Date : 12.02.2024

Theory of computation (TOC)

Book : An Introduction to Formal Language and Automata
– Peter Linz.

- Theory of computation
  - Theory → complexity, capabilities
  - computation → machine

Set :-

A set is a collection of elements, without any structure other than membership.

$\therefore S = \{1, 2, 3\}$ $\quad [2^n = 2^3 = 8]$

$2^S = \{\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{\phi\}\}$

- $\cup$ = Union (everything)
- $\cap$ = Intersection (only common)
- $-$ = Difference

compliment = $\overline{S}$

Universal set = U

$A = \{1,2,3\}$, $B = \{1,2\}$.

$\therefore A - B = \{1,2,3\} - \{1,2\}$

$= \{3\}$

Graph Vs Tree

Graph
- Cycle
- Set of vertices/nodes and edges

Tree
- No cycle
- Set of nodes and edges.

Symbol :- A small unit of letter/character. Alpha numeric, individual letter.

 # Alphabet :- Set of possible Symbol (Σ).

 # Language :- Consists of all possible set of string from Alphabet.

 # Grammer :- Set of string from Alphabet

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

$L_1 = \{$All possible strings that starts with a$\}$

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

∴ Infinite Language.

2 kind of language —

① Finite language.

② Infinite language.

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

$L = \{$All strings over Σ that lengths 2$\}$

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

∴ Finite Language.

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

$L^1 = \{a, b\}$

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

$L^3 = \{aba, aab, aaa, bbb \ldots\ldots\}$

$L^1 = 1$ length-ৰ string

$L^2 = 2$ . . .

$L^3 = 3$ . . .

$L^4 = 4$ . . .

String a = "hello"

String b = "naeem"

a+b = hello naeem..

A={1,2,3}, B={2,3}

∴ A×B = (1,2), (1,3), (2,2), (2,3), (3,2), (3,3)

↓ cartesian Product

$L_1 . L_2 = \{a, b\}$

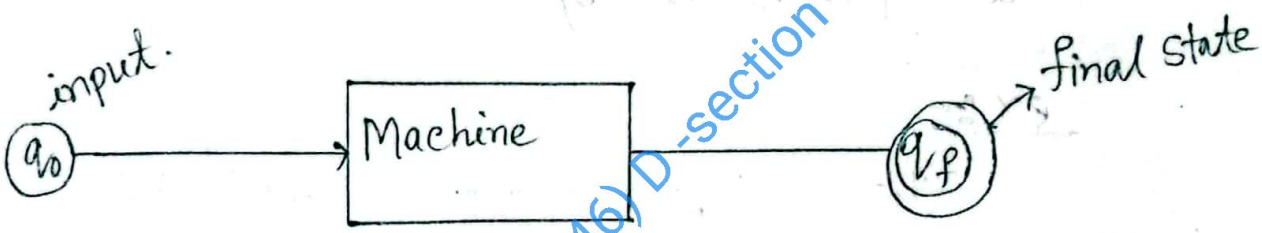
$L_1 = \{a\}$, $L_2 = \{b\}$

- L^* → kleene closure → All possible String including empty String.
- $L^* \Longrightarrow \{a, b, ab, aa, ba, bb, \in/\lambda\}$
- L^+ ⟶ All possible String excluding empty String
- $L^+ = \{a, b, ab, aa, ba, bb\}$

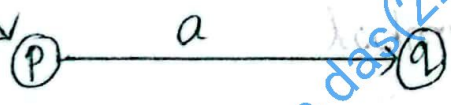
$L_1 \times L_2$ = Cartesian Product.

$L_1 . L_2$ = Concatent অর্থাৎ, যোগ।
1st string-এর আগে 2nd string বা, পরের টা যোগ হয়ে যাবে।

Automata :- One kind of machine that can takes input and go to the final state.



$\delta(p,a) = q$



δ = Transition State

q_0 = initial state / starting state

q_f = final state

* যে state থেকে return করা যায় না, তাকে dead state বলে।

Symbol → basic black of TOC.

Alphabet → Set of infinite Symbol.

String → Sequence of Symbol from Alphabet

Language → Set of all possible String

Date - 15.02.2024

Automata :-

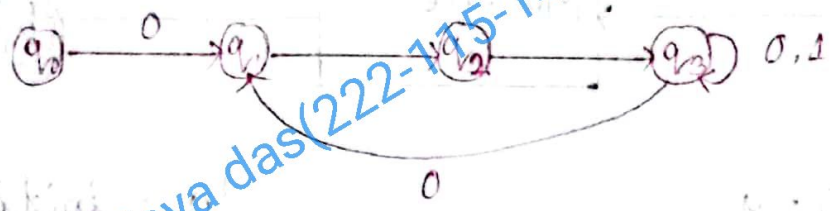
A finite automata consist of 5 tuples.

$(Q, \Sigma, \delta, q_0, F)$.

$Q \rightarrow$ set of all possible state.

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

ex :-



$\Sigma \rightarrow$ Set of Input Symbol

$= \{0, 1\}$

$\delta =$ Transition state

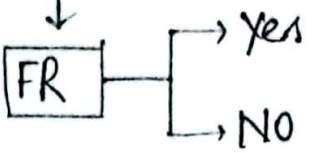
$q_0 =$ Initial state / Starting state.

$F =$ Final state

$\downarrow$

q_3.

Infinite Language



Finite Automata 2 types —

(একটা State একাধিক)

① Deterministic finite Automata (DFA)

② Non-deterministic finite Automata (NFA)

(একের অধিক State.)

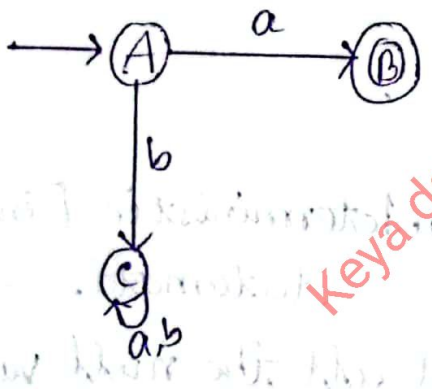
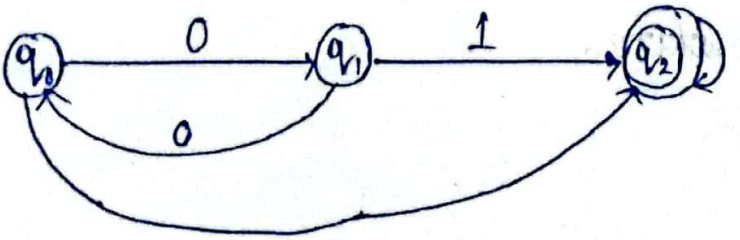


Fig:- Transition

Transition table:

Ps	a	b
→ A	*B	C
+B	∈	∈
C.	C	C

Final State বুঝানোর জন্য

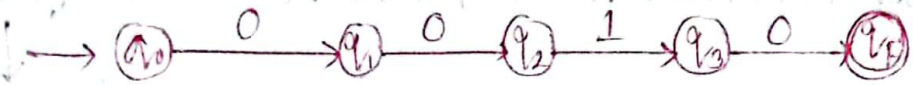


(NFA Diagram)

Transition table

Ps	0	1
→ q_0	q_1	$*q_2$
→ q_1	q_0	$+q_2$
↔ q_2	$*q_2$	$*q_2$

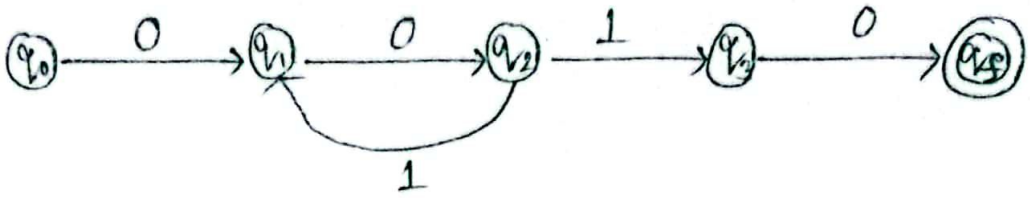
Alphabet ও String দেওয়া থাকবে,
$\Sigma = \{0,1\}$, 0010, DFA or NFA তৈরি হবে।



