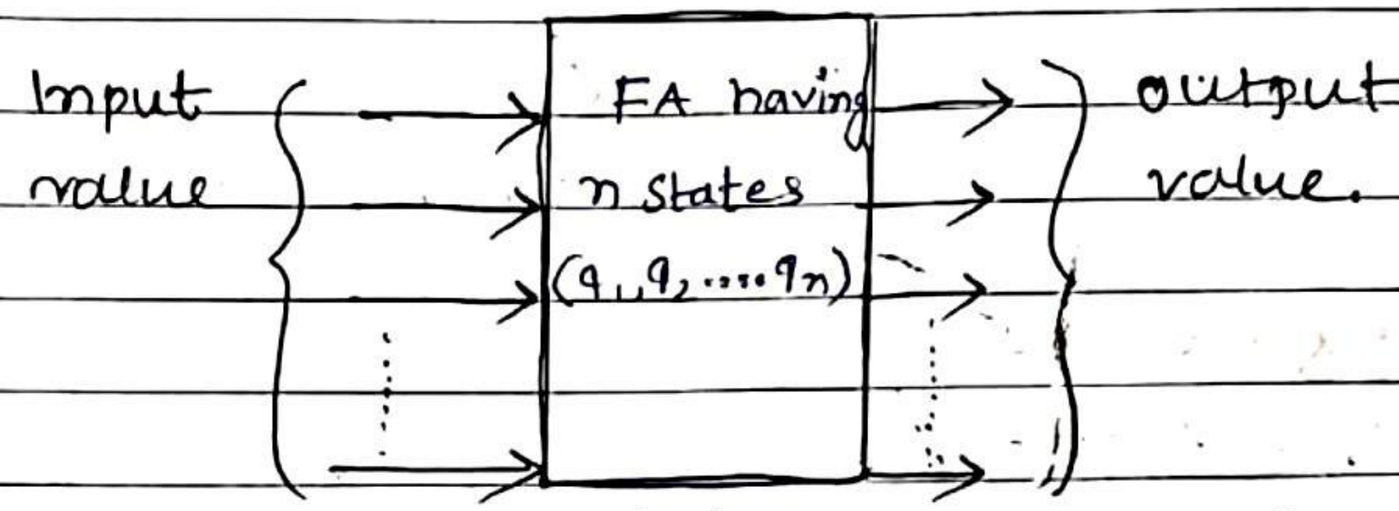
Regular Expression and Finite Automata.

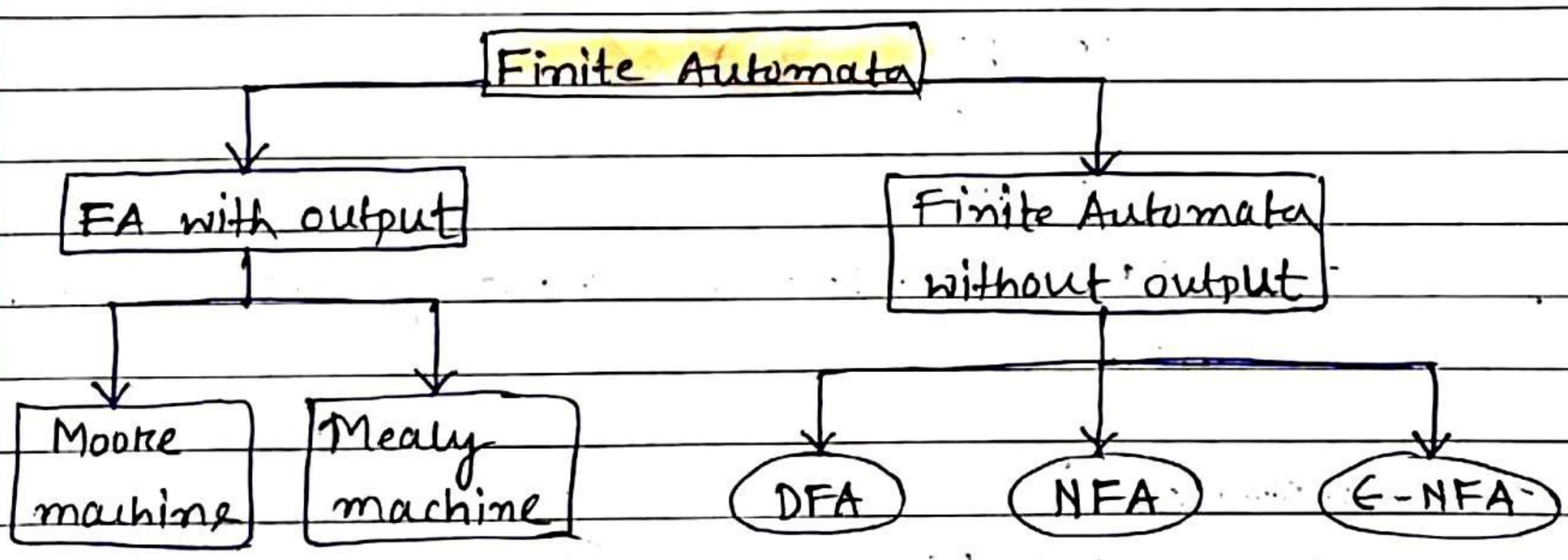
- **Finite Automata:-**

→ A Finite state Machine (FSM) or finite state automata is an abstract machine used in the study of computation and language that has only a finite, constant amount of memory.



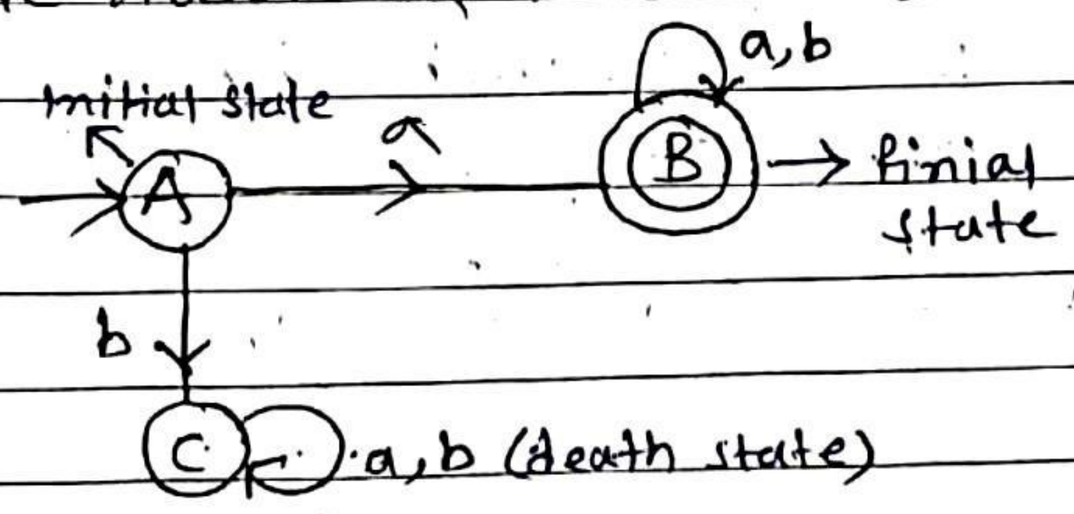
Model of finite automata

- **Types of Finite Automata:**



- **DFA (Deterministic Finite Automata):-**

→ A finite Automata is defined as 5-tuples $(Q, \Sigma, \delta, q_0, F)$.



Where,

Q = set of all states $\Rightarrow \{A, B, C\}$

Σ = Input $\Rightarrow \{a, b\}$

q_0 = Initial state $\Rightarrow$ 'A'.

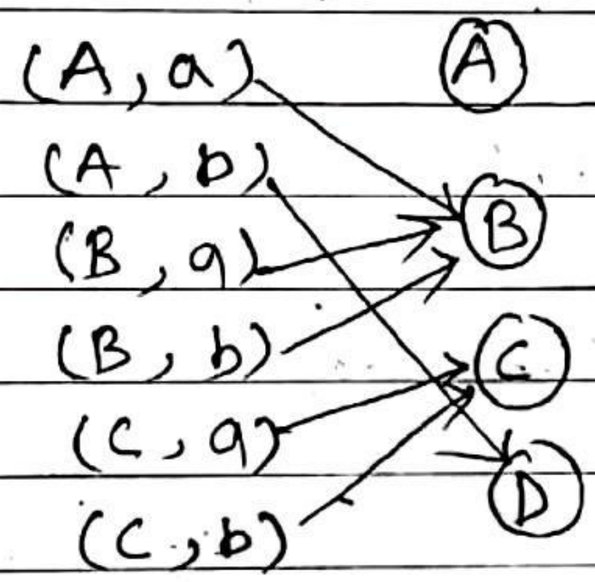
F = finial state $\Rightarrow \{B\}$
↳ set of.

→ Q is the superset of F. ($F \subseteq Q$)

δ = transation function which maps $Q \times \Sigma$ into Q.

$$\boxed{\delta : Q \times \Sigma \rightarrow Q}$$

$\{A, B, C\} \times \{a, b\}$



→ In DFA, a state with 1 Input go into another one state.

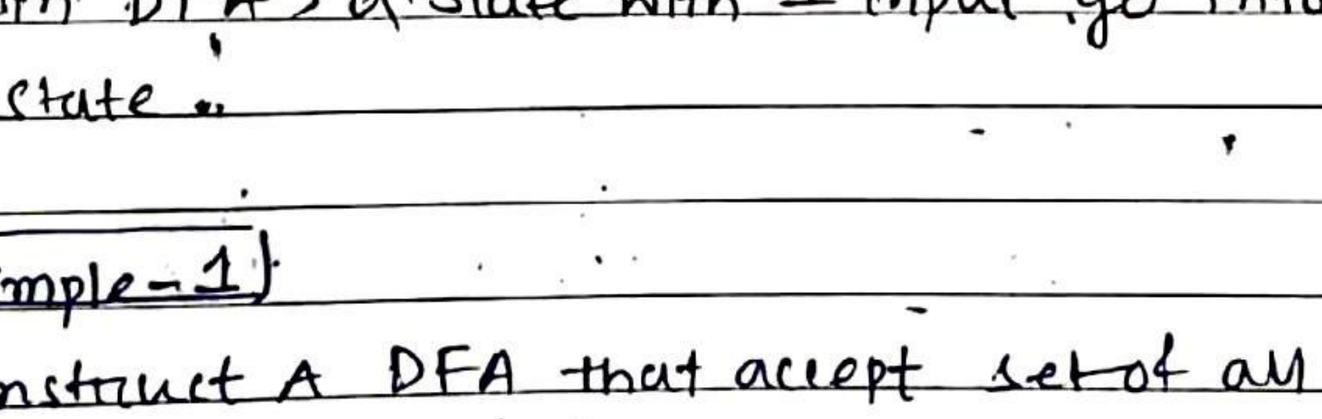
Example-1

Construct A DFA that accept set of all string over $\{a, b\}$ of length '2'.

→ $\Sigma = \{a, b\}$.

language, $L_1 = \{aa, ab, ba, bb\}$

State transation Diagram -



- String accept → Scan the entire string, if we reach in a finial state from initial state. Then string will be accepted by FA.

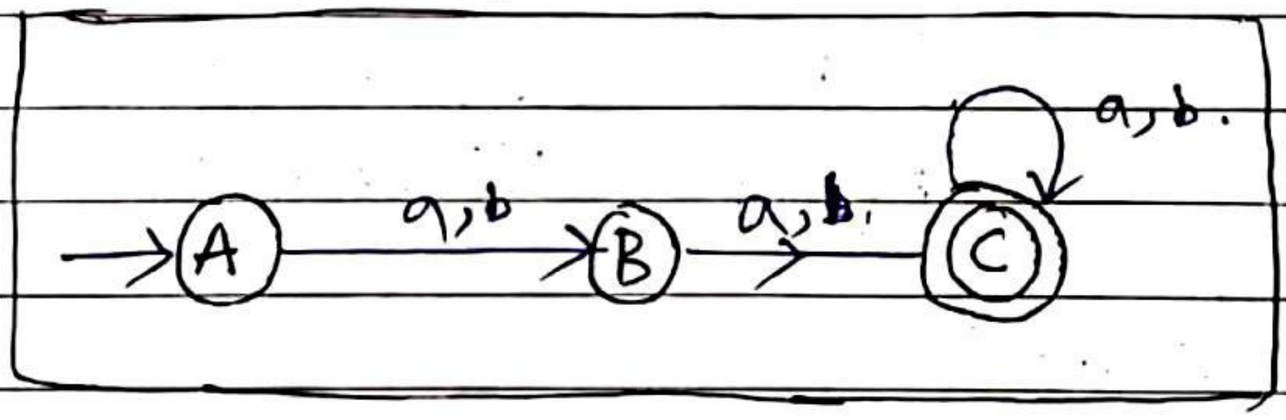
- language accept → A Finite Automata is said to accept a language if all the string in the language are accepted and the string not in the language are rejected.

EX-2

Construct a DFA which accepts all the String {a,b} where the string length is ~~accept~~ atleast '2'. $|w| \geq 2$

→ $\Sigma = \{a, b\}$

$1 = \{aa, ab, ba, bb, aaa, aba, baa, bba \ldots\ldots\ldots\}$



State transition Diagram,

EX-3

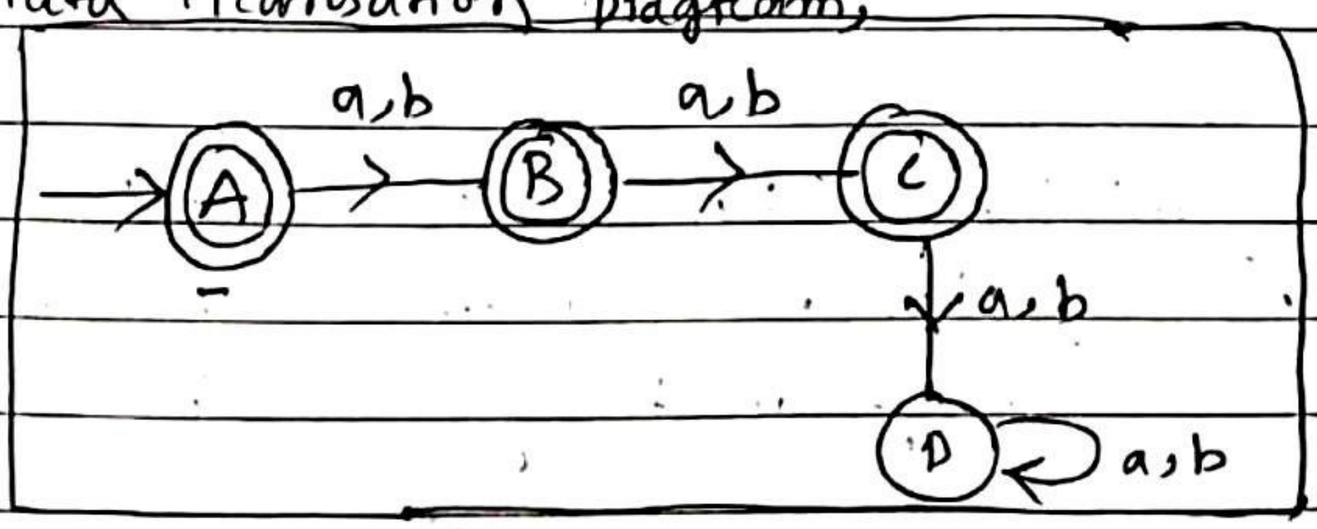
∴ Construct a DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}$, $|w| \leq 2$.

∴

→ $\Sigma = \{a, b\}$

$1 = \{\epsilon, a, b, aa, ab, ba, bb\}$

State transition Diagram,



Example-3 $w \rightarrow$ string

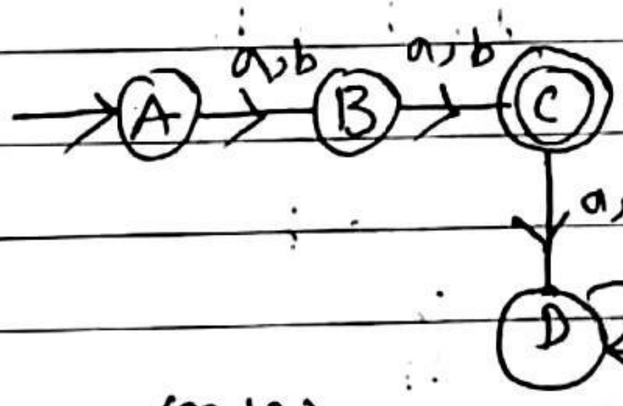
(*) $|w|=2$, $|w| \geq 2$, $|w| \leq 2$ .. (gate)

→ When, string length $|w| = n$, then
no of state required $(n+2)$.
when, $|w| \geq n$, then
no of state required $(n+1)$.

When, $|w| \leq n$, then
no of state required $(n+2)$

• $|w|=2$

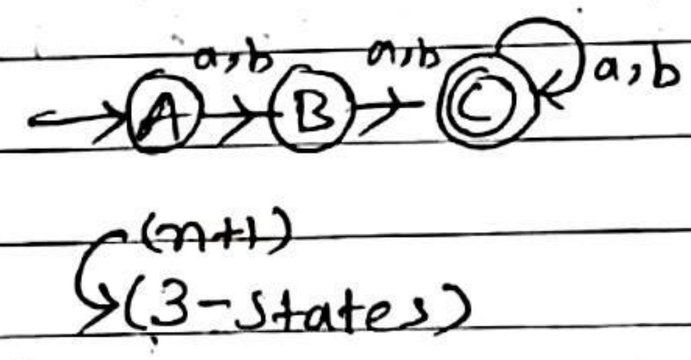
$L = \{aa, ab, ba, bb\}$



(n+2)
(4-states)

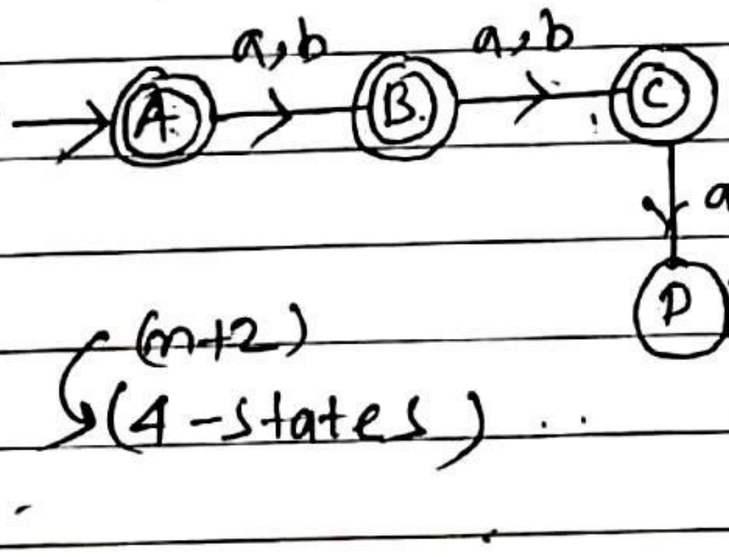
• $w \geq 2$

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



• $|w| \leq 2$

$L = \{\epsilon, a, b, aa, ab, ba, bb\}$



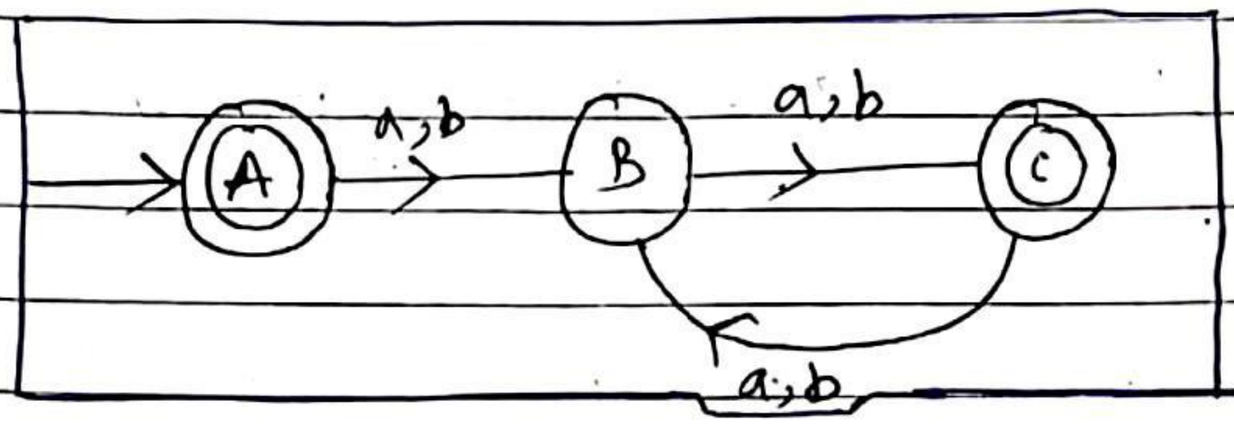
Example-4

Construct a minimal DFA which accept all strings over $\{a,b\}$, $|w| \bmod 2 = 0$ (means the length of the string ↳ length of string will be even)

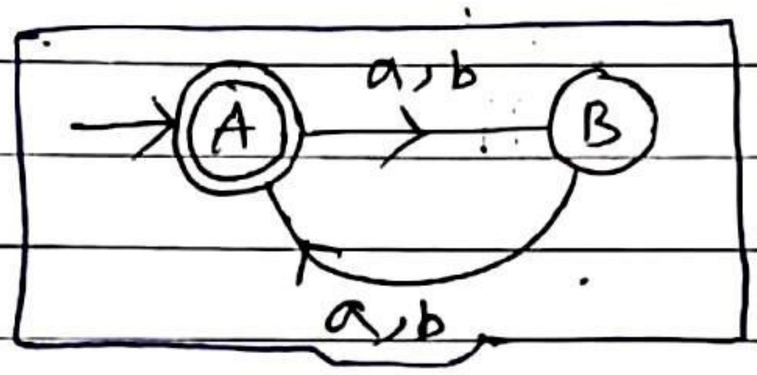
→

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

$L = \{\epsilon, aa, ab, ba, bb, aaaa, \text{----}, bbbb, \text{-----}\}$



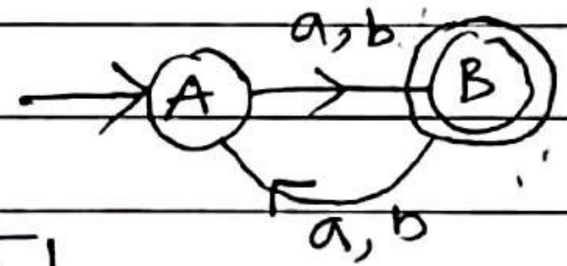
OR



(State transation Diagram)

Ex-5, $|w| \bmod 2 = 1$

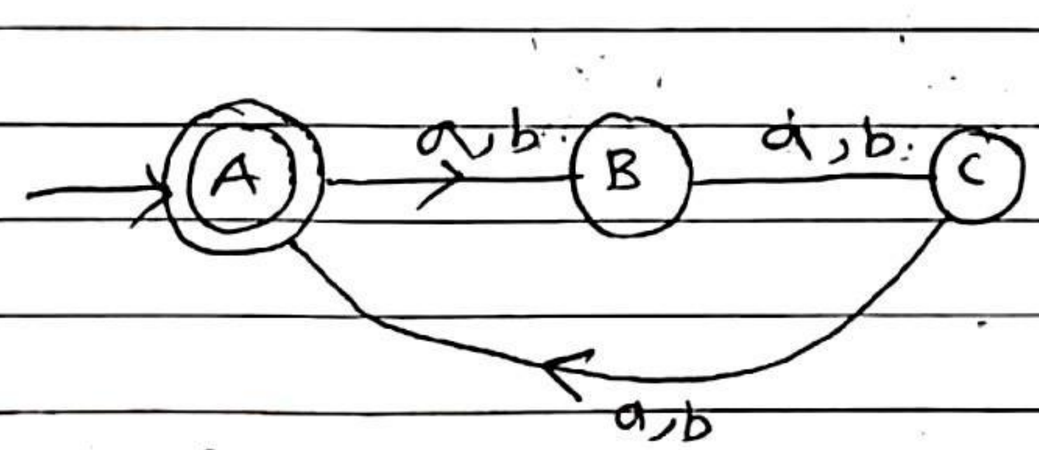
→ $L = \{a, b, aaa, bbb, \cdots aaaaa, \cdots\}$



Ex-6,

$|w| \bmod 3 = 0$

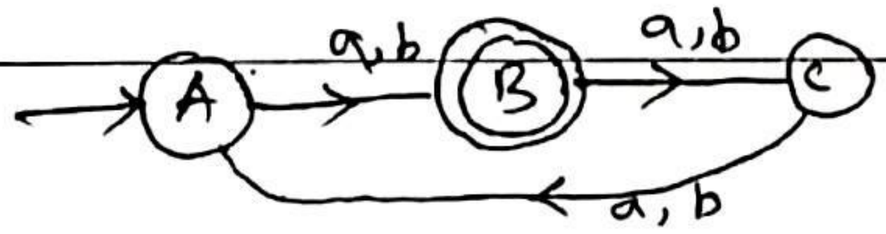
→ $L = a(aa)^* \{\epsilon, aaa, aba, bbb \cdots\}$



finial state 'A',

Ex-7, $|w| \bmod 3 = 1$

→ $L = \{a, b, aaaa, bbbb, \cdots\cdots\}$



** If $|w| \bmod n = 0$, then no of states are 'n'.

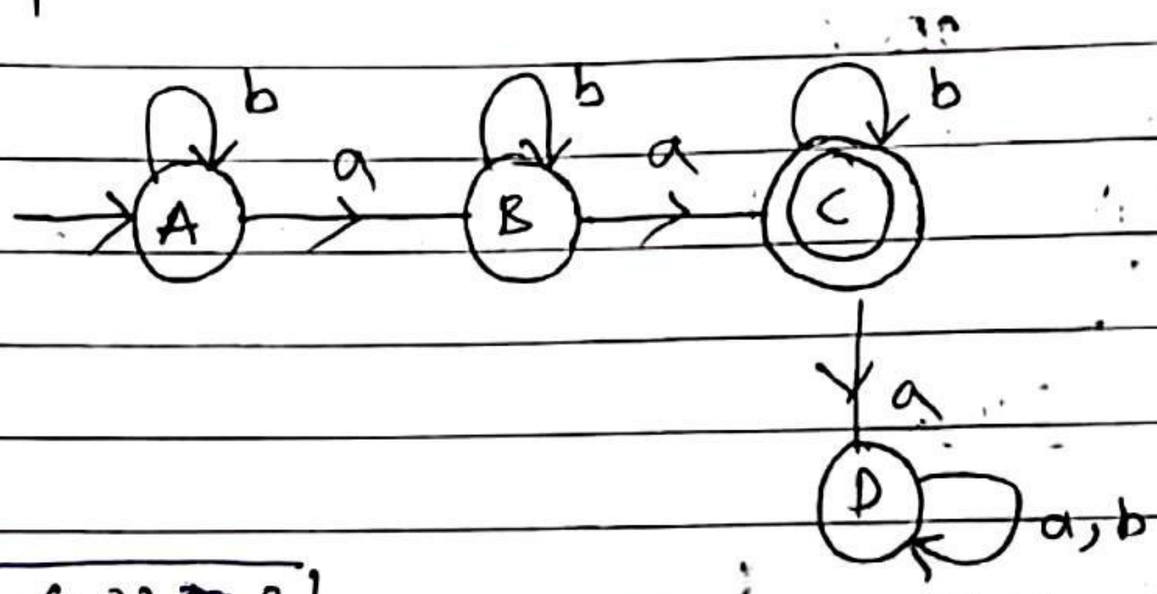
Example - 8

Construct a minimal DFA, that accept $w \in \{a,b\}^*$

1. $n_a(w) = 2$

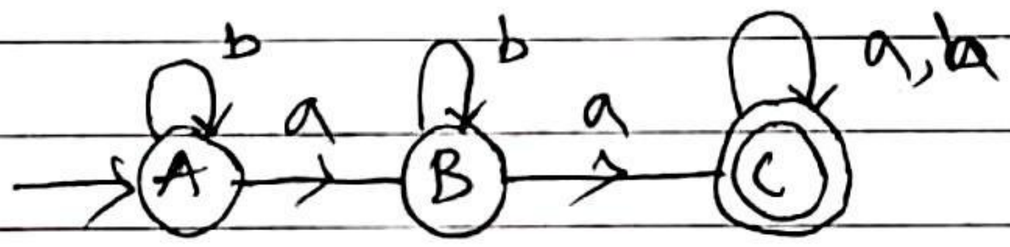
→

$L = \{aa, aab, baa, aba, bbaa \ldots\ldots\}$



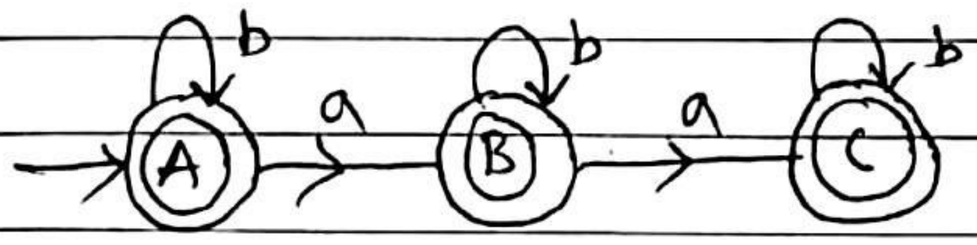
2. $n_a(w) \geq 2$

→ $L = \{aa, aaab, aaaaba \ldots\ldots\}$



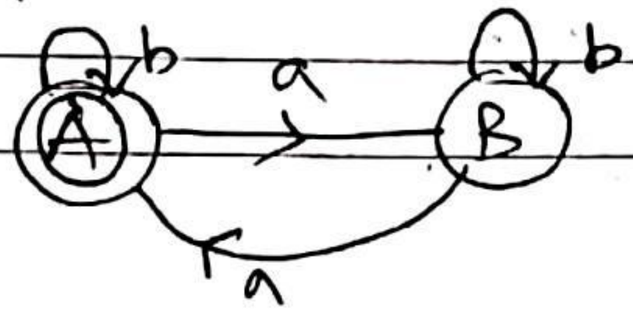
3. $n_a(w) \leq 2$

→ $L = \{\epsilon, a, b, aa, ab, ba, bb\}, baabb, \ldots\ldots\}$



4. $n_a(w) \bmod 2 = 0$

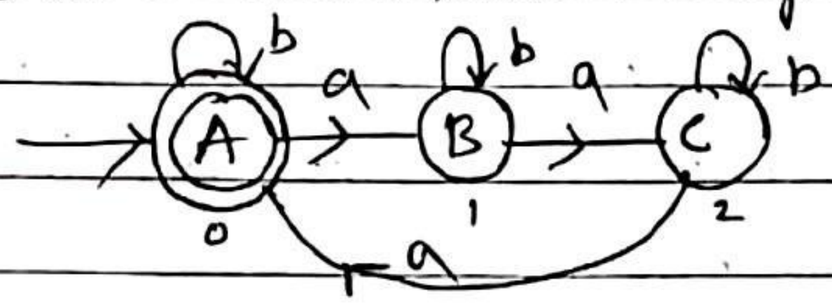
→ $L = \{aa, aab, aabb, aaaa, aaaaaa \ldots\ldots\}$



5. $\boxed{n_a(w) \bmod 3 = 0}$

→ $L = \{aaa, \ldots aaaaaaa \ldots\}$

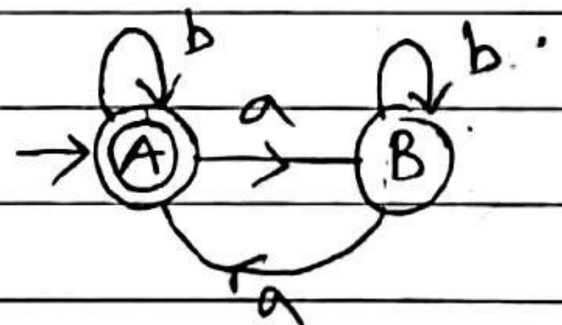
State transition diagram -



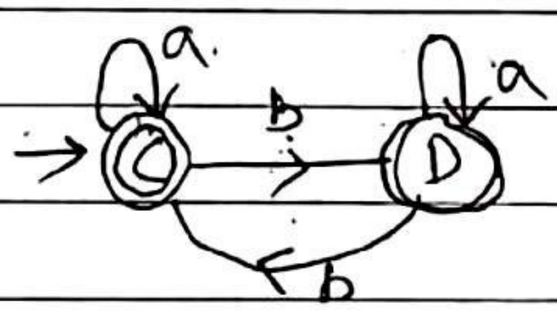
6. Construct a minimal DFA, where $w \in \{a,b\}^*$

$$\boxed{\begin{array}{l} n_a(w) \cong 0 \bmod 2 \\ \quad \& \& \\ n_b(w) \cong 0 \bmod 2 \end{array}}$$

→ $L = \{\epsilon, aa, bb, aabb, abab, bbaaaa, \ldots bbbb \ldots\}$



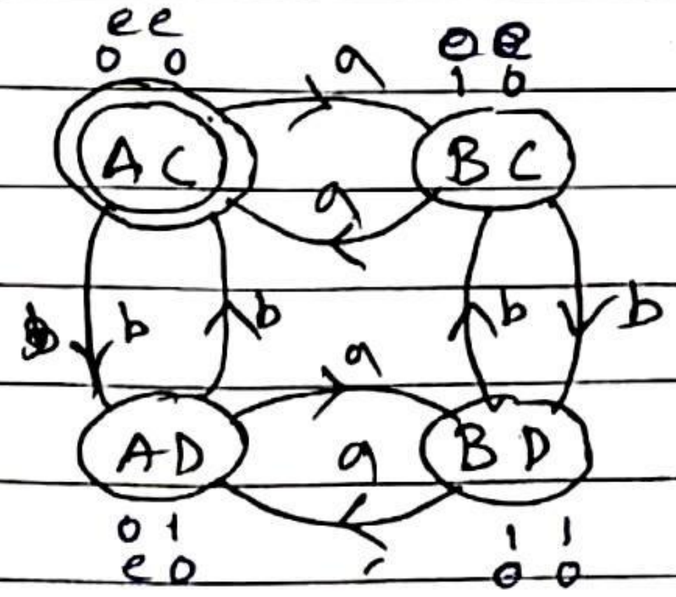
# of state {A, B}



# of state {C, D}

Cross product method -

$$\{A,B\} \times \{C \times D\} = \{AC, AD, BC, BD\}$$



A	a	B	A	b	A
C		C	C		D

e or e = {ee, eo, oe} ~~state~~ (finial state).

