abb, aaab \}$

$w = aaab$, 5

$\underset{x}{aaa}\underset{y}{b}$ $\quad z = \epsilon, y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aaabab \notin L$

$\underset{x}{aa}\underset{y}{a}\underset{z}{b}$. $\quad y \neq \epsilon$ ✓

[partially regular] $\quad |xy| \leq n$ ✓

$aaaaab \in L$

is a regular language

Ex:- 1, $L = \{a^n / n > 0\}$

2, $L = \{a^{n^2} / n > 0\}$

① $L = \{a, aa, aaa, ----\}$

$w = \underset{x}{a}\underset{y}{aaa}\underset{z}{a}$ $\quad$ $w = \underset{y}{aaa}\underset{z}{a}$

$xy^1z = aaaaaa$ $\quad$ $xy^1z = aaaaaaaa$

Regular language.

② $L = \{a, aaaa, aaaaaaaaaa, ----\}$

$w = \underset{x}{aaa}\underset{y}{a}$ $\quad$ $w = \underset{x}{a}\underset{y}{aa}\underset{z}{a}$

$xy^1z = aaaaaa$ $\quad$ $xy^1z = aaaaaa$

$\times$ $\quad$ $\times$

Not regular language.

Ex:- $L = \{a^n b^m / n > m$ where $n, m \in [1, 10]\}$

is a regular language.

* Closure properties of Regular sets :- [explain with examples]

1, The union of two regular languages is regular

2, Intersection of two regular languages is regular

3, Complement of a regular language is regular

$[\Sigma^* - L = L^1]$

4, Difference of two regular languages is regular.

$[L - M \neq M - L]$ $\quad$ $[L - M = L - (L \cap M)]$

5, Reversal or Transpose of a regular language is also regular.

[If abc regular cba also regular]

6, The closure of regular language is regular.

[closure means *]

[if r is regular, r^* will also be regular]

7, Concatenation of regular languages is regular.

Ex:- L & M regular ⇒ LM is regular.

$[LM \neq ML]$

8, Homomorphism of a regular language is regular.

9, Inverse Homomorphism of regular language is also regular.

⇒ if we get repeated patterns we can give the repeated patterns a new name — This is called as Homomorphism.

⇒ This reduces the length.

Ex:- $\underbrace{aaa}_{A}\underbrace{bbb}_{B}\underbrace{ccc}_{C}c = \underbrace{ABCc}$ — Homomorphism

inverse homomorphism.

Unit-3

Regular Grammars

* Regular Grammar tuples are

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

(NT)

V = set of non-terminal symbols or variables

T = set of terminal symbols (Σ)

P = Production rules

S = Initial non-terminal symbol.

Ex:- $\underset{A}{a}\underset{B}{a}\underset{C}{bb}\underset{A}{cc}aa$

$V = \{S, A, B, C\}$

$T = \{a, b, c\}$

$P :$ $S \rightarrow ABCA$

$A \rightarrow aa$

$B \rightarrow bb$

$C \rightarrow cc$

$S = S$

Ex:- The string ends with 'a'

$T = \{a\}$

$L = \{a, aa, aaa, \ldots\}$

$S \rightarrow a/aA$

$A \rightarrow aA/\epsilon$

Ex:- starts with a: $T = \{a, b\}$

$L = \{ a, aa, ab, aaa, abb, \ldots \}$

$S \rightarrow a/aA$

$A \rightarrow [a/b/AA]\ aA/bA/\epsilon \Rightarrow (a+b)^*$

universal expression

Ex:- starts with a ends with b

$L = \{ ab, aab, abb, \ldots \}$

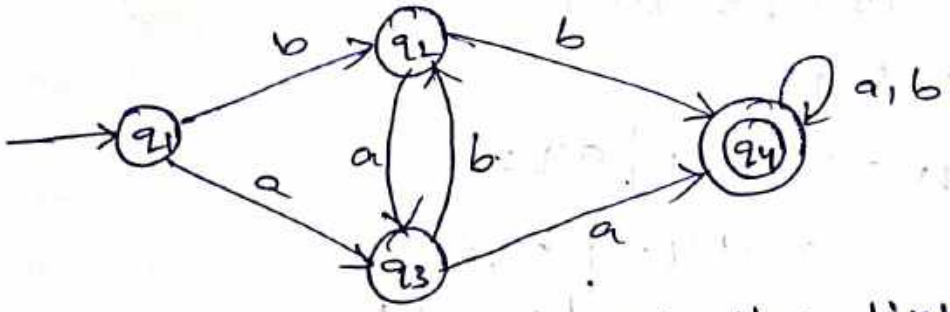
$S \rightarrow [ab]/aAb$

$A \rightarrow aA/bA/\epsilon$

→ P: $A \rightarrow Aa/a$ — left linear grammar.

$A \rightarrow aA/a$ — Right linear grammar.