Finite Automata :-

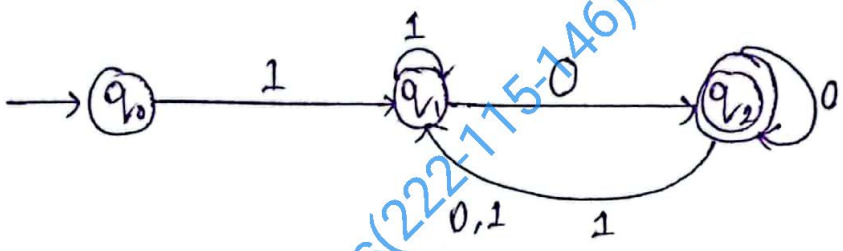
- DFA (Deterministic Finite Automata)
  - refers uniqueness of the computation.
  - Doesn't accept the Null value.
  - Support one state.

- NFA (Non-deterministic Finite Automata.
  - Accept all the null value
  - support one or more state.



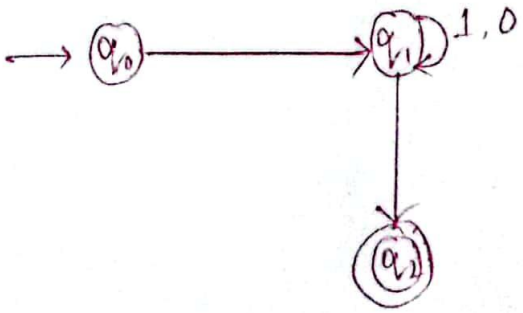
Ex:- DFA : $\Sigma = \{a, b\}$

L = set of all strings start with a.



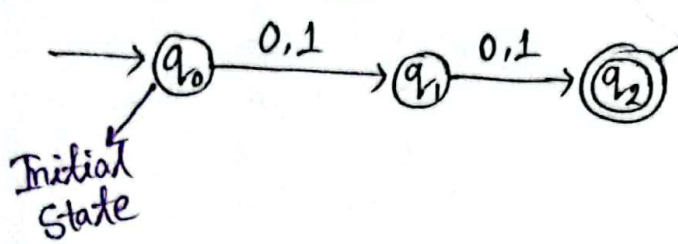
NFA Ex:- Design DFA that $\Sigma = \{0, 1\}$ accept the input starts with 1 and ends with 0.

Soln:-



Date: 22-02-2024

NFA Figure:-



NOTE:-

* DFA-তে একটি input-এর জন্য একটি state থাকতে হবে।
* NFA-তে একটি input-এর জন্য একাধিক state থাকতে হবে

Σ → Set of Alphabet

↓

ফিজমা

Transition table:-

Ps	0	1
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
$*q_2$	ϵ	ϵ

Ex:-1: L_1 = {set of all strings that ends with '0'}

Draw the NFA.

Soln:- NFA design:-

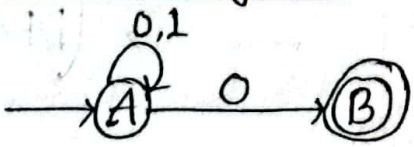
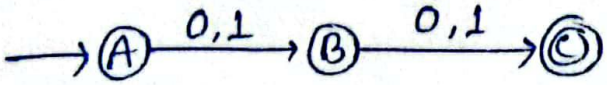


table:-

Ps	0	1
$\rightarrow A$	A, B	A
$*B$	ϵ	ϵ

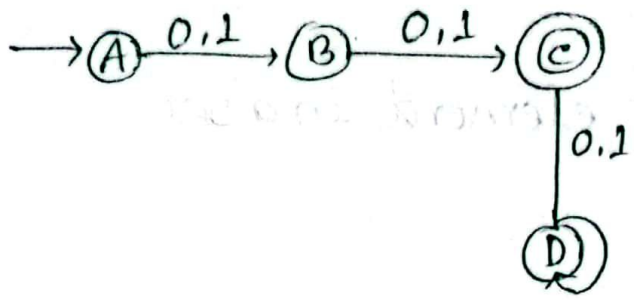
Ex-2:-



Now design the DFA Diagram?

Soln:-

DFA Diagram:-



Powers of Σ (Sigma):-

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

Σ^0 = {set of all strings of length 0}

$\therefore \Sigma^0 = \{\in\}$ → epsilon.

Σ^1 = {set of all strings of length 1}

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

Σ^2 = {set of all strings of length 2}

$\therefore \Sigma^2 = \{00, 01, 10, 11\}$

Σ^3 = {set of all strings of length 3}

$\therefore \Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

⋮

Σ^n = {set of all strings of length n}

Cardinality :-

- number of element in a set.
- $\Sigma^n = 2^n$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots\ldots$ infinite set

$= \{\epsilon\} \cup \{0,1\} \cup \{00, 01, 10, 11\} \cup \{\ldots\ldots\}$

= Set of all possible strings of all lengths over alphabet {0,1}

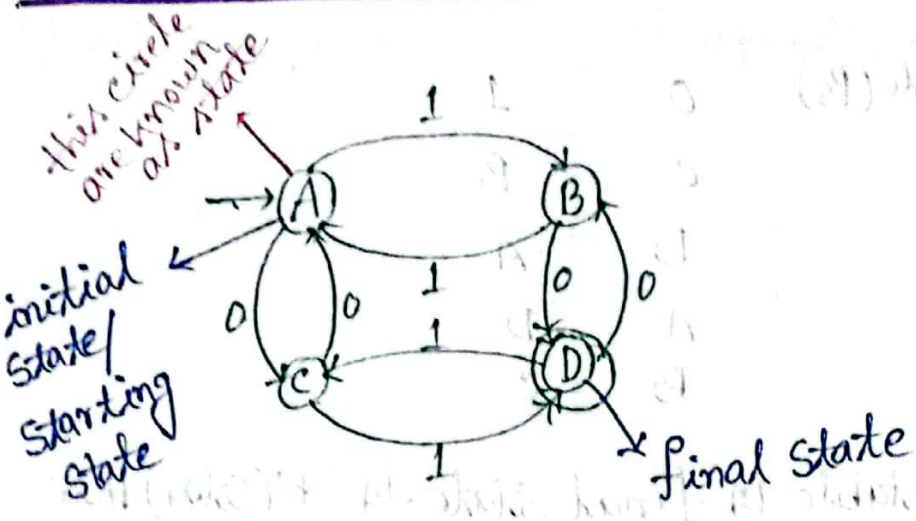
∴ this is an infinite set.

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup \ldots\ldots\ldots$

DFA (Deterministic Finite Automata) :-

- It is the simplest model of computation.
- It has a very limited memory.

Transition Diagram :-



* Whenever, you see a double circle, it means that is the final state of DFA

- Every DFA can be defined using 5 tuples,

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

$Q \longrightarrow$ Set of all states

Σ = inputs

q_0 = initial state/starting point/state

F = Final states

δ = Transition function from $Q \times \delta \longrightarrow Q$

Here,

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

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

$q_0 = A$

$F = \{D\}$

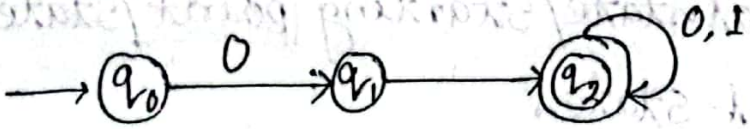
Transition Table :-

Present state (Ps)	0	1
→ A	C	B
B	D	A
C	A	D
*D	B	C

* Transition table-ত্ব final state-ক '*' (star) চিহ্ন দেয়া হয়।

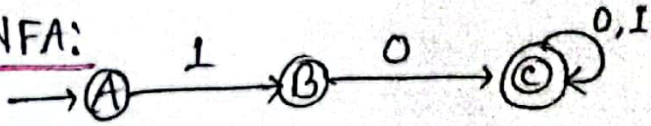
Ques:-01 :- L = { set of all strings that starts with '0' }. Draw the NFA?