**

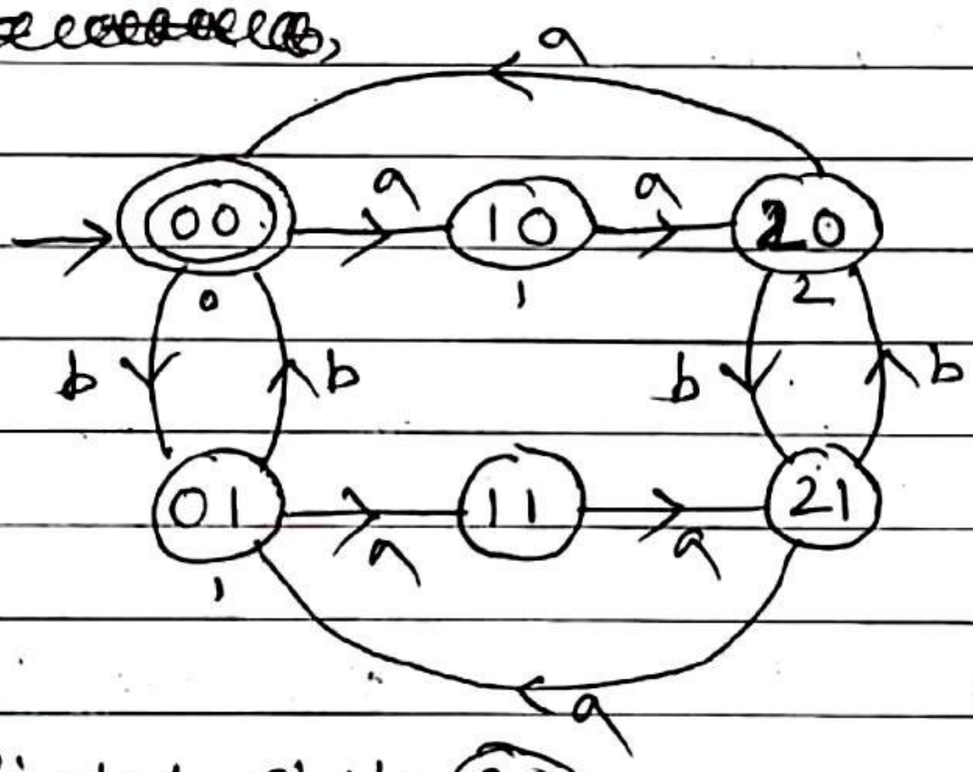
→ If one Automata contain 'n' state and the other automata contain 'm' number of states then there cross product going to contain, (mxn) states.

⑦ Construct a minimal DFA which accepts set of all strings over {a, b} in which no of a's are divisibe by 3 and no of b's are divisible by 2.

→ $w \in \{a,b\}^*$

$n_a(w) \equiv 0 \bmod 3$

$n_b(w) \equiv 0 \bmod 2$



finial state (00).

When, $n_a(w) \bmod 3 \neq n_b(w) \bmod 2$

finial state = {10, 20, 11, 21}

**

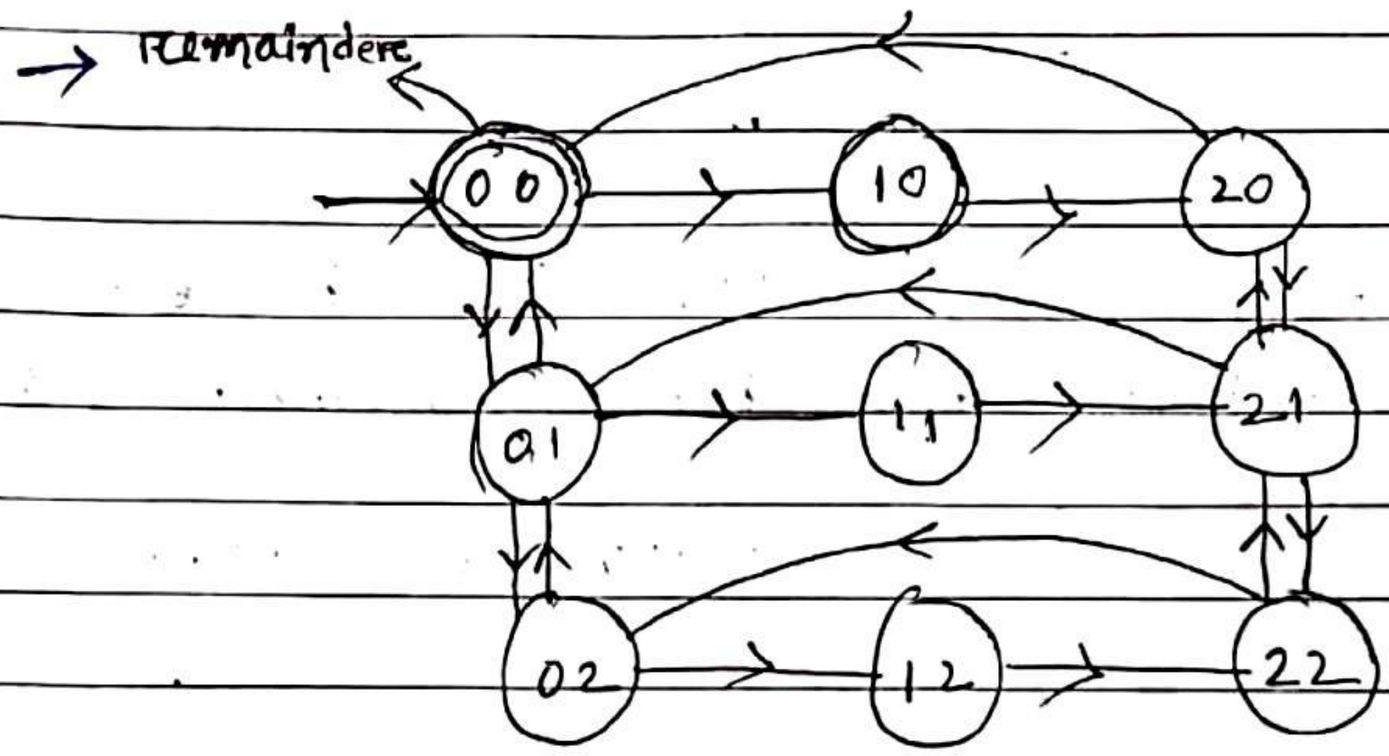
When, $n_a(w) \bmod 3 = n_b(w) \bmod 2$

finial state = {00, 11}.

(8) Construct a minimal DFA, $w \in \{a,b\}^*$,

$n_a(w) \cong 0 \bmod 3$

$n_b(w) \cong 0 \bmod 3$



When; $n_a(w) \bmod 3 = 1$

&

$n_b(w) \bmod 3 = 2$, then

final state = {1, 2}

When, $n_a(w) \bmod 3 > n_b(w) \bmod 3$, then

final state = {10, 20, 21}

→ $n_a(w) \cong 0 \bmod n$

$n_b(w) \cong 0 \bmod n$

Then the minimum no of states in automata is 'n x n'.

Example-2] — (1)

Construct a minimal DFA, which accepts set of all string over {0,1}, which when interpreted as binary number is divisible by '2'.

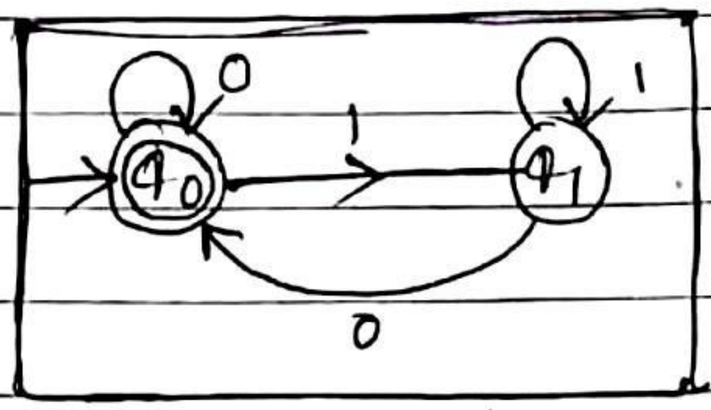
→ $\Sigma = \{0,1\}$

$w \in \{0,1\}^*$

If we divided any no by 2 reminder can be '0' or '1'.

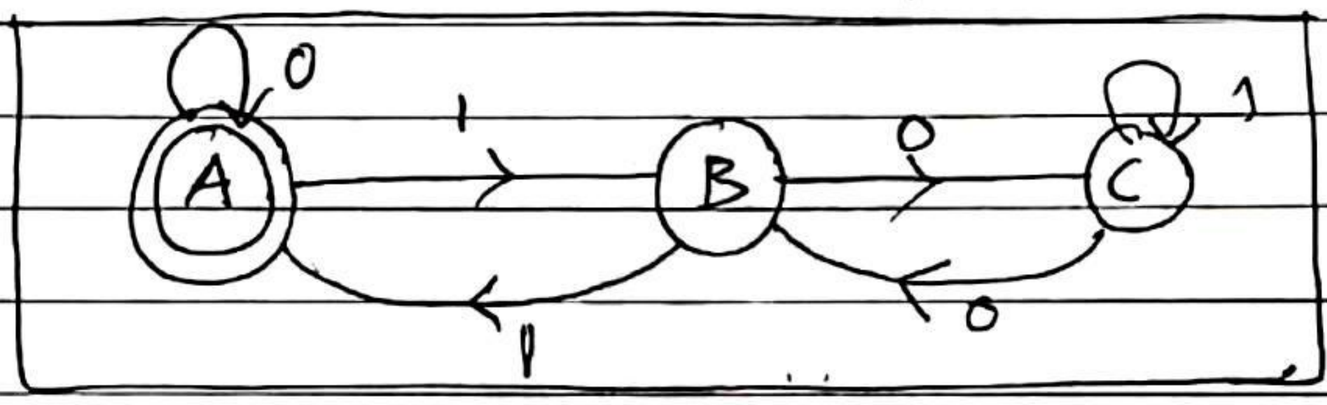
32 16 8 4 2 1
1 1

$L = \{0, 00, 000, 0000, \ldots, 10, 100, 110, \ldots\}$



(2) construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$ when interpreted as binary number is divisibe by 4.

→ $L = \{0, 00, 000 \ldots, 11, 110, 1100, 1111 \ldots\}$



→ Finite Automata Can represent in two ways
- state ~~Diagram~~ transation Diagram.
- State transation table.

State transtion table,

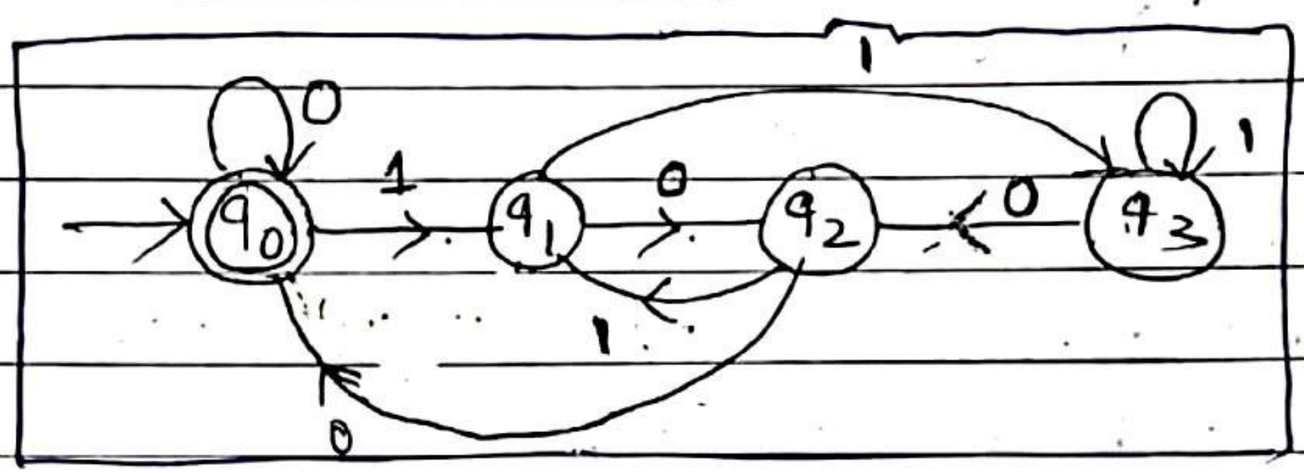
	0	1
→*A	A	B
B	C	A
C	B	C

16 8 4 2 1

(3) Construct a minimal DFA, which accept ~~all~~ set of all string over $\{0,1\}$ which when interpreted as a binary number is divisible by '4'.

→ $\Sigma = \{0,1\}$, $\forall w \in \{0,1\}^*$

$L = \{0, 100, 000, \cdots, 100, 1000, 1100, 10000, 10100, \cdots\}$



State transaction table -

	0	1
* → q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3

→ If the reminder is = 0
then FS → q_0
reminder is = 1
FS → q_1
reminder is = 2
FS → q_2
reminder is = 3
FS → q_3

Example - 10

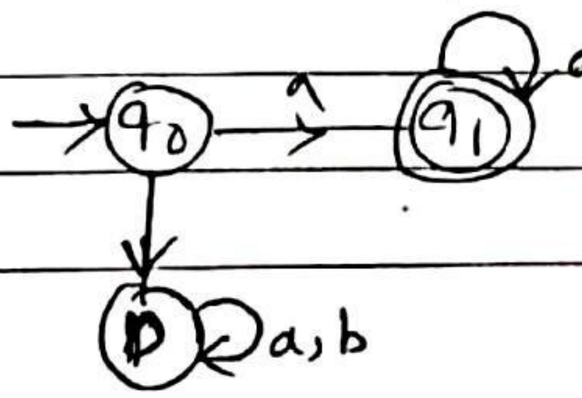
Construct a minimal DFA, which accepts set of all strings, over $\{a,b\}$ where each string starts with an 'a'.

→ $\Sigma = \{a, b\}$

$w \in \{a, b\}^*$

language, $L_1 = \{a, aa, ab, aaa \ldots\}$

state transaction Diagram,

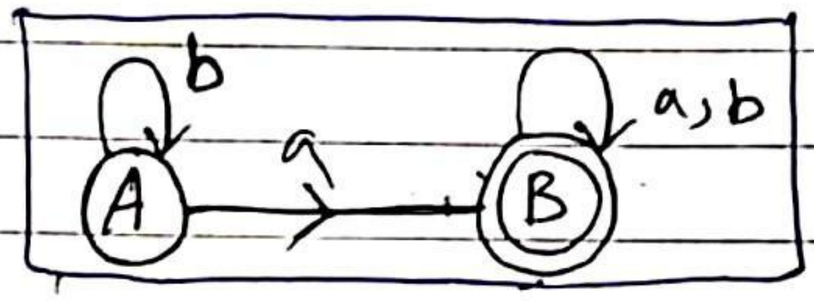


→ If the language is infinite then take the smallest string and make its skeleton.

Example-11

Construct ~~mi~~ a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$ "where each string contains 'a'."

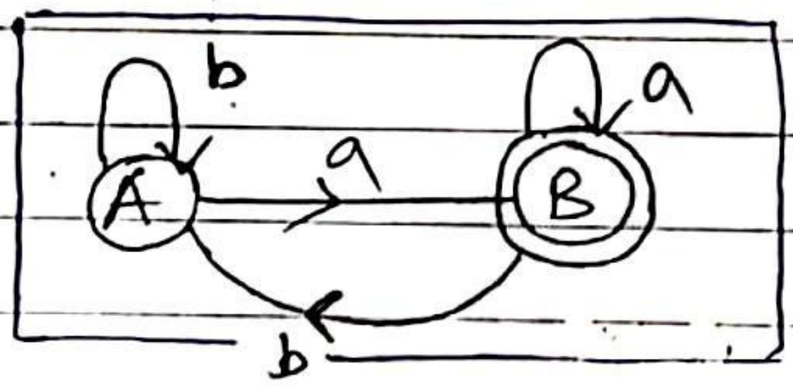
→ $L = \{a, aa, ab, ba, aaa, \ldots\}$



Example-12

Construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$ String ends with an 'a'.

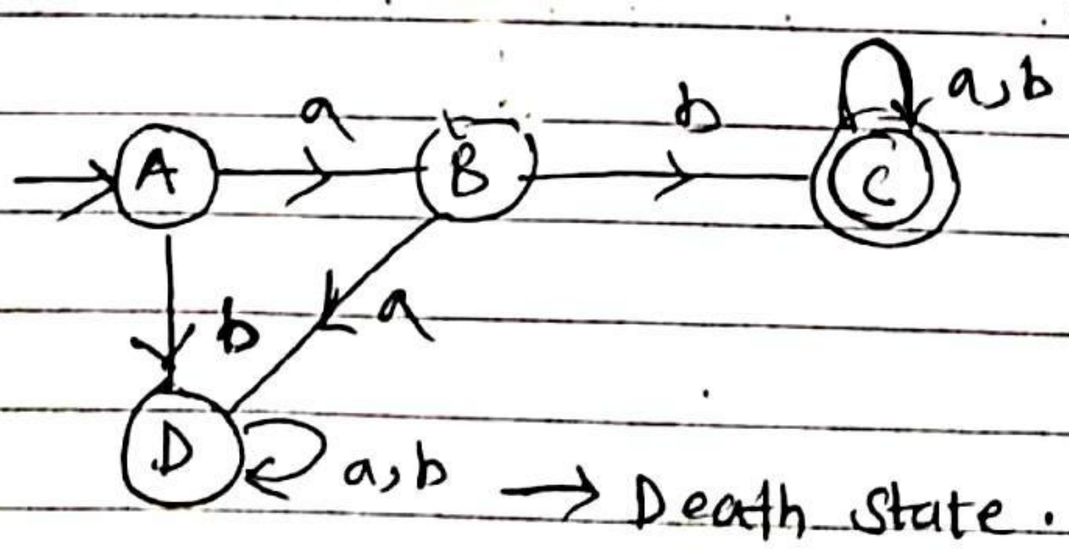
→ $L = \{a, aa, ba, aaa, bba, baa, bbba \ldots\}$



Example-13

Construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$ "each string start with 'ab'".

→ $L = \{ab, aab, abb, \ldots\}$

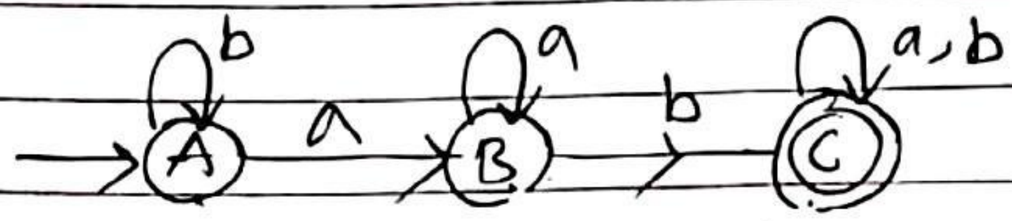


Example - 14

DFA, containing ab.

→ Here $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$

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

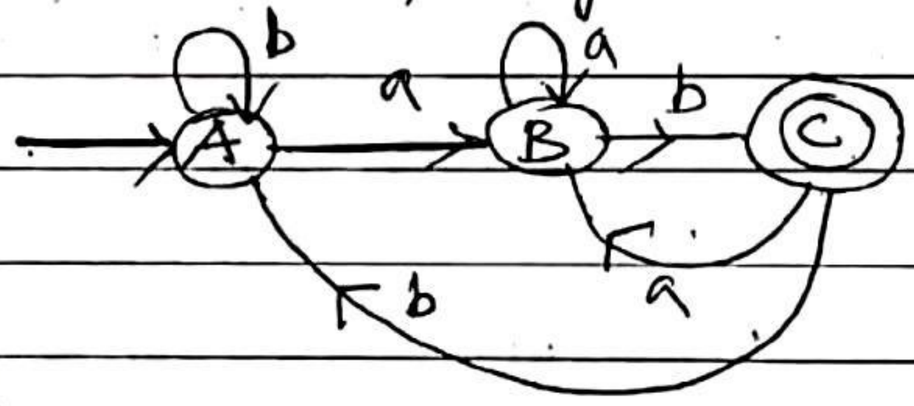


Example - 15

~~end~~ Construct a minimal DFA, end's with "ab".

→ $L = \{ab, bbab, aaab \ldots\}$

State transition diagram-

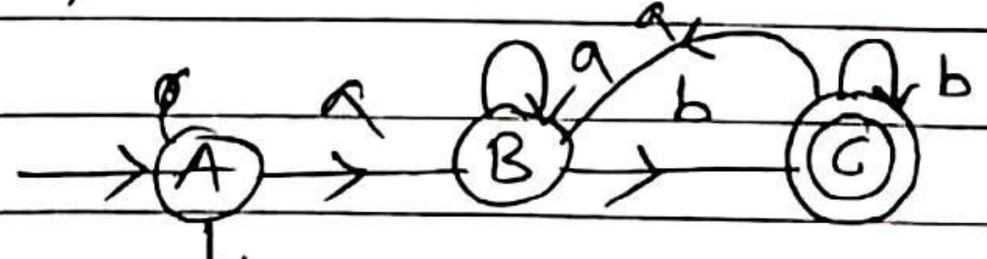


Ex - 16

construct a minimum DFA, starts with 'a' and ends with 'b'.

→ $\Sigma = \{a, b\}^*$, $w \in \{a, b\}^*$

$L = \{ab, aabb, abab, abbab \ldots\}$

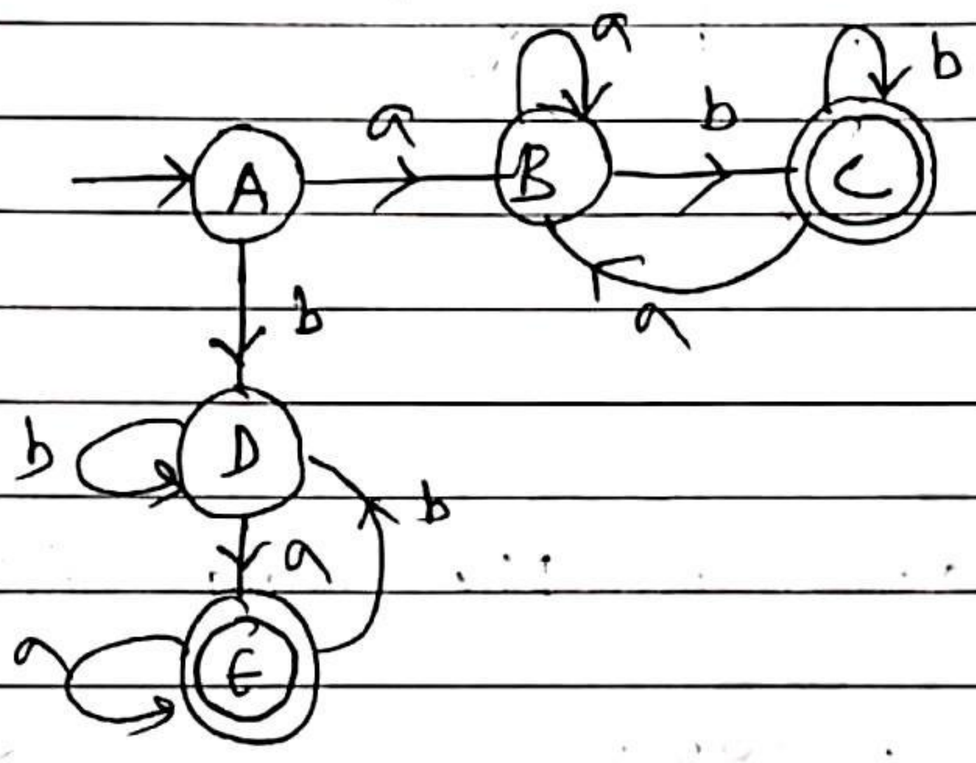


Ex-17

Construct a DFA, $\Sigma=\{a,b\}$, $w\in\{a,b\}^*$
" start and ends with different symbol".

→ $\Sigma=\{a,b\}$, $w\in\{a,b\}^*$.

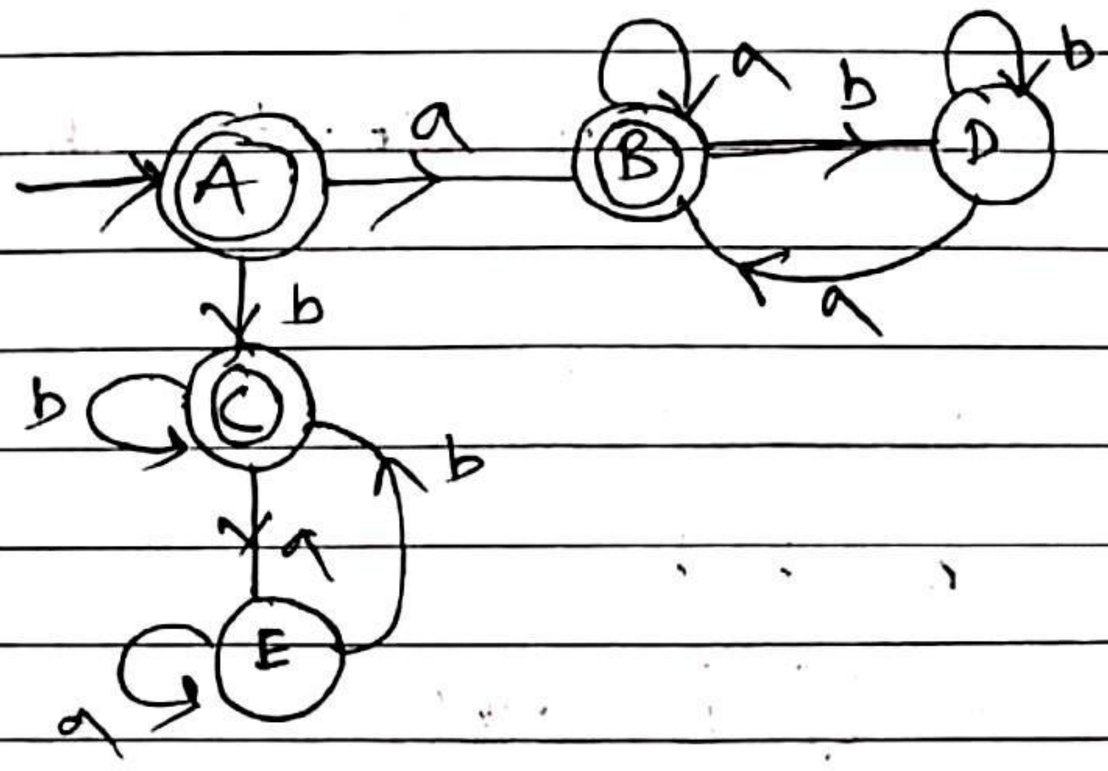
$L=\{ab, ba, abb, bba.....\}$



Ex-18

Construct a DFA, $\Sigma=\{a,b\}$, $w\in\{a,b\}^*$.
" start and ends with same symbol".

→ $L=$ ~~aa~~ $\{\epsilon, a, b, aa, bb...\}$



→ here, Example - 17 and 18 are compliment of each other.

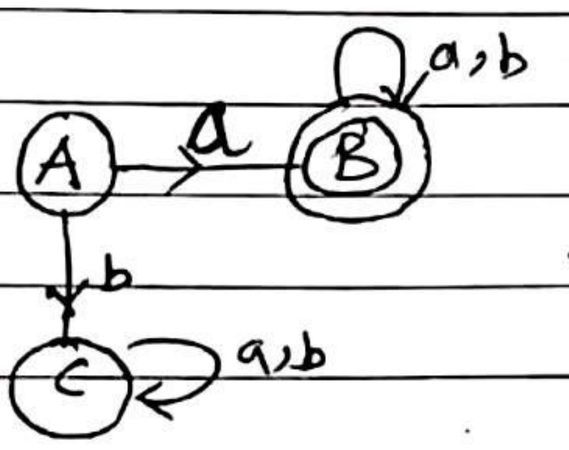
Ex-18

→ L_1 and L_2 are 2-language, then L_1 is said complement of L_2,

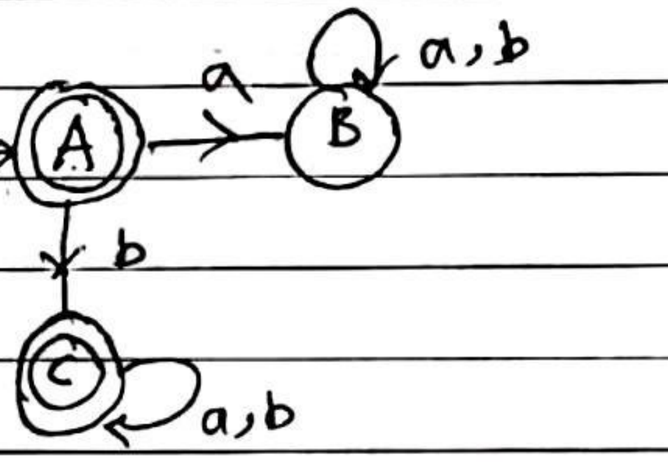
$$L_1 = \Sigma^* - L_2$$

- Complementation of DFA :

L_1 = {Starting with 'a'}



L_2 = {not starting with 'a'}



$\Leftrightarrow$

$L_1 = \overline{L_2}$

→ Complimentation methode apply for only DFA not for NFA.

→ Every thing has to be same only ~~finial~~ finial state will be changed ^

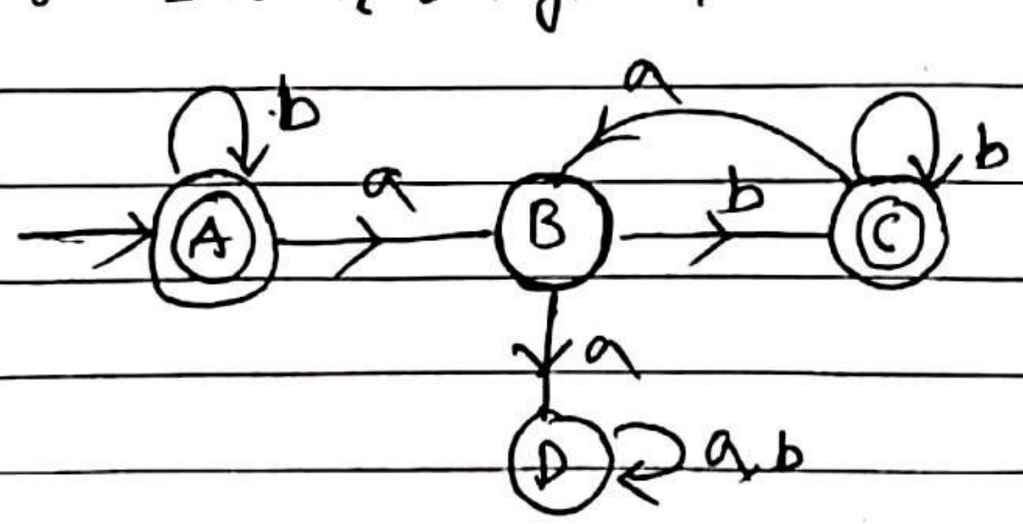
Ex-19

Construct a DFA, $w \in \{a,b\}^*$, F

Every 'a' should be followed by a 'b'.

