

Theory of Computation

Gate Notes

For more visit www.gatenotes.in

25/06/2018 (Revision) 2 Day Review

Serial Number	Date	Title	Page Number	Teacher's Sign/Remarks
✓ 1.		• Introduction of T.O.C. ①		
✓ 2.		• Regular expression and finite Automata. ②		
✓ 3.		• <u>Regular Language and context free language.</u> ② ③		
✓ 4.		• Context free grammar and push down automata. - ③		
✓ 5.		• Turing Machine. ④		
✓ 6.		• Closure properties of language. ①, ②, ③, ④		
✓ 7.		• Uncountability. ④		
✓ 8.		• Undecidability. ④		
✓ 9.		• pumping lemma. ②		

INTRODUCTION OF TOC

- Automata :- Study of abstract computing devices or machines.
- Symbol :- A symbol is an abstract entity.
Example: $a, b, 0, 1, \dots$
- Alphabet :- An Alphabet is a finite collection of symbols.
(Σ) Example: $S = \{a, b, c\}$
 $\Sigma = \{0, 1\}$
- String :- It is a sequence of symbols.
Example :- $\Sigma = \{a, b\}$
strings :- $\{\underline{a}, \underline{b}, \underline{aa}, \underline{ab}, \underline{ba}, \underline{bb}\}$ hence have '6' string.
null length-2
→ Empty string can be denoted by (ϵ or 1 or λ).
[If the length of string is zero, such string is called empty string]
- Language :- Collection of string.
[Where strings is restricted over given alphabet]
→ $\Sigma = \{a, b\}$
 $L_1 = \text{set of all string of length 2}.$
 $= \{aa, ab, ba, bb\}$
 where language L_1 is finite language.
- $\Sigma = \{a, b\}$
 $L_2 = \text{set of all string where each string starts with } a.$
 $= \{a, aa, ab, aaa, aab, aba, \dots\}$
 where, L_2 is infinite language.
- Length of string :- No of symbols contained in a string.

$$\{ab\} \Rightarrow \text{length } 2 \cdot 1 \in \{ \Rightarrow \text{length } 0 \cdot$$

- Prefix of a string :- If $w = xy$, for some string y , then x is a prefix of w .

example - $w = \{0011\}$

$$\Rightarrow \left\{ \frac{00}{x} \frac{11}{y} \right\} \Rightarrow \left\{ \frac{00}{x} \frac{11}{y} \right\} \Rightarrow \{0, 00, 001\}$$

- Suffix of a string :- If $w = xy$, for some string x , then y is a suffix of w .

example - $w = \{0011\}$

$$\Rightarrow \left\{ \frac{00}{x} \frac{11}{y} \right\} \Rightarrow \{1, 11, 00\}$$

suffix.

- Kleene closure :-

→ It is denoted by * (asterisk) after the name of the alphabet - is Σ^* . This notation also known as the Kleene Star.

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

Σ^0 = set of all strings of length '0'. = $\{\epsilon\}$.

Σ^1 = set of all strings of length '1' = $\{a, b\}$

$\Sigma^2 = \Sigma \cdot \Sigma = \{aa, ab, ba, bb\}$

= $\{aa, ab, ba, bb\}$ = set of all strings of length '2'.

$\Sigma^3 = \Sigma \Sigma \Sigma$ = set of all strings of length '3'.

$|\Sigma|^3 = 8$ (no of strings)

$|\Sigma|^n = 2^n$ = set of all strings of length 'n'.

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

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

Σ^* = set of all strings possible over $\{a, b\}$ [Universal set]

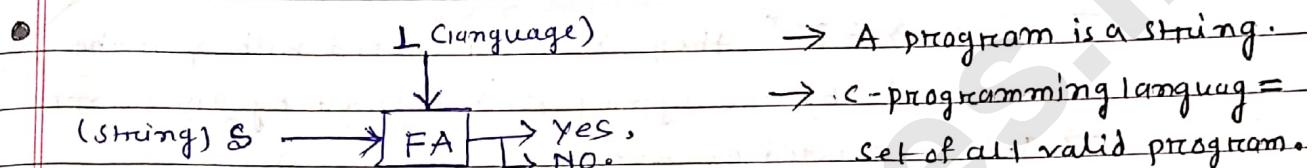
$$\boxed{\Sigma^* = \Sigma^+ \cup \{\epsilon\}}$$

$$\Sigma^* = \boxed{L_1 \ L_2 \\ L_3}$$

$$\begin{aligned} L_1 &\subseteq \Sigma^* \\ L_2 &\subseteq \Sigma^* \\ L_3 &\subseteq \Sigma^* \end{aligned}$$

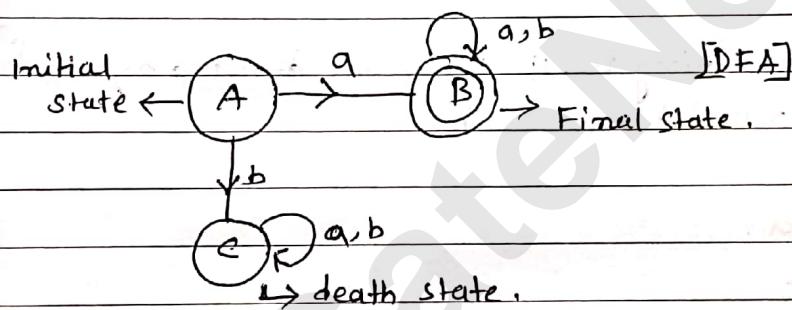
→ no of string possible over Σ^* is infinite.

→ no of language possible over Σ^* is infinite.



L = set of all strings which starts with 'a'.

$$L = \{ a, aa, ab, aab, \dots \}$$



String = aab (present in language or not)

this string are accepted by FA. $A \xrightarrow{a} B \xrightarrow{a} B \xrightarrow{b} C$

→ the string will be accepted by FA, if after scanning entire string we reach \rightarrow in final state from initial state.

String = bbba

this string are not accepted by FA. $A \xrightarrow{b} C \xrightarrow{b} C \xrightarrow{a} C$

positive closure:

→ The '+' (plus operation) is sometimes called positive closure.

→ If $\Sigma = \{a, b\}$, then,

$$\Sigma^+ = \{ a, aa, ab, ba, bb, \dots \}$$

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

• Substring of a string :-

→ A string ' w ' = $a b c$

here, ab , bc are substring of w .

But ac are not substring of w .

• Concatenation of two strings :-

→ If $x, y \in \Sigma^*$, then x concatenated with y is the word formed by the symbol of ' x ' followed by the symbols of ' y '.

→ This is denoted by $x \cdot y$, it is same as xy .

• Reversal of a string :-

→ Given a string w , its reversal denoted by w^R is the string spelled back wards.

$$w = ab$$

$$w^R = ba.$$

• Grammar :-

→ It enumerates strings of the language.

→ It is a finite set of rules defining a language.

→ A grammar is defined as 4-tuples (V, T, P, S)

where, $V \xrightarrow{\text{set}} \text{of non terminals}$

$T(\Sigma) \xrightarrow{\text{set}} \text{Set of input terminals}$

$P \rightarrow \text{finite set of production rule}$

$S \rightarrow \text{start of symbol}$

→ $\xrightarrow{\text{start}(S)}$

example :-

$$(production) P = \begin{cases} (S) \rightarrow aSB \\ S \rightarrow aB \\ B \rightarrow b \end{cases}$$

here, $V = \{S, B\}$

$T = \{a, b\}$

→ Getting a string from a grammar is called Derivation.

Derivation on aabb

$$S \rightarrow aSB$$

$$\rightarrow aaBB \quad [S \rightarrow aB] \quad \text{entire process called derivation.}$$

$$\rightarrow aabB \quad [B \rightarrow b] \quad \text{sequential forms.}$$

$$\rightarrow \boxed{aabB} \quad [B \rightarrow b]$$

Q-1

Construct a grammar given the following language,

$$L = \{ \text{Set. of all strings length } 2 \}$$

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

$$\rightarrow L = \{aa, ab, ba, bb\} \quad \frac{(a+b)}{A} \frac{(a+b)}{A}$$

$$\boxed{\begin{array}{l} S \rightarrow aa/ab/ba/bb \\ S \rightarrow AA \\ A \rightarrow a/b \end{array}} \quad \text{production rules.}$$

Q-2

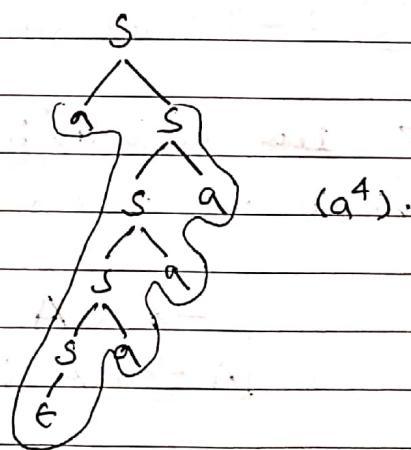
construct grammar, given the following language,

$$L = \{a^n \mid n \geq 0\}$$

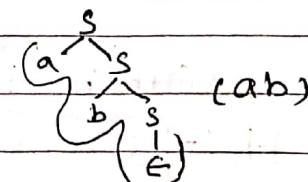
$$\rightarrow L = \{\epsilon, a, aa, aaa, \dots\}$$

P.R.

$$\boxed{\begin{array}{l} S \rightarrow aS/\epsilon \\ S \rightarrow Sa/\epsilon \end{array}}$$

**Q-3** construct grammar to generate (ab)^{*}.

$$\rightarrow \boxed{S \rightarrow aS/bS/\epsilon}$$



Q-4) construct a grammar, given the following language,
 $L = \{ \text{set of all string of length at least } '2' \}$

$$\rightarrow L = \{ aa, ab, ba, bb, aaa, aab, \dots \} \\ (a+b)^* (a+b)^* (a+b)^*$$

production rules,

$S \rightarrow AAB$
$A \rightarrow a/b$
$B \rightarrow aB / bB / \epsilon$

Q-5) c.A.Q, given the following language,
 $L = \{ \text{string of length at most } '2' \}$.