Soln :- NFA :-



Ques-02 :- Set of all strings that starts with '10'. Design an NFA.

Soln :- NFA :

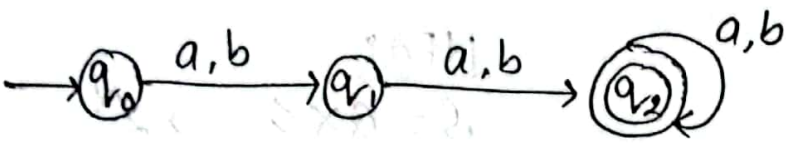


Date- 4 mareh. 24

L = {w/w is a string of length at least 2}

Soln: $\Sigma = \{a, b\}$ $|w| \geq 2$

$L = \{aa, ab, ba, bb, aaa, aab, aba, bbb, \ldots\}$



Transition table:

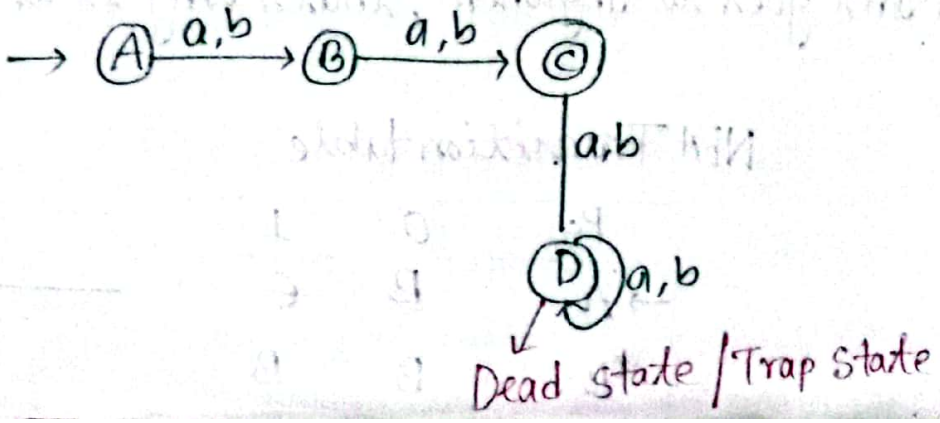
PS	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
$*q_2$	q_2	q_2

L = {w/w is a string of length at most 2}.

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

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

DFA:



Conversion of NFA TO DFA :-

- Every DFA is an NFA, but every NFA is not DFA
- Every NFA converted to DFA.

DFA :-

$\delta = Q \times \Sigma \rightarrow Q$

NFA :-

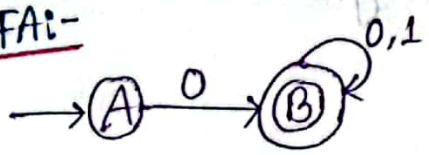
$\delta = Q \times \Sigma \rightarrow 2^Q$

NFA $\cong$ DFA

Ex-01 :- L = {Set of all strings with $\Sigma = \{0,1\}$ that starts with '0'}

Soln :- $\Sigma = \{0,1\}$

NFA :-



Here I can't mention for the state A if input 1 in taken it can't goes to anywhere, that's why it is a NFA.

NFA Transition table

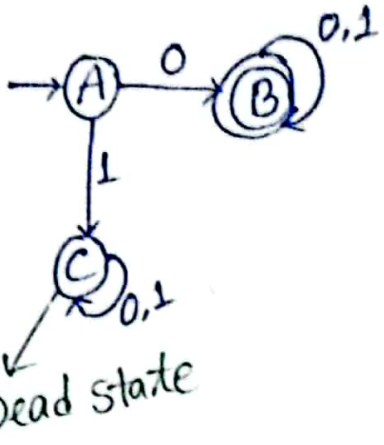
Ps	0	1
→ A	B	∈
*B	B	B

DFA Transition table :-

Ps	0	1
→A	B	C
*B	B	B
C	C	C

C - Dead state / Trap state

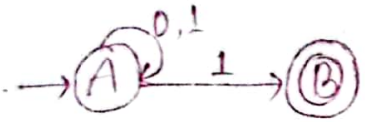
DFA Diagram :-



Ex-1 :- L = {Set of all strings over (0,1) that ends with '1'}

Soln :- $\Sigma = \{0, 1\}$

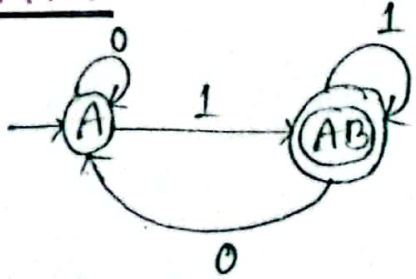
NFA :-



Transition table :-

Ps	0	1
→A	{A}	{A,B}
*B	∈	∈

DFA :-



Transition table :-

Ps	0	1
A	{A}	{AB}
AB	{A}	{AB}

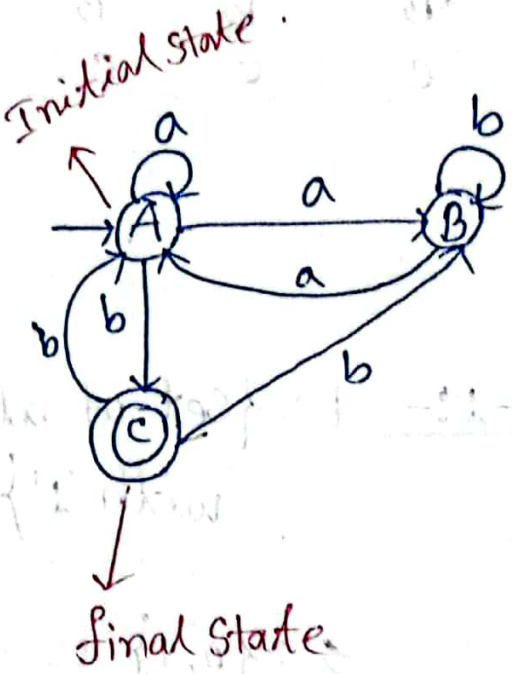
AB is not a state. It is a single state.

Date:- 06 march, 24

Find the equivalent DFA for the NFA given by $M = [\{A, B, C\}, (a, b), \delta, A, \{C\}]$. Where δ is given by.