→ $L = \{\epsilon, ab, abb, abab, babb, \ldots, b, bbb, \ldots\}$

State transation Diagram -



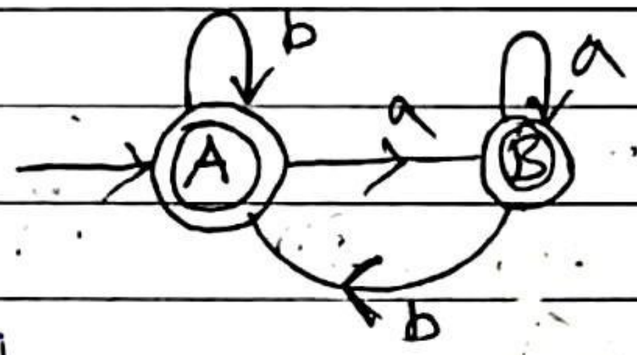
Ex-20

Construct a DFA, where, $w \in \{a,b\}^*$

Every 'a' should not never followed by a 'b'.

→ $L = \{\epsilon, a, b, aa, ba, bb, aaa, bba, \ldots\}$

State transation Diagram-



Ex-21

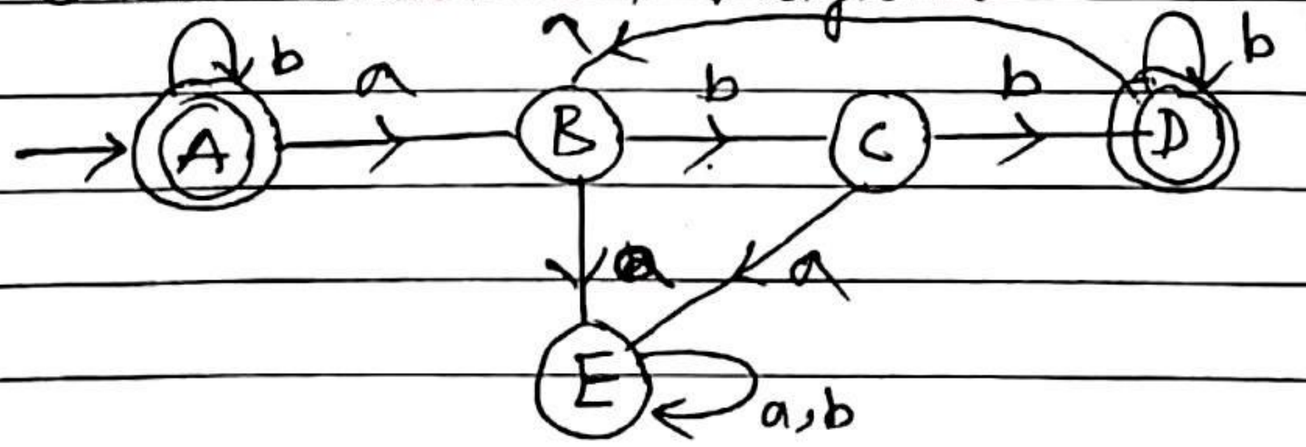
construct a minimal DFA, where

$w \in \{a,b\}^*$

every 'a' should be followed by 'bb'.

→ $L = \{\epsilon, abb, \ldots\}$

State transation Diagram-



final state = {A, D}.

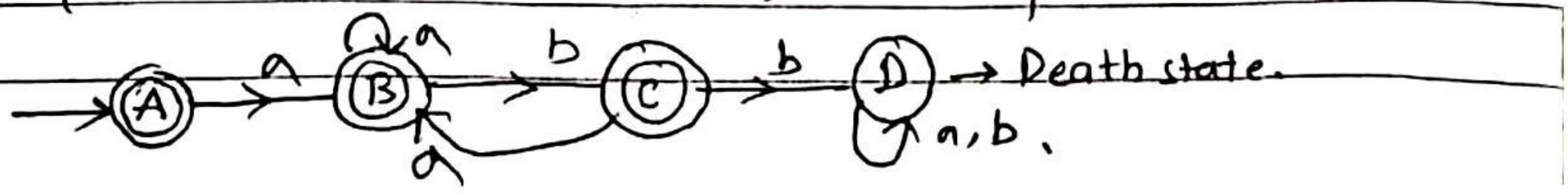
Ex-22

Construct a DFA, where.

$w \in \{a,b\}$

every 'a' should never be followed by a 'bb'.

→ $L = \{\epsilon, a, aa, ab, ba, bb, aaa, \ldots\}$

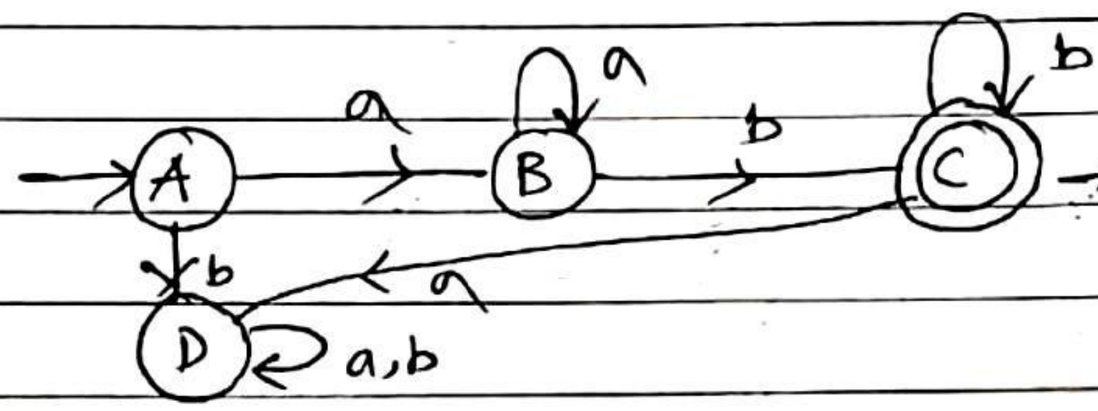


Ex-23

Construct a minimal DFA which accepts, $L = \{a^n b^m / n, m \geq 1\}$

→ $L = \{ab, aab, aabb, aabb \cdots\}$

State transcation diagram-

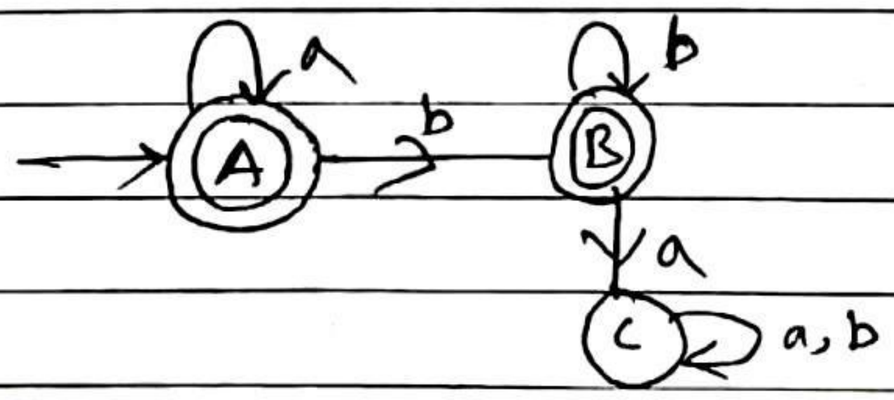


Ex-24

Construct a minimal DFA, which accepts $L = \{a^n, b^m / n, m > 0\}$

→ $L = \{\epsilon, a, aa, aaa, \cdots, b, bb, bbb \cdots, ab, aabb \cdots\}$

State transcation Diagram-

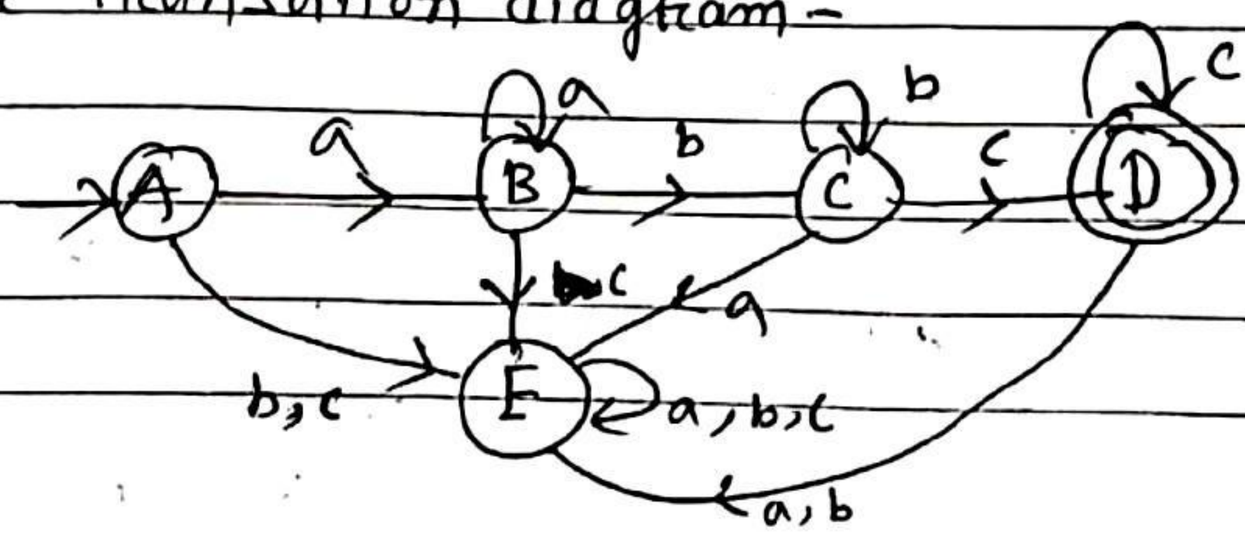


Ex-25

Constructs a minimal DFA, which accepts $L = \{a^n b^m c^l / n, m, l \geq 1\}$

→ $L = \{abc, aabbcc \cdots\}$

State transcation diagram-

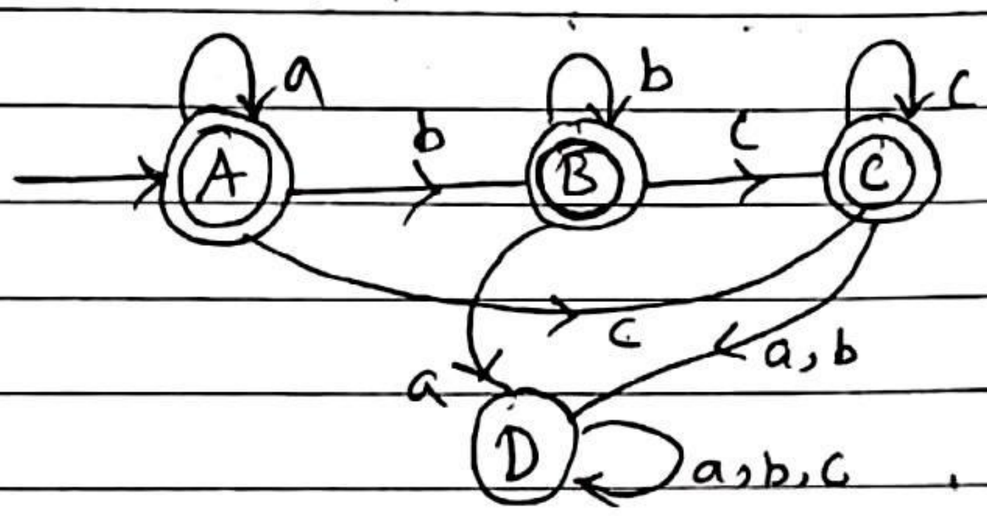


Ex-26

Construct a minimal DFA, which accepts,
$L = \{a^n b^m c^l / n, m, l \geq 0\}$.

→ $L = \{\epsilon, a, aa \cdots, b, bb \cdots, c, cc \cdots, abc, aabbcc\}$

State transation Diagram –

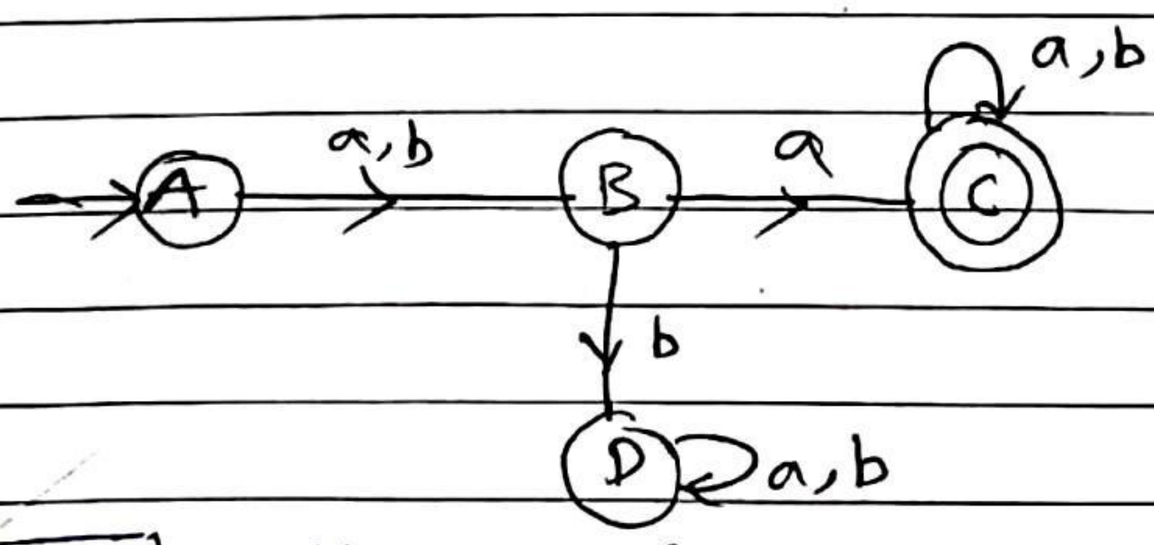


Ex-27

Construct a minimal DFA which accepts set of all strings over $\{a, b\}$ such that second symbol from left is 'a'.

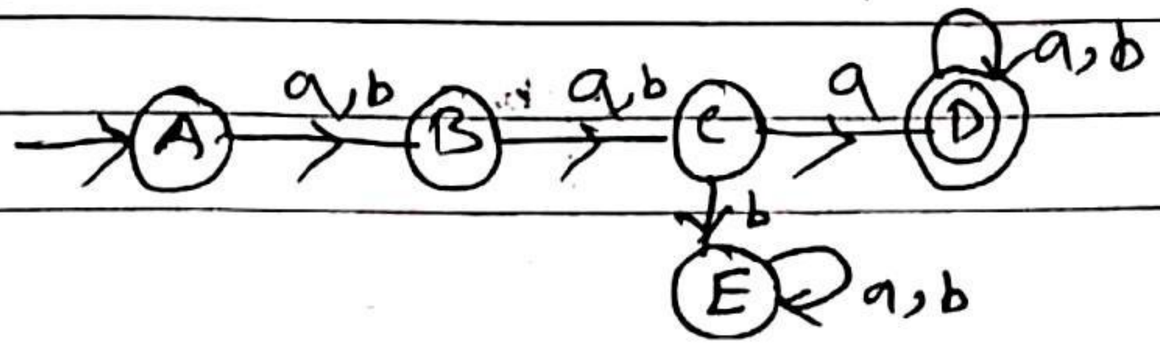
→ $\Sigma = \{a, b\}$

$L = \{aa, ba, aaa, baa, bba, \cdots\cdots\}$



Ex-28 3rd symbol from left is a.

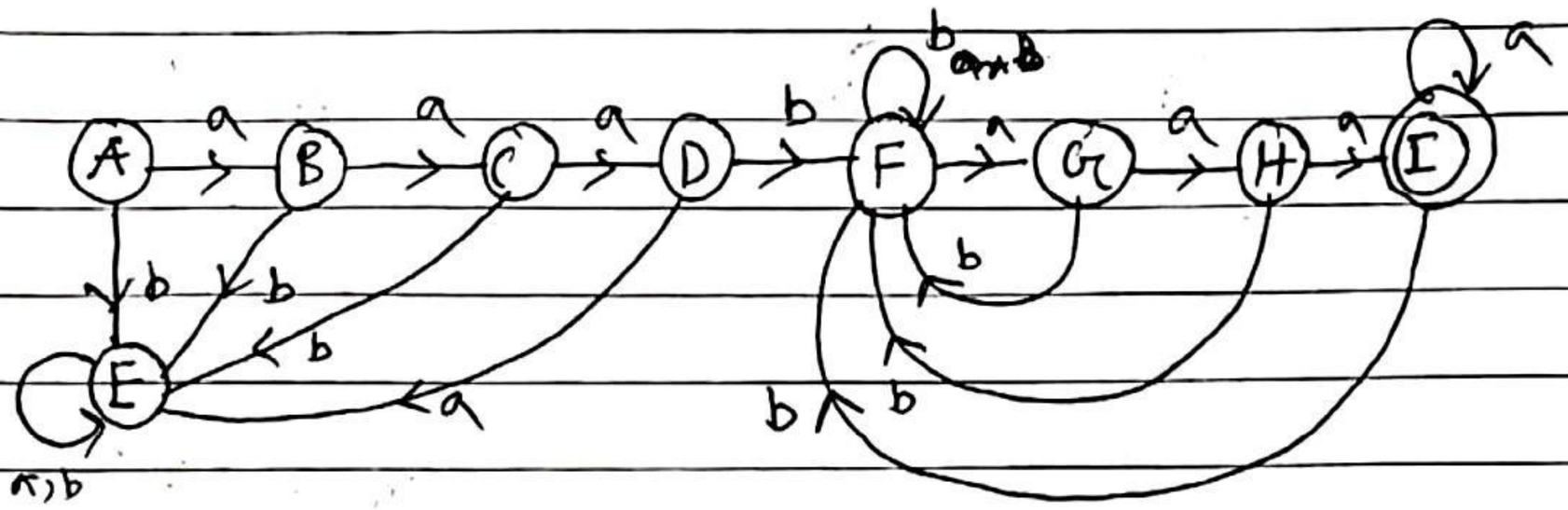
→ $L = \{aaa, aba, baa, bba, aaa, abaa \cdots\}$



Ex-29 construct a DFA which accepts set of all string over $\{a,b\}$ where strings are of the form 'a^3bwa^3'.

where 'w' is any string over $\{a,b\}$.

→ $L = \{a^3b \in a^3, a^3b\,a\,a^3, a^3b\,b\,a^3, \ldots\}$



- OPERATION ON DFA like –

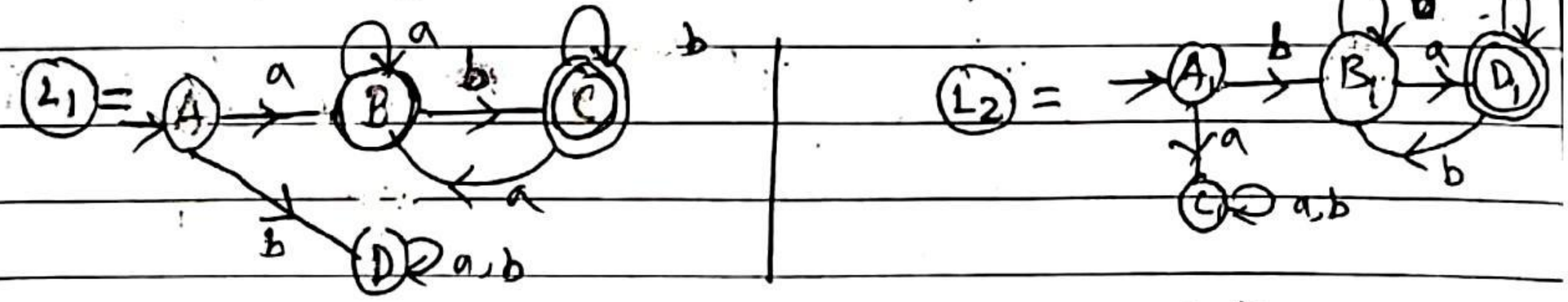
(I) Union (II) Concatenation (III) cross product (IV) Complementation. (V) Reversal.

(1) Union :-

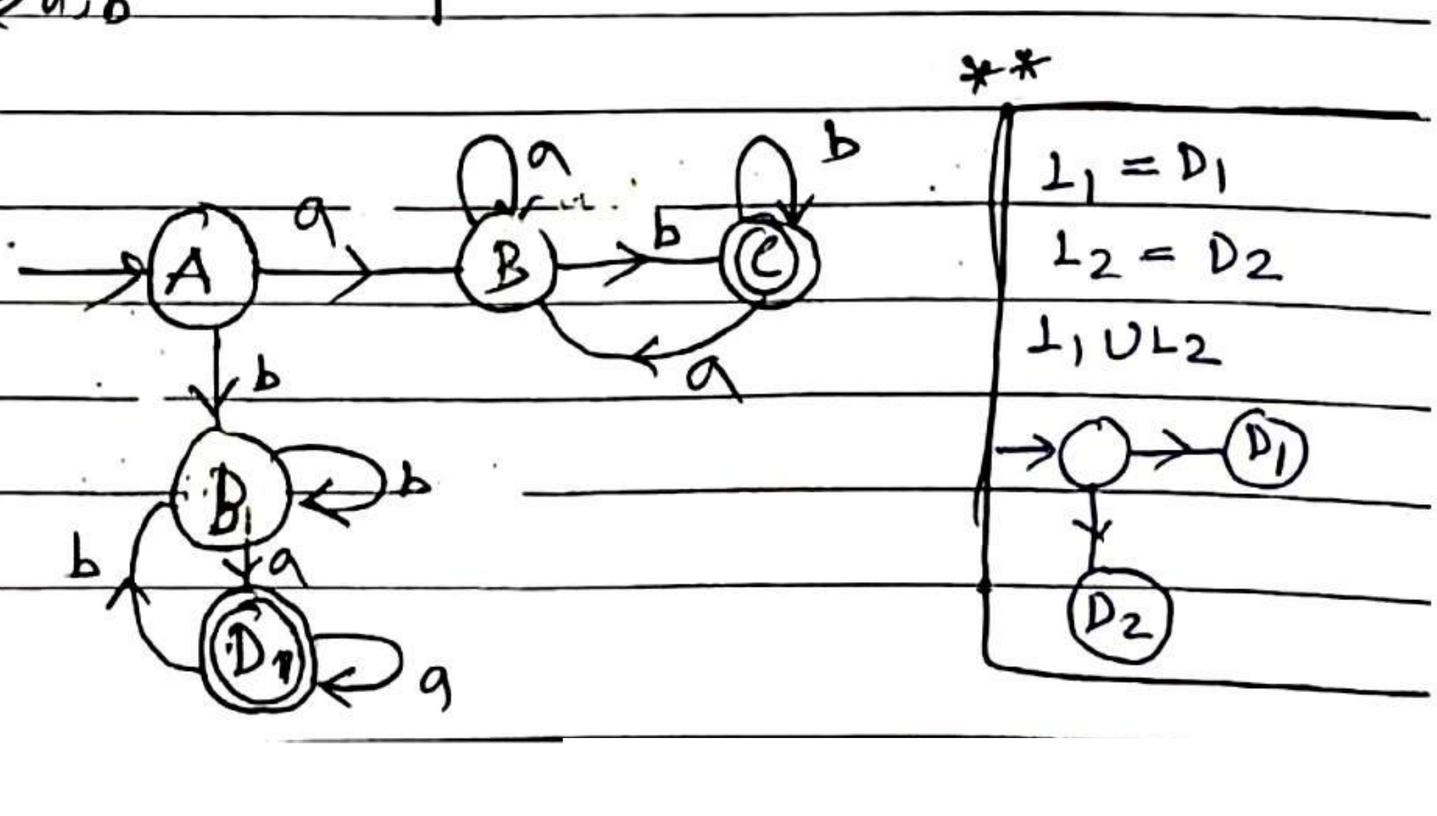
Ex- starts and ends with different symbol.

→ $L_1 = \{ab, aab, aba, abb, \ldots\}$

$L_2 = \{ba, baa, bba, \ldots\}$



$L_1 \cup L_2$ =



② Concatenation — $L_1 = D_1$, $L_2 = D_2$ / $L_1 \cdot L_2 = D_1 \cdot D_2$.

→ starting with 'a' and ending with 'b'.

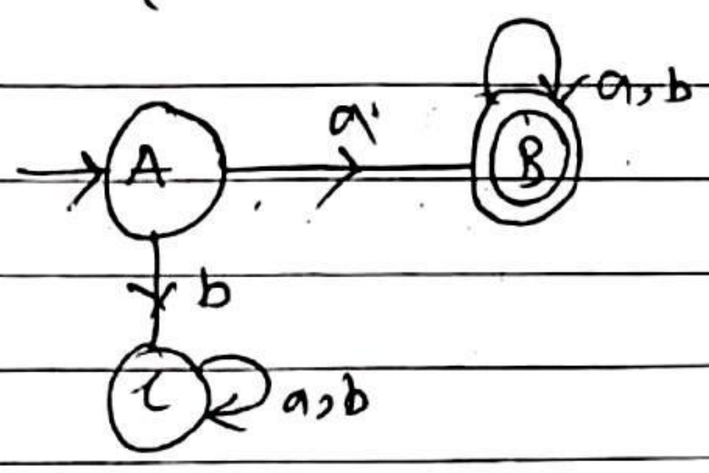
(starting with a)

$L_1 = \{a, aa, ab, aaa \cdots\}$

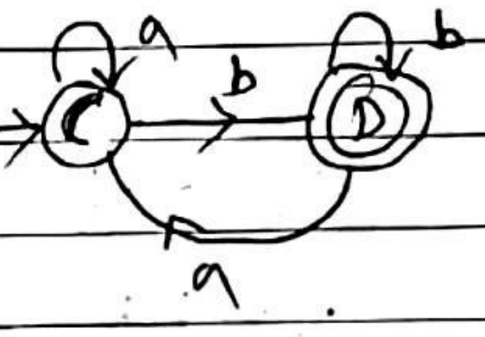
Ending with b)

$L_2 = \{b, ab, bb, aab, bab, bbb, \cdots\}$

$L_1 =$

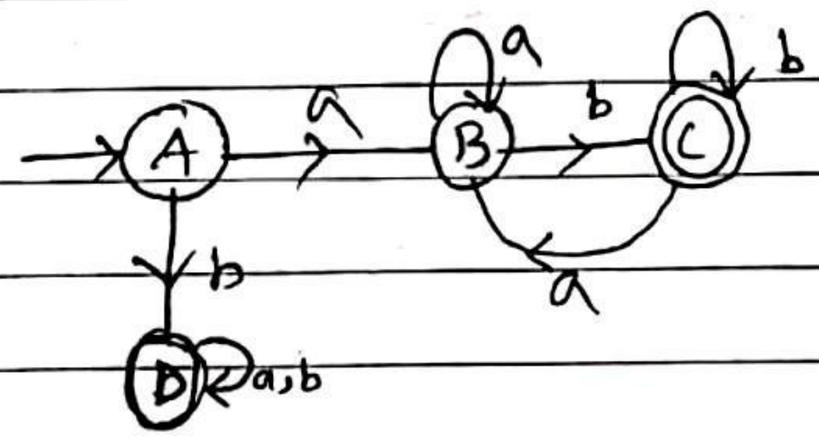


$L_2 =$



Concatenation,

$L_1 \cdot L_2 =$



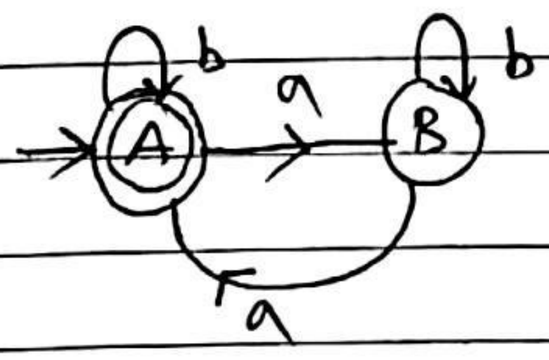
③ Cross product methode —

→ even no of 'a's and even num of b's.

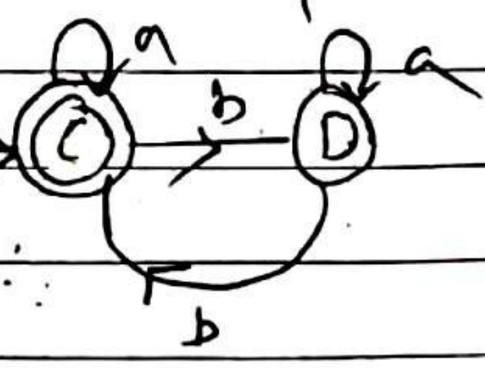
$L_1 = \{aa, aab, aaaa, aaaaaa \cdots\}$

$L_2 = \{bb, bba, bbbb, bbbbbb \cdots\}$

$L_1 =$



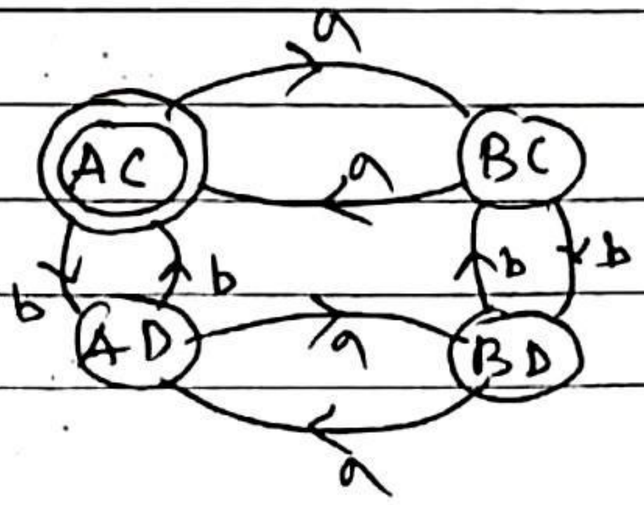
$L_2 =$



$L_1 \times L_2$

$= \{A, B\} \times \{C, D\}$

$= \{AC, AD, BC, BD\}$



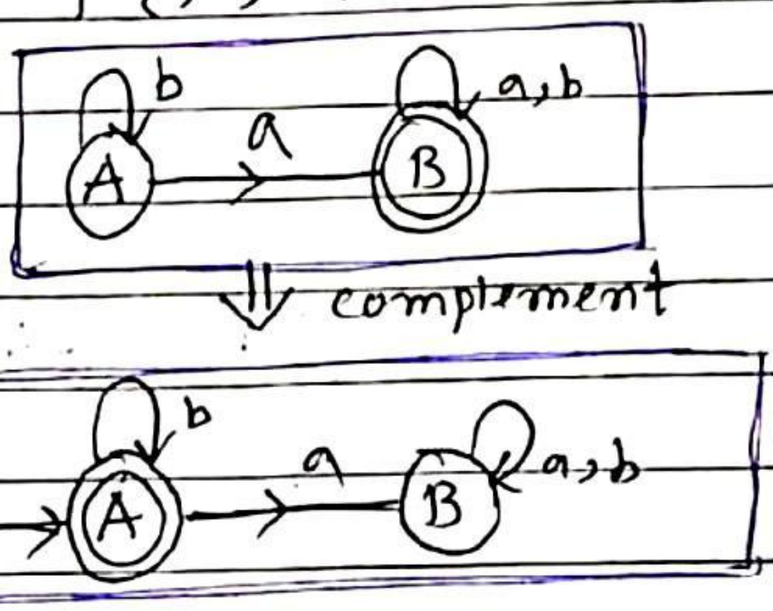
④ Complement :-

→ Does not contain 'a'.

$L_1 = \{\text{containing 'a'}\}$

$= \{a, aa, ab, ba, aaa \cdots\}$

$\bar{L}_1 = \{\epsilon, b, bb, bbb, bbbb \cdots\}$



⑤ Reversal :

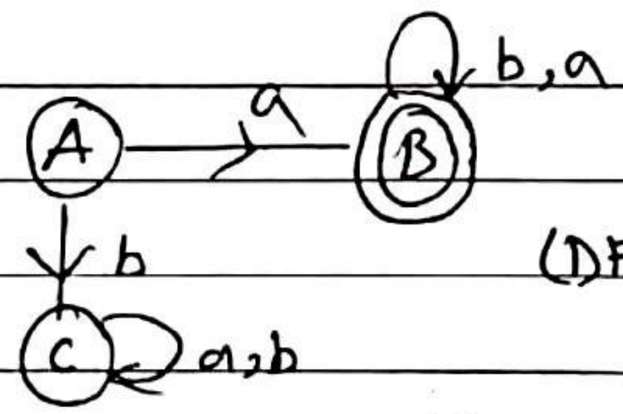
→ $aWb \Leftrightarrow bWa$.