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

prv $(a+b+\epsilon)^* (a+b)^*$

production rules,

$$S \rightarrow AA$$

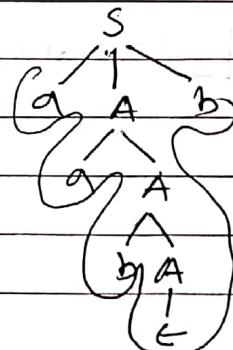
$$A \rightarrow a/b/\epsilon$$

Q-6) $L = \{ \text{start with } 'a' \text{ and end with } 'b' \}$.

$$\rightarrow L = a (a+b)^* b$$

production rules,

$S \rightarrow aAb$
$A \rightarrow aA / bA / \epsilon$



(aabbb)

Q-7) $L = \{ \text{set of all string starting and ending with different symbol} \}$.

$$\rightarrow a(a+b)^* b + b(a+b)^* a$$

PR:- $S \rightarrow aAb / bAa$

$A \rightarrow aA / bA / \epsilon$

(Q-8) $L = \{ \text{Set of all strings starting and ending with same symbol} \}$

$$\rightarrow a(a+b)^*a + b(a+b)^*b.$$

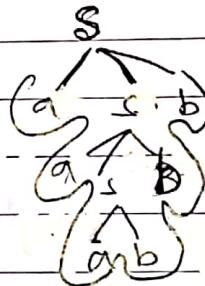
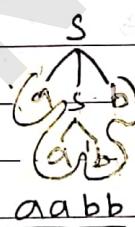
$$\boxed{S \rightarrow aAa / bAb / b / a / \epsilon} \\ A \rightarrow aA / bA / \epsilon}$$

(Q-9) construct a grammar, given the following language.

$$L = \{a^m b^m / m \geq 1\}$$

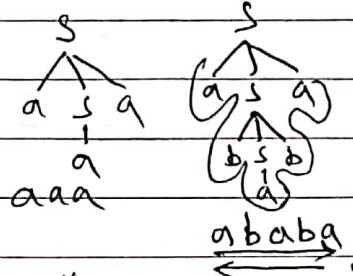
$$\rightarrow L = \{a^m b^m / m \geq 1\} \\ = \{ab, aabb, \dots\}$$

$$\boxed{S \rightarrow aSb / ab / \epsilon}$$



(Q-10) construct grammars that generate set of all palindrom - $w w^R, w a w^R, w b w^R$

$$\rightarrow S \rightarrow aSa / bSb / a / b / \epsilon$$



(Q-11) construct grammar that generate "even length string".

$$\rightarrow L = \{ \underline{aa}, \underline{ab}, \underline{ba}, \underline{bb}, \underline{aaaa}, \dots \} \\ ((a+b)(a+b))^*$$

$$\boxed{S \rightarrow BS / \epsilon} \\ B \rightarrow AA \\ A \rightarrow a / b}$$

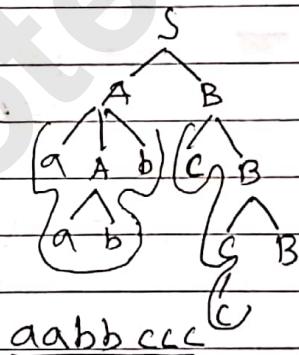
$$(Q-12) \quad L = \{a^n b^m / n, m \geq 1\}$$

$$\rightarrow L = \{ab, aab, aabb, abb, \dots\}$$

$$\begin{array}{|c|c|} \hline S & \rightarrow AB \\ \hline A & \rightarrow aA / a \rightarrow a^n \\ \hline B & \rightarrow bB / b \rightarrow a^m \\ \hline \end{array}$$

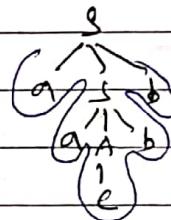
$$(Q-13) \quad L = \{a^n b^m c^m / n, m \geq 1\}$$

$$\rightarrow \begin{array}{|c|c|} \hline S & \rightarrow AB \\ \hline A & \rightarrow aAb / ab \\ \hline B & \rightarrow cB / c \\ \hline \end{array}$$



$$(Q-14) \quad L = \{a^n c^m b^m / n, m \geq 1\}$$

$$\rightarrow \begin{array}{|c|c|} \hline S & \rightarrow aSb / aAb \\ \hline A & \rightarrow cA / c \\ \hline \end{array}$$



aac cb

$$(Q-15) \quad L = \{a^n b^m c^m d^m / n, m \geq 1\}$$

$$\rightarrow \begin{array}{|c|c|} \hline S & \rightarrow AB \\ \hline A & \rightarrow aAb / ab \rightarrow a^n b^m \\ \hline B & \rightarrow c.Bd / cd \rightarrow c^m d^m \\ \hline \end{array}$$

$$(Q-16) \quad L = \{a^n b^{2n} / n \geq 1\}$$

$$\rightarrow S \rightarrow aSbb / abb$$

(Q-17) $L = \{a^{m+n} b^m c^n / m, n \geq 1\}$

$\{a^m b^m c^n a^m / m, n \geq 1\}$ X

$\rightarrow L = \{a^m \underline{a^m b^m} c^n / m, n \geq 1\}$

$S \rightarrow aSc / aAac$
 $A \rightarrow aAb / ab$

(Q-18) $L = \{a^n b^{n+m} c^m / n, m \geq 1\}$

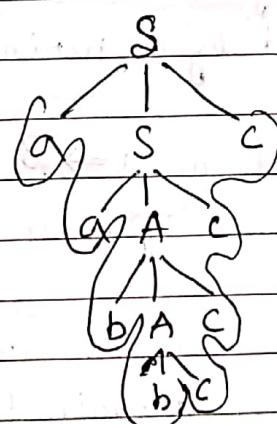
$\rightarrow L = \{a^n b^m \underline{b^m c^m} / n, m \geq 1\}$

$S \rightarrow A:B$
 $A \rightarrow aAb / ab$
 $B \rightarrow bBc / bc$

(Q-19) $L = \{a^n b^m c^{n+m} / n, m \geq 1\}$

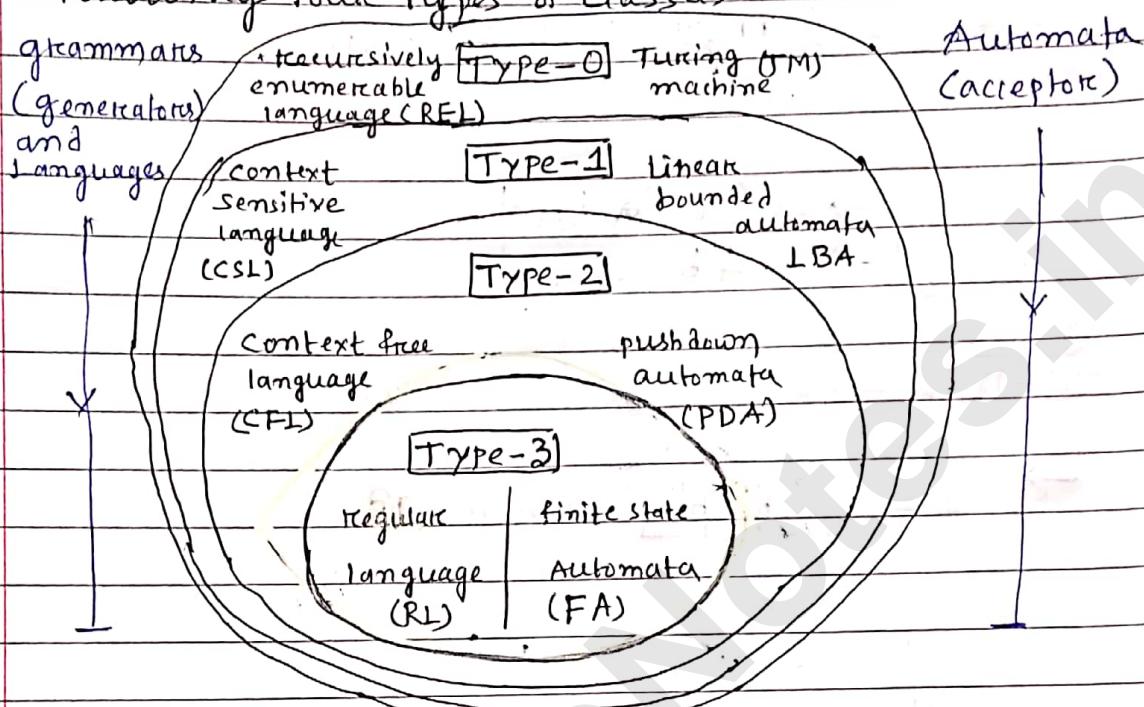
$\rightarrow L = \{a^n b^m \underline{c^m} \underline{a^n} / n, m \geq 1\}$

$S \rightarrow aSc / aAc$
 $A \rightarrow bAc / bc$



$$\begin{aligned} aabbccccc &\Rightarrow a^2 b^2 c^4 \\ &\Rightarrow \underline{a^2} \underline{b^2} c^4 \end{aligned}$$

- Chomsky Hierarchy : Chomsky hierarchy consists of following four types of classes -



- Types of Grammars :-

① TYPE-0 Grammar (Unrestricted Grammar) :

→ These are Unrestricted grammar which include all formal language.

→ This grammar generate exactly all language that can be recognized by a turing machine.

→ Rules are of the form $\alpha \rightarrow \beta$

(α must have at least 1 non-terminal.)

→ Where, α and β are arbitrary sequence of terminals and non-terminals and $\alpha \neq \epsilon$.

ex- $A \rightarrow aA \mid bA$ | $a \rightarrow Ba$

(2) TYPE - 1 Grammars (Context Sensitive Grammars):

→ language defined by type-1 grammars are accepted by the linear bounded automata.

→ Rules are of the form, $|d| \leq |B|$
length of 'd' is less than or equal to length of 'B'.

$A \rightarrow \epsilon$ is not allowed unless A is a start symbol.

$\times \boxed{AB \rightarrow a}, \checkmark \boxed{A \rightarrow aB}$

(3) TYPE - 2 Grammars (Context ~~sensitive~~ ^{free} Grammars)

→ Language defined by type-2 grammars are accepted by push down automata.

→ Rules are of the form $d \rightarrow B$.
where, $\bullet d = \text{single nonterminal}$.

$\boxed{A \rightarrow a}, \checkmark \boxed{Aa \rightarrow bB} \times$

(4) TYPE - 3 Grammars (Regular Grammars)

→ language defined by type-3 grammars are accepted by finite state automata.

→ Regular grammars can follow either right linear or left linear.

(right linear grammar)		example,
$A \rightarrow \alpha \boxed{B} / B$		$A \Rightarrow aB$
$A, B \in V$ (non terminal)		$B \rightarrow aB / bB / a / b$
$\alpha, \beta \in T^*$ (terminal)		

left linear grammar		example,
$A \rightarrow \boxed{B} \alpha / \beta$		$A \rightarrow Ba$
$A, B \in V$		$B \rightarrow Bay / Bb / a / b$
$\alpha, \beta \in T^*$		

If, $\boxed{A \rightarrow Ba / a}$ → not Type 3 grammar.

- Chomsky hierarchy Examples-

• Identify grammar :-

1) Example :-

$$S \rightarrow aSb | aA | B$$

$$B \rightarrow aA | a$$

$$C \rightarrow cD | D$$

2) Example

$$S \rightarrow a | aA | Bb$$

\rightarrow (Type-2).

It will be accepted by

\rightarrow ans is (Type-0)

Push Down Automata.

it will be accepted by
Turing machine.

3) Example -

$$S \rightarrow a\$b | ab \rightarrow$$
 (Type-2) grammar.

- Type-0 class is also called as :

\rightarrow Unrestricted Grammar.