NFA:

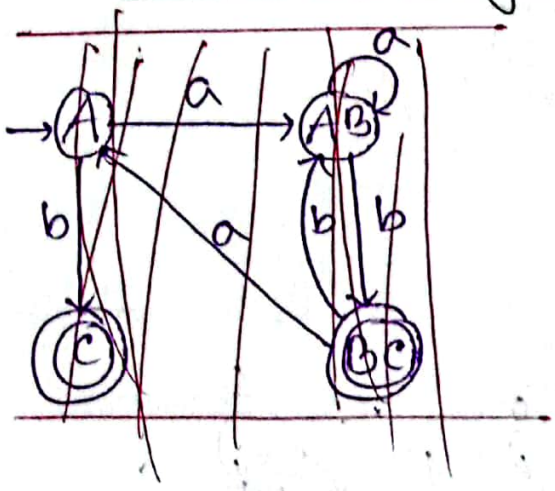
	a	b
→A	A, B	C
B	A	B
*(C)	∈	A, B



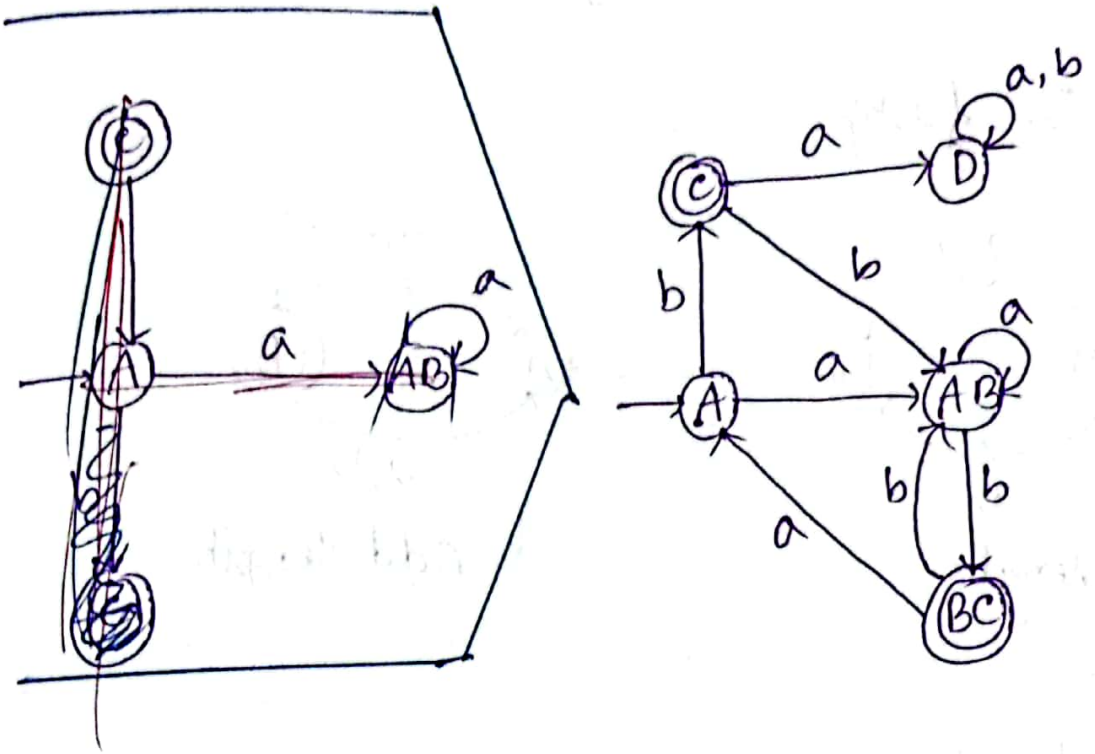
DFA Transition Table:-

Ps	a	b
→A	AB	C
AB	AB	BC
(BC)	A	AB
*(C)	D	AB
D	D	D

Transition Diagram:-



Transition Diagram :-

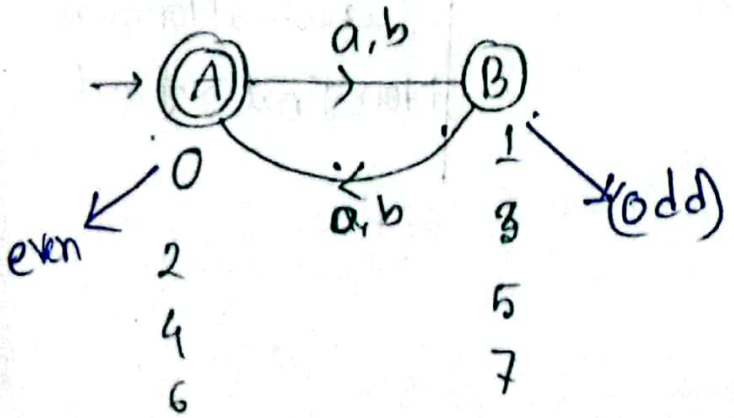


Minimal DFA :

Construct a minimal DFA which accepts set of all strings over {a,b} which is divisible by 2}

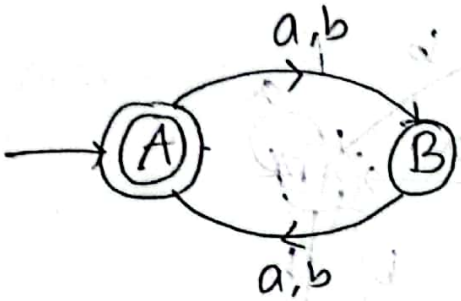
Soln :- $L = \{\epsilon, ab, aa, ba, bb, aaaa, aaab, abaa \ldots\}$

যখন language infinite হবে, তখন সবচেয়ে ছোট string নিয়ে কাজ করতে হবে

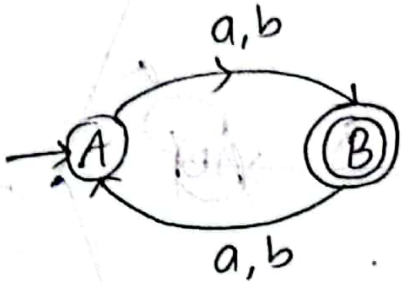


L={w/w has an odd and even length of String}

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



even length

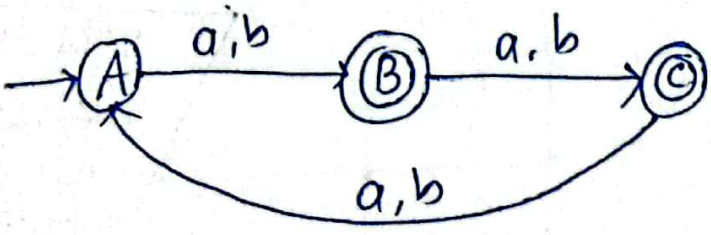


odd length

L={w the length of string (w) is not divisible by 3}

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

$L = \{\epsilon, aaa, aab, bbb, bba, aba\}$



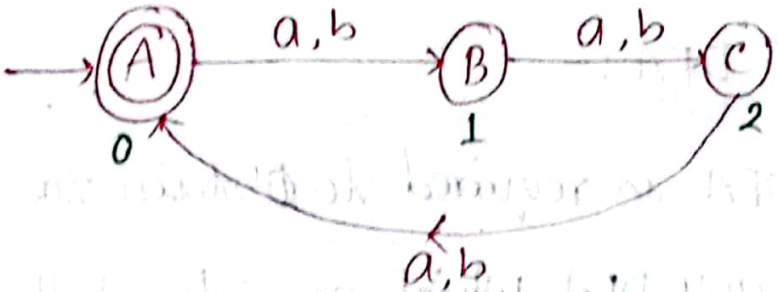
Compliment:

Final → Non-final
Non-final → final

Ⓐ $L(M) = \{w \mid$ the length of string (w) is divisible by $3\}$
/multiple of 3
$|w| \bmod 3 = 0$

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

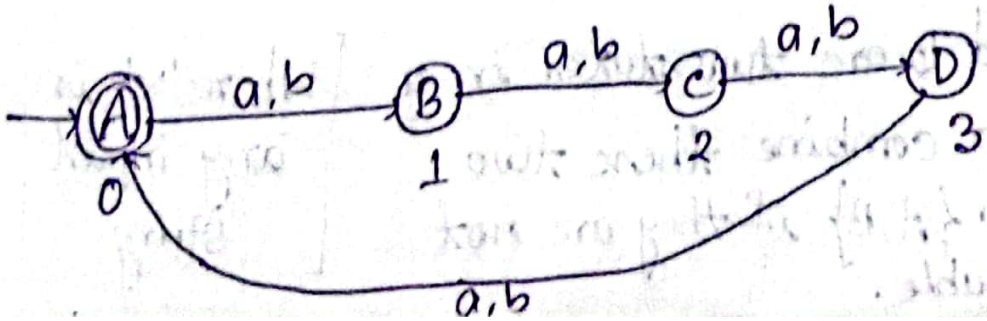
$L = \{\epsilon, aaa, bbb, aab, aba, bab, bba, aaaaaa, bbbbbb, \ldots\}$