$L_1 = \{\text{start with a}\}$

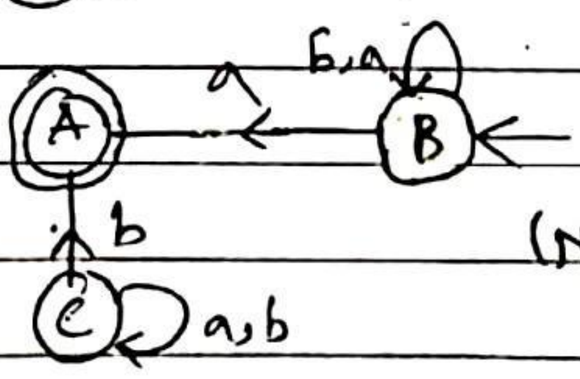
$= \{a, aa, ab, aaa, aba, aaaa \cdots\}$

$L_1^R = \{a, aa, ba, aaa, aba, aaaa \cdots\}$

$L_1 =$



$L_1^R =$



**

$L_1 \rightarrow (DFA)^R \rightarrow L_1^R \rightarrow$ NFA / DFA

(30) Ex-30

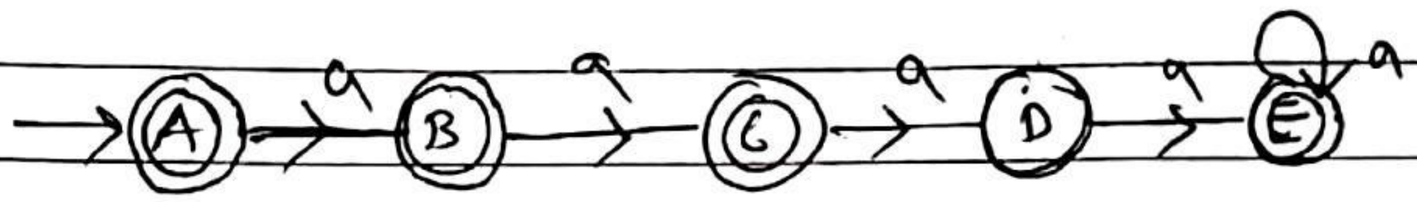
Construct a minimal DFA over $\{a\}$.

1. For $\{a^n / n \geq 0, n! = 3\}$
2. For $\{a^n / n \geq 0, n! = 2, n! = 4\}$.

→ $\Sigma = \{a\}$

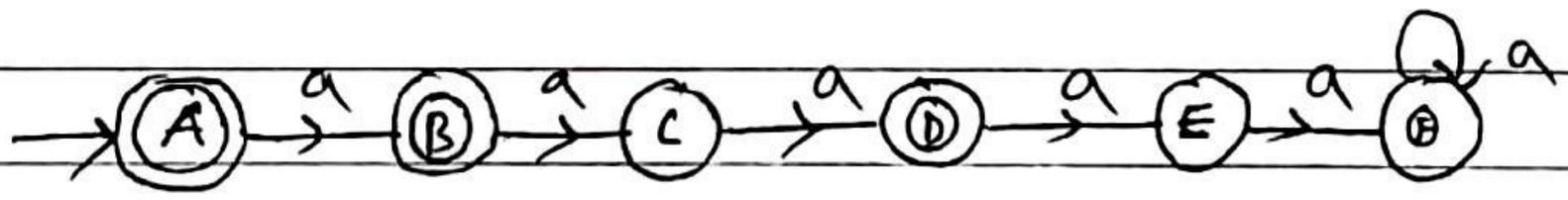
(1) $L = \{a^n / n \geq 0, n! = 3\}$

$L = \{\epsilon, a, aa, aaaa, \cdots\}$



(2) For $L = \{a^n / n \geq 0, n! = 2, n! = 4\}$

$= \{\epsilon, a, aaa, aaaaa, \cdots\}$



→ In NFA, not need to show death state.

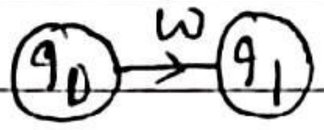
NFA-1

- **NFA (Non-Deterministic Finite Automata):**

→ Every DFA is NFA.

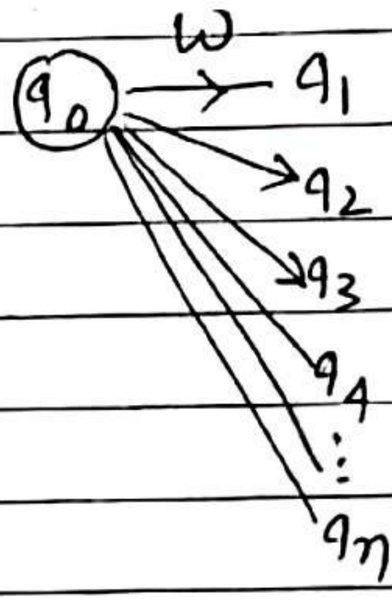
$\delta: Q \times \Sigma \to Q$

DFA



$\delta: Q \times \Sigma \to 2^Q$

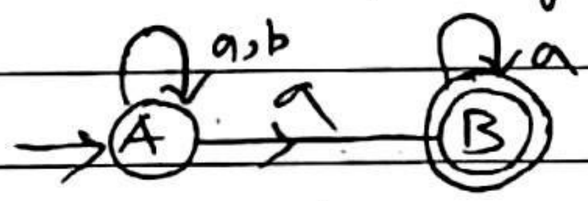
NFA

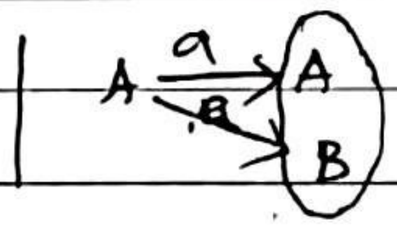


EX-1 Construct a NFA, ~~where~~ which accepts set of all string over {a,b} such that L = {ends with 'a'}

→ L = { a, aa, ba, aaa, baa .... }

State transation Diagram,

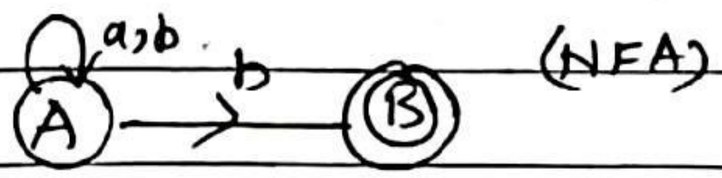




→ [The string 'a' is accepted by NFA, if start with initial state and ~~end~~ if reach at least 1 state is final]

EX-2 L = {ending with b}

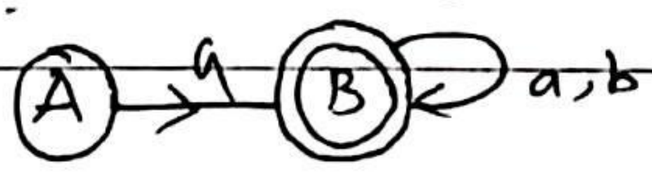
→ L = {b, ab, abb ....}



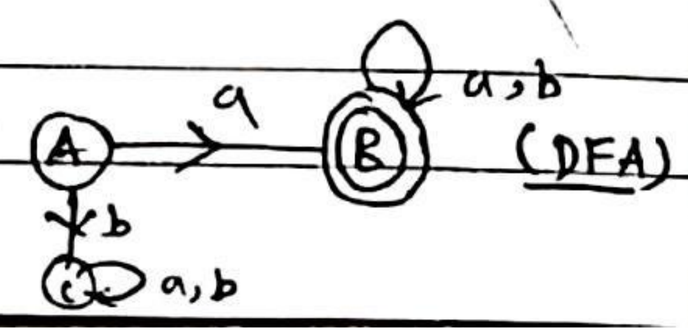
(NFA)

EX-3 L = ~~starting~~ {string starts with 'a'}

→ L = {a, aa, ab - .. }



(NFA)



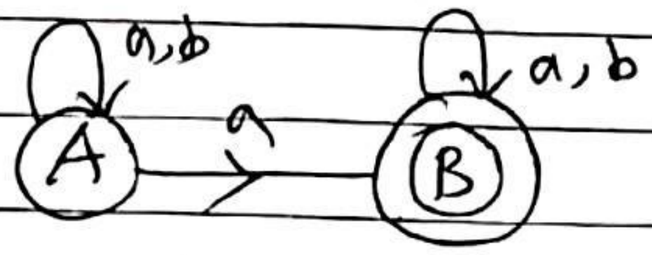
(DFA)

ababa

Ex-3

L = { set of all strings containg 'a' }.

→ $L = \{a, aa, ab, ba, aaa, aab ----\}$

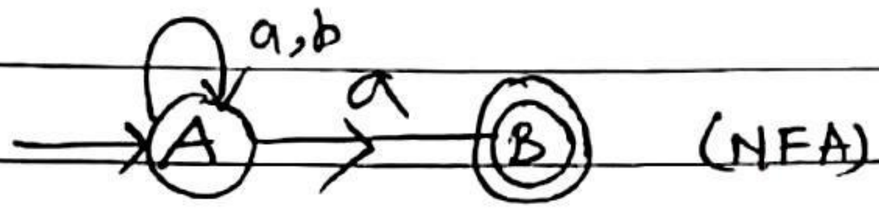


(NFA)

Ex-4

L = { set of all strings ends with 'a' }.

→ $L = \{a, aa, ba, \ldots\}$

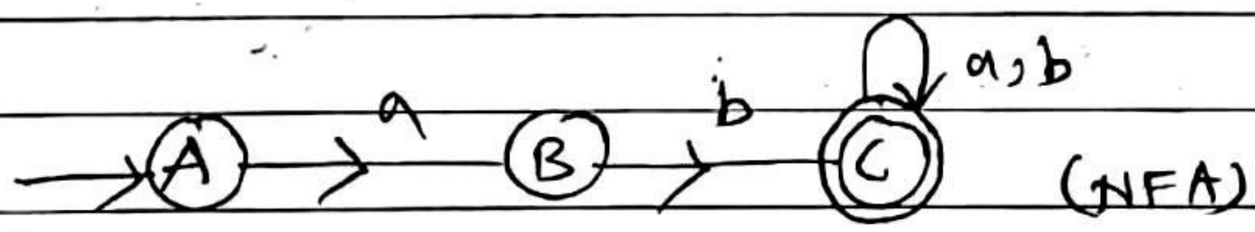


(NFA)

Ex-5

L = { all string Start with 'ab' }

→ $L = \{ab, abb, abab, \ldots\}$

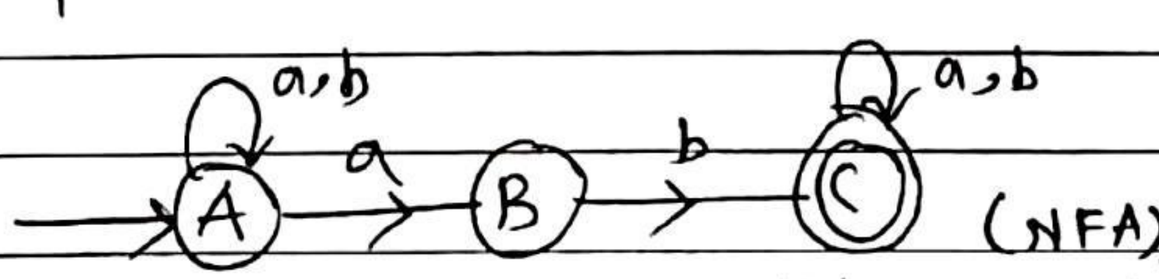


(NFA)

Ex-6

L = { set of all strings contains 'ab' }

→ $L = \{ab, abb, aabb ---\}$

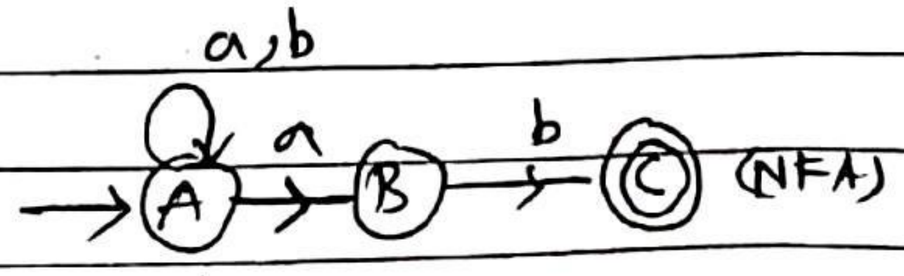


(NFA)

ex-7

L = { ending with 'ab' }

→ $L = \{ab, aab, bab, \ldots\}$

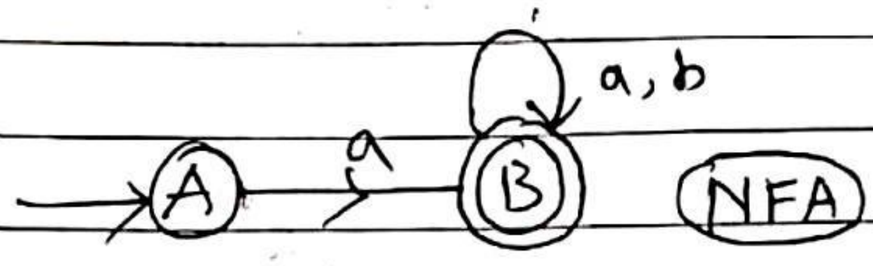


(NFA)

- CONVERSION of NFA to DFA for the Example (1) "all strings start with a".

→ L = {a, aa, ab, aab, abq, ...}

State transcation Diagram of NFA.



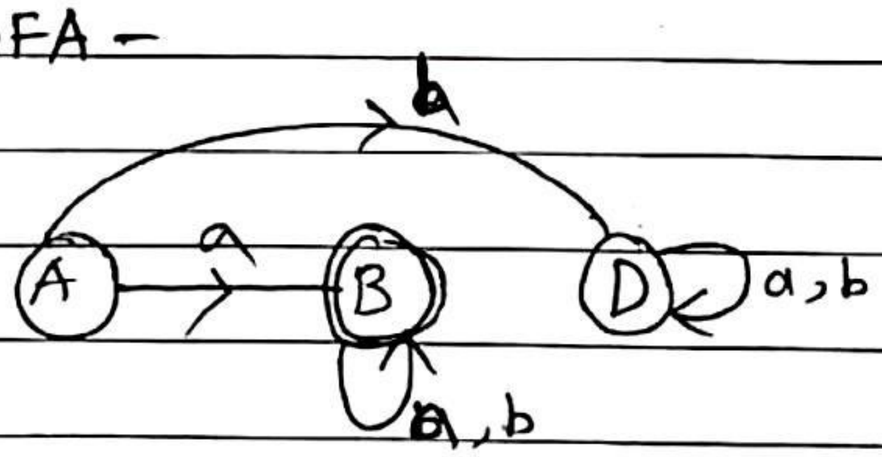
STT of NFA -

	a	b
→ A	B	∅
* B	B	B

STT of DFA -

	a	b
→ A	B	D
* B	B	B
D	D	D

STD of DFA -



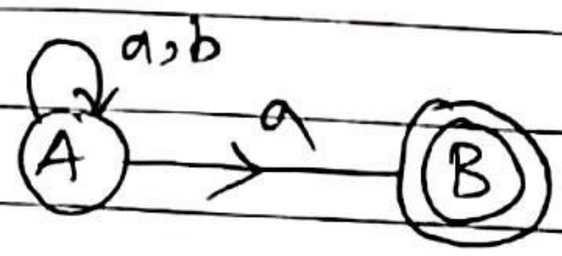
Example-2

Conversion of NFA to DFA:

L = "all strings ends with a".

→ { a, aa, abba, aaa, aba, ... }

ST-Diagram of NFA -



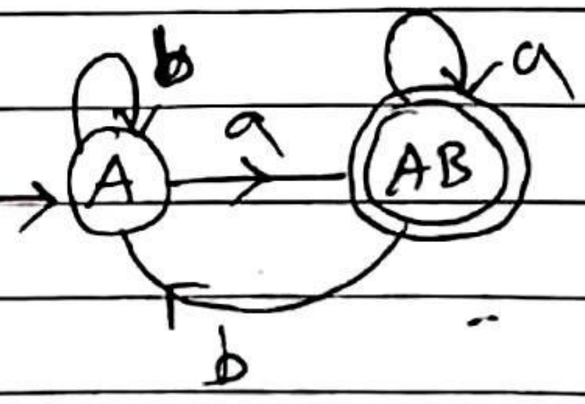
ST-table of NFA

	a	b
A	{A,B}	{A}
B	{∅}	{∅}

ST-Table of DFA

	a	b
→ A	[AB]	[A]
* [AB]	[AB]	[A]

ST-Diagram of DFA



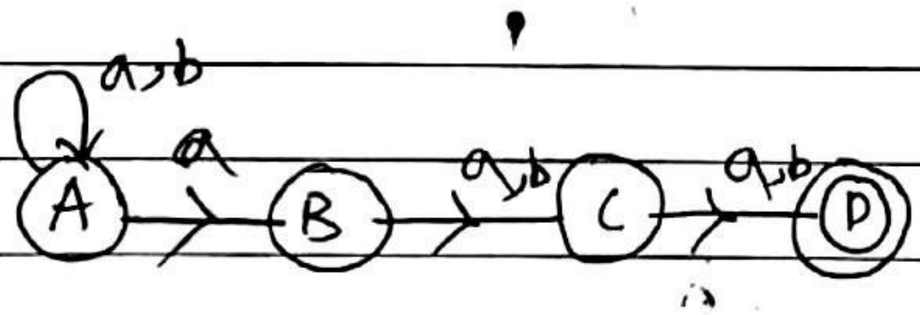
EX-2

Conversion of NFA to DFA,

$\perp = \{$ all strings in which third symbol from R.H.S is 'a' $\}$

$\rightarrow \perp = \{aaa, abb, aba, baaa, ----\}$

ST-D of NFA



ST-Table of NFA

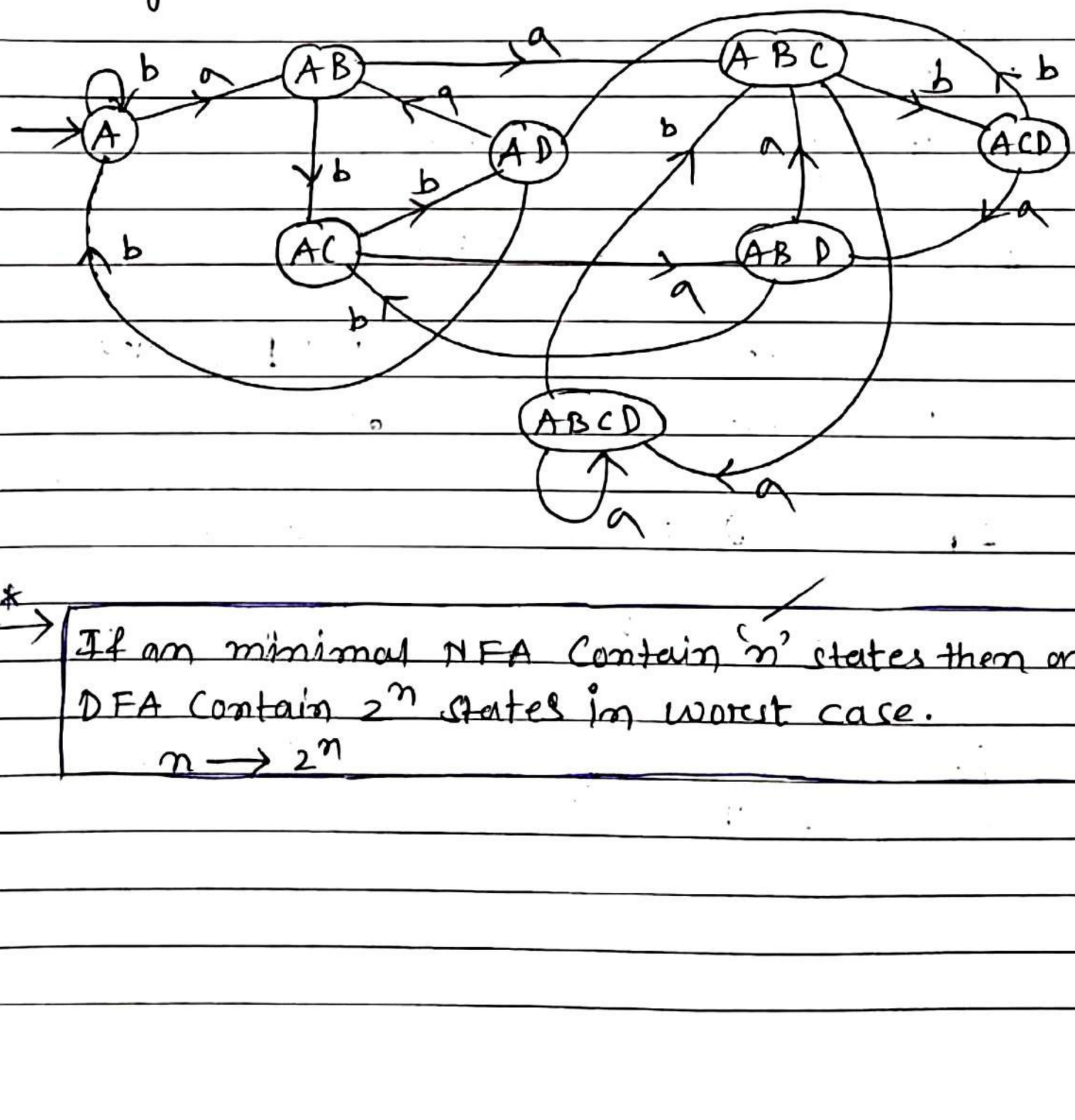
	a	b
→ A	{A,B}	{A}
B	{C}	{C}
C	{D}	{D}
* D	{∅}	{∅}

↓T- Table of DFA from ↓-TT of NFA -

	a	b
→ [A]	[A, B]'	[A]'
[A B]	[A B C]'	[A C]'
[A C]	[A B D]'	[A D]'
*[A D]	[A B]'	[A]'
[A B C]	[A B C D]	[A C D]'
*[A B D]	[A B C]'	[A C]'
*[A C D]	[A B D]'	[A D]'
*[A B C D]	[A B C D]'	[A C D]'

dony state symbol D

ST-Diagram of DFA from above STT of DFA -

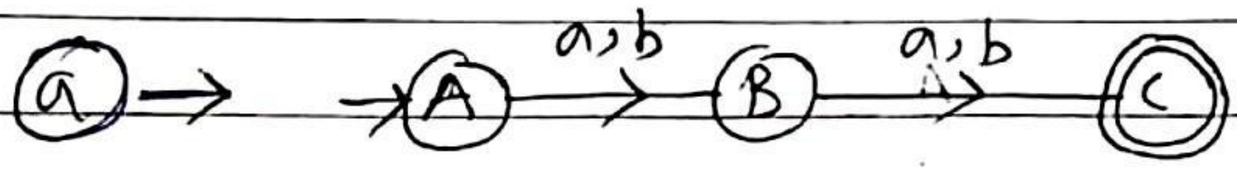


*** → If an minimal NFA Contain 'n' states then an DFA Contain 2^n states in worst case.

$n \longrightarrow 2^n$

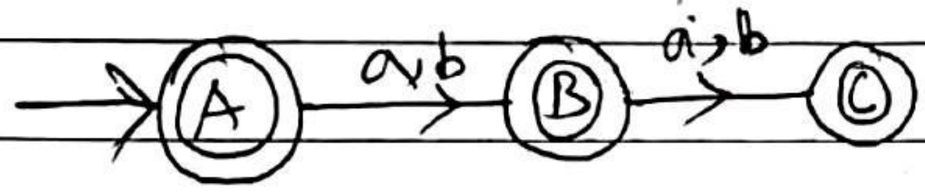
• [Ex-] NFA for string of length -

a) exactly 2. b) at most 2 c) atleast 2

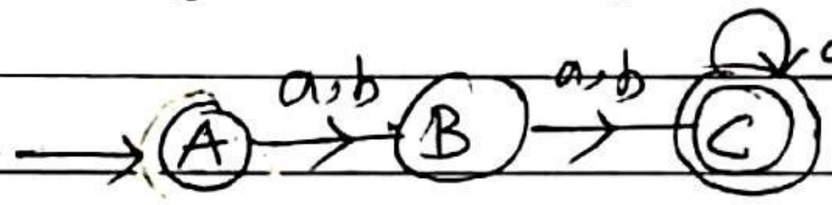


$L = \{aa, ab, ba, bb\}$

(b) → $L = \{\epsilon, a, b, aa, ab, ba, bb\}$



(c) → $L = \{aa, ab, ba, bb, aaa, \ldots\}$



**

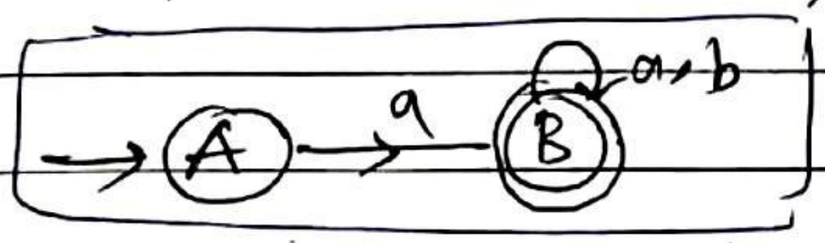
→ If we have ~~a~~ string of length 'n' then for NFA it is going to be 'n' states.

• COMPLEMENTATION of NFA -

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

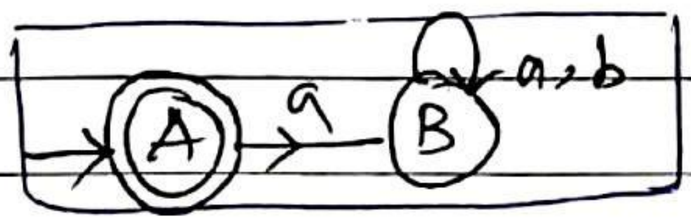
$L = \{$ starts with a $\}$

$L_1 = \{a, aa, ab, \ldots\}$



Then

$I =$



$L \neq I$ (in NFA)

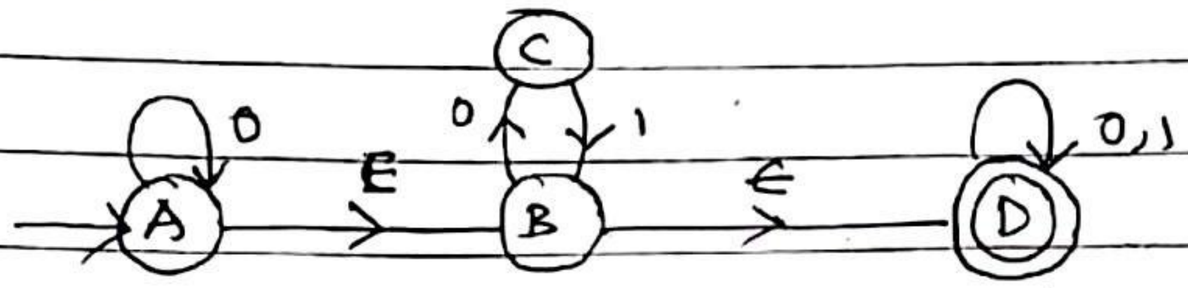
• [Q-1] What is the language accepted by the complement of NFA.

→ $\{\epsilon\}$.

• [Q-2] What is the complement of language accepted by NFA.

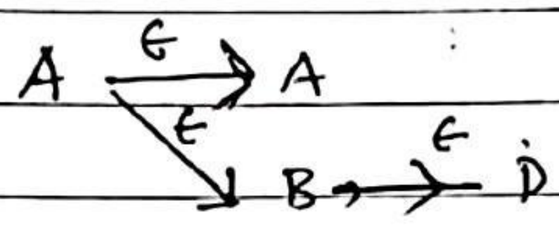
→ $\{\epsilon, b, bb, bbb, \ldots, ba\}$.

- **Epsilon NFA :-** (ϵ-NFA)



→ For ϵ-NFA : $Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

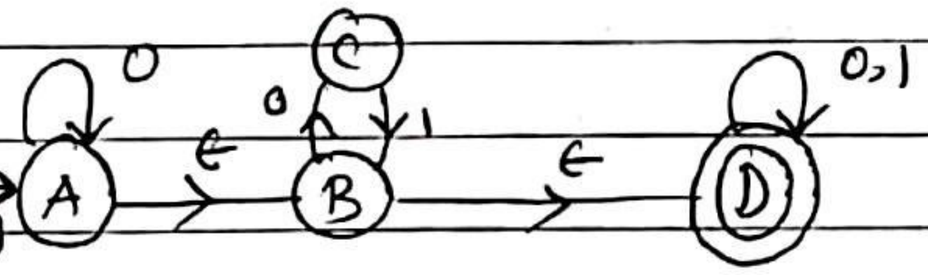
→ ϵ-closure (A) = {A, B, D}



Ex-1

- **Conversion ϵ-NFA to NFA :**

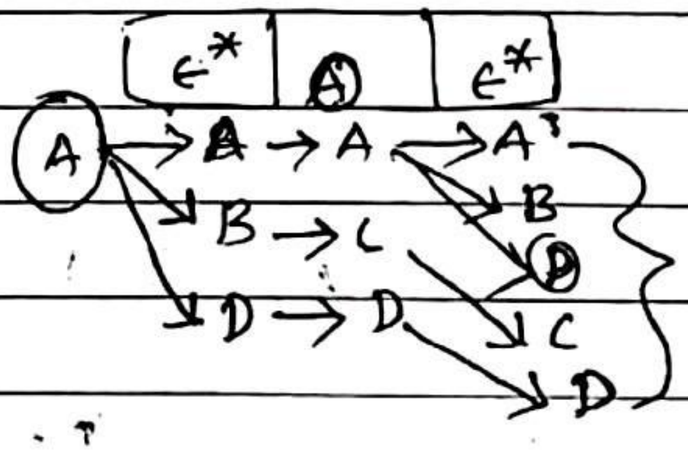
[State transition Diagram of ϵ-NFA]

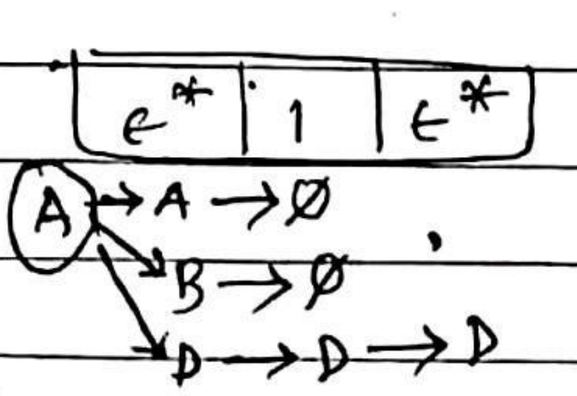


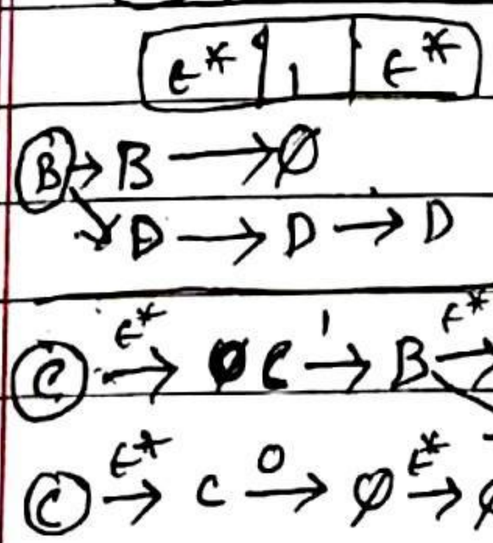
⇓ conversion NFA

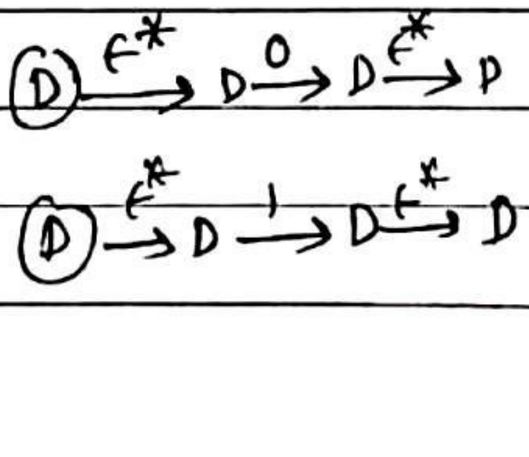
State transition table of NFA :

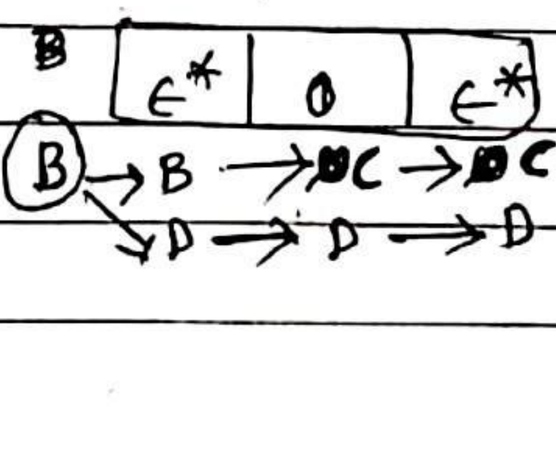
	0	1
A	{A, B, C, D}	{D}
B	{C, D}	{D}
C	{∅}	{B, D}
D	{D}	{D}



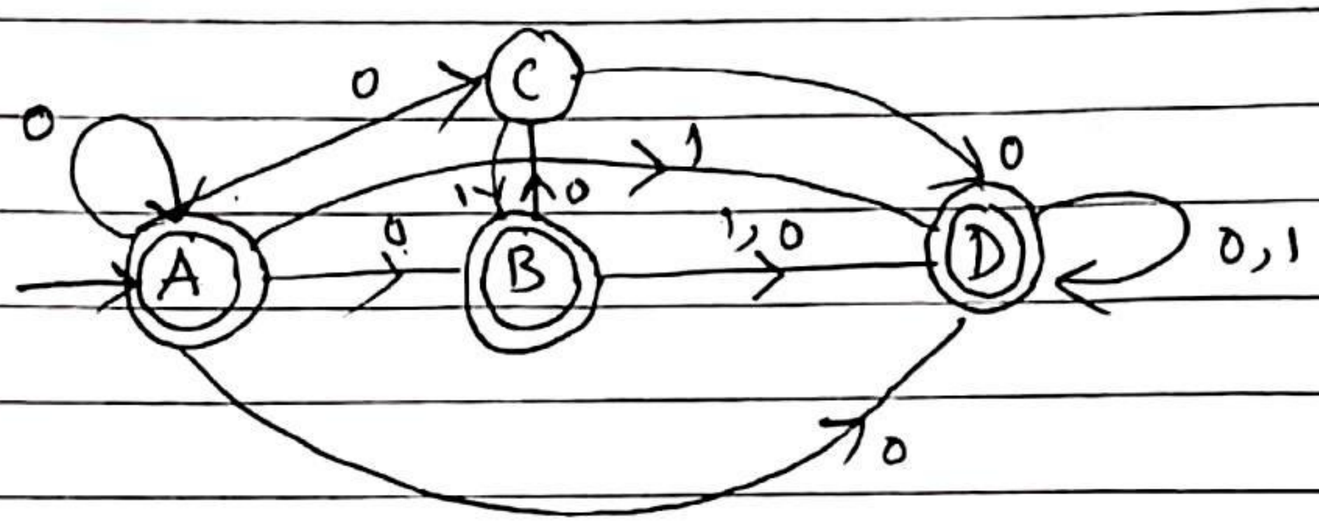








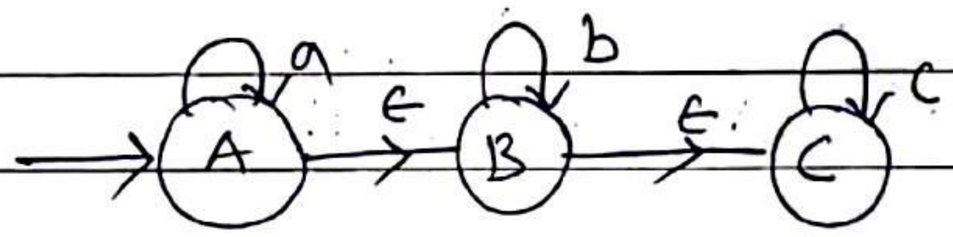
from the state ~~table~~ transaction Diagram - ~~we~~



→ 'A' is going to 'D' by seeing 'ϵ' so 'A' will be the finish state.
→ 'B' is going to 'D' " " 'ϵ' so 'B' " " " .