\rightarrow Recursively Enumerable languages.

\rightarrow Turing Machine.

- Type-1 class is also called as :

\rightarrow Context Sensitive grammar.

\rightarrow Context Sensitive language.

\rightarrow Linear Bounded Automata.

- Type-2 class is also called as :

\rightarrow Context Free Grammar.

\rightarrow context Free language.

\rightarrow push down automata.

- Type-3 class is also called as :

\rightarrow Regular Grammar.

\rightarrow Regular language.

\rightarrow finite automata.

- Finite Automata (FA):

- Machines with fixed amount of unstructured memory, accepts regular language.
- Application of FA: useful for modeling chips, communication protocols, adventure games, some control systems, etc.

- Push Down Automata (PDA):

- Finite Automata with unbounded structured memory in the form of a push down stack, accepts CFL.
- Application of PDA: Compilers useful for modeling parsing, compilers, programming language design.

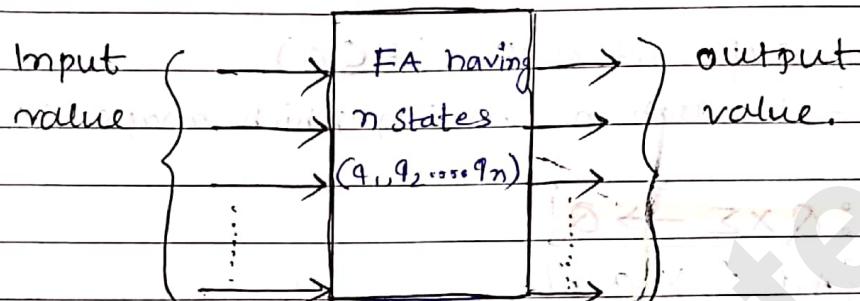
- Turing Machine (TM):

- Finite automata with unbounded tape accepts or enumerates recursively enumerable language.
- equivalent to RAM's and various programming language models.
- Application of TM: Model for general sequential computation (real computer).

Regular Expression and Finite Automata

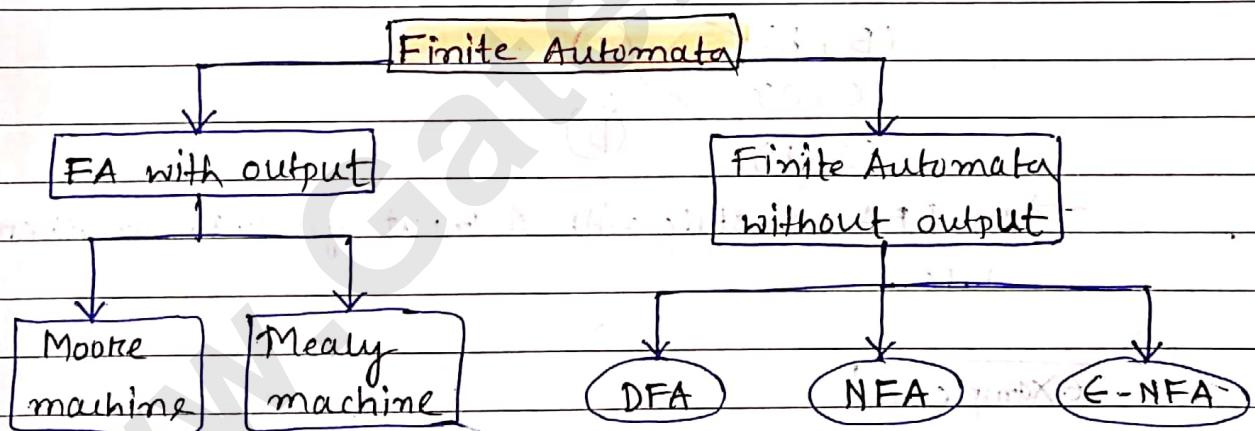
Finite Automata :-

→ 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.



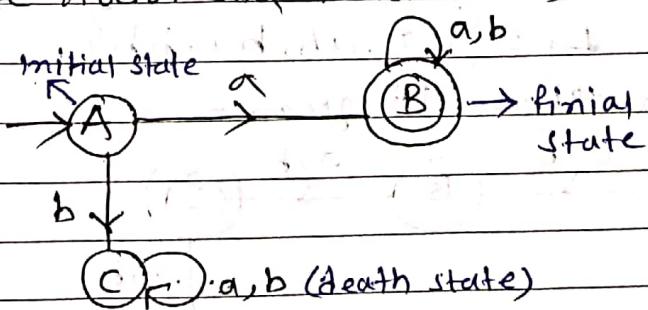
Model of finite automata

Types of Finite Automata :-



DFA (Deterministic Finite Automata) :-

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



where,

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

$\Sigma = \text{Input} \Rightarrow \{a, b\}$

$q_0 = \text{Initial state} \Rightarrow 'A'$

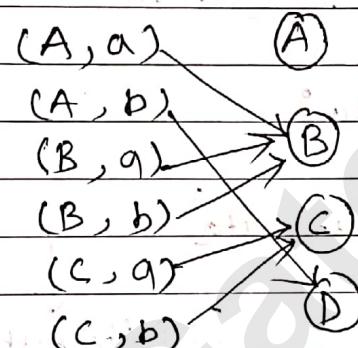
$F = \text{final state} \Rightarrow \{B\}$
↳ set of.

$\rightarrow Q$ is the superset of F . (FCQ)

$\delta = \text{transition function which maps } Q \times \Sigma \text{ into } Q$.

$$\delta : Q \times \Sigma \rightarrow Q$$

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



\rightarrow In DFA, a state with 1 input go into another one state.

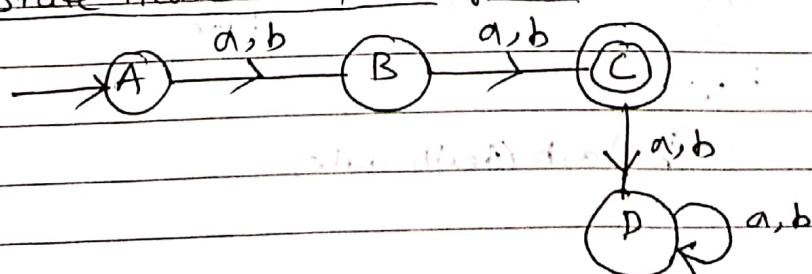
Example - 1

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

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

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

state transition Diagram -



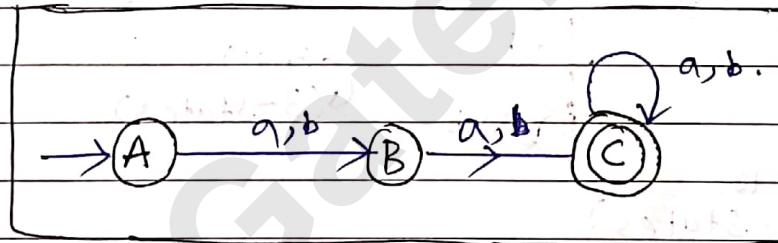
- String accept \rightarrow Scan the entire string, if we reach in a final state from initial state. Then string will be accepted by FA.
- Language accept \rightarrow A Finite Automata is said to accept a language if all the strings in the language are accepted and the strings not in the language are rejected.

[Ex-2]

Construct a DFA which accepts all the strings $\{a, b\}$ where the string length is ≥ 2 . $|w| \geq 2$

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

$$L = \{aa, ab, ba, bb, aaa, aba, baa, \dots \dots \}$$



State transition Diagram,

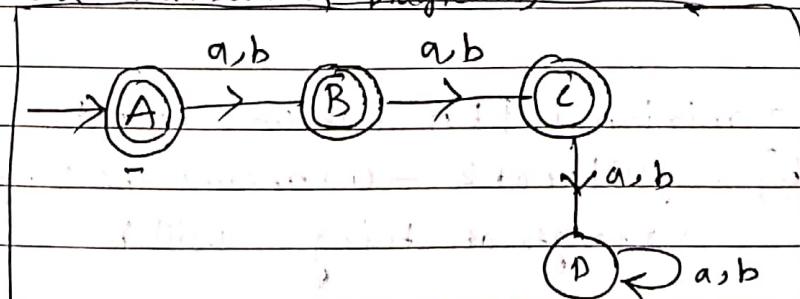
[Ex-3]

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

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

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

State transition Diagram,



Ex-9 $w \rightarrow \text{string}$

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

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

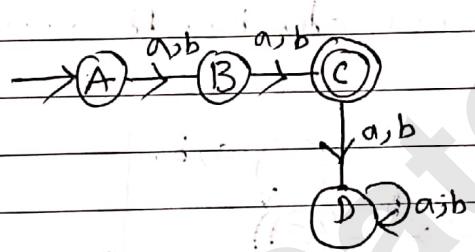
When, $|w| > n$, then
no of state required $(n+1)$.

When, $|w| \leq n$, then

no of state required $(n+2)$

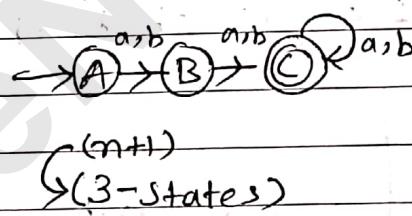
• $|w|=2$

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



• $w \geq 2$

$$L = \{aa, ab, aua\dots\}$$

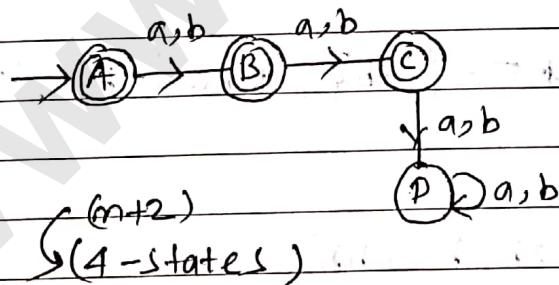


$(n+2)$

↙ (4-states)

• $|w| \leq 2$

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



$(n+2)$

↙ (4-states)

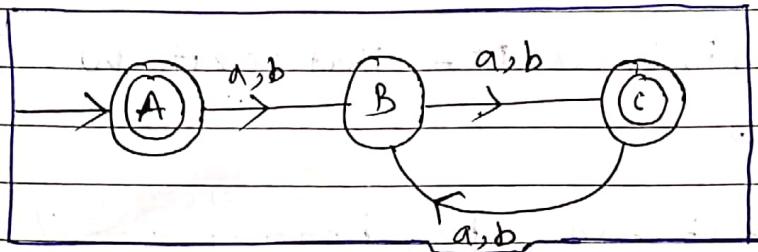
Example)-4

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

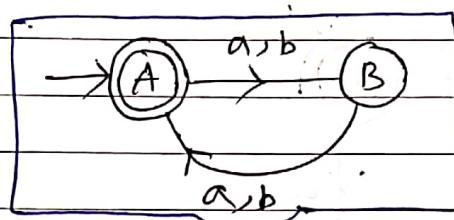


$$\Sigma = \{a, b\}$$

$$L = \{a^i b^j \mid i, j \geq 0, i \neq j\}$$



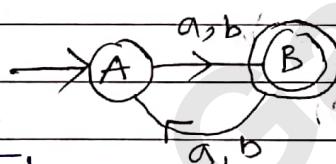
Q76



(State transition Diagram)

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

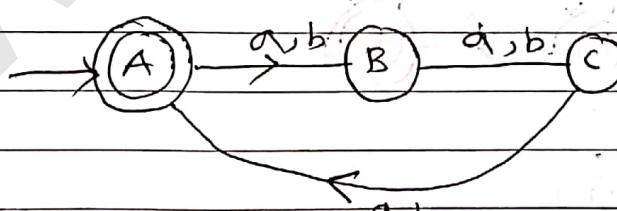
$$\rightarrow L = \{a^i b^j, aaaa, bbbb, \dots, aaaaa, \dots\}$$



[Ex-6],

$$[|w| \bmod 3 = 0]$$

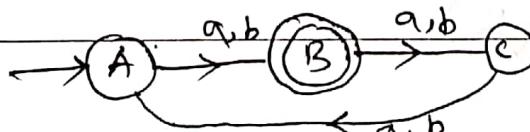
$$\rightarrow L = \{aabb, aaaa, aba, bbbb, \dots\}$$



final state 'A'.