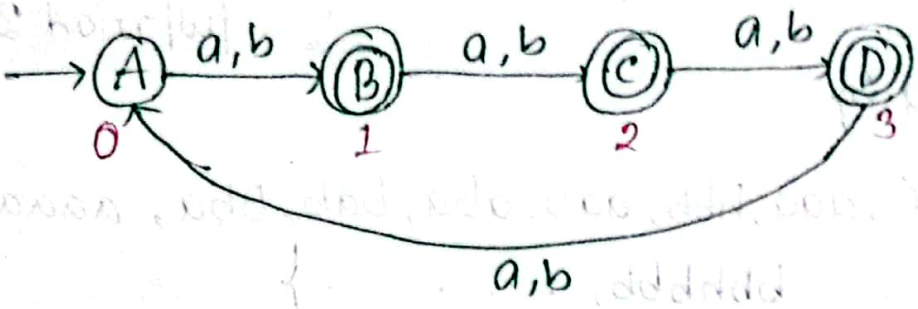
Ⓑ $L(M) = \{w \mid$ the length of string (w) is divisible by $4\}$
/multiple of 4
$|w| \bmod 4 = 0$

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

$L = \{\epsilon, aaaa, bbbb, aabb, \ldots\}$



not divisible by 4 :-



Date - 07 march, 2024

Minimization of DFA :-

- Minimization of DFA is required to obtain the minimal version of any DFA which consists of the minimum number of states possible.

- Two states 'A' and 'B' are said to be equivalent if →

$\delta(A,x) \longrightarrow F$ and $\delta(B,x) \longrightarrow F$

$\delta(A,x) \not\longrightarrow F$ or and $\delta(B,x) \not\longrightarrow F$

* If A and B are two states in a DFA, we can combine these two states into {A,B} if they are not distinguishable.

[Where 'x' is any input string]

If $|x| = 0$, then A and B are said to be 0 equivalent,

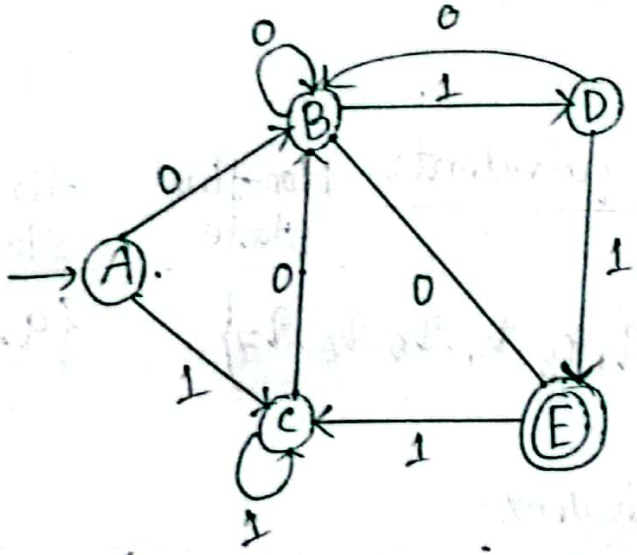
If $|x| = 1$, then A and B are said to be 1 equivalent.

If $|x| = 2$, then A and B are said to be 2 equivalent.

.....

If $|x| = n$, then A and B are said to be n equivalent.

Example :- 1 DFA Minimization



Ps	0	1
→A	B	C
B	B	D
C	B	C
D	B	*E
(E)	B	C

equivalence-0 :- Non-final State {A, B, C, D} Final State {E}

1-equivalence :- {A, B, C} {D} {E}

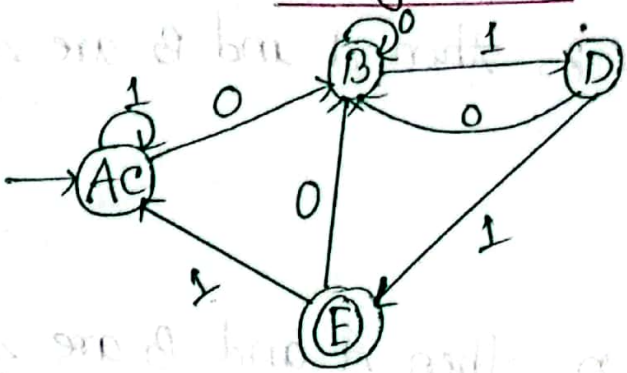
2-equivalent :- {A, C} {B} {D} {E}

3-equivalent :- {A,C} {B}, {D} {E}

Transition Table:

Ps	0	1
→AC	B	C
B	B	D
D	B	*E
*E	B	C

Diagram:



Construct a minimum DFA equivalent to the DFA described by :-

Ps	0	1
→q_0	q_1	q_5
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_0
q_4	q_7	q_5
q_5	q_2	q_0
q_6	q_6	q_4
q_7	q_6	q_2

0-equivalent: Non-final State | final State

$\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$ $\{q_2\}$

1-equivalent:-

$\{q_0, q_4, q_6\}$, $\{q_1, q_7\}$ $\{q_3, q_5\}$ $\{q_2\}$

2-equivalent:-

$\{q_0, q_4\}$ $\{q_6\}$ $\{q_1, q_7\}$ $\{q_3 q_5\}$ $\{q_2\}$

3-equivalent:-

$\{q_0 q_4\}$ $\{q_6\}$. $\{q_1, q_7\}$ $\{q_3 q_5\}$ $\{q_2\}$

Transition table:-

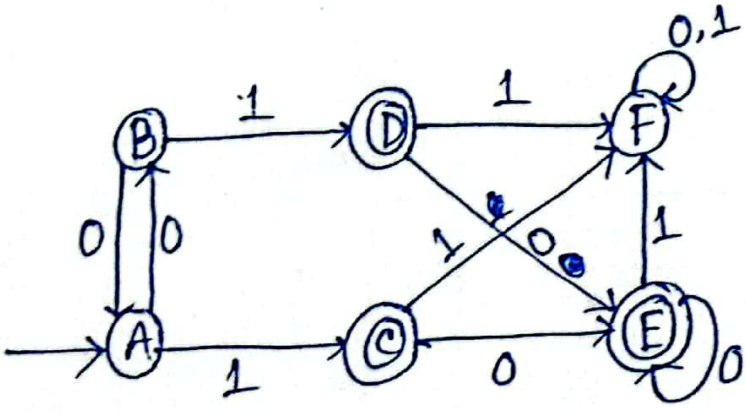
Ps	0	1
$\rightarrow\{q_0 q_4\}$	$\{q_1 q_7\}$	$\{q_3, q_5\}$
$\{q_6\}$	$\{q_6\}$	$\{q_0, q_6\}$
$\{q_1, q_7\}$	$\{q_6\}$	$\{q_2\}$
$\{q_3 q_5\}$	$\{q_2\}$	$\{q_0\}$
$*\{q_2\}$	$\{q_0 q_4\}$	$\{q_2\}$

Diagram:

DFA Minimization (with Multiple final state)

When there are more than one Final state involved.