EX-2

conversion ϵ NFA ⟶ NFA.

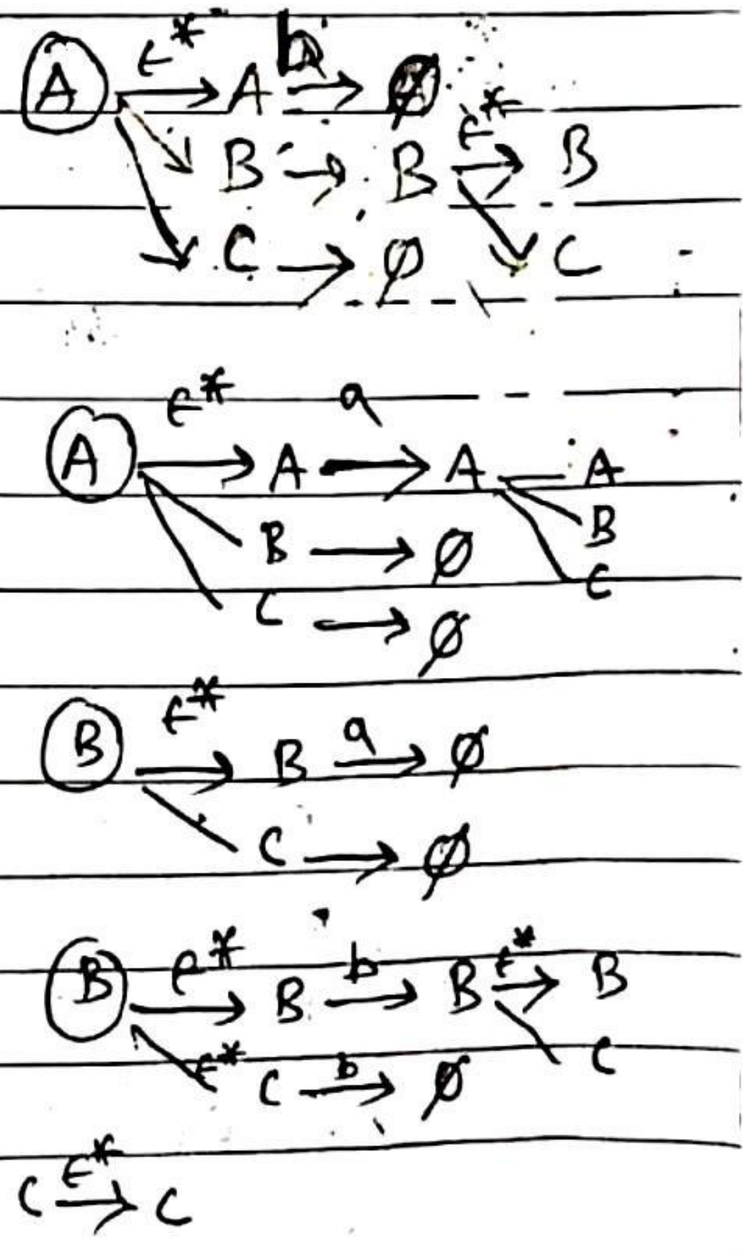


..... (ϵ NFA)

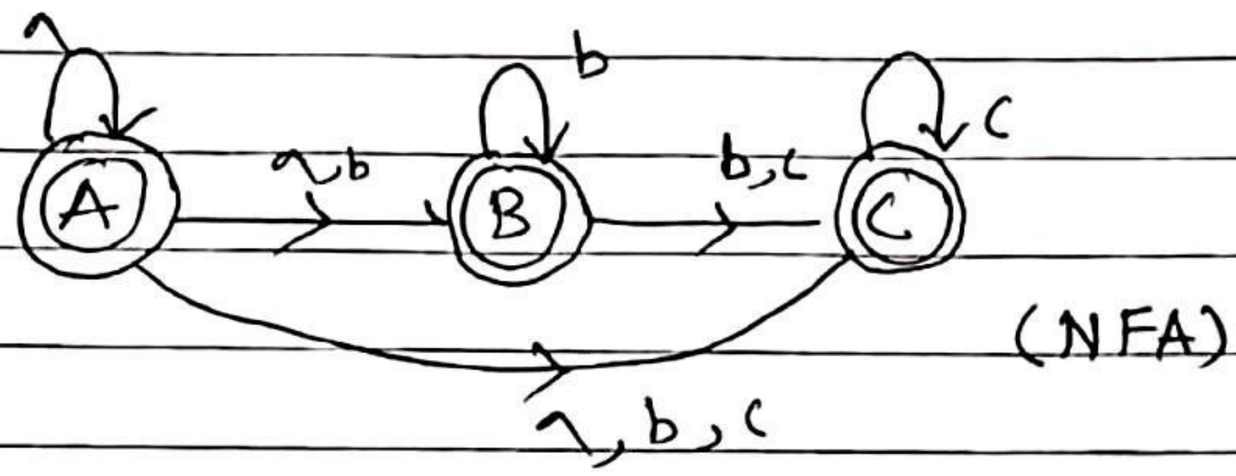
↓ conversion.

transaction table :-

		a	b	c
→	A	{A, B, C}	{B, C}	{C}
	B	{∅}	{B, C}	{C}
	C	{∅}	{∅}	{C}



State transaction Diagram ~~for~~ from state transation table:



(NFA)

→ all the DFA, NFA and ϵ-NFA are equal in power.

Minimisation of DFA:

** → Two state called equivalent,

(P,Q) equivalent,

$\delta(P,W) \in F$

$\Rightarrow \delta(Q,W) \in F$

OR

$\delta(P,W) \notin F$

$\Rightarrow \delta(Q,W) \notin F$

When,

length of string $|W|=0$, 0 equivalent.

$|W|=1$, 1 equi.

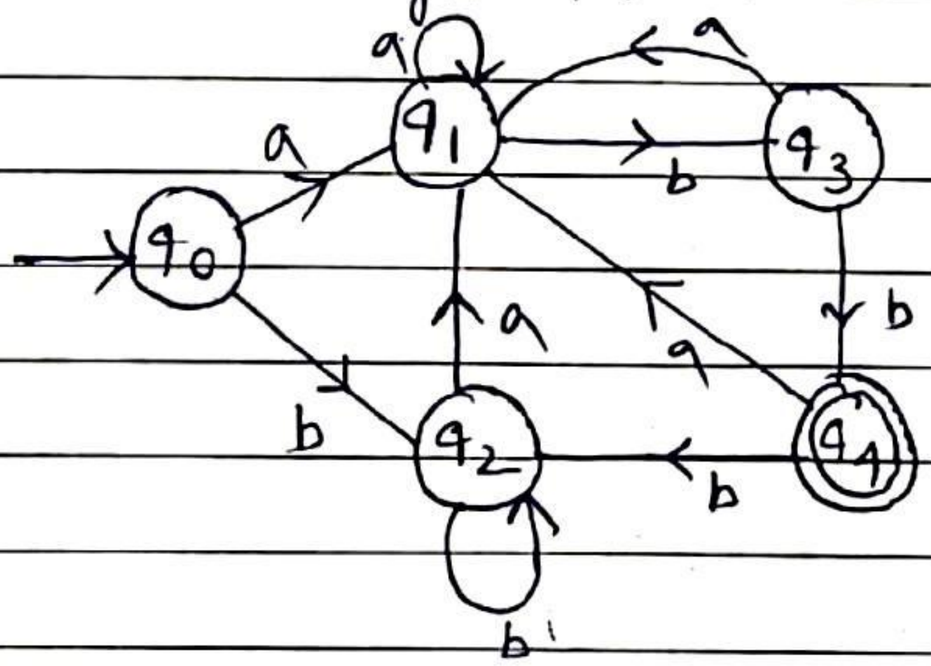
$|W|=2$, 2 equi.

⋮

$|W|=n$, n equivalent.

Example - 1

Minimise the given DFA -



→ Step-1 : Identify the Initial state and final state.

Step-2 : Delete all the state that not reachable from initial state. in final stat

Step-3 : Draw State transation table.

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

(1) Find out '0' equivalent State,

$[q_0, q_1, q_2, q_3]$ $[q_4]$

(2) Find out '1' equivalent State,

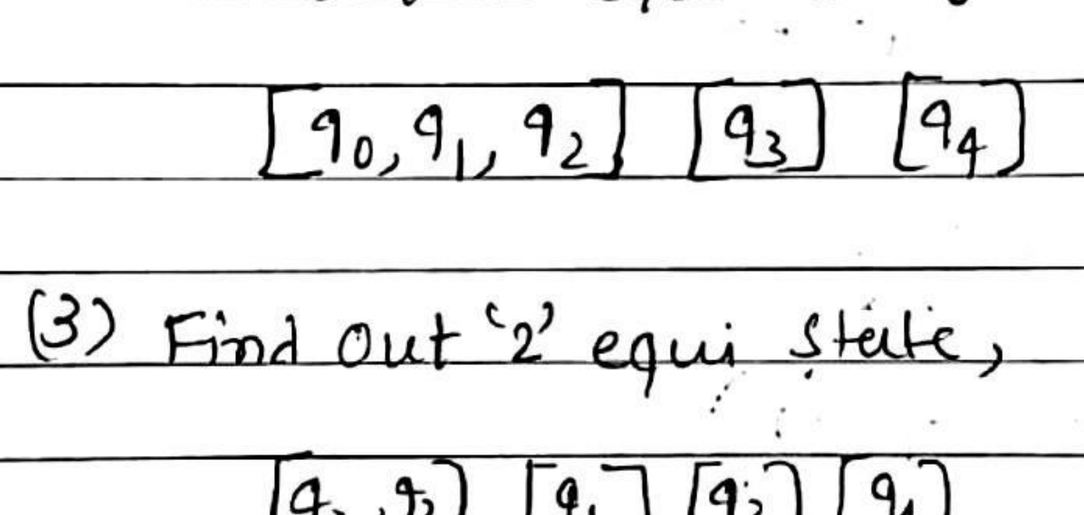
$[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

(3) Find out '2' equi state,

$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

(4) Find out '3' equi state,

$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

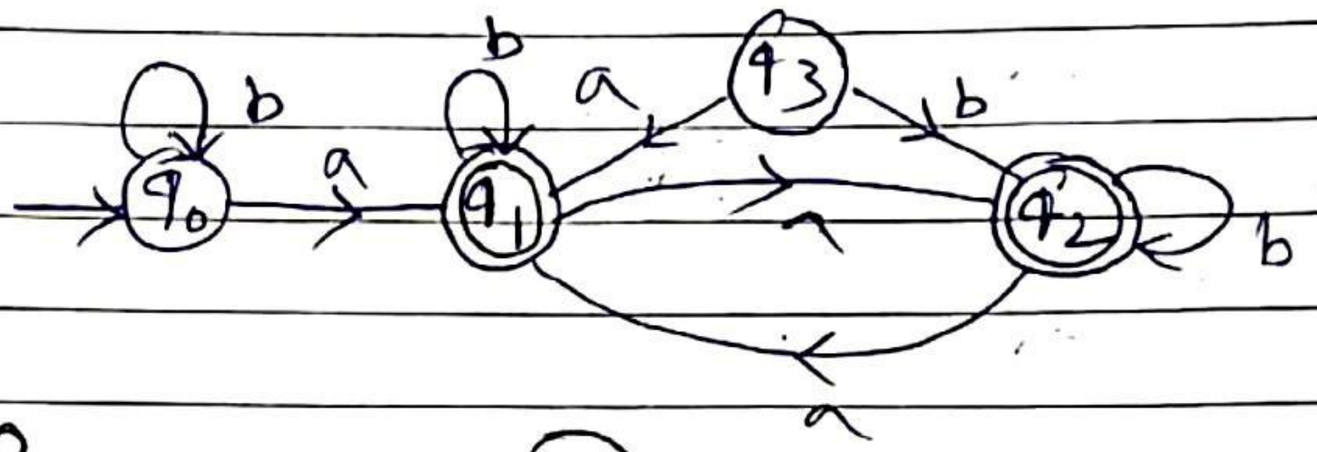


→ finial diagram.

** → State can be decrease, If you find out some state that are equivalent.

- **Example - 2** (gate)

Minimice the given DFA -



→ Remove state (q_3), because we can't reach in q_3 state frome initial State (q_0).

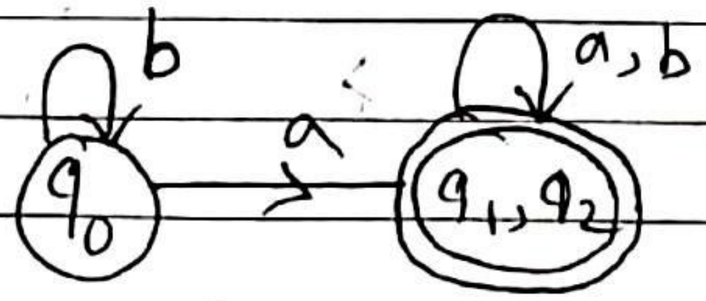
→ State Transation Table -

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

(1) Findout '0' equivalent state, $[q_0]$ $[q_1, q_2]$
Non-F F

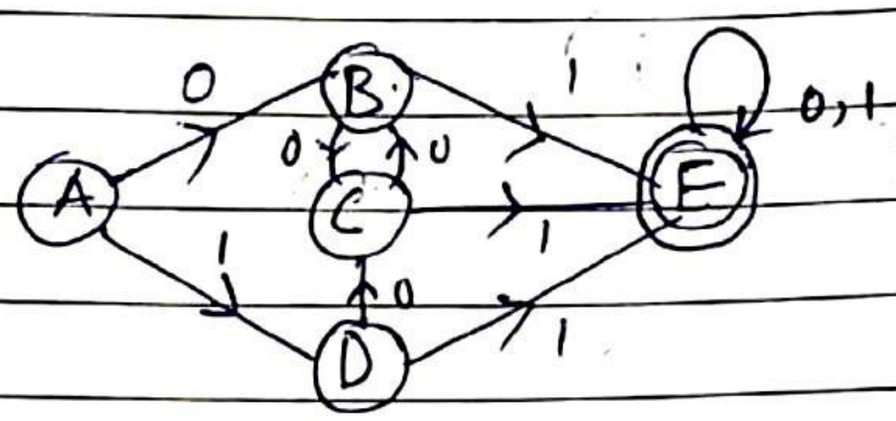
(2) " '1' " " → $[q_0]$ $[q_1, q_2]$

finial STD of DFA -



Example-3

Minimize given DFA -



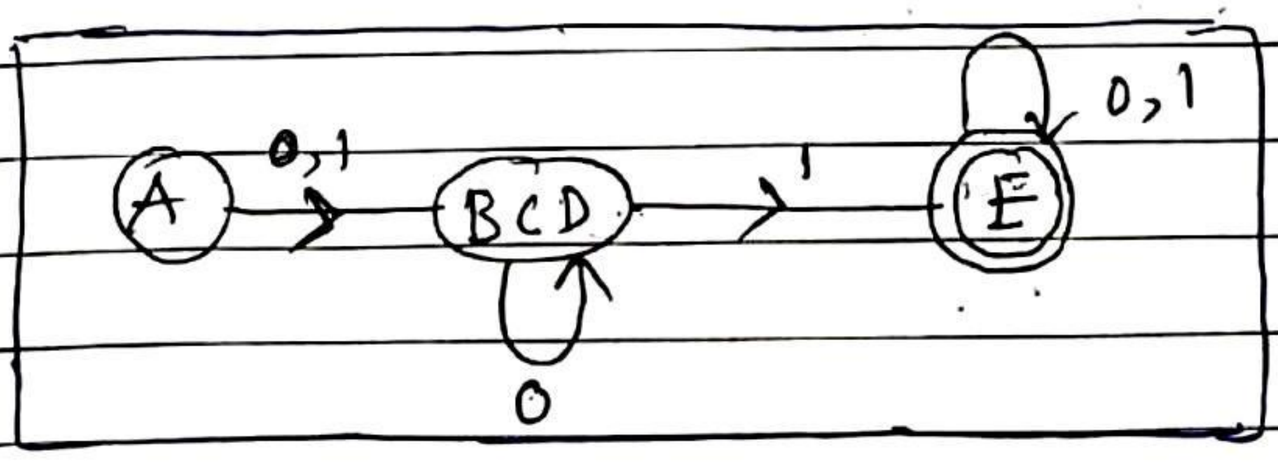
→		0	1
	A	B	D
	B	C	*E
	C	B	*E
	D	C	*E
	*E	*E	*E

'0' equivalent, [A, B, C, D] [E].

'1' equivalent, [A] [B, C, D], [E]

'2' equivalent, [A] [B, C, D] [E]

∴ Final State transition Diagram -



- **Equivalence of DFA and NFA:**

⇒ In contrast to NFA, the DFA has:

→ No ε-transition.
→ For every (q, a) with $q \in Q$ and $a \in \Sigma$ at most one successor state.
→ DFA can simulate the behaviour of NFA by increasing no of state.
→ In DFA for each state, there is at most one transition for each possible Input.

→ In NFA there can be more than one transition from a given state for a given possible Input.

→ All DFA are NFA but not all NFA are DFA.
→ All NDFA (NFA) and DFA have the same power.

→ Processing of an Input string is more time consuming when NFA is used and less time consuming when DFA is used.

- **MOORE AND MEALY MACHINE:**

- **Introduction of moore and mealy machines-**

→ FA with Output Capabilities-
There are two types of machines: Moore machine and mealy machine.

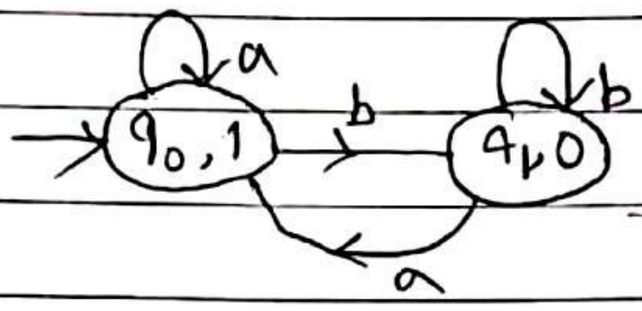
- **Moore machine** → It is a FA in which output is associated with each state.
- **Mealy machine** → It is a FA in which output depends upon both the present Input and present state

n i/p = (n+1) o/p .

n i/p = n o/p

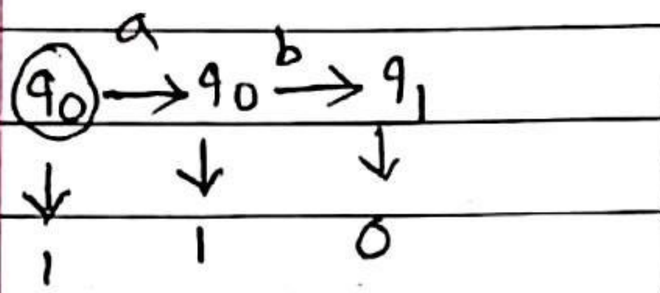
FA with output (O/P).

Moore m/c



$\lambda : Q \to \Delta$

$q_0 \to 1,\ q_1 \to 0$



$(Q, \Sigma, \delta, q_0, \Delta, \lambda)$

$Q \to$ finite set of state.

$\Sigma \to$ i/p alphabet.

$\delta \to$ Transition function

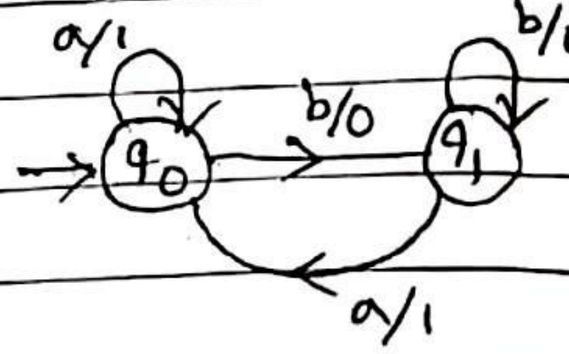
$Q \times \Sigma \to Q$

$q_0 \to$ initial state

$\Delta \to$ O/P alphabet.

$\lambda \to$ O/P function.

Mealy m/c



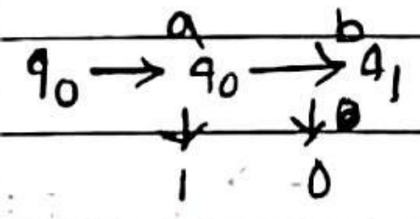
$\lambda : Q \times \Sigma \to \Delta$

$(q_0, a) \to 1$

$(q_0, b) \to 0$

$(q_1, a) \to 1$

$(q_1, b) \to 0$



- **Example - 1**

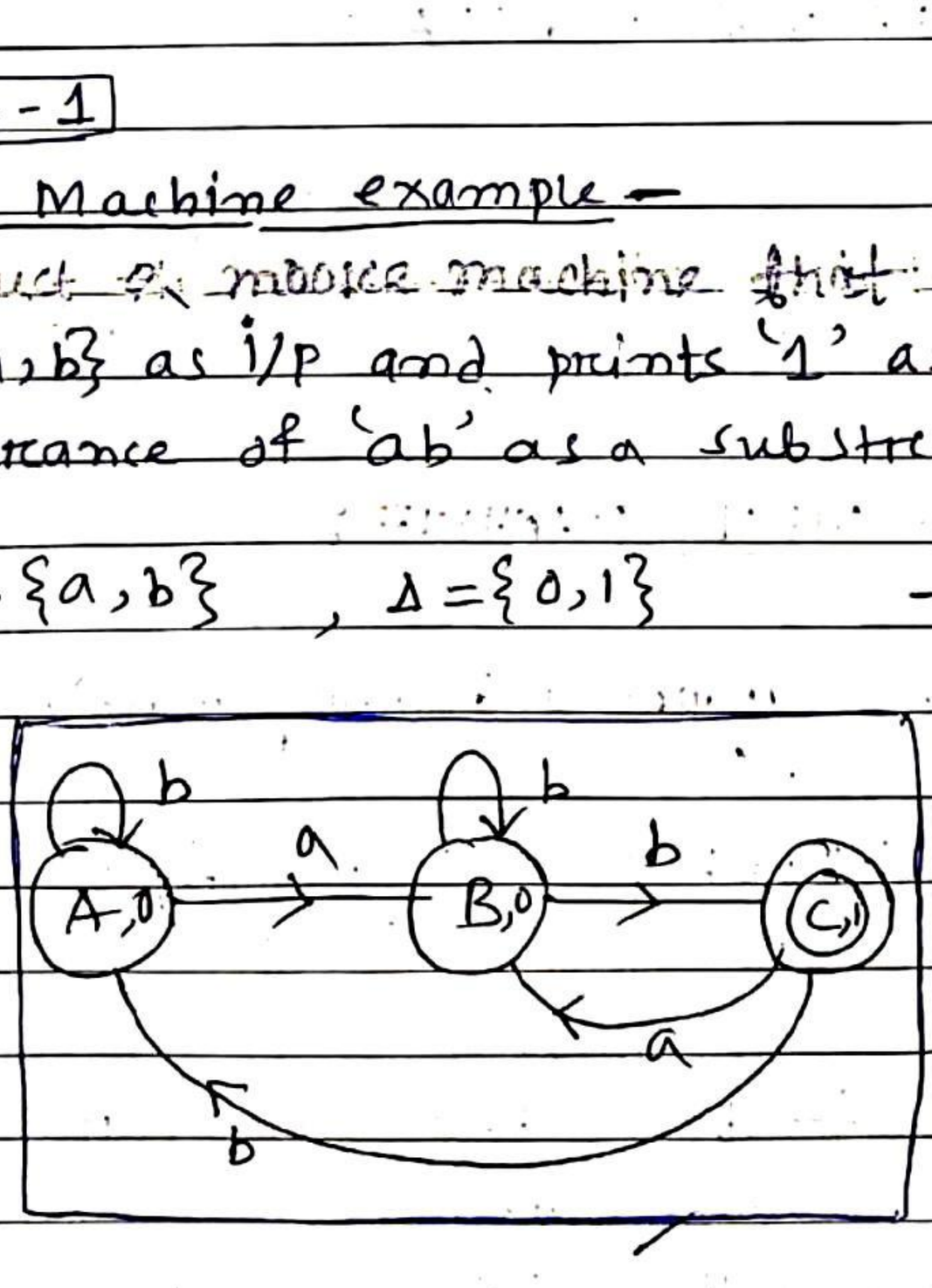
Moore Machine example –

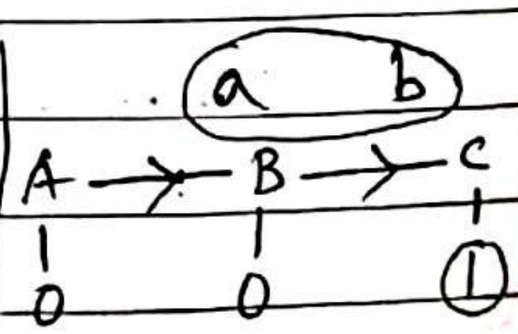
Construct a moore machine that take set of all strings over {a, b} as i/p and prints '1' as O/p for every occurance of 'ab' as a substring.

→ $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$ → for every ab it is going to print '1'.

DFA

(ending with ab)



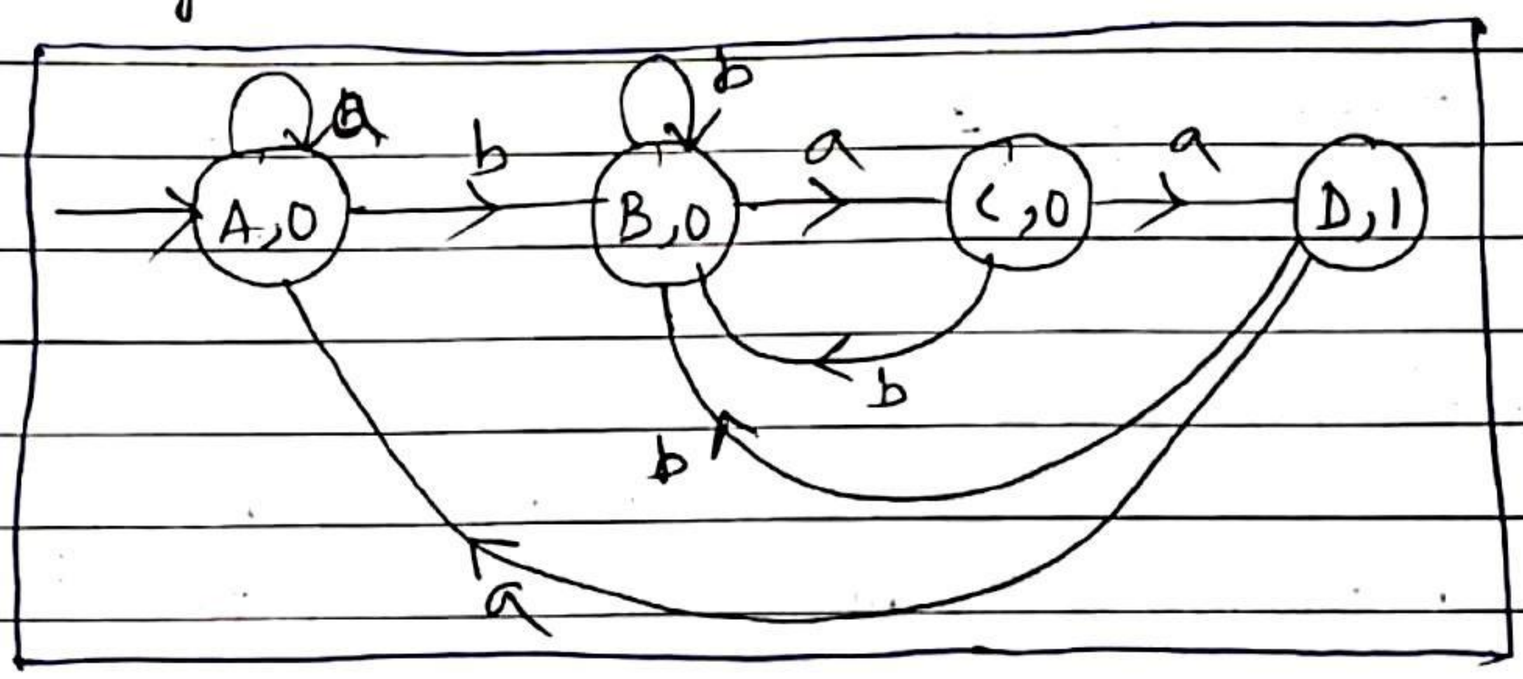


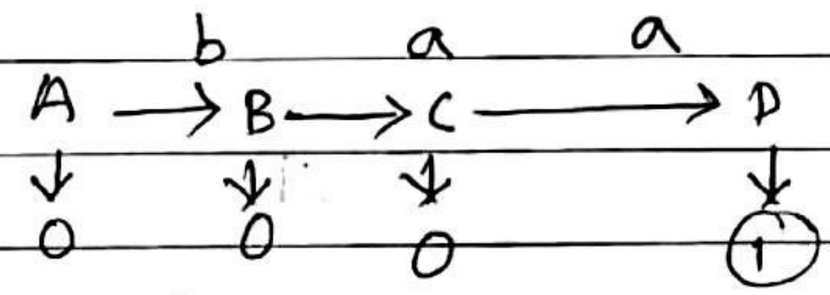
Ex-2

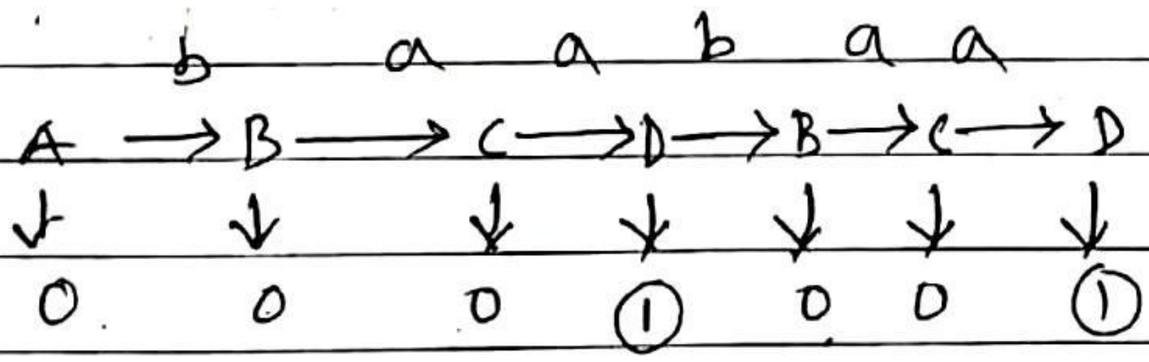
Construct a moore m/c that take set of all strings over {a, b} and counts no of occurrences of string 'baa'.