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

$$\rightarrow L = \{a, b, aaaa, bbbb, \dots\}$$



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

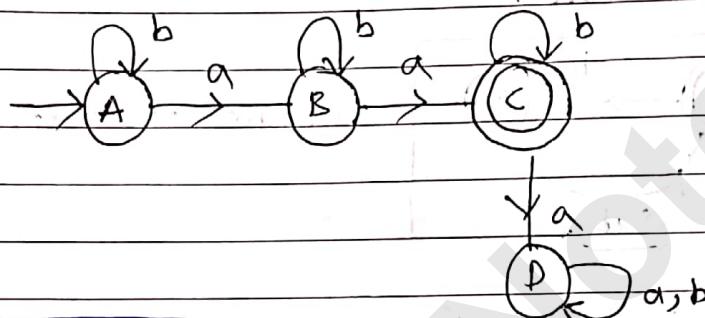
Example - 8

construct a minimal DFA, that accept $w \in \{ab\}^*$

1. $n_a(w) = 2$

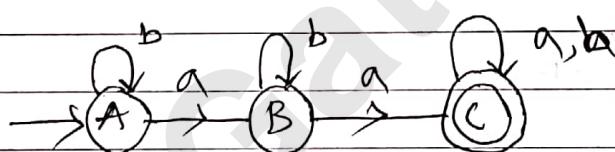


$L = \{aa, aab, baa, aba, bbaa, \dots\}$



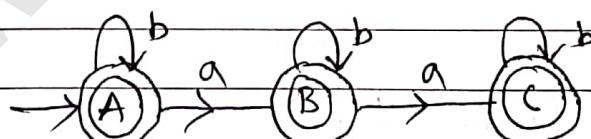
2. $n_a(w) \geq 2$

→ $L = \{aa, aaab, aaaaaba, \dots\}$



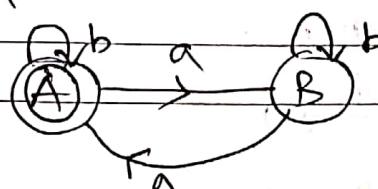
3. $n_a(w) \leq 2$

→ $L = \{e, a, b, aa, ab, ba, bb, aabb, \dots\}$



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

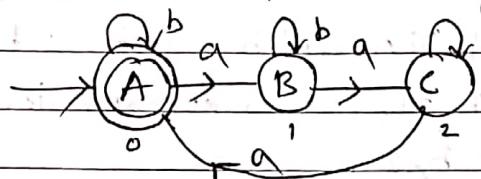
→ $L = \{aa, aab, aabb, aaaa, aaaaaa, \dots\}$



5. $na(w) \bmod 3 = 0$

$\rightarrow L = \{aaa, \dots, aaaaaaaaa, \dots\}$

state transition diagram -

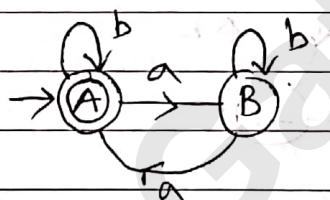


6. Construct a minimal DFA, where $w \in \{a, b\}^*$

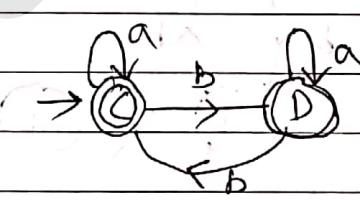
$$\left[\begin{array}{l} na(w) \equiv 0 \pmod{2} \\ \& \& \end{array} \right]$$

$$\left[\begin{array}{l} nb(w) \equiv 0 \pmod{2} \end{array} \right]$$

$\rightarrow L = \{\epsilon, aa, bb, aabb, abab, bbaaaaa, \dots, bbbb\dots\}$



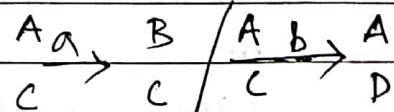
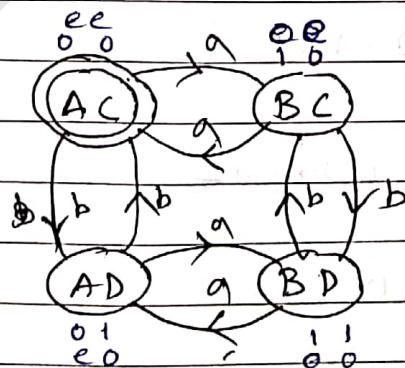
of state {A, B}



of state {C, D}

Cross product method -

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



$e_{ave} = \{ee\}$ (final state)

$e_{one} = \{ee, e0, 0e\}$ (final state).

**

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

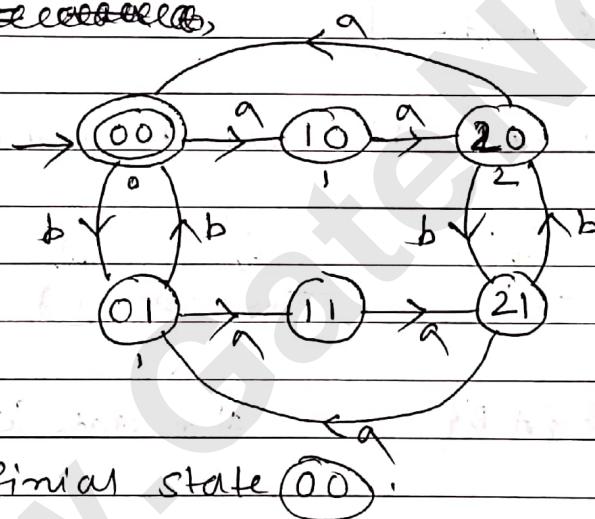
- ⑦ Construct a minimal DFA which accepts set of all strings over $\{a, b\}$ in which no of 'a's are divisible by 3 and no of 'b's are divisible by 2.

$$\Rightarrow W \in \{a, b\}^*$$

$$n_a(W) \equiv 0 \pmod{3}$$

$$n_b(W) \equiv 0 \pmod{2}$$

~~Experiments,~~



$$\text{When, } n_a(W) \pmod{3} > n_b(W) \pmod{2}$$

$$\text{final state} = \{10, 20, 11, 21\}$$

or

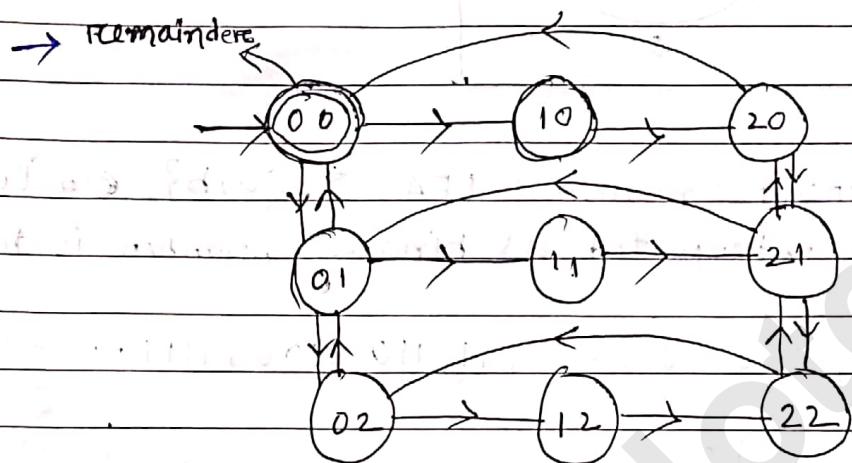
$$\text{When, } n_a(W) \pmod{3} = n_b(W) \pmod{2}$$

$$\text{final state} = \{00, 11\}$$

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

$$n_a(w) \equiv 0 \pmod{3}$$

$$n_b(w) \equiv 0 \pmod{3}$$



when, $n_a(w) \pmod{3} = 1$

&

$n_b(w) \pmod{3} = 2$, then

final state = $\{1, 2\}$

when, $n_a(w) \pmod{3} > n_b(w) \pmod{3}$, then

final state = $\{10, 20, 21\}$

→ $n_a(w) \equiv 0 \pmod{n}$

$n_b(w) \equiv 0 \pmod{m}$

Then the minimum no of states in automata is ' $m \times n$ '.

[example-9] - ①

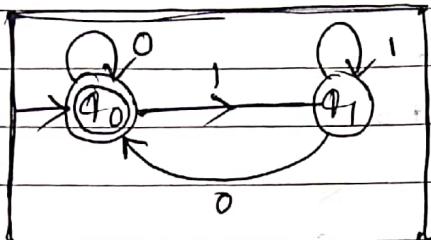
construct a minimal DFA, which accepts set of all string over $\{0, 1\}^*$, which when interpreted as binary number is divisible by '2'.

→ $\Sigma = \{0, 1\}^*$

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

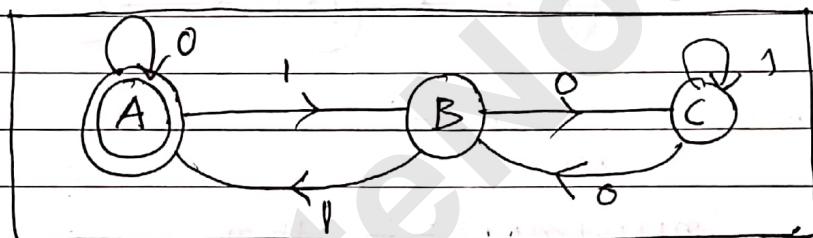
If we divide any no by 2 remainder can be '0' or '1'.