Minimize the following DFA:-



Transition Table

Ps	0	1
$\rightarrow$ A	B	C
B	A	D
*C	E	F
*D	E	F
*E	E	F
F	F	F

Non-final State

final State

0-equivalent: {A, B, F}

{C, D, E}

1-equipvalence :-

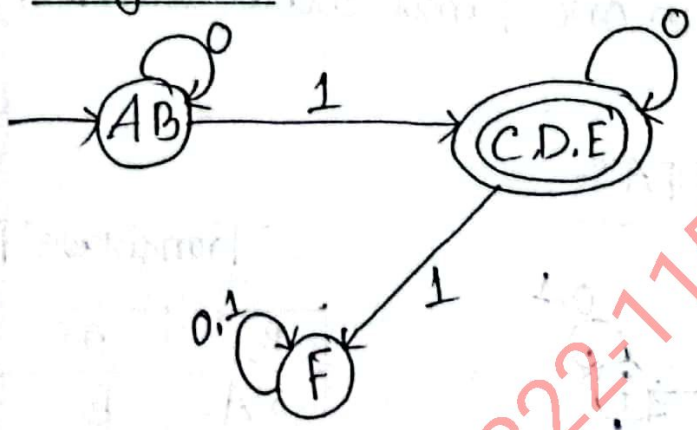
{A, B} {F} {C, D, E}

2-equipvalence :- {A, B}, {F}, {C, D, E}

New DFA Transition Table

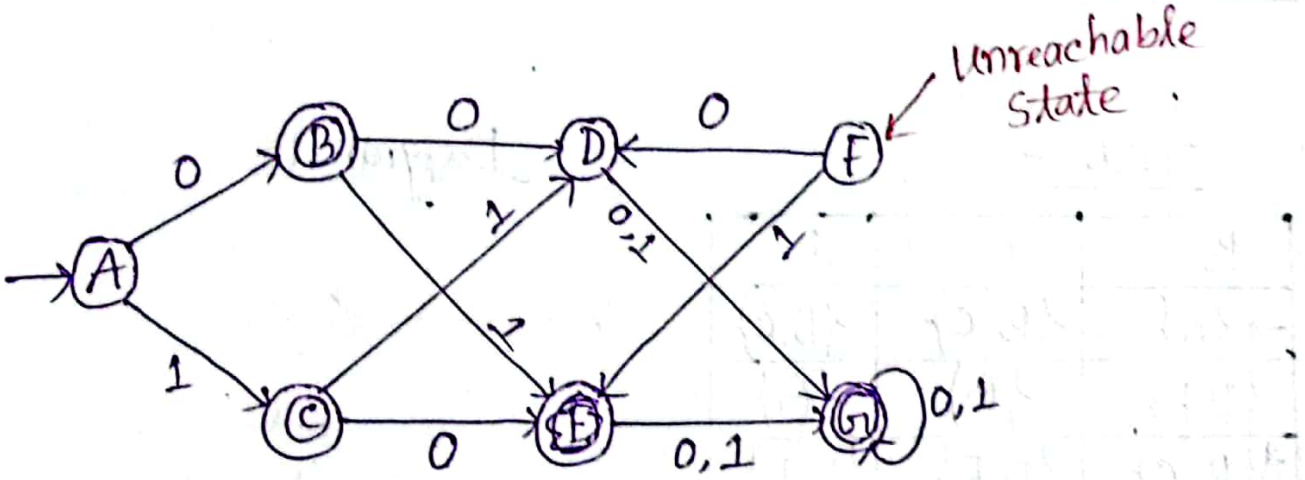
Ps	0	1
→{A, B}	{A, B}	{C, D, E}
{F}	F	F
*{C, D, E}	{C, D, E}	{F}

Diagram :-



DFA Minimization for Unreachable State :-

Unreachable State:- A state is said to be unreachable if there is no way it can be reached from the initial state.



Transition Table:

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

0-equivalence :-

Non-final state {A, D, E}

Final state {B, C, G}

1-equivalence:- {A,D,E} {B,C}, {G}

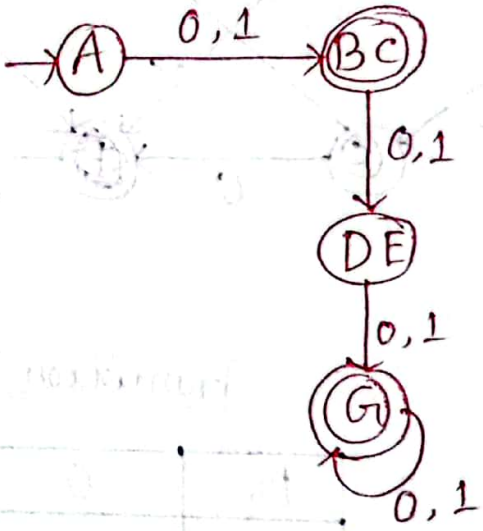
2-equipvalence:- {A}, {D,E}, {B,C}, {G}

3-equipvalence:- {A}, {D,E}, {B,C}, {G}

Table:-

Ps	0	1
→{A}	{B,C}	{B,C}
{D,E}	{G}	{G}
*{B,C}	{D,E}	{D,E}
*{G}	{G}	{G}

Diagram:-



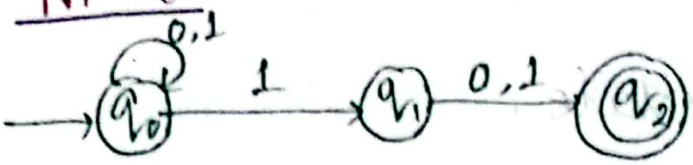
Date:- 11 march, 24

NFA to DFA Conversion:-

step-1:- Null value আছে কি না।

step-2:- Multiple state আছে কি না।

NFA :-



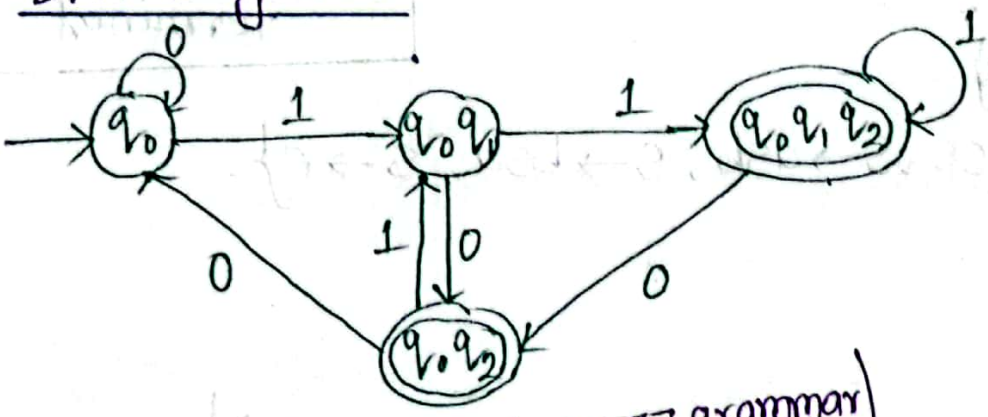
NFA Transition Table :-

Ps	0	1
$\rightarrow q_0$	q_0	$\{q_0 q_1\}$
q_1	q_2	q_2
$*q_2$	ϵ	ϵ

DFA

Ps	0	1
$\rightarrow q_0$	q_0	$\{q_0 q_1\}$
$\{q_0 q_1\}$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$
$*\{q_0 q_1 q_2\}$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$
$*\{q_0 q_2\}$	q_0	$\{q_0 q_1\}$

DFA Diagram :-



(language-টা ভালভাবে grammar এর করা হয়)