Ex:- L = Substring aa/bb.



consider out going links. If the link reaches final state write only terminal. We can get Right linear grammar from automata

$V = \{q_1, q_2, q_3, q_4\}$

$T = \{a, b\}$

$S = \{q_1\}$

P: $q_1 \rightarrow bq_2/aq_3$

$q_2 \rightarrow aq_3/bq_4/b$

$q_3 \rightarrow bq_2/aq_4/a$

$q_4 \rightarrow aq_4/bq_4/a/b$

Let us consider the string "aaba".

Start with S

$q_1 \rightarrow aq_3$

$\rightarrow aaq_4$

$\rightarrow aabq_4$

$\therefore$ string is derivable.

for abab

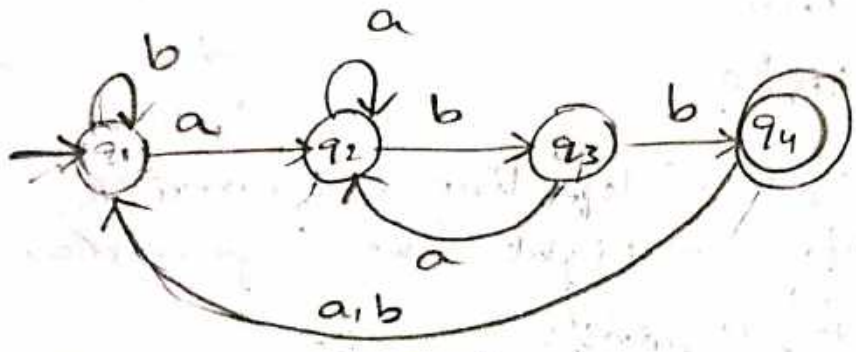
$q_1 \to aq_3$

$\to abq_2$

$\to abaq_3$

$\to ababq_2$ $\Longrightarrow$ still there are non-terminals so string is not derivable.

ex:- $(a+b)^*abb$.



$V = \{q_1, q_2, q_3, q_4\}$

$T = \{a, b\}$

P: $q_1 \to bq_1 \mid aq_2$

$q_2 \to aq_2 \mid bq_3$

$q_3 \to aq_2 \mid bq_4 \mid b$

$q_4 \to aq_1 \mid bq_1$

ababb

$q_1 \to aq_2$

$\to abq_3$

$\to abaq_2$

$\to ababq_3$

$\to ababb \to$ acceptable.

abab

$q_1 \to aq_2$

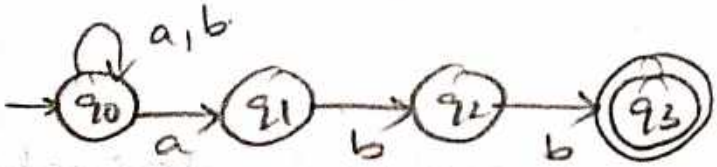
$\to abq_3$

$\to abaq_2$

$\to ababq_3$

not acceptable

(or)