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



- (2) construct a minimal DFA, $\Sigma = \{a, b\}$, ~~where~~ $w \in \{a, b\}^*$
when interpreted as binary numbers is divisible by 8.

$$\rightarrow L = \{0, 00, 000, \dots, 11, 110, 1100, 1111, \dots\}$$



→ Finite Automata Can represent in two ways
 ↗ state Diagram transaction Diagram.
 ↗ State transition table.

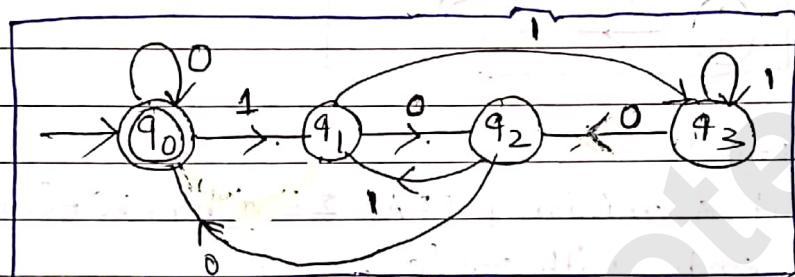
State transition table,

	0	1
→ *A	$A \rightarrow B$	
B,	$C \rightarrow A$	
C	$B \rightarrow e$	

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

$$\rightarrow \Sigma = \{0, 1\}, \text{ where } w \in \{0, 1\}^*$$

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



State transition table -

	0	1	→ If the remainder is = 0 then FS → q0
q0	q0 → q1		
q1	q2 → q3		remainder is = 1 FS → q1
q2	q0 → q1		remainder is = 2 FS → q2
q3	q2 → q3		remainder is = 3 FS → q3

Example-10

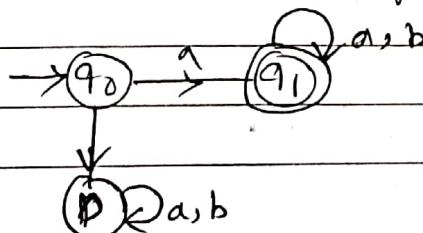
- Construct a minimal DFA, which accepts set of all strings over $\{a, b\}$, where each string starts with an 'a'.

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

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

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

state transition Diagram,

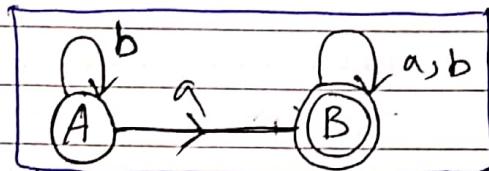


→ If the language is infinite
then take the smallest string and
make its skeleton.

[Example - 11]

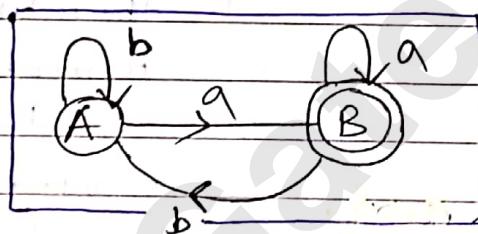
Construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$
such that each string contains 'a'.

$$\rightarrow L = \{a, aa, ab, ba, aaa, \dots\}$$

**[Example - 12]**

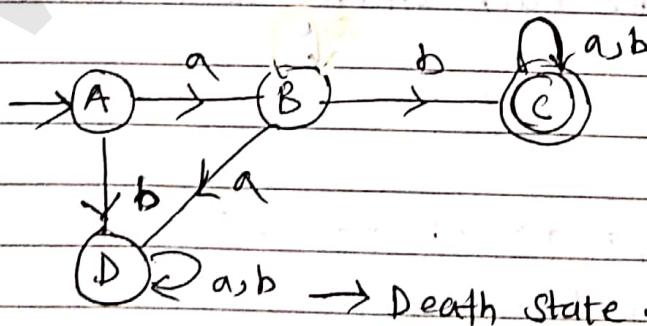
Construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$
String ends with an 'a'.

$$\rightarrow L = \{a, aa, ba, aaa, bba, baa, bbb, \dots\}$$

**[Example - 13]**

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

$$\rightarrow L = \{ab, aab, abb, \dots\}$$

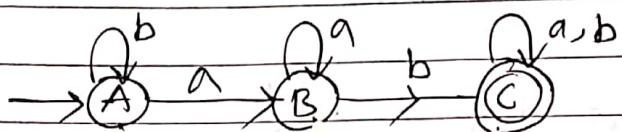


Example - 14

DFA, containing ab.

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

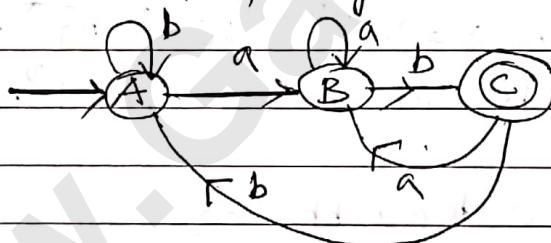
$$L = \{ab, abb, bab, \dots\}$$

**Example - 15**

~~end~~ construct a minimal DFA, ends with "ab".

$$\rightarrow L = \{ab, bbab, cab, \dots\}$$

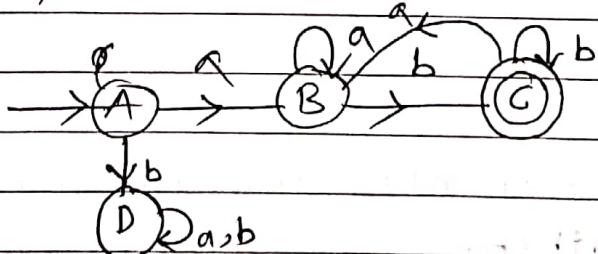
State transition diagram-

**Ex - 16**

construct a minimal DFA. starts with 'a' and ends with 'b'.

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

$$L = \{ab, aabb, abab, aabbab, \dots\}$$



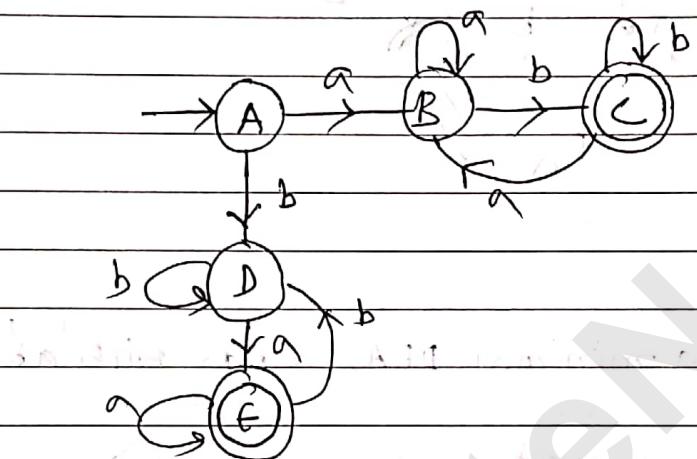
[Ex-17]

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

cl start and ends with different symbol".

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

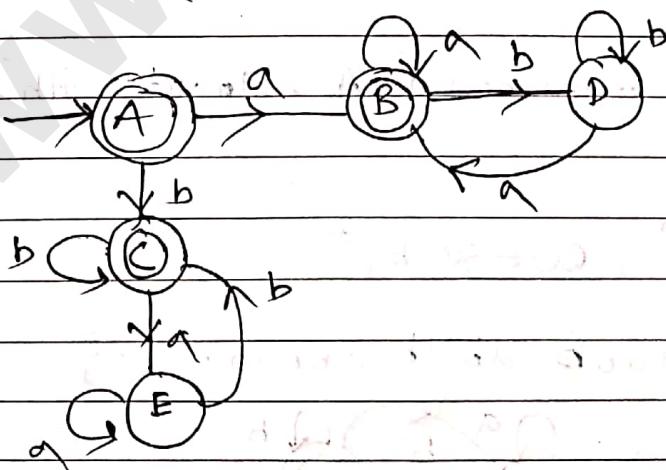
$$L = \{ab, ba, a bb, bba, \dots\}$$

**[Ex-18]**

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

cl start and ends with same symbol".

$\rightarrow L = \{aa, \dots, a, b, a a, b b, \dots\}$



\rightarrow here, Example - 17 and 18 are complements of each other.

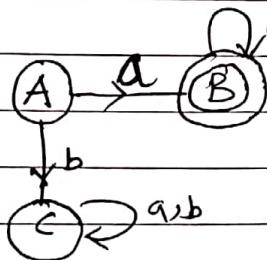
Ex-19

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

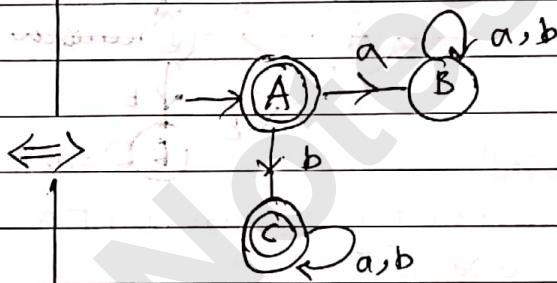
$$L_1 = \Sigma^* - L_2$$

Complementation of DFA :

$L_1 = \{ \text{Starting with } 'a' \}$



$L_2 = \{ \text{not starting with } 'a' \}$



$$L_1 = L_2$$

→ Complementation method apply for only DFA not for NFA.
will be changed

→ Every thing has to be same only changing final state.

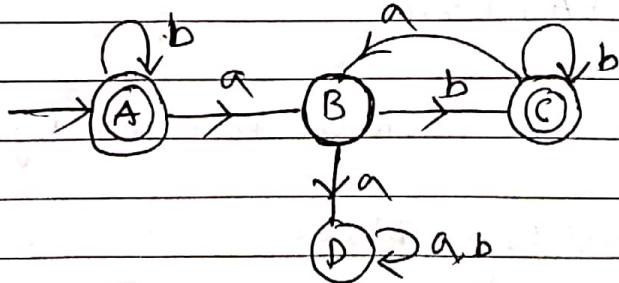
Ex-19

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

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

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

State transition Diagram -



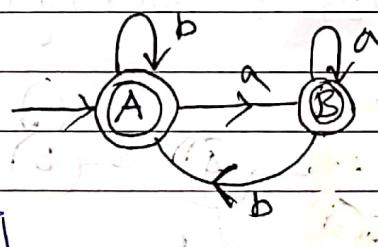
EX-20

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

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

$$\rightarrow L = \{\epsilon, a, b, aa, ba, bb, aaa, bba, \dots\}$$

State transition diagram -

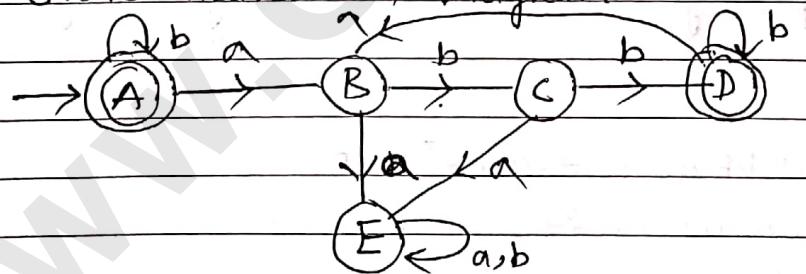
**EX-21**

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

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

$$\rightarrow L = \{\epsilon, abb, \dots\}$$

State transition diagram -



$$\text{final state} = \{A, D\}$$

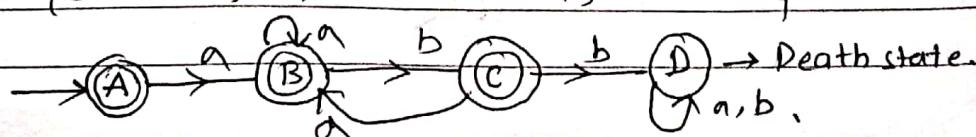
EX-22

Construct a DFA, where.