→ $\Sigma = \{a, b\}$,
$\Delta = \{0, 1\}$

ending with 'baa'.





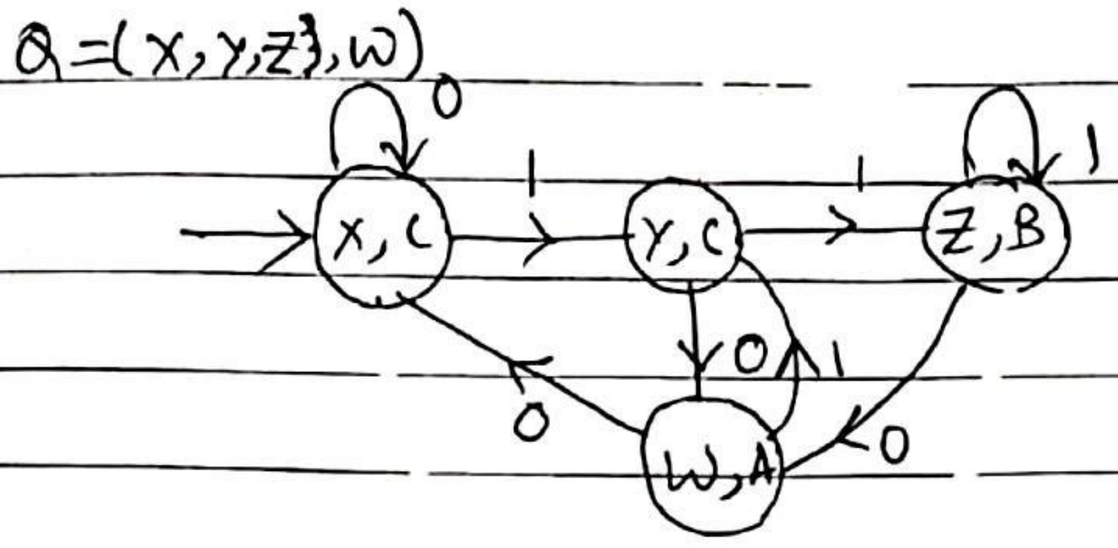


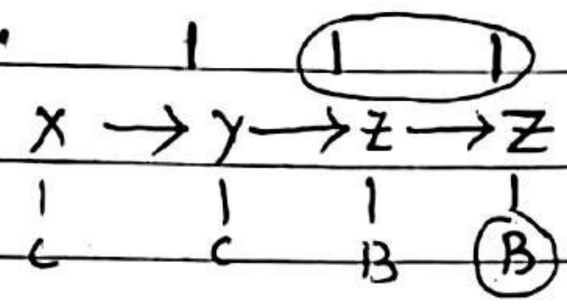
Ex-3

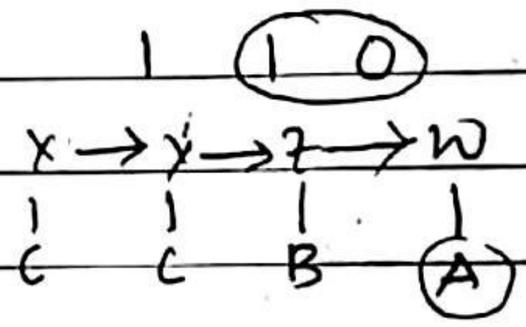
Construct a moore machine that takes set of all strings over {0,1} and produces 'A' as o/p if i/p ends with '10' or produces 'B' as o/p it ends with '1' otherwise produces 'c'.

10 → A
11 → B

→ $\Sigma = \{0, 1\}$
$\Delta = \{A, B, C\}$







Ex-4

Construct a moore machine that takes binary no's as i/p and produces residue modulo '3' as o/p.

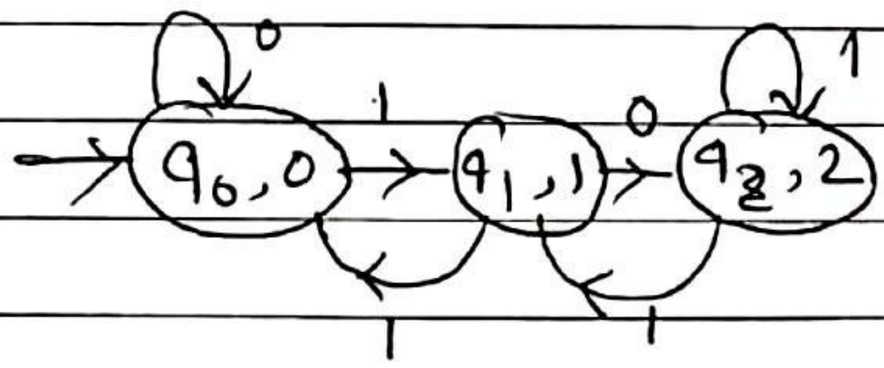
→ $\Sigma = \{0,1\}$

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

ST-Table

	0	1	Δ
→ q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

ST-Diagram



Construct a ~~m/m~~ that moore machine that takes base 4 ~~as~~. no's as i/p and produces residue modulo '5' as o/p.

→ $\Sigma = \{0,1,2,3\}$

$\Delta = \{0,1,2,3,4\}$

ST-Table -

	0	1	2	3	Δ
→ q_0	q_0	q_1	q_2	q_3	0
q_1	q_4	q_0	q_1	q_2	1
q_2	q_3	q_4	q_0	q_1	2
q_3	q_2	q_3	q_4	q_0	3
q_4	q_1	q_2	q_3	q_4	4

Example-5

Mealy machine example.

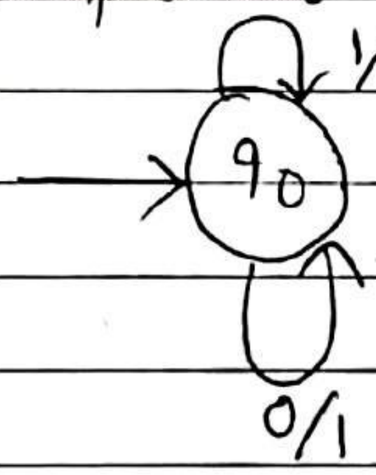
- Construct a mealy m/c that takes binary numbers as input and produces 2's complement of that number as O/P Assume the string is read LSB to MSB and ends. Carry is discarded.

→ For every state, for every input there will be a out-put.

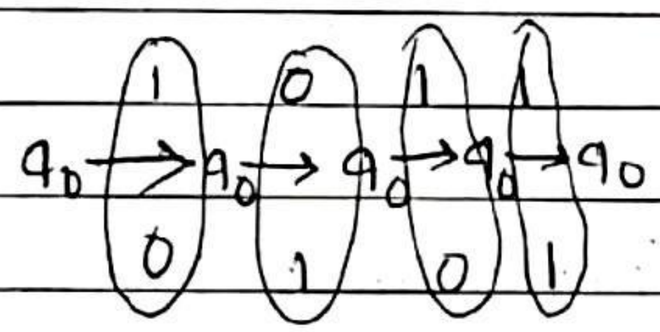
$\Sigma = \{0, 1\}$

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

1's complement -



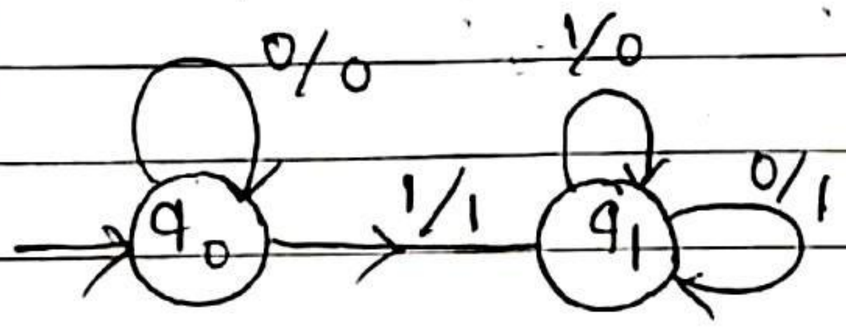
0 1
1 0
+ 1
0 1



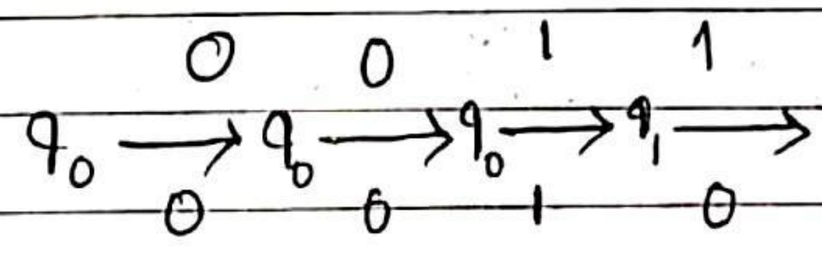
1 1 1 0 1 1 0 0
1's - 0 0 0 1 0 0 1 1
+ 1
2's - 0 0 0 1 0 1 0 0

1 1 1 0 0
1's - 0 0 0 1 1
+ 1
2's - 0 0 1 0 0

2's complement -

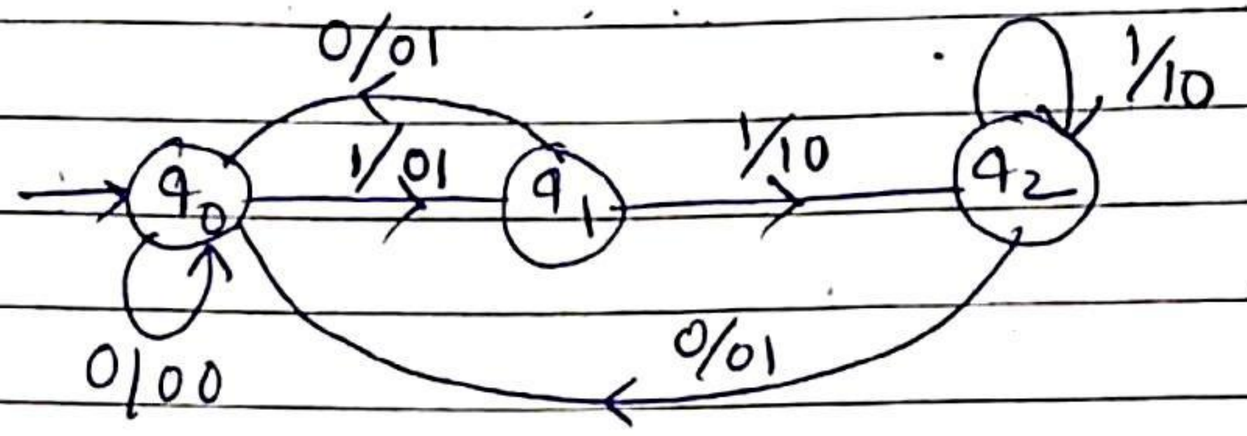


1's - 1 1 0 0
0 0 1 1
+ 1
O/P - 0 1 0 0



Ex-6

What is the O/p produced by the following state m/c



(a) 11 → 01 (b) 10 → 00

(c) Sum of present & previous bits.

(d) Not.

→ (0 + 1 + 0 + 1 + 1 + 0)

$q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0$

01 01 01 10 o/p

- Conversion of Moore machine to Mealy machine.

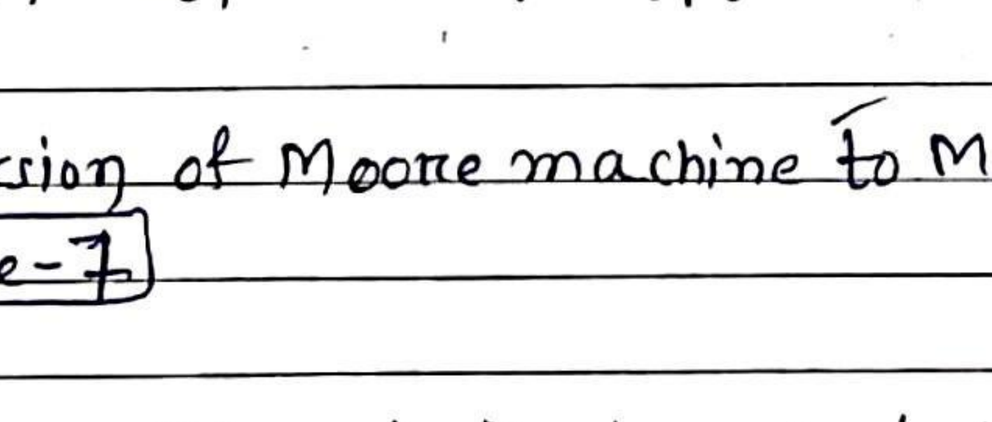
Example-7

→ Moore M/c and Mealy m/c both are same in power.

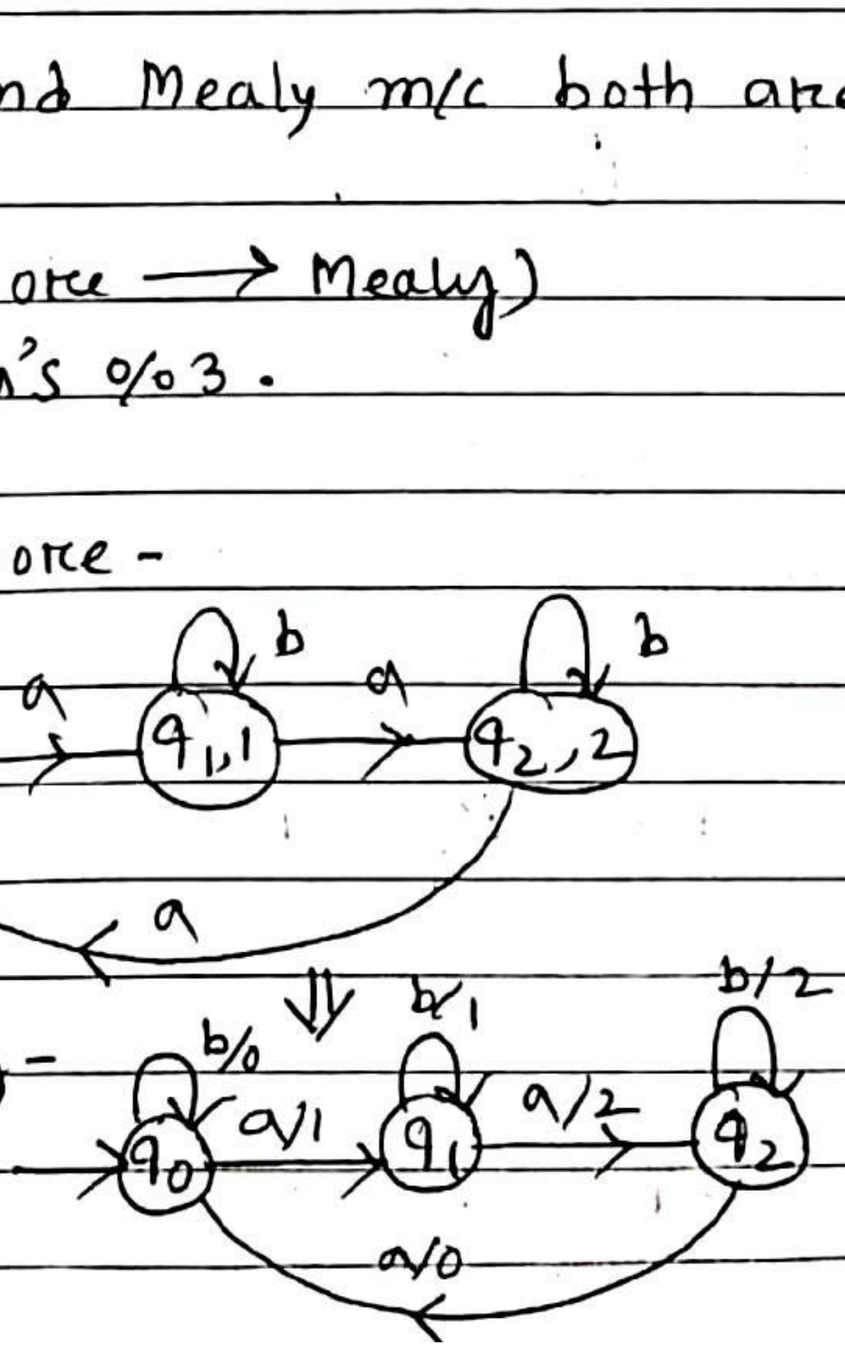
(Moore → Mealy)

Count no of a's %3.

ST-Diagram of Moore -



ST-Diagram of Mealy -



ST-Table of Moore-

	a	b	Δ
q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_0	q_2	2

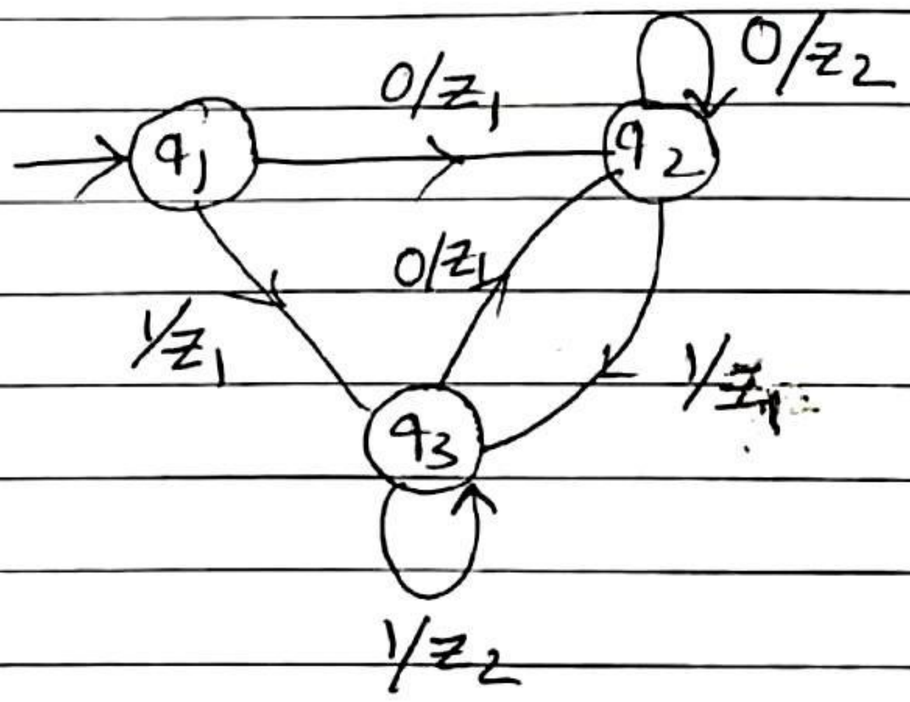
⇒

ST-Table of Mealy-

	a	b
q_0	$(q_1,1)$	$(q_0,0)$
q_0	$(q_2,2)$	$(q_1,1)$
q_2	$(q_0,0)$	$(q_2,2)$

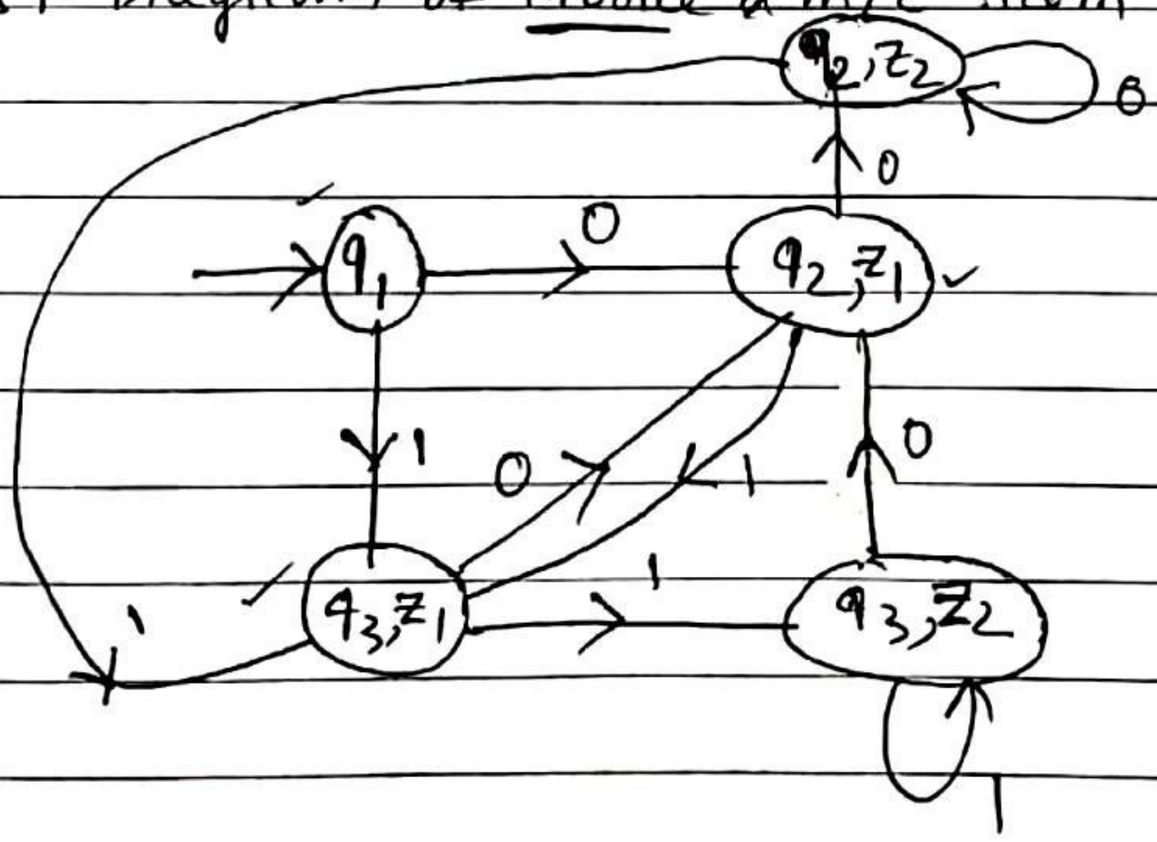
Ex-8 Conversion ~~Moore to~~ Mealy to Moore Machine.

ST-Diagram of Mealy m/c -



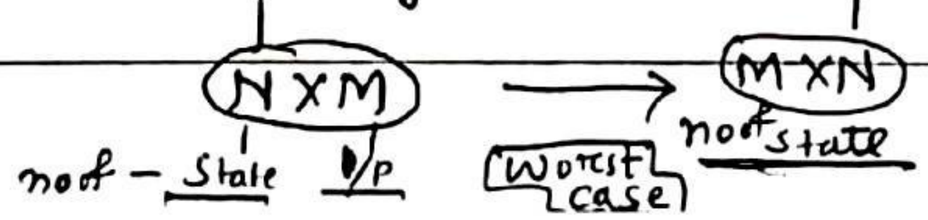
⇓

ST-Diagram of Moore m/c from STD of Mealy machine-

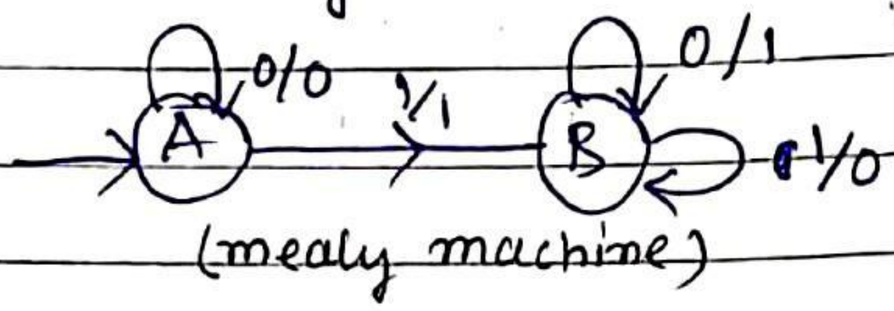


⊛ ⇒ Moore m/c ⟶ mealy m/c. (no of state same)

⊛ Mealy m/c ⟶ moore m/c. (no of state may-increase)



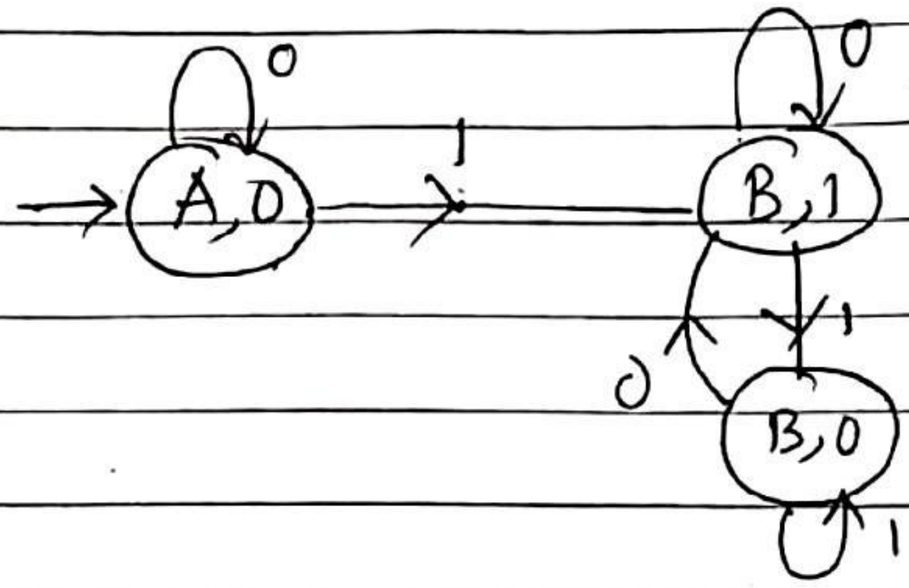
Ex-9 Mealy m/c ⟶ Moore m/c



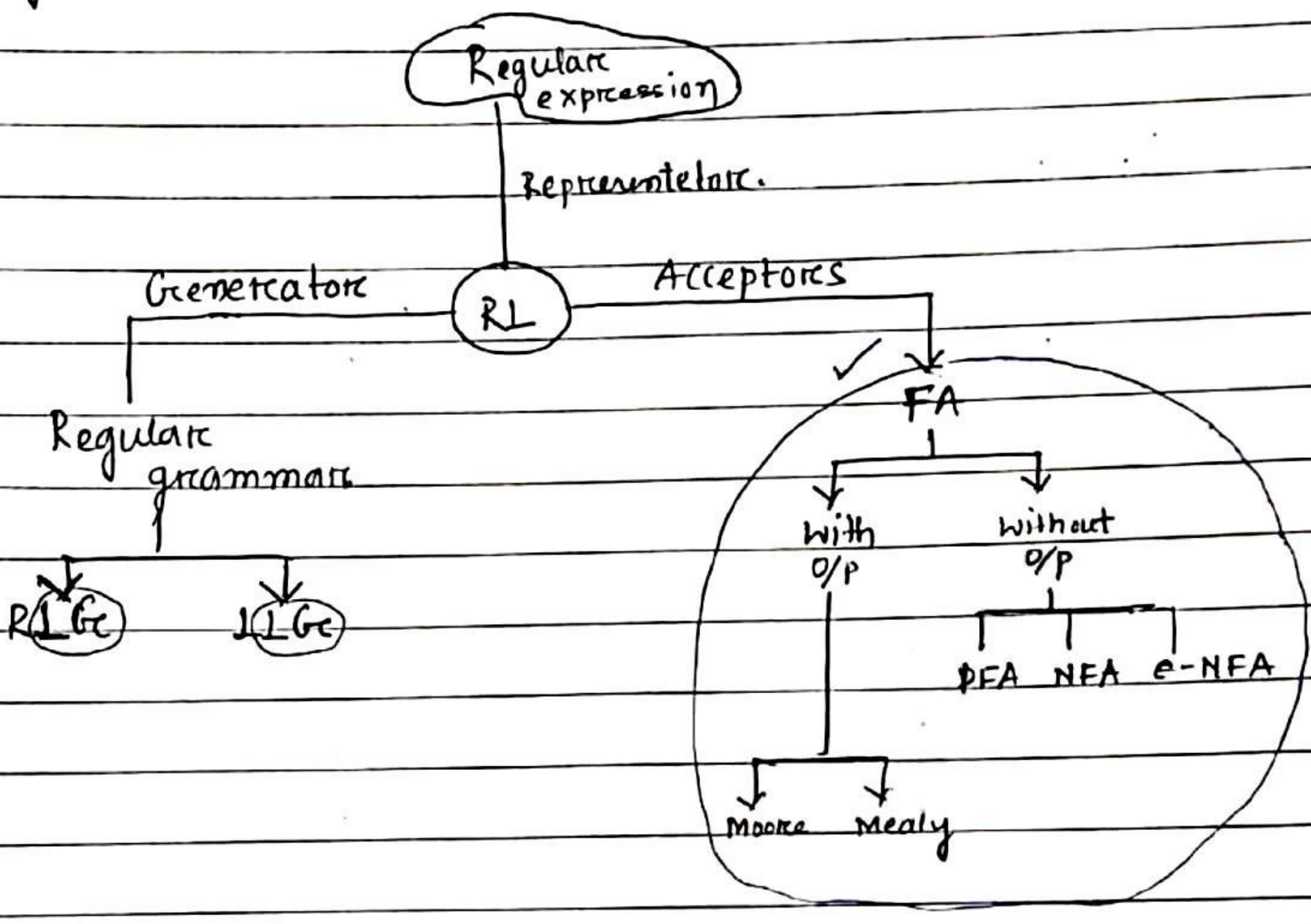
(mealy machine)

In mealy machine -
no state = 2
no of i/p = 2

2×2 = 4 state in Moore m/c in worst case.



• Regular Expression:



→ Regular expression mean to represent certain set of strings in some algebric fashion.

→ Regular expression contain 3- operations:-

(i) + (union)

(ii) . (concatination)

(iii) * (Kleene closure)

A RE over the alphabet Σ is defined as follows -

(a) $\emptyset$, ϵ, $a \in \Sigma$ (primitive)

$\{\}$ $\{\epsilon\}$ $\{a\}$ $\{b\}$.

(b) $r_1 + r_2$, $r_1 \cdot r_2$, r_1^*.

→ A set represented by a regular expression is called Regular set.

Different of regular sets and their Expression —

Regular expression	Regular set
$\emptyset$	$\{\}$
ϵ	$\{\epsilon\}$
a	$\{a\}$
a^*	$\{\epsilon, a, aa, aaa, \dots\}$
a^+	$\{a, aa, aaa, aaaa \dots\}$
$(a+b)^*$	$\{\epsilon, a, b, aa, ab, ba, bb, \dots\}$
$(a+b)$	$\{a, b\}$

Ex-1

$\Sigma = \{a, b\}$, ↑

① Find out regular expression for set of all strings whose length is exactly '2'.

→ $L = \{aa, ab, ba, bb\}$

$= aa + ab + ba + bb$

$= a(a+b) + b(a+b)$

$= \underbrace{(a+b)\ (a+b)}$

length exactly '2'

② Find out RE for set of all strings whose length is exactly '3'.

→ $L = \{aaa, aba, baa, bba, aab, abb, bab, bbb\}$

$= \underbrace{(a+b)\ (a+b)\ (a+b)}$

length exactly '3'

③ Find out RE for set of all string whose length ~~ex~~ atleast '2'.