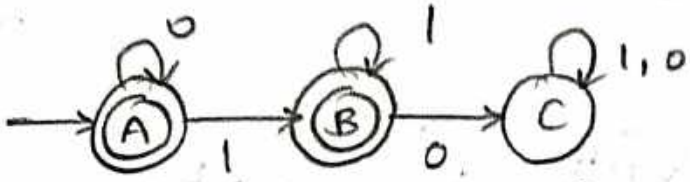
P: $q_0 \rightarrow aq_0 | bq_0 | aq_1$

$q_1 \rightarrow bq_2$

$q_2 \rightarrow bq_3 | b$

$q_3 \rightarrow \epsilon$

Ex:-



Language - 0^*1^*

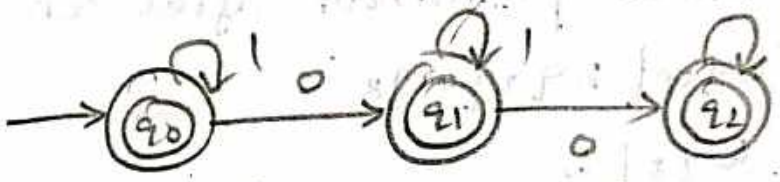
$V = \{A, B, C\}$ $T = \{0, 1\}$

P: $A \rightarrow 0A, 1B | 0 | 1$

$B \rightarrow 1B, 0C | 1$

$C \rightarrow 0C | 1C |$

Ex:-



$V = \{q_0, q_1, q_2\}$ $T = \{0, 1\}$

P: $q_0 \rightarrow 1q_0 | 0q_1 | 0 | 1$

$q_1 \rightarrow 1q_1 | 0q_2 | 0 | 1$

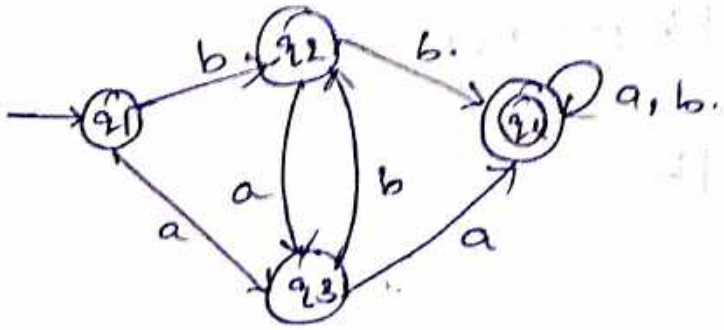
$q_2 \rightarrow 1q_2 | 1$

Language:- string with atmost 2 0's

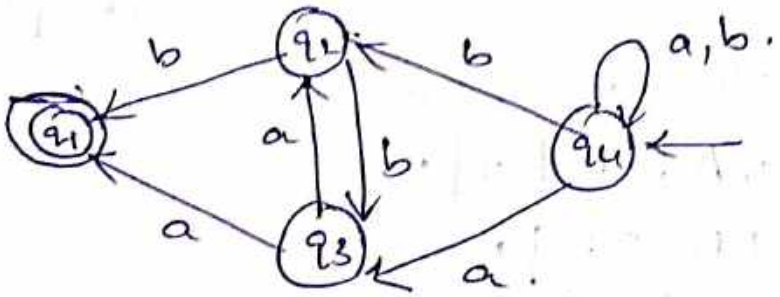
* Right linear grammar to left linear grammar:-

1. Interchange the initial & final states
2. Change the directions of transition links
3. Construct right linear grammar for the given finite automata.
4. Now, construct left linear form Right linear.

Ex:-



Step-1&2



Step-3 :- Right linear grammar after conversion.

$q_4 \rightarrow aq_4 | bq_4 | bq_2 | aq_3$

$q_2 \rightarrow bq_1 | bq_3 | b$

$q_3 \rightarrow aq_2 | aq_1 | a$

$q_1 \rightarrow \lambda$ [null production rule]

Step-4 :- Left linear grammar for initial automata.

$q_4 \rightarrow q_4a | q_4b | q_2b | q_3a$

$q_2 \rightarrow q_1b | q_3b | b$

$q_3 \rightarrow q_2a | q_1a | a$

$q_1 \rightarrow \lambda$

Right linear grammar for initial automata:-

$q_1 \rightarrow bq_2 | aq_3$

$q_2 \rightarrow bq_4 | aq_3 | b$

$q_3 \rightarrow bq_2 | aq_4 | a$

$q_4 \rightarrow aq_4 | bq_4 | a | b$

consider string bb

right linear verification

bb $q_1 \rightarrow bq_2$

$\rightarrow bb$

acceptable

left linear verification

$q_u \rightarrow q_2 b$.

$\rightarrow bb$

acceptable.

* left linear to right linear

1, Construct right linear grammar equivalent to given left linear grammar. [interchange terminal & non terminal]

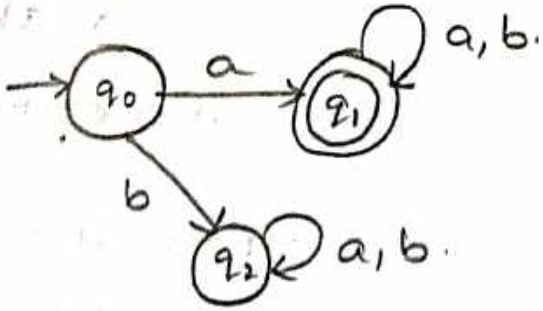
2, Construct finite automata from above intermediate right linear grammar.

3, Interchange initial & final states

4, change direction of transition links.

5, Construct exact right linear grammar form the above automata.

Ex:- Construct the left linear grammar for the language which accepts the strings starts with 'a'. $\Sigma = \{a, b\}$



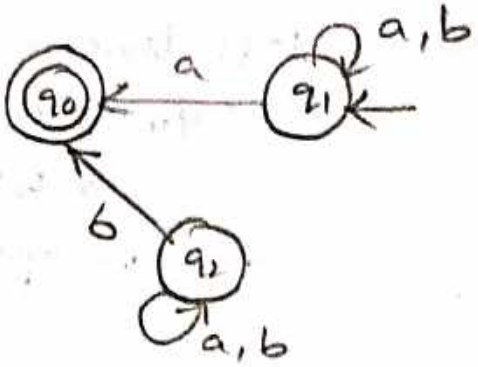
Right linear grammar

$q_0 \rightarrow aq_1 | bq_2 | a$

$q_1 \rightarrow aq_1 | bq_1 | a | b$

$q_2 \rightarrow aq_2 | bq_2 | a | b$

Step-1,2 :-



Step-3 :- intermediate right linear grammar

$q_2 \rightarrow aq_2 | bq_2 | bq_0 | b$

$q_1 \rightarrow aq_1 | bq_1 | aq_0 | a$

$q_0 \rightarrow \lambda$

Step-4 :- left linear grammar.

$q_2 \rightarrow q_2a | q_2b | q_0b | b$

$q_1 \rightarrow q_1a | q_1b | q_0a | a$

$q_0 \rightarrow \lambda$

consider string aba

Right linear verification

$q_0 \rightarrow aq_1$

$\rightarrow abq_1$

$\rightarrow aba$

acceptable

left linear verification

$q_2 \rightarrow q_2a$

$\rightarrow q_1ba$

$\rightarrow aba$

acceptable.

$q_1 \rightarrow q_1a$

$\rightarrow q_1ba$

$\rightarrow aba$

acceptable