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

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

$$\rightarrow L = \{\epsilon, a, aa, ab, ba, bb, aaa, \dots\}$$

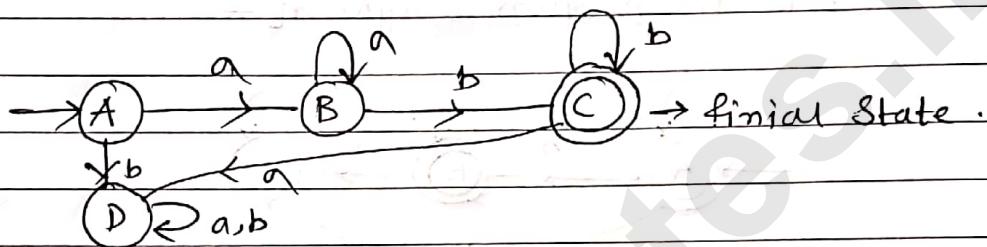


Ex-23

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

$$\rightarrow L = \{ab, aab, aabb, aabb\dots\}$$

State transition diagram-

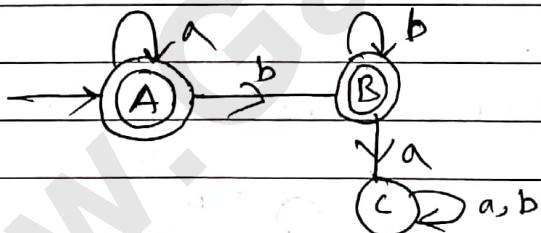


Ex-24

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

$$\rightarrow L = \{e, a, aa, aaa, \dots, b, bb, bbb\dots, ab, aabb\dots\}$$

State transition Diagram-

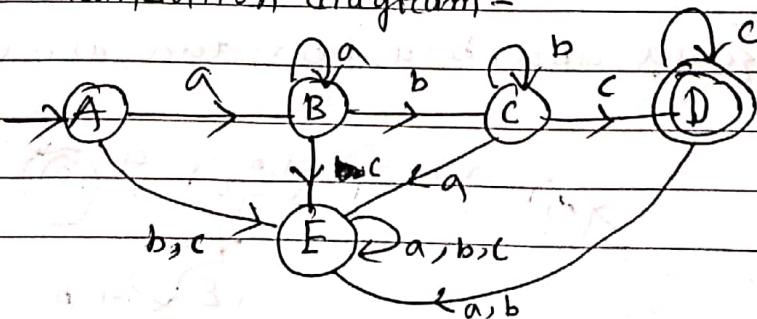


Ex-25

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

$$\rightarrow L = \{abc, aabbcc\dots\}$$

State transition diagram-

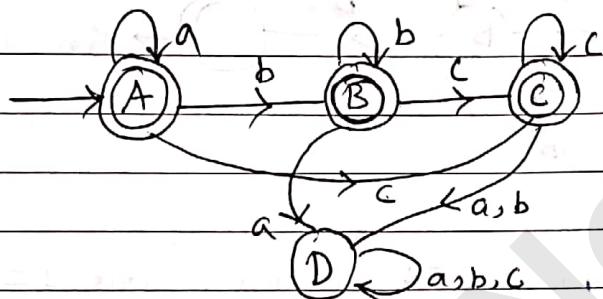


Ex-26

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

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

State transition Diagram -

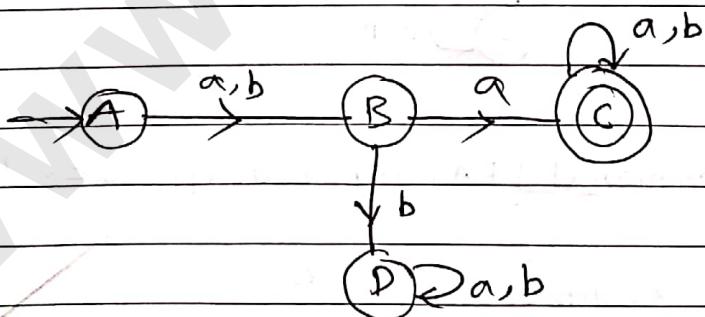


Ex-27

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

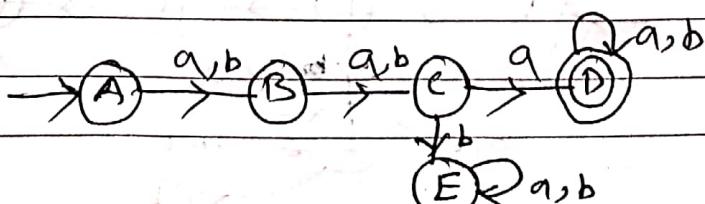
$$\rightarrow \Sigma = \{a, b\}$$

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



Ex-28 3rd symbol from L.H.S is 'a'.

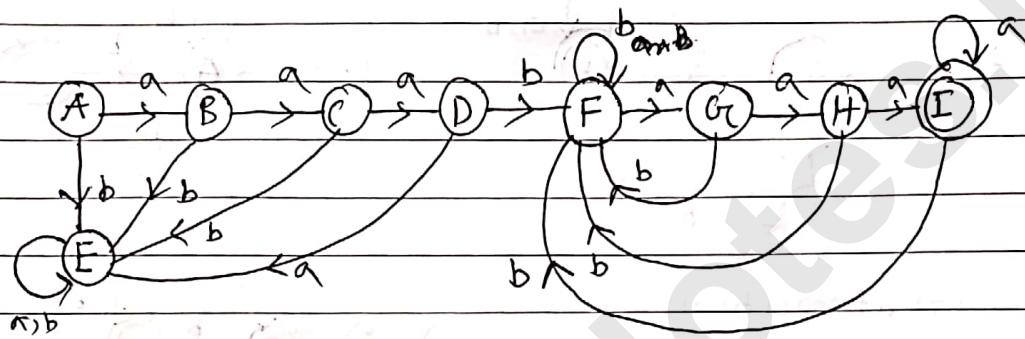
$$\rightarrow L = \{aaa, aba, baa, bba, aaa, abaa, \dots\}$$



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

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

$$\rightarrow L = \{a^3b \in a^3, a^3baa^3, a^3bba^3, \dots\}$$



• OPERATION ON DFA like -

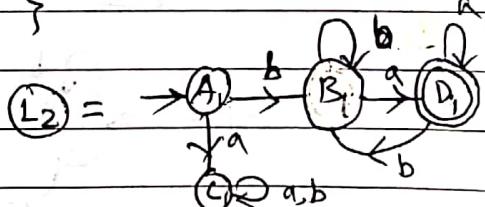
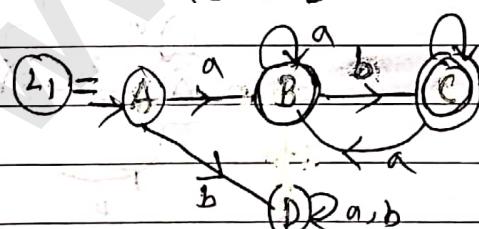
- (i) Union (ii) Concatenation (iii) cross product (iv) complementation.
- (v) Reversal.

① Union :-

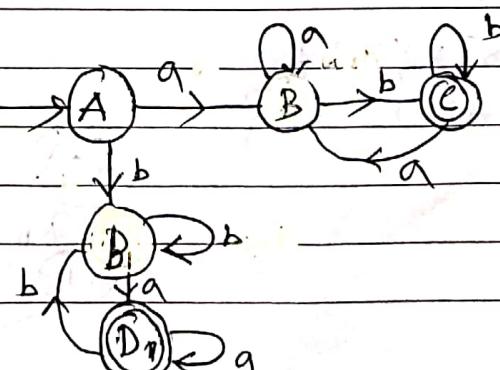
Ex - starts and ends with different symbol.

$$\rightarrow L_1 = \{ab, aab, aba, abb, \dots\}$$

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



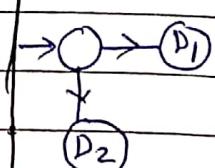
$$L_1 \cup L_2 =$$



$$L_1 = D_1$$

$$L_2 = D_2$$

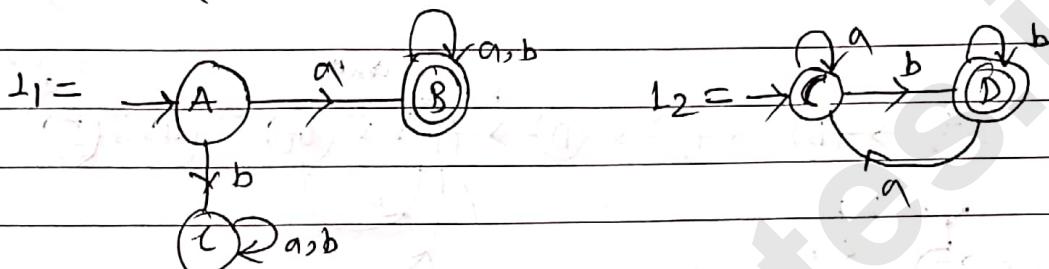
$$L_1 \cup L_2$$



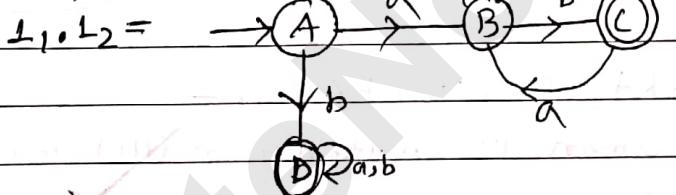
② Concatenation — $L_1 = D_1, L_2 = D_2 / L_1 \cdot L_2 = D_1 \cdot D_2$.
 → starting with 'a' and ending with 'b'.

$$L_1 = \{a, aa, ab, aab, \dots\}$$

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



Concatenation,



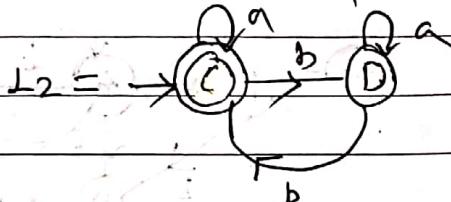
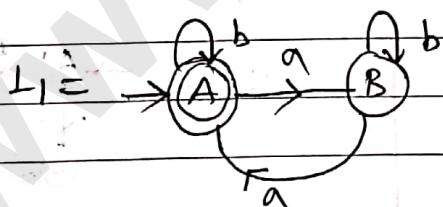
③ Cross product method —

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

H

$$L_1 = \{aa, taa, aab, aaaa, aaaaa, \dots\}$$

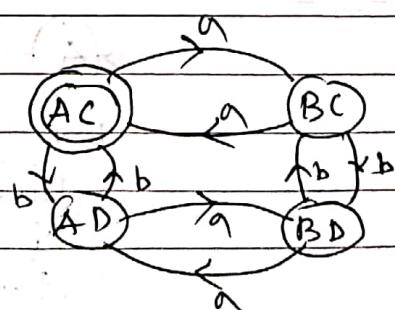
$$L_2 = \{t, bb, bba, bbbb, bbb.bbbb, \dots\}$$



$L_1 \times L_2$

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

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



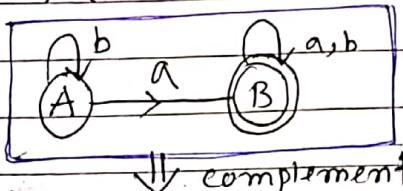
(4) Complement :-

→ Does it not contain 'a'.