→ $L = \{aa, ab, ba, bb, aaa \ldots\ldots\}$

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

④ Find out RE for set of all string whose length at most '2'.

→ $L = \{\underset{0}{\epsilon}, \underset{1}{a}, \underset{1}{b}, \underset{2}{aa}, \underset{2}{ab}, \underset{2}{ba}, \underset{2}{bb}\}$

$= (\epsilon + a + b + aa + ab + ba + bb)$

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

Ex-2

① $\Sigma = \{a, b\}$

Find out Regular expression for set of all strings of even length.

→ $L = \{$ set of all strings of even length $\}$

$= \{\epsilon, aa, ab, ba, bb, \ldots\ldots\}$

$= \boxed{((a+b)(a+b))^*}$

$((a+b)^2)^*$
$= (a+b)^{2n}$ / $n \geq 0$

② Find out Regular expression for set of all strings of odd length.

→ $L = \{a, b, aaa, \ldots\}$.

$= \boxed{((a+b)(a+b))^*(a+b)}$

$(a+b)^{2n+1}$ / $n \geq 0$

Ex-3

① Find out RE for set of all string whose length is divicible by '3'.

→ $\perp$ = 0, 3, 6, 9, 12, ---- length of the string divicible by 3.

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

② Find out RE for set of string whose length $\cong 2 \bmod 3$.

→ $(a+b)^{(3n+2)} / n \geq 0$

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

Ex-4

① Find out RE for set of all strings in which no of 'a's are ~~exactly~~ exactly '2'.

$b^n a b^n a b^n / n \geq 0$

→ $b^* a b^* a b^*$

$\perp = \{aa, babab, bbabb, abb \cdots\}$.

② no of a's atleast '2'.

→ $b^* a^{2} b^* a (a+b)^*$

③ no of a's atmost '2'.

→ $b^* (\epsilon + a) b^* (\epsilon + a) b^*$.

④ a's are even.

a^{2n}

→ ~~(a+b)~~* $(b^* a b^* a b^*)^* + b^*$.

Ex-5

① Find out RE for set of all String which starts with 'a'.

→ $a(a+b)^*$

② set of all string which end's with 'a'.

→ $(a+b)^*a$.

③ set of all string containing 'a'.

→ $(a+b)^*a(a+b)^*$.

④ set of all string starting and ending with diff symbols.

→ $a(a+b)^*b + b(a+b)^*a$

⑤ starting and ending with the same symbol.

→ and $L = \{\epsilon, a, b, aa, bb, aaa, bbb, \dots\}$

Regular expression, = $a(a+b)^*a + b(a+b)^*b + a + b + \epsilon$.

Ex-6

① Find out Regular ~~language~~ expression for set of all string does't contain two a's together.

→ $L = \{\epsilon, b, bb, bbb, aba, a, abab, bba, ba \dots\}$

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

$= (b+ab)^*(\epsilon+a)$.

② Find out a RE for all set of all string ∑ does't contain no 2 a's and 2 b's together.

→ $L = \{\epsilon, a, b, ab, ba, aba, bab, \ldots\}$

RE (Regular expression) :

RE	Starts with	ends with	Language
$(ab)^*a / a(ab)^*$	a	a	→ {a, aba, abab, baba}
$(ba)^*b / b(ab)^*$	b	b	→ {b, bab, babab}
$a(ba)^*b / (ab)^*$	a	b	→ {ab, abab}
$b(ab)^*a / (ba)^*$	b	a	→ {ba, baba}

$RL = \underline{(ab)^*}a + b(ab)^* + \underline{(ab)^*} + b(ab)^*a$.

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

$= (\epsilon+b)(ab)^*(a+\epsilon)$.

$= (\epsilon+a)(ba)^*(\epsilon+b)$.

- **IDENTITIES OF RE :-**

→ $\emptyset + R = R + \emptyset = R$

→ $\emptyset \cdot R = R \cdot \emptyset = \emptyset$

→ $\epsilon \cdot R = R \cdot \epsilon = R$

→ $\epsilon^* = \epsilon$

→ $\emptyset^* = \epsilon$

→ $\epsilon + RR^* = R^*R + \epsilon = R^*$

→ $(a+b)^* = (a^* + b^*)^*$

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

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

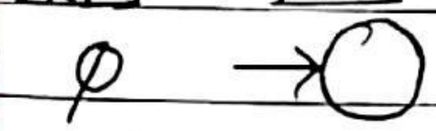
$= (a + b^*)^* = a^*(ba^*)^* = b^*(ab^*)^*$.

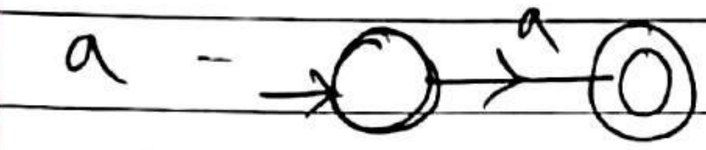
→ $RR^* = R^+$

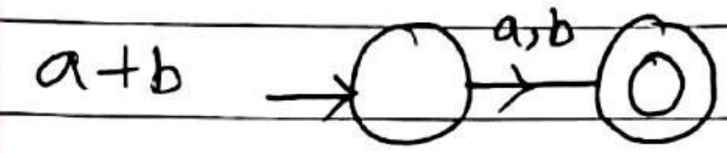
→ $R^*R^* = R^*$

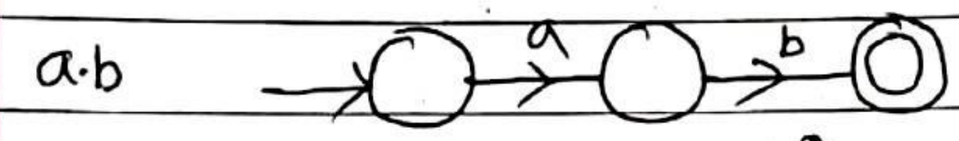
- Conversion RE → FA :.

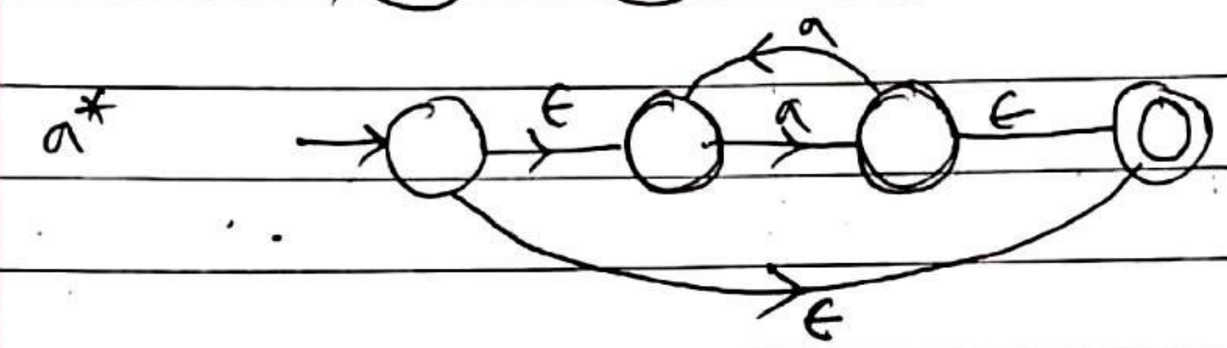
RE → FA.

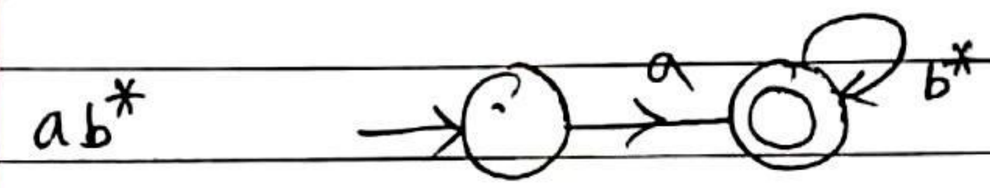


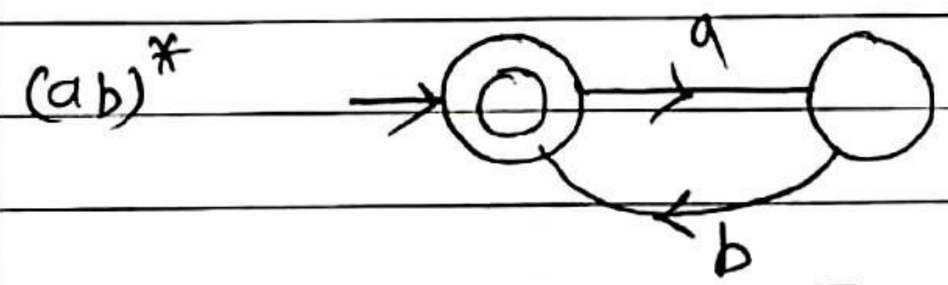


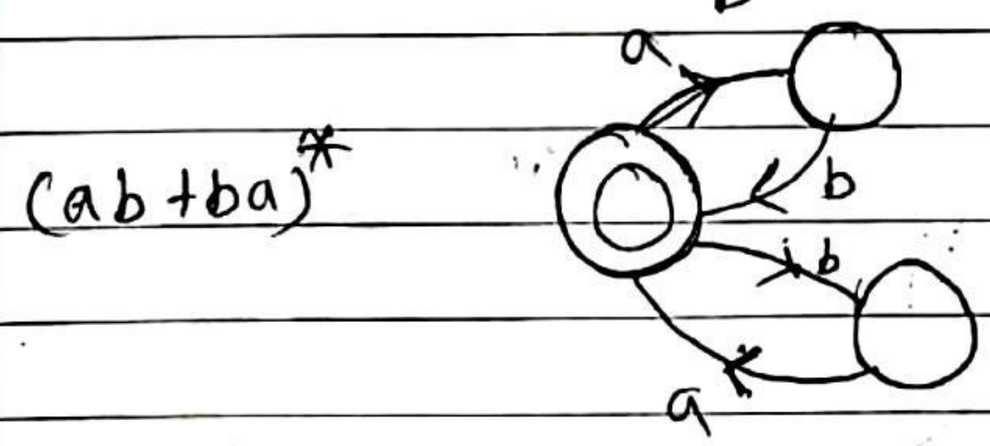








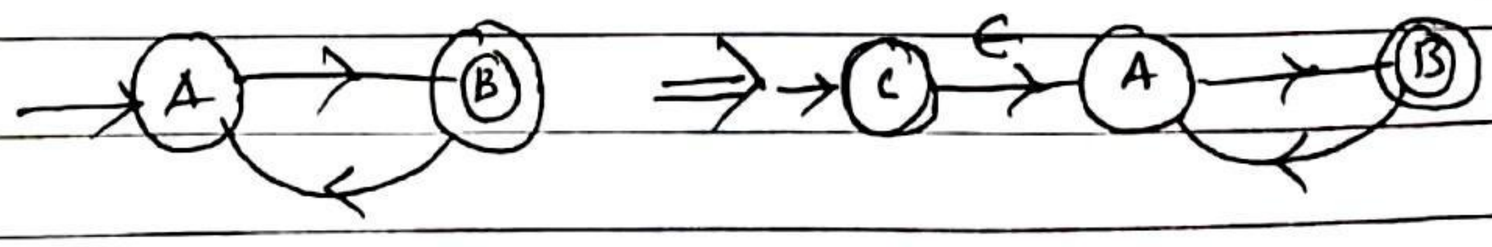




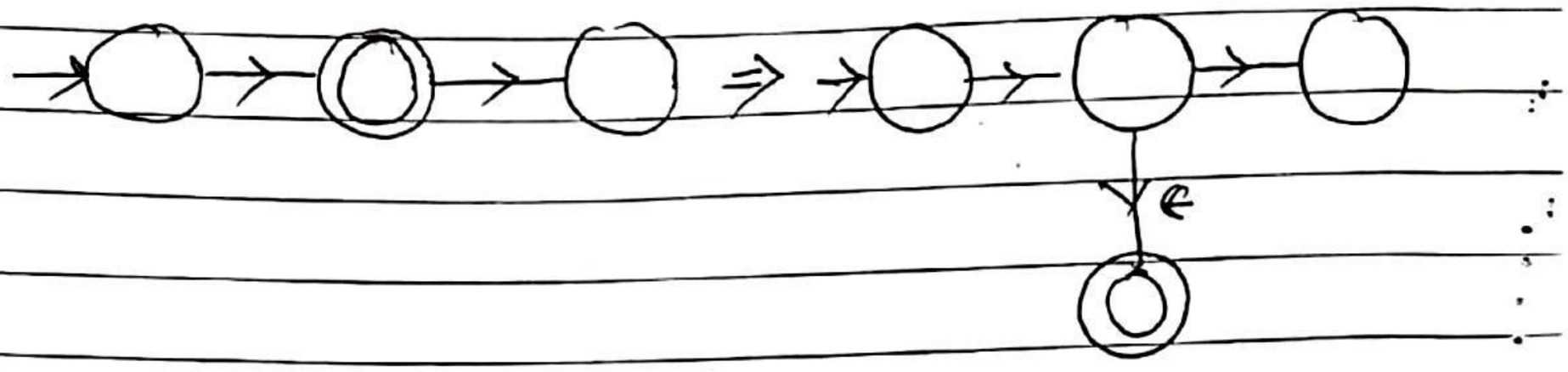
- Conversion FA → RE :

- State Elimination Methode :- (3 rules)

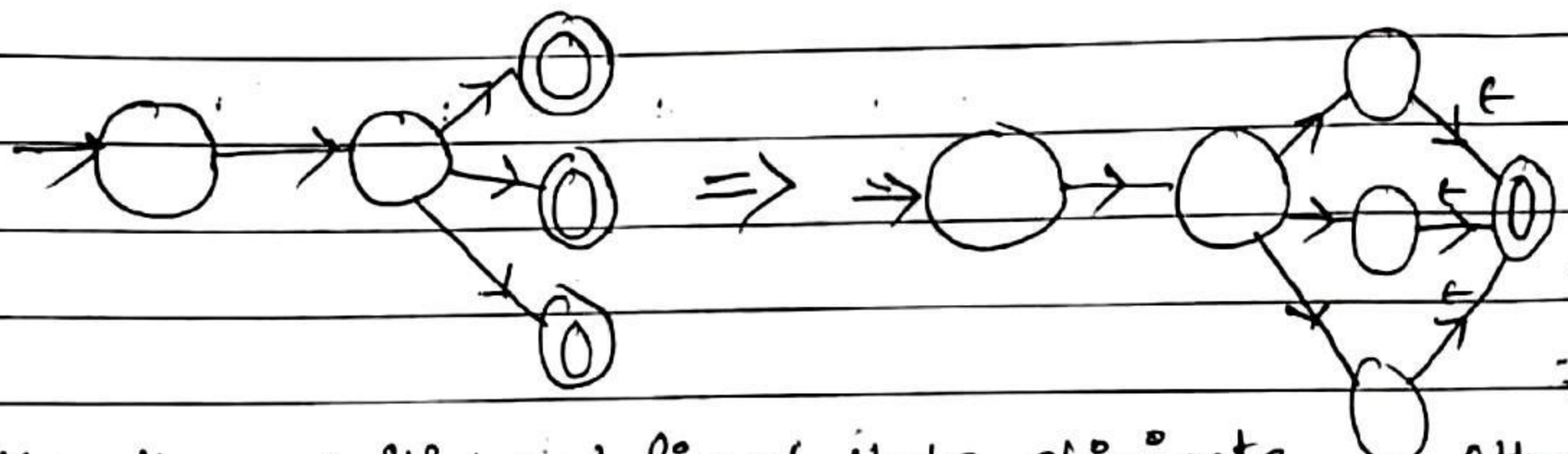
(1) Initial State should not have any incoming edge.



(2) - The finial state should not have any outgoing edges.



- If have more than one finial state then make a new fininal state then remove previous finial states.

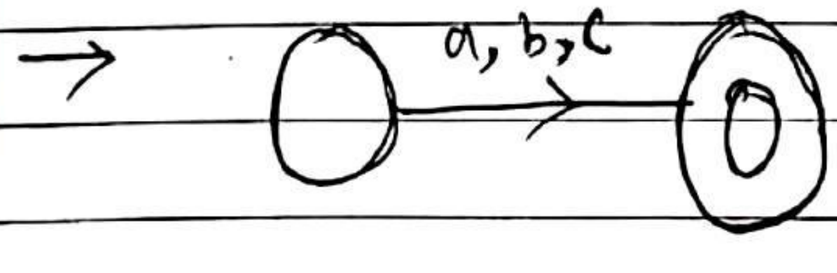


(3) → Other than Initial and final state eliminate ~~other~~ remaining ~~this~~ states one by one.

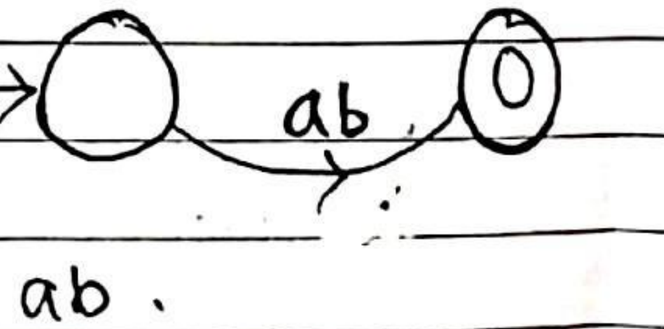
Ex-1

Find RE from FA:

(1)



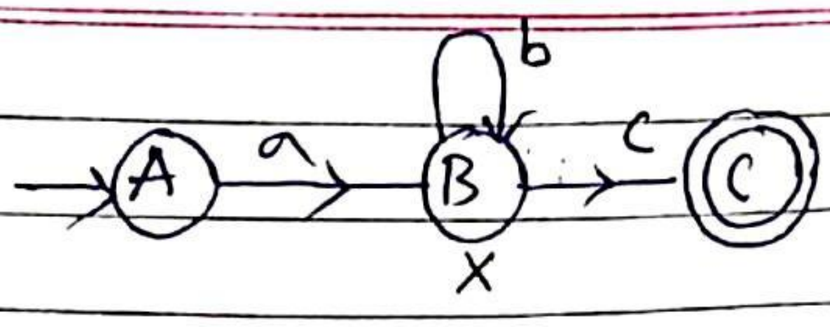
(11)

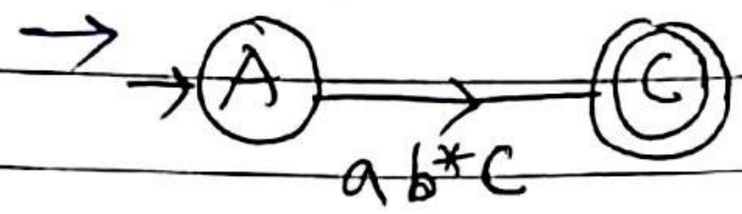


RE = ab.

Regular Expression = (a+b+c)

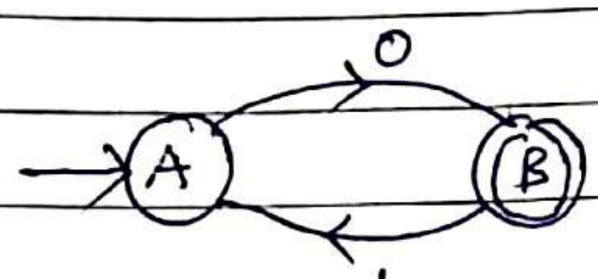
(iii)

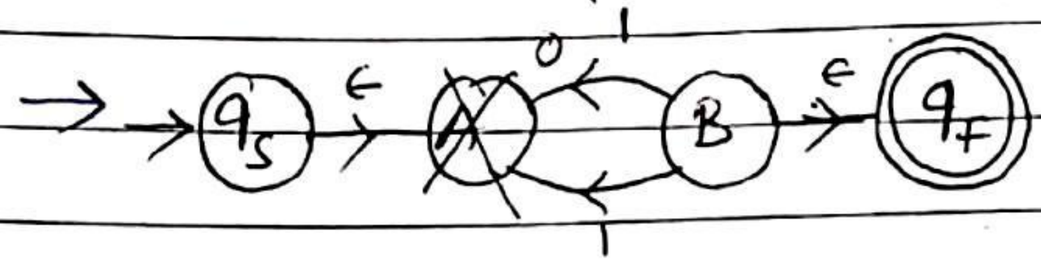


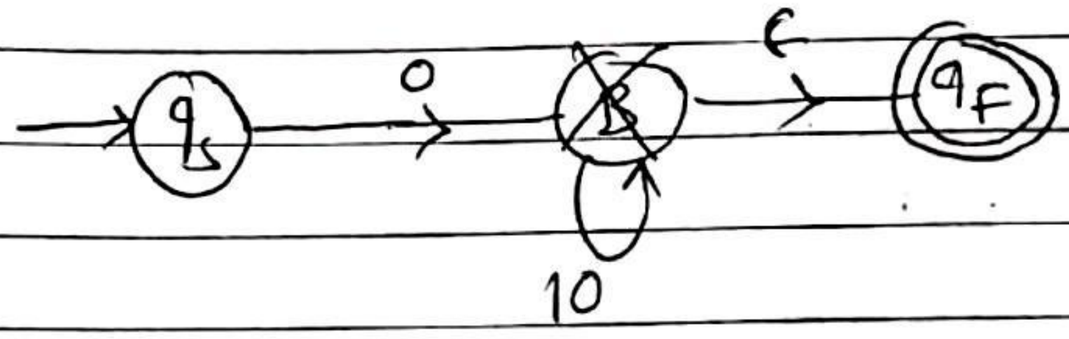


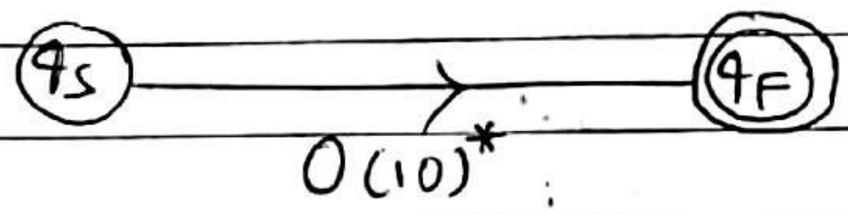
So, RE is = ab^*c.

(iv)



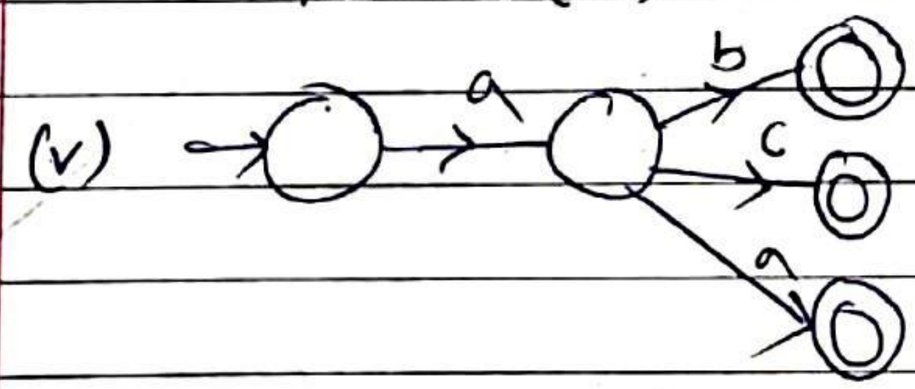


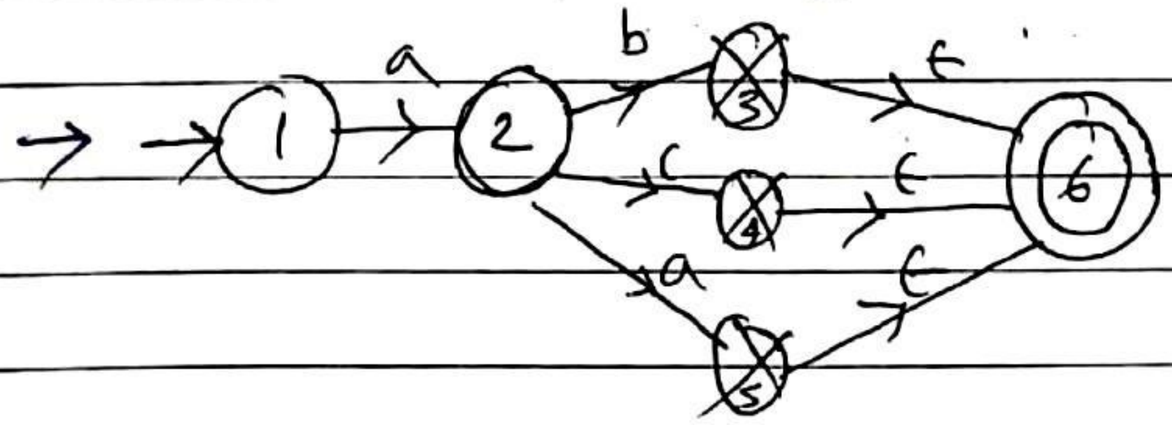


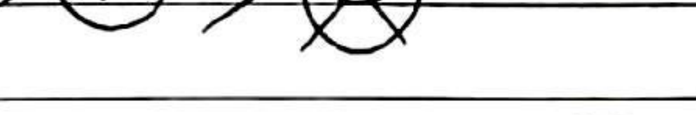


RE = $0(10)^*$.

(v)

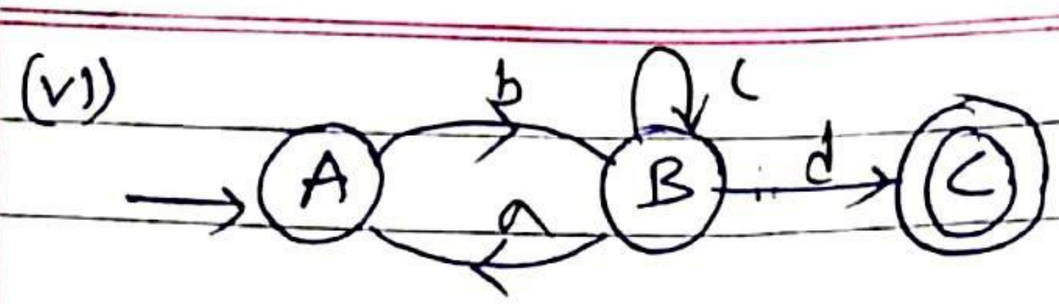


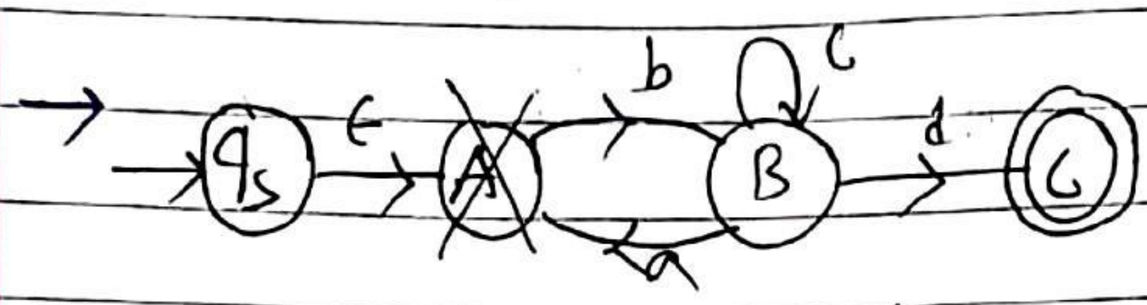


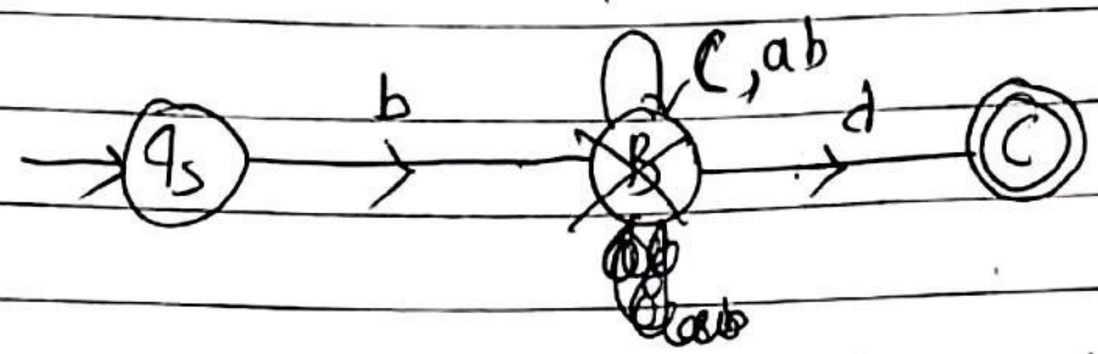


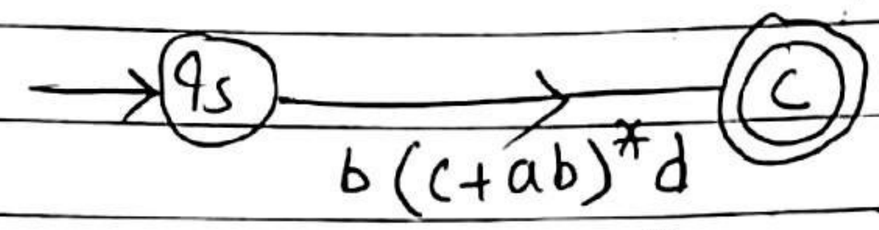
RE = a(b+c+d).

(vi)



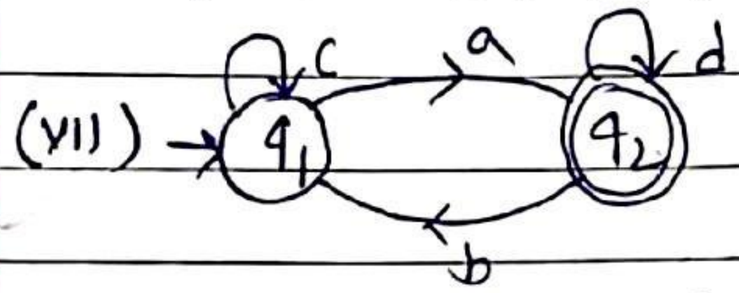


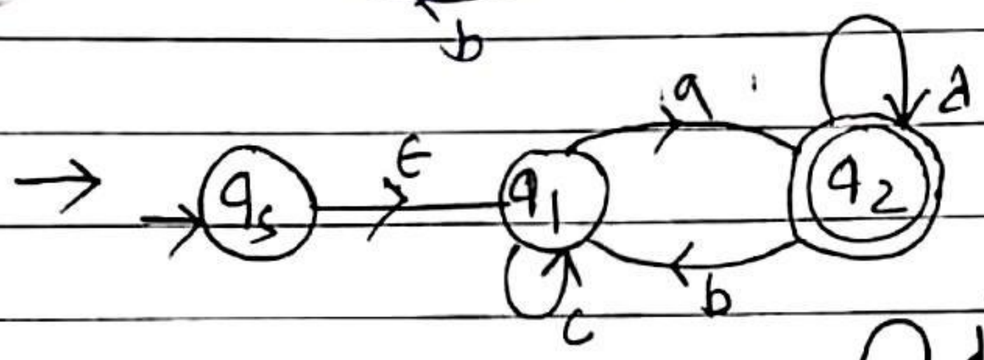


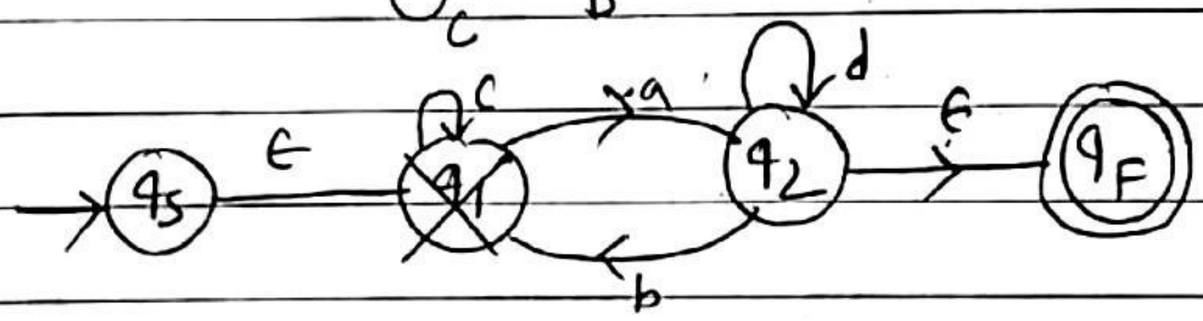


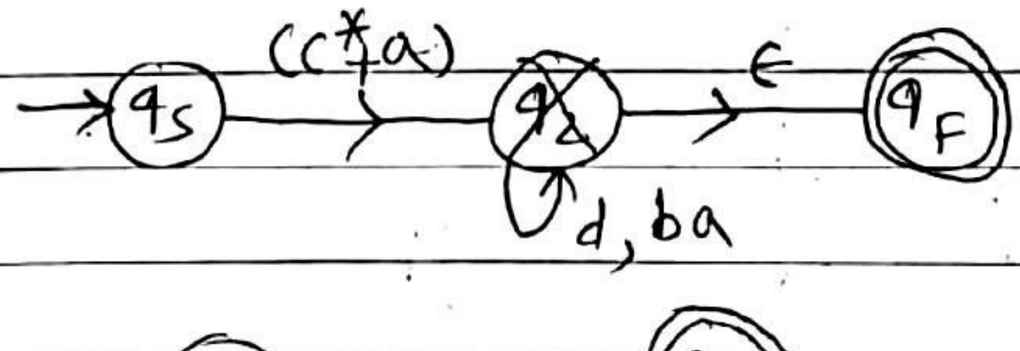
Finay RE = b(c+ab)* d.