Grammar :-

— A Grammar 'G' is defined as quadruple-

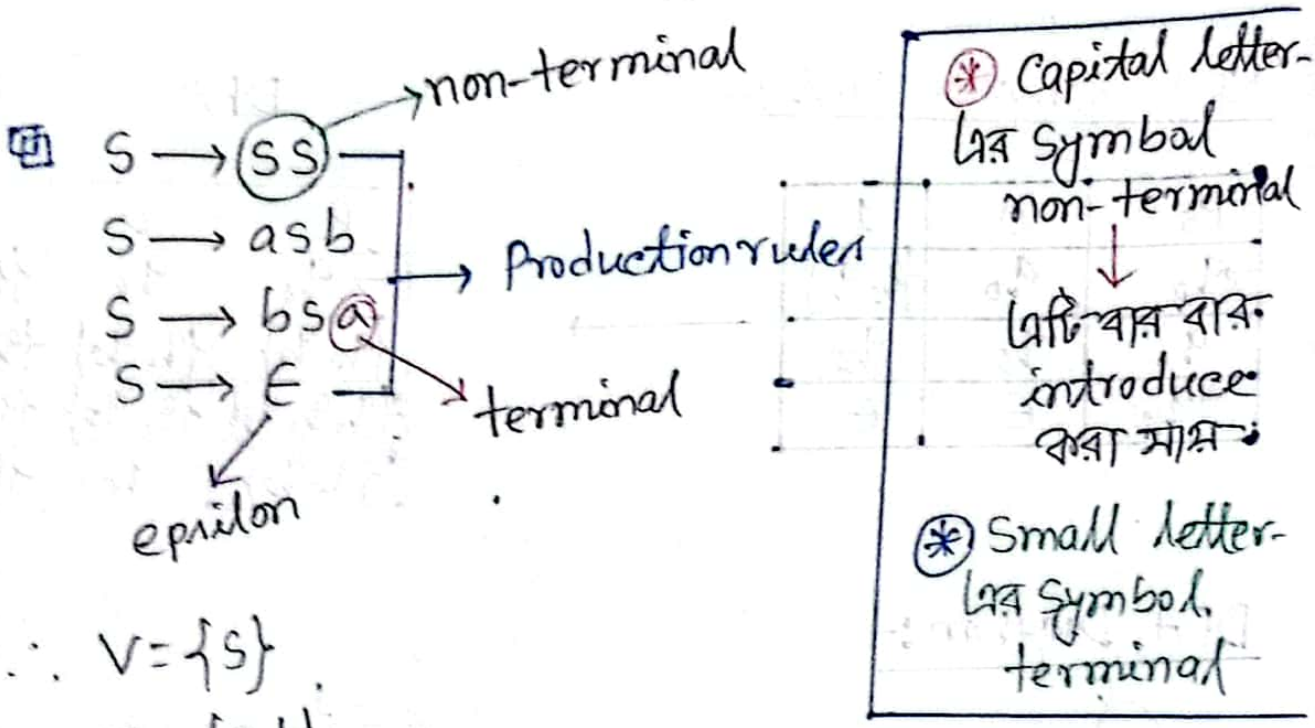
$$G(V, T, P, S)$$

V = Finite Non-terminal Symbol.

T = Finite Terminal Symbol

P = Production rules

S = Starting Symbol/state.



∴ V = {S}

T = {a, b}

P = {S → SS ; S → asb , S → bsa , S → ϵ}

S = S

Example-1:- $G = \{(S, A), (a, b), (S \to asb/\epsilon)\}$

Soln:-

$S \to asb$

$\to ab$ [by $S \to \epsilon$]

$S \to a\underline{S}b$

$\to aasbb$ [by $S \to asb$]

$\to aabb$

$S \to a\underline{S}b$

$\to aa\underline{S}bb$ [by $S \to asb$]

$\to aaasbbb$ [by $S \to asb$]

$\to aaabbb$ [by $S \to \epsilon$]

$\rightarrow a^3 b^3 \Longrightarrow a^n b^n$

$\therefore L(S) = [a^n b^n , n \geq 0]$

'O' length. মাঝখানে ϵ দিয়ে রাখতে হয়

Ex-2:- $G = \{b^{2n} ; \text{where } n \geq 0\}$

$S \rightarrow \epsilon$; $S \rightarrow bbS$
$\rightarrow bb$

$S \rightarrow bbS$
$\rightarrow bbbbS$
$\rightarrow bbbb$

$S \rightarrow bbS$
$\rightarrow bbbbS$
$\rightarrow bbbbbbS$
$\rightarrow bbbbbb$

$\therefore L(G) = \{\epsilon, bb, bbbb, bbbbbb, \ldots\}$

Ex-3:- $G_3 = [\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AB, A \rightarrow aA/a, B \rightarrow bB/b\}]$

Soln:- $S \rightarrow AB$
$\rightarrow ab$

$S \rightarrow AB$
$\rightarrow aAbB$
$\rightarrow aabb$

$S \rightarrow AB$
$\rightarrow aAb$
$\rightarrow aab$

$S \rightarrow AB$
$\rightarrow abB$
$\rightarrow abb$

$\therefore L(G_3) = \{ab, aabb, aab, abb \ldots\}$
$= \{ab, a^2b^2, a^2b, ab^2 \ldots\}$

$= \{a^m, b^n \mid m \geq 0, \text{ and } n \geq 0\}$

Ans:

Ex-4:- Consider the grammar, $G4 = [\{S, A\}, \{a, b\}, S, \{S \to aAb, aA \to aaAb, A \to \epsilon\}]$

Soln:- $S \to aAb$

$\to aaAbb$ [by $aA \to aaAb$]

$\to aaaAbbb$ [by $aA \to aaAb$]

$\to aaabbb$ [by $A \to \epsilon$]

$\therefore L(G_4) = \{aaa\ bbb\}$

Ans:

CT Ques:-

Ans. to the Ques. No 1.

Finite Automata :- Finite Automata is the simplest machine to recognize patterns. A finite Automata is a collection of 5 tuples $(Q, \Sigma, \delta, q_0, F)$

Where,

Q = Finite set of all states

Σ = Finite set of the input symbol.

δ = Transition function. $[Q \times \delta \rightarrow Q]$

q_0 = inital state / start state

F = Final State.

Ans. to the Ques. No 2

Kleene closure :-

- Kleene closure denoted by *
- Set of all posible strings including empty string.
- It contain the empty string and allows for zero

or more repetitions of string from the original language.

- Using asterrisk(*) symbol after the expression such as L^*.

- Example :- If $L=\{a,b\}$ then

$L^* = \{\epsilon, abb, aa, ab, ba, bb, aaa, aab, bba, aba, \ldots\}$

positive closure :-

- Positive closure denoted by +.
- Set of all possible strings excluding empty string.
- It doesnot contain empty string and allows for one or more repetitions of string from the original language.
- Using the plus (+) symbol, after the expression such as L^+.

Example: If $L=\{a,b\}$ then,

$L^+ = \{a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bbb, \ldots\}$

Ans: to the Quest. No 3 (a)

$L_1 = \{a^n b^m ; n \geq 3, m \geq 2\}$ Where $\Sigma = \{a, b\}$

$S \rightarrow aaaB$, $B \rightarrow bB \mid bbC$, $C \rightarrow \epsilon \mid bC$

$\therefore S \rightarrow aaaB$
$\rightarrow aaabB$ [by $B \rightarrow bB$]
$\rightarrow aaabbbC$ [by $B \rightarrow bbC$]
$\rightarrow aaabbb$ [by $C \rightarrow \epsilon$]

$S \rightarrow aaaB$
$\rightarrow aaabBC$
$\rightarrow aaabbbC$
$\rightarrow aaabbbbC$
$\rightarrow aaabbbbb$

$S \rightarrow aaaB$
$\rightarrow aaabbC$ [by $B \rightarrow bbC$]
$\rightarrow aaabb$ [by $C \rightarrow \epsilon$]

$S \rightarrow aaaB$
$\rightarrow aaabbC$
$\rightarrow aaabbbbC$
$\rightarrow aaabbbbbC$
$\rightarrow aaabbbbbb$

$\rightarrow aaa\,bbbbb$

$L_1 = \{a^3b^3, a^3b^4, a^3b^5, a^3b^5 \ldots\}$

$= \{a^n.b^m \mid n \geq 3, m \geq 2\}$

Ans:

Ans. to the Quer. No 3(b)