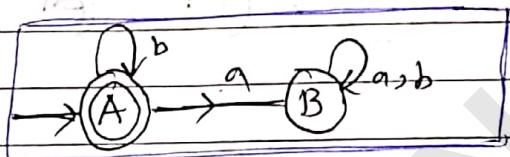
$$L_1 = \{ \text{containing } 'a' \}$$

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

$$\bar{L}_1 = \{ e, b, bb, bbb, bbbb \dots \}$$



↓ complement

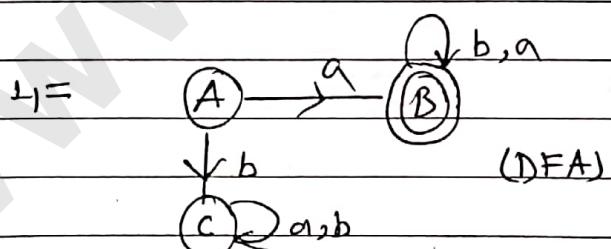
(5) Reversal :-

→ $awb \Leftrightarrow bwa$.

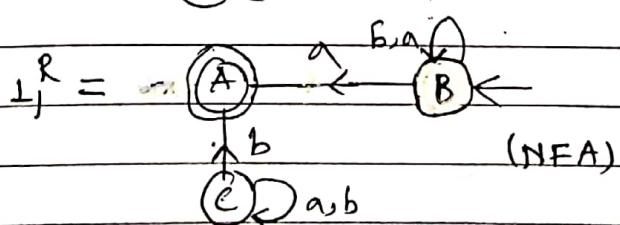
$$L_1 = \{ \text{start with } a \}$$

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

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



(DFA)



(NFA)

**

$$L_1 \rightarrow \$(\text{DFA})^R \rightarrow L_1^R \rightarrow \text{NFA/DFA}$$

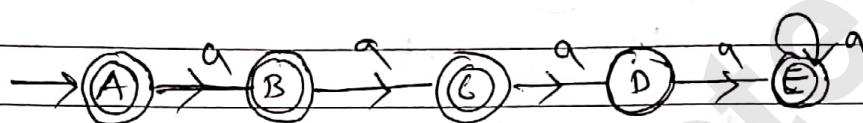
(30) Ex-30

Construct a minimal DFA over $\{\alpha\}$.1. For $\{a^n / n \geq 0, n! = 3\}$ 2. For $\{a^n / n \geq 0, n! = 2, n! = 4\}$.

$$\rightarrow \Sigma = \{\alpha\}$$

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

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

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

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



→ In NFA, not need to show death state.

classmate

Date _____
Page 37

NFA-1

• NFA (Non-Deterministic Finite Automata):

→ Every DFA is NFA.

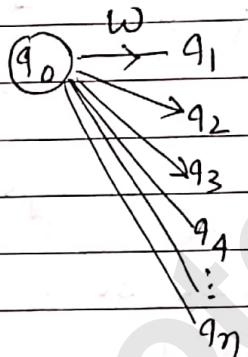
$$S: \alpha^* \rightarrow Q$$

DFA



$$S: \alpha^* \rightarrow 2^Q$$

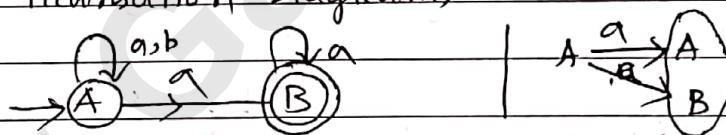
NFA



Ex-1 Construct an NFA which accepts all strings over {a, b} such that L = {ends with 'a'}

$$\rightarrow L = \{a, aa, ba, aaa, baa, \dots\}$$

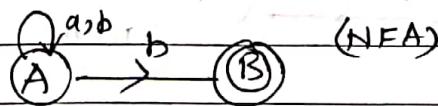
State transition Diagram,



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

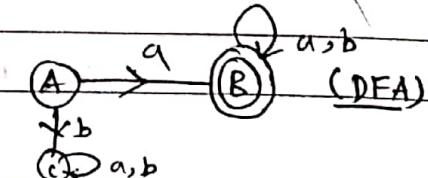
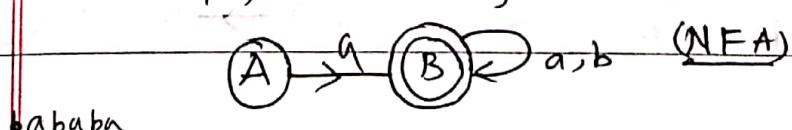
Ex-2 L = {ending with b}

$$\rightarrow L = \{b, ab, abb, \dots\}$$



Ex-3 L = starting string starts with 'a'

$$\rightarrow L = \{a, aa, ab, \dots\}$$

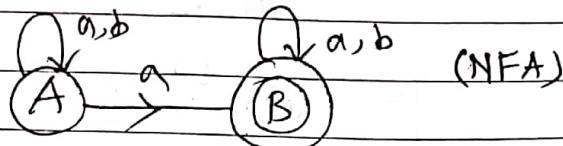


bababa

[Ex-3]

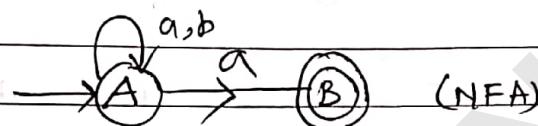
$L = \{ \text{set of all strings containing } 'a' \}$.

$$\rightarrow L = \{ a, aa, ab, ba, aaa, aab, \dots \}$$

**[Ex-4]**

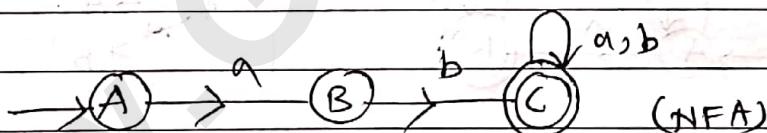
$L = \{ \text{set of all strings ends with } 'a' \}$.

$$\rightarrow L = \{ a, aa, ba, \dots \}$$

**[Ex-5]**

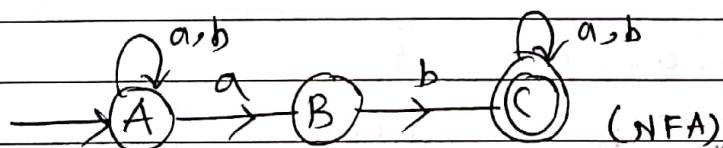
$L = \{ \text{all string start with } 'ab' \}$.

$$\rightarrow L = \{ ab, abb, abab, \dots \}$$

**[Ex-6]**

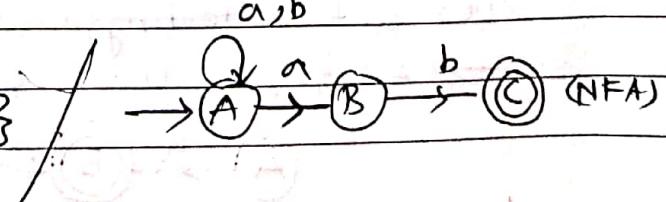
$L = \{ \text{set of all strings contains } 'ab' \}$.

$$\rightarrow L = \{ ab, abb, aabb, \dots \}$$

**[Ex-7]**

$L = \{ \text{ending with } 'ab' \}$.

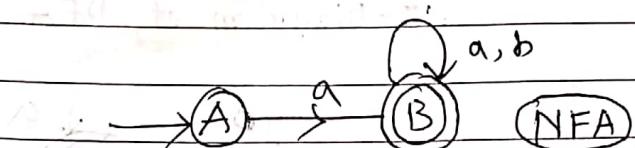
$$\rightarrow L = \{ ab, aab, bab, \dots \}$$



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

$$\rightarrow L = \{ a, aa, ab, aab, aba, \dots \}$$

State transition Diagram of NFA.



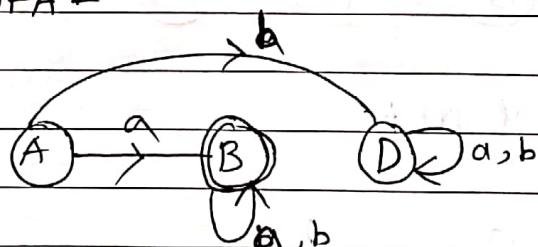
STT of NFA -

	a	b
$\rightarrow A$	B	\emptyset
* B	B	B

STT of DFA -

	a	b
$\rightarrow A$	B	D
* B	B	B
D	D	D

STD of DFA -

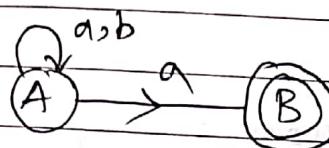


Example - 2

Conversion of NFA to DFA :

$L = \text{"all strings ends with } a\text{"}$.