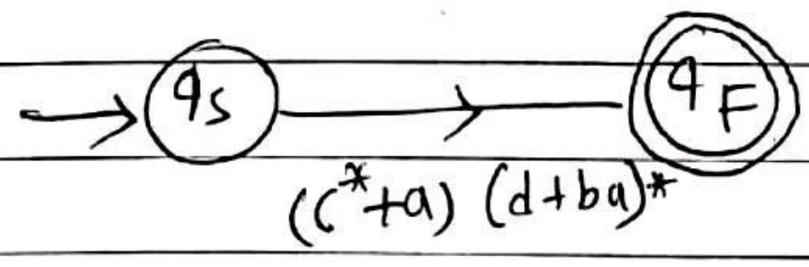
(vii)





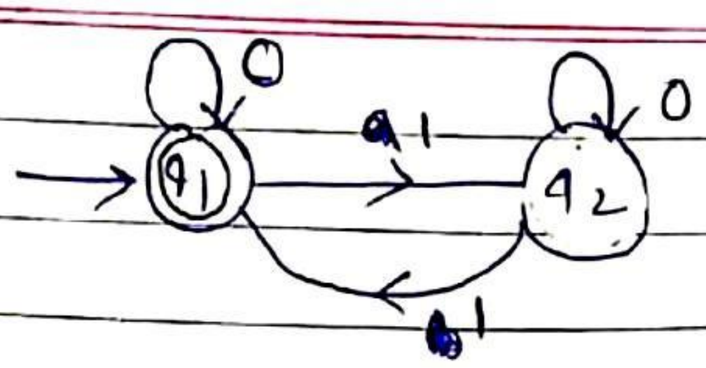


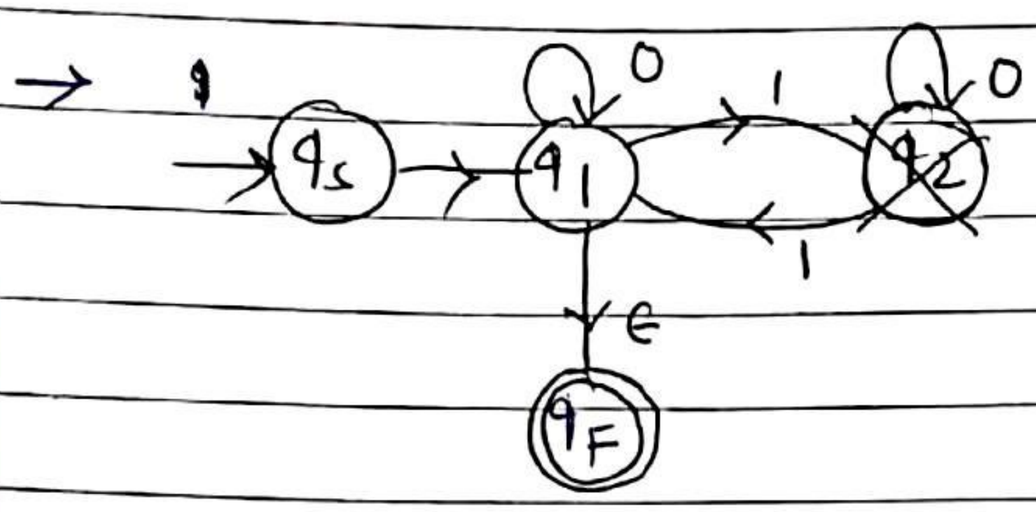


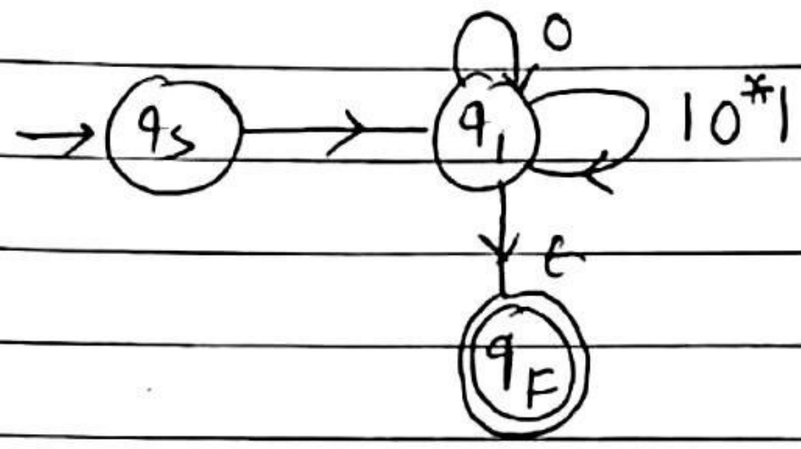


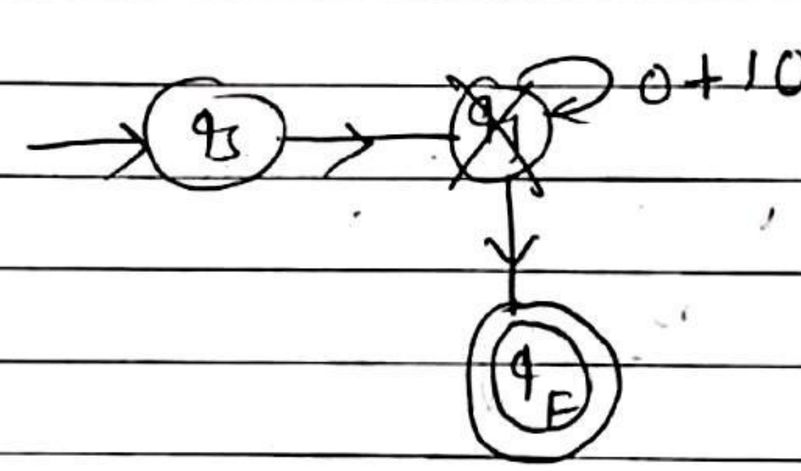
RE = (c*+a) (d+ba)*.

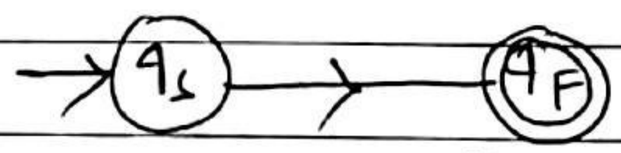
(VIII)





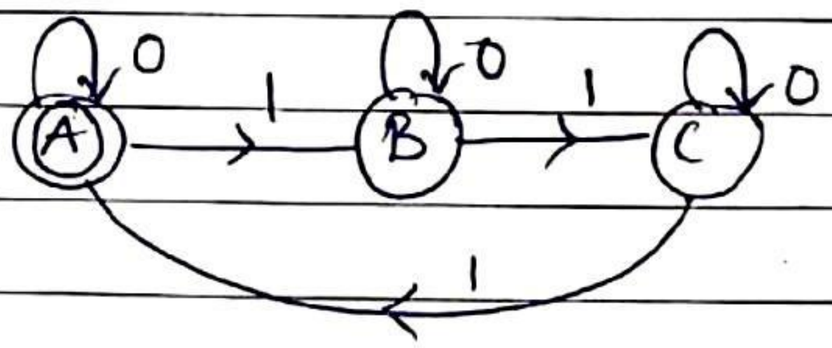


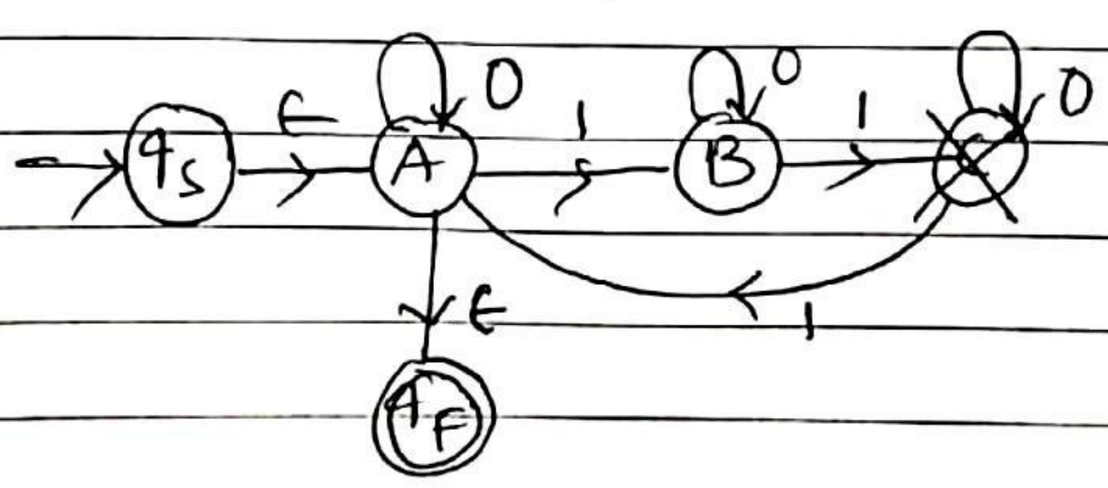


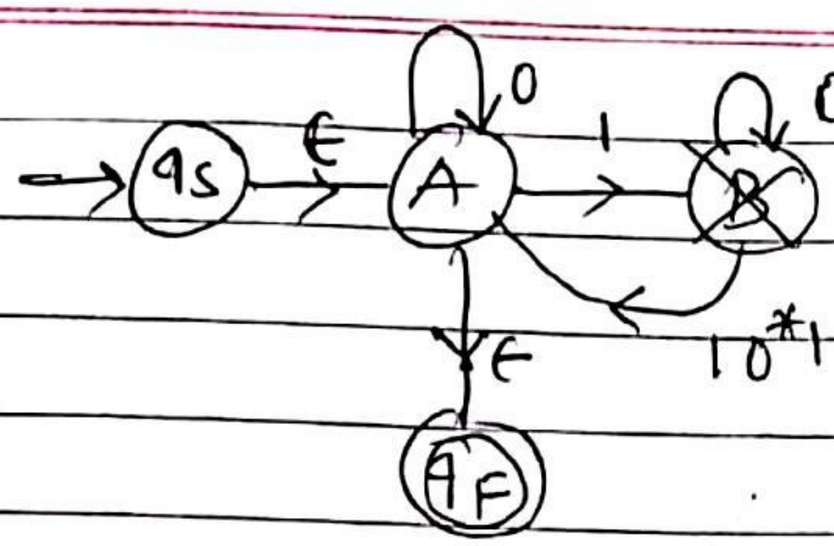


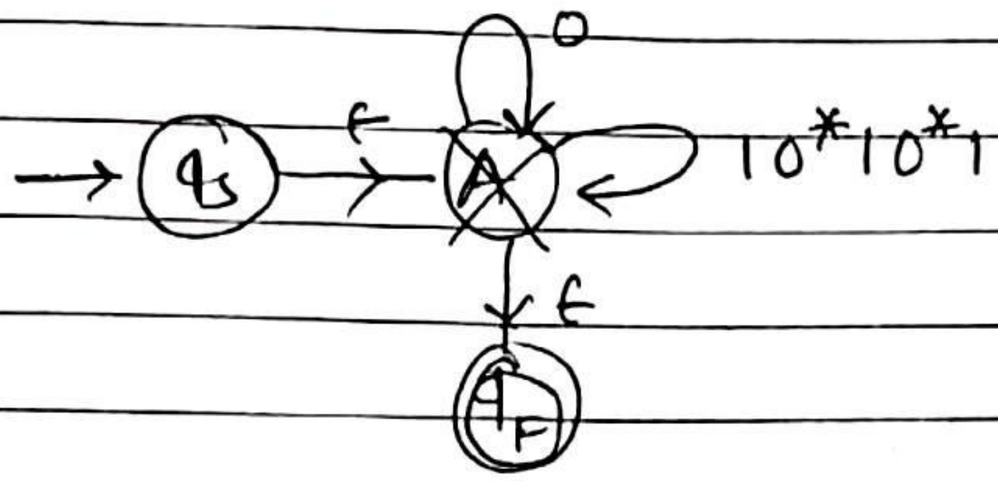
$RE = (0+10^*1)^*$

(IX)









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

Arden's Theorem:

* $R = Q + RP$ Where, P, Q, R are RE.

the above equation has a unique solution,

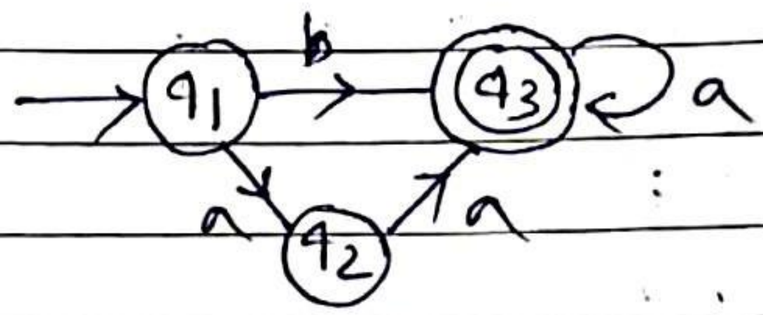
$\boxed{R = QP^*}$ where, $\lambda \notin P$

→ Arden's theorem is used to determine regular expression requested by a transition diagram.

$R = Q + RP \Rightarrow Q + QP^*P$

$\Rightarrow Q(1 + P^*) \Rightarrow QP^*$.

EX-1 Find RE from FA using Arden's theorem :-



→ $q_1 \Rightarrow \lambda$

$q_2 \Rightarrow q_1 \cdot a \Rightarrow \lambda a$

$q_3 \Rightarrow q_1 \cdot b + q_2 q_1 \cdot a + q_3 \cdot a \Rightarrow \lambda b + \lambda aa + q_3 a$

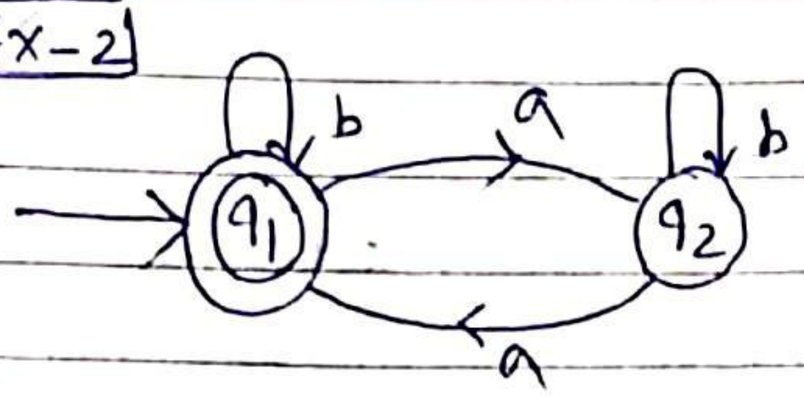
$\underset{r}{q_3} \Rightarrow \underset{q}{(b + aa)} + \underset{r}{q_3} \underset{p}{a}$.

$\therefore r = qp^*$

$q_3 \Rightarrow \boxed{(b + aa)\, a^*}$.

→ If here more than 1 finiat state, then find RE for every finial state then add every RE.

Ex-2



$$q_1 = \lambda + q_1 \cdot b + q_2 \cdot a \quad \text{—(I)}$$

$$q_2 = q_2 \cdot b + q_1 \cdot a \quad \text{—(II)}$$

use Arden's theorem on eq-(II)

$\therefore R = Q + RP$

$$q_2 = q_1 a + q_2 b$$

$$q_2 = (q_1 a) b^*$$

put the value of q_2 in q_1 equation —

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

$$q_1 = \lambda + q_1 (b + ab^*a) \quad \text{(again used Arden theorem)}$$

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

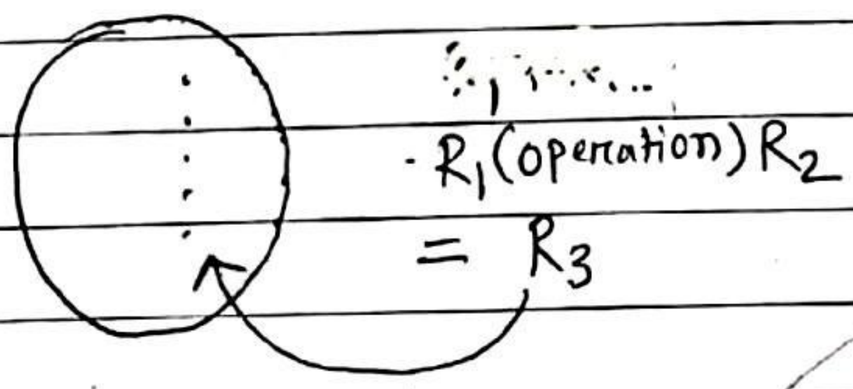
$$q_1 = (b + ab^*a)^*$$

(RE)

• Properties of Regular language:

→ Regular languages are closed under following Operations —

1. Union
2. Intersection
3. Concatenation.
4. Complementation.
5. Kleene closure.
6. Difference
7. Reversal.
8. Homomorphism.
9. Inverse Homomorphism.
10. Quotient Operation.
11. INIT Operation.
12. Substitution.
13. not under Infinite Union. (13)



1. $L_1 \cup L_2$ → $R_1 + R_2$
2. $L_1 \cap L_2$ $= \overline{\overline{L_1} \cup \overline{L_2}}$
3. $L_1 \cdot L_2$ → $R_1 \cdot R_2$
4. L_1^* → R_1^*
5. $\overline{L_1} = \Sigma^* - L_1$
6. $L_1 - L_2 = L_1 \cap \overline{L_2}$
7. DFA $\xrightarrow{(R)}$ FA (L^R)

(8) The function $h: \Sigma \to \Gamma^*$ is called homomorphism.
RL are closed under homomorphism.

Ex: $\Sigma = \{a, b\}$ $\Gamma = \{0, 1, 2\}$

$h(a) = 01$

$h(b) = 112$

homomorphic Image of String are.

$h(aba) = 01\ 112\ 01$

homomorphic Image of language =
$L_1 = a^* b = (01)^* 112$.

(9) Inverse homomorphism —
→ The inverse of homomorphic image of a language L is called IH.

$\bar{h}^1(L) = \{x / h(x) \text{ is in } L\}$
for a string w, $\bar{h}^1(w) = \{x / h(x) \in L\}$

Ex- Let $\Sigma = \{0, 1, 2\}$ and $\Delta = \{a, b\}$ Define 'h' by
$h(0) = a$, $h(1) = ab$, $h(2) = ba$.
Let, $L_1 = \{ababa\}$

$\Rightarrow \bar{h}^1(L) = \{110, 022, 102\}$

ababa
0 1 0

ab aba
0 2 2

abab a
1 0 2

Let, $L_2 = a(ba)^*$
$= \{a, aba, ababa, \ldots\}$

$\bar{h}^1(L_2) = \{0, 10, 011, 022, 102, \ldots\}$
02 022
012

ab abab a
1 1 1 0

ab abab a
0 2 2 2

ab ab ab a
1 0 2 2

$1^*0 + 02^* + 10^*2^*$

$h^{-1}(L_2) = \{1^*02^*\}$.

(Ex-2) Let $\Sigma = \{0,1\}$ and $\Delta = \{a,b\}$.
Define 'h' by $h(0) = aa$, $h(1) = aba$,
Let, $L = (ab + ba)^*a$.

$\rightarrow L = \{a^*, \underbrace{aba}_{1}, baa, abaaba, baabaa \ldots\ldots\}$

$h^{-1}(L) = \{1\}$

$h(h^{-1}(L)) = \{aba\}$
$\neq L$

** $h(h^{-1}(L)) \subseteq L$

(10) Let L_1 and L_2 be languages on the ~~same~~ same alphabet. Then the right ~~alphabet~~ quotient of L_1 with L_2 is defined as,

$L_1/L_2 = \{x : xy \in L_1 \text{ for some } y \in L_2\}$

(Ex-) $L_1 = \{01, 001, 101, 0001, 1101\}$
$L_2 = \{01\}$
$L_1/L_2 = \{\epsilon, 0, 1, 00, 11\}$.

Ex- $L_1 = 10^*1$, $L_2 = 0^*1$

$L_1/L_2 = ?$

$L_1 = \{11, 101, 100, 1, 10001, \cdots\}$

$L_2 = \{1, 01, 1001, 0001, 00001, \cdots\}$

$L_1/L_2 = \{1, 10, 100, 1000, \cdots\}$

$= 10^*$

(11) INIT Operation:

Ex: $L = \{a, ab, aabb\}$ (apply Prefix operation)

$Init(L) = \{\underline{\epsilon}, \underline{a}, \overset{x}{\epsilon}, \overset{x}{a}, ab, \overset{x}{\epsilon}, \overset{x}{a}, aa, aab, aabb\}$

(12) Under substitution:

Ex:- $\Sigma = \{a, b\}$

$f(a) = 0^*$, $f(b) = 01^*$

$L = a + b^*$

$\Rightarrow f(L) = 0^* + (01^*)^*$

(13) $L_1 = \{a^1 b^1\}$ $L_1 \cup L_2 \cup L_3 \cdots$

$L_2 = \{a^2 b^2\}$ $= \{a^n b^n / n \geq 1\}$ X

$L_3 = \{a^3 b^3\}$

:

so, RL not closed under Infinite Union.

REGULAR GRAMMAR

- Type 3 Grammar called Regular grammar.

→ If the grammar has all the production of the form –

$A \longrightarrow \alpha B / \beta$ here, $A, B \in V$ (variables) & $\alpha, \beta \in T^*$ (Terminals)	→ Right linear grammar (RLGr)

$A \longrightarrow B\alpha / \beta$ here, $A, B \in V$. $\alpha, \beta \in T^*$	→ Left linear grammar (LLGr).

Ex of LLGr –

here, $A \rightarrow B\alpha / \alpha$

$B \rightarrow B\alpha / Bb / a / b$

Ex of RLGr –

$A \rightarrow aB / a$

$B \rightarrow aB / bB / a / b$

→ Either they should be RLGr or LLGr, they should not be both of them.

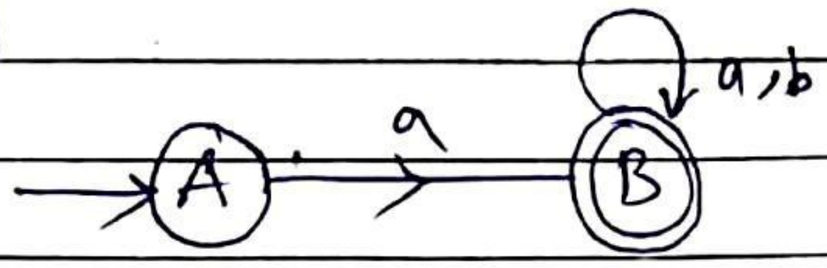
Ex 01 –

$A \rightarrow Ba / a$ $B \rightarrow aB / a$	→ not type-3 grammar.

→ Whatever language are generated by type-3 grammar they are Regular Language.

Conversion FA to RG (Regular grammar): (Type-3)

Ex-1

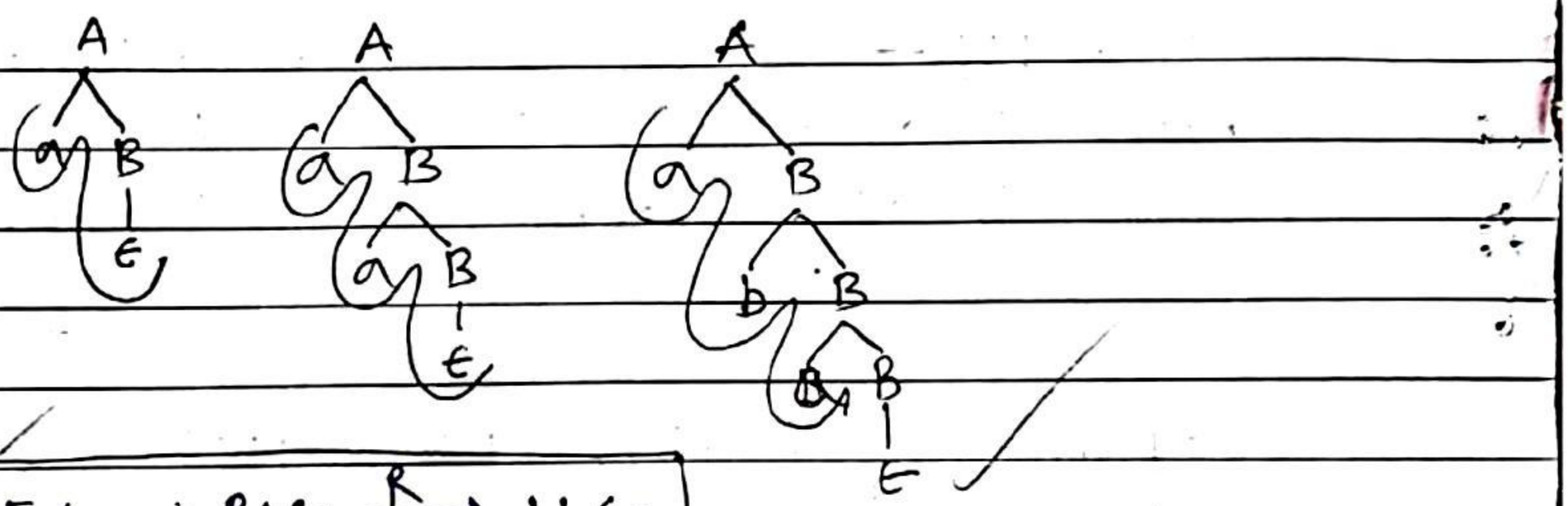


→ FA represent set of all string start with 'a'.

$A \rightarrow aB$
$B \rightarrow aB / bB / \epsilon$

— RLG.

→ for the give FA it is the RG.



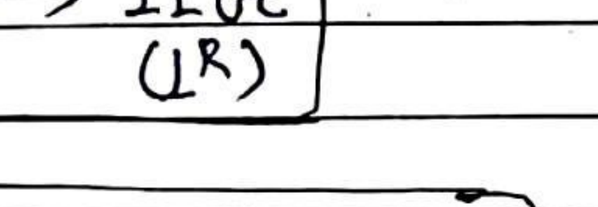
(*) FA → RLG → LLG
(L) (L) (L^R)

(*) FA → FA → RLG → LLG → Conversion FA to LLG.
(L) (L^R) (L^R) (L^R)^R
(I) (II) (III) =(L)

Ex-2

FA = (L)

Reversal of FA = (L^R)



(R-FA → RLG), (L^R)

$B \rightarrow aB / bB / a$
$A \rightarrow \epsilon$

— RLG of FA.

RGr ⇄ FA

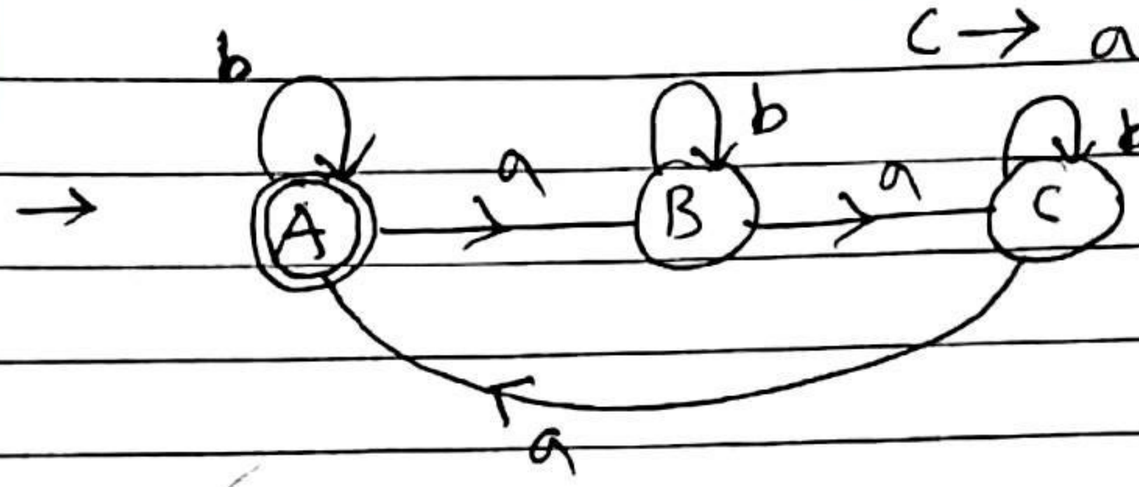
Now, Reversal of RLGr,

$$\boxed{\begin{array}{l} B \to Ba / Bb / a \\ A \to \epsilon \end{array}} \quad \text{LLGr} \qquad (1^R)^R = 1$$

- Conversion RGr to FA :

[Ex-1] given RLGr are,

$A \to aB / bA / (b)$
$B \to aC / bB$
$C \to aA / bC / (a)$.



[Ex-2] Converting LLGr → FA

**

$$\text{LLGr} \xrightarrow{R} \text{RLGr} \longrightarrow \text{FA} \xrightarrow{R} \text{FA}$$
$$(L) \qquad (L^R) \qquad L^R \qquad (L^R)^R = L$$

→ RGr and FA both are equivalent in power.

- Decidable problem on RL :-

1. Emptiness problem of FA:

Algo -

S-1) select ~~All~~ the states which are not reachable in final state from initial state and delete those states and corresponding transitions.

S-2) In the remaining FA, if we find at least one final state, then the language accepted by ~~the~~ given FA is non empty Otherwise empty.

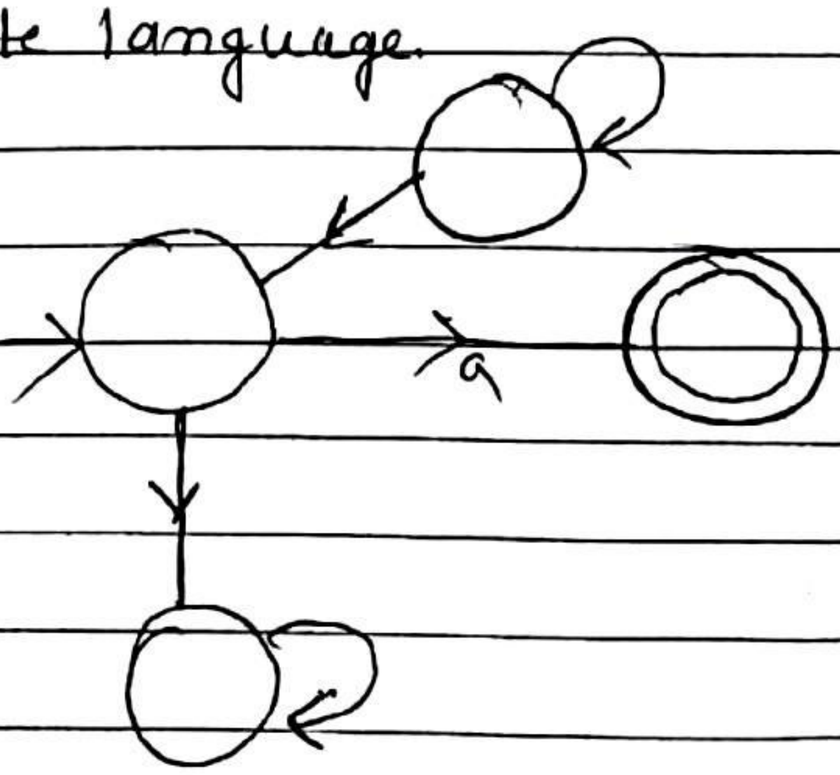
2. Infiniteness problem of RL :

Algo -

S-1) Remove the states which are not reachable from Initial state and corresponding transition.

S-2) Delete the states and corresponding transitions, from which we cannot reach final state.

S-3) In the remaining FA, if we find at least one loop and one final state then it is accepting infinite language.



atlantis

Date
Page

3. equality problem :-

Is $L_1 = L_2$

↓ ↓

DFA_1 DFA_2

↓ ↓

DFA_{m_1} DFA_{m_2}

→ Given Languages are Grammars convert into DFA, then minimize them & After minimization compare them to check both are equal or not.