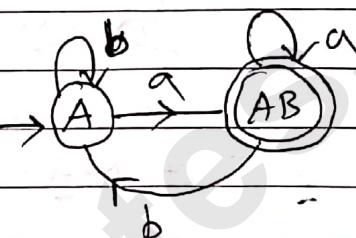
$$\rightarrow \{ a, aa, abba, aad, aba, \dots \}$$

ST-Diagram of NFA -ST-table of NFA

	a	b
A	{A, B}	{A}
B	{Ø}	{Ø}

ST-Table of DFA

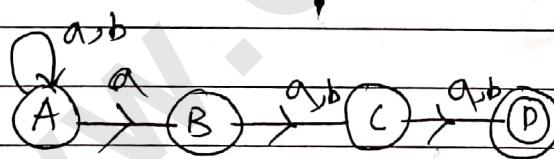
	a	b
→ A	[AB]	[A]
* [AB]	[AB]	[A]

ST-Diagram of DFA(Ex-2)

Conversion of NFA to DFA,

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

$$\rightarrow L = \{ \text{aaa, abb, aba, baa, ---} \}$$

ST-D of NFAST-Table of NFA

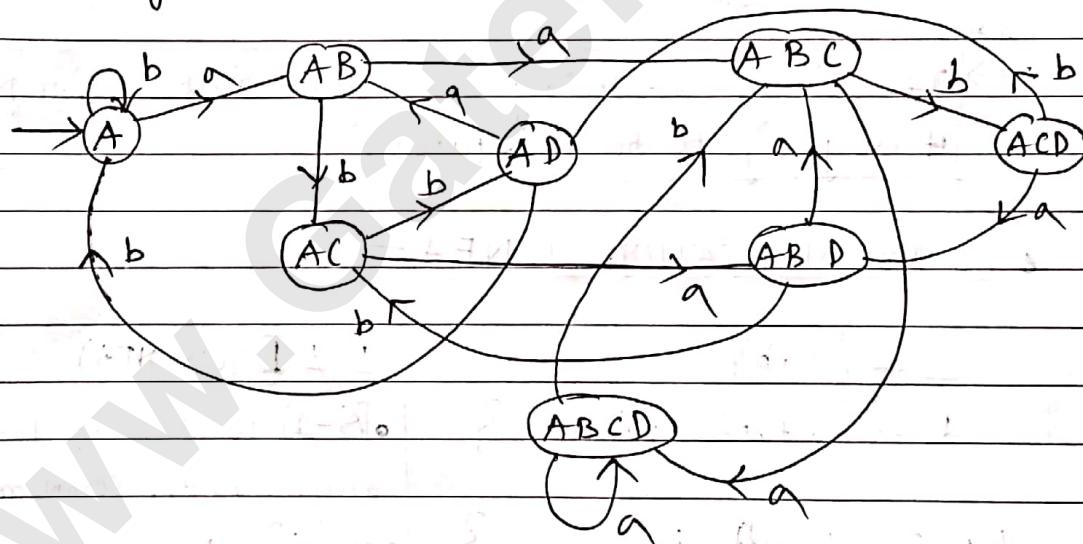
	a	b
→ A	{A, B}	{A}
B	{C}	{C}
C	{D}	{D}
* D	{Ø}	{Ø}

JT-Table of DFA from S-TT of NFA -

	a	b
$\rightarrow [A]$	$[A, B]'$	$[A]'$
$[A, B]$	$[ABC]'$	$[AC]'$
$[AC]$	$[ABD]'$	$[AD]'$
* $[AD]$	$[AB]'$	$[A]'$
$[ABC]$	$[ABCD]'$	$[ACD]'$
* $[ABD]$	$[ABC]'$	$[AC]'$
* $[ACD]$	$[ABD]'$	$[AD]'$
* $[ABCD]$	$[ABCD]'$	$[ACD]'$

Ans
Y/N
contd

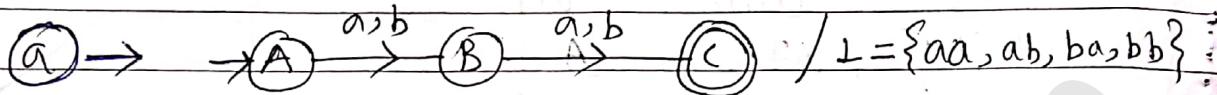
ST-Diagram of DFA from above JTT of DFA -



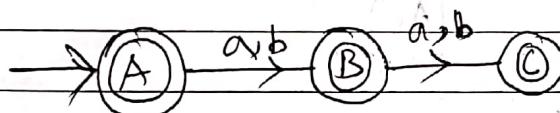
** If an minimal NFA contain ' n ' states then an DFA contain 2^n states in worst case.
 $n \rightarrow 2^n$

Ex-1 NFA for string of length -

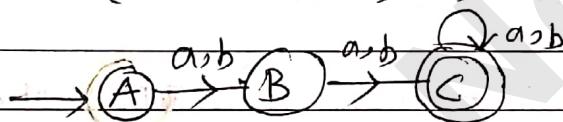
- a) exactly 2 . b) at most 2 c) at least 2



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



$\text{c) } \rightarrow L = \{aa, ab, ba, bb, aaa, \dots\}$



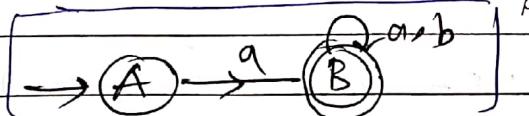
** \rightarrow If we have a string of length 'n' then for NFA it is going to be 'n' states.

COMPLEMENTATION of NFA -

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

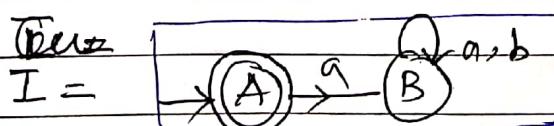
$$L = \{ \text{starts with } a \}$$

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



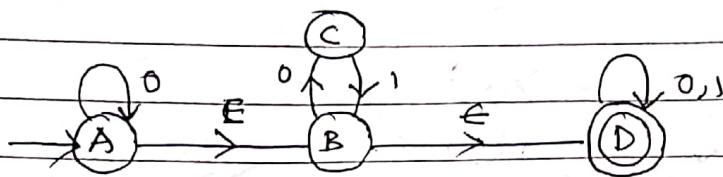
$$L \neq L_1 (\text{in NFA})$$

Q-1] What is the language accepted by the complement of NFA.
 $\rightarrow \{\epsilon\}$



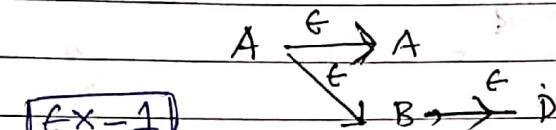
Q-2] What is the complement of language accepted by NFA.
 $\rightarrow \{\epsilon, b, bb, bbb, \dots, ba\}$

Epsilon NFA :- (ε-NFA)

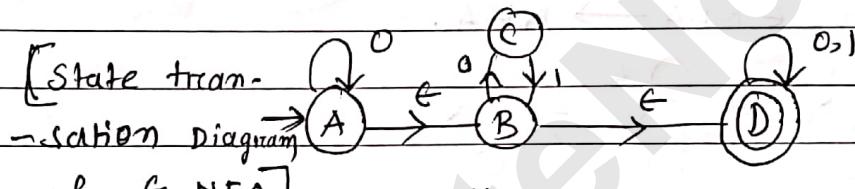


→ For ϵ -NFA : $Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

→ ϵ -closure (A) = {A, B, D}



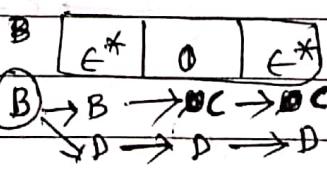
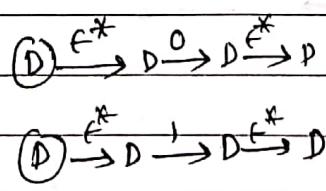
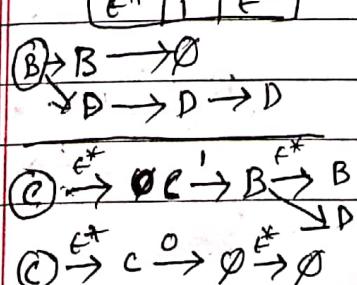
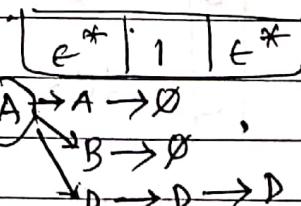
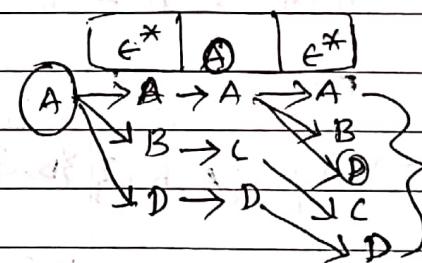
• Conversion ε-NFA to NFA:



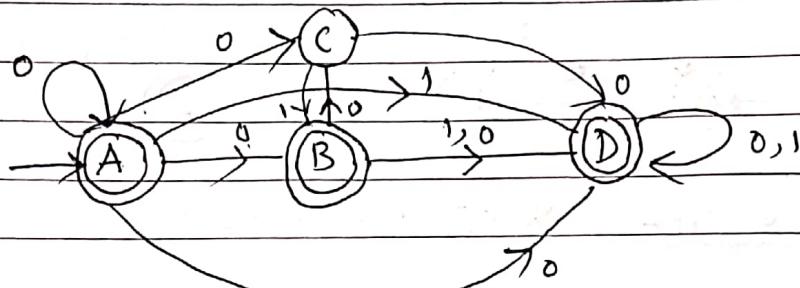
↓ conversion NFA

State transition table of NFA:

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



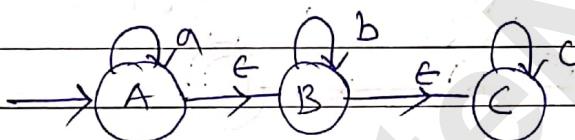
from the state transition diagram - we



- 'A' is going to 'D' by seeing '0' so 'A' will be the final state.
- 'B' is going to 'D' " " " 0 so 'B' " " " .

Ex-2

conversion ϵ NFA \rightarrow NFA.

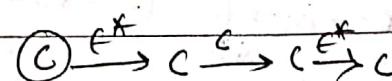
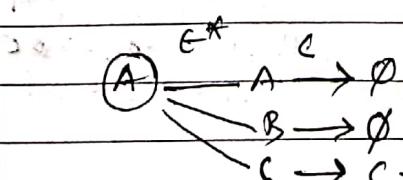
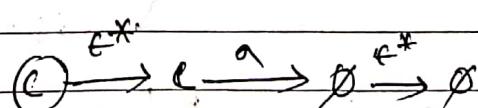
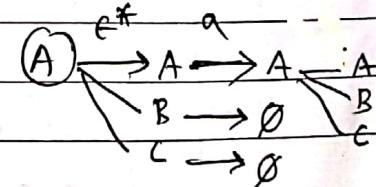
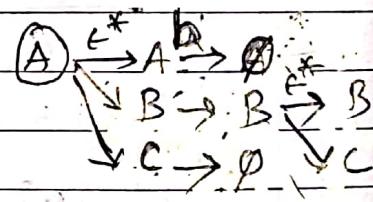


(ϵ NFA)

↓ conversion.

transformation table :-

	a	b	c
A	{A, B, C}	{B, C}	{C}
B	{Ø}	{B, C}	{C}
C	{Ø}	{Ø}	{C}



To Get Full Content

Click of the below link given
in this page



www.GateNotes.in

or

visit:www.gatenotes.in