$L_2 = \{w : |w| \bmod 3 = 0\}$ Where $\Sigma = \{0\}$

$S \rightarrow aaaS \mid \lambda \mid \epsilon$

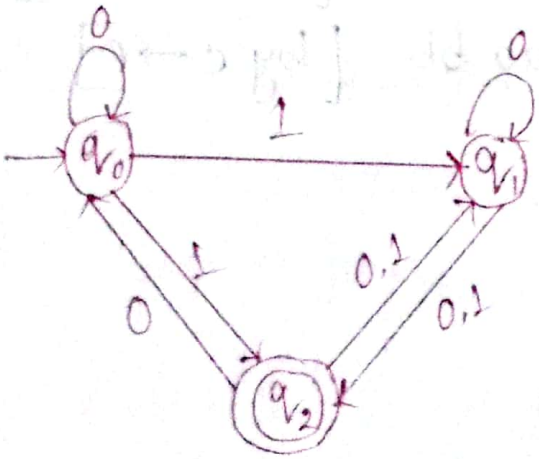
$\rightarrow aaa$

$S \rightarrow aaaS$

$\rightarrow aaa\,aaaS$

$\rightarrow aaa\,aaa$

Ans. to the Quer. No 4



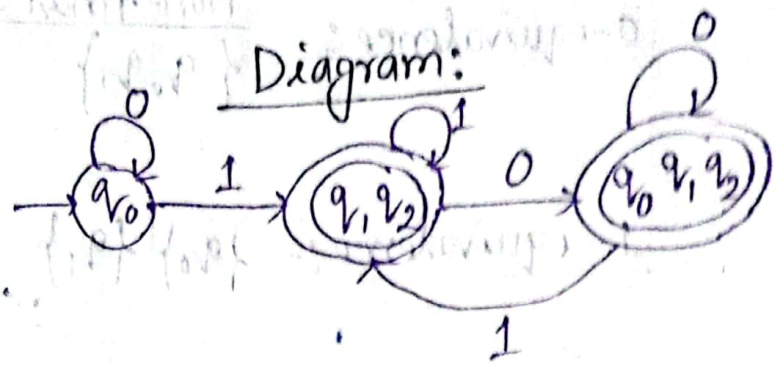
NFA Transition Table

PS	0	1
$\rightarrow q_0$	q_0	q_1, q_2
q_1	q_1, q_2	q_2
$*q_2$	q_0, q_1	q_1

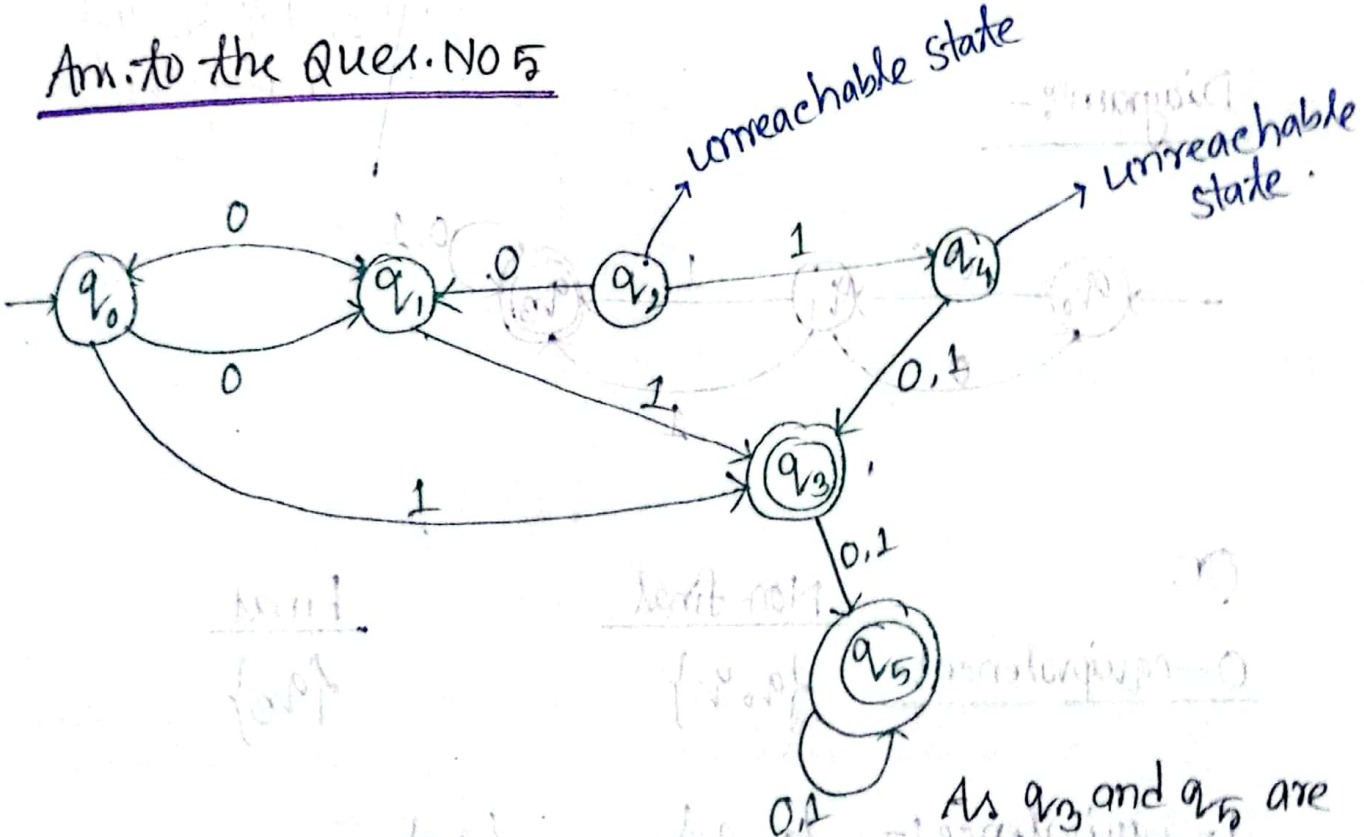
DFA Transition table

PS	0	1
$\rightarrow q_0$	q_0	$q_1 q_2$
$*q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2$
$*q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2$

Diagram:



Ans. to the Quer. No 5



Transition Table

PS	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$*q_3$	q_5	q_5
$*q_5$	q_5	q_5

As q_3 and q_5 are same for input 0 and 1. so that, remove q_5 because they are same. then replace by q_3

PS	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$* q_3$	q_3	q_3

0-equivalence: Non-final state $\{q_0 q_1\}$ Final state $\{q_3 q_5\}$

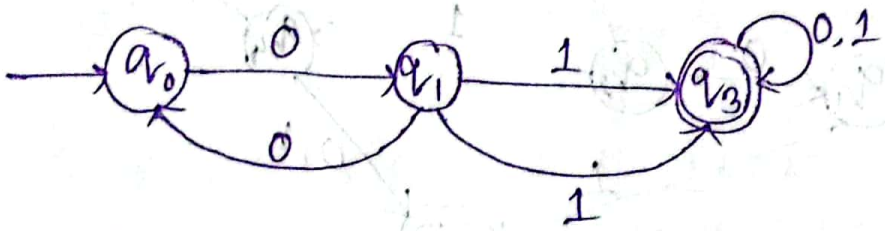
1-equivalence:- $\{q_0\}\{q_1\}$ $\{q_3 q_5\}$

q_3 and q_5 to the same state on 0 and 1

So, skip q_5 and replace q_5 by q_3

2-equivalence:- $\{q_0\}\{q_1\}$ $\{q_3\}$

Diagram:-

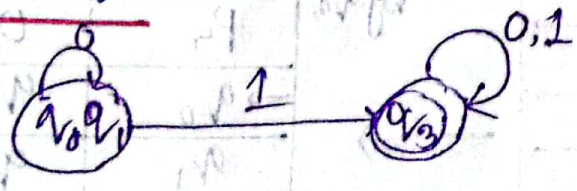


Or,

0-equivalence:- Non-final $\{q_0 q_1\}$ Final $\{q_3\}$

1-equivalence:- $\{q_0 q_1\}$ $\{q_3\}$

Diagram:-



ε- NFA TO DFA Conversion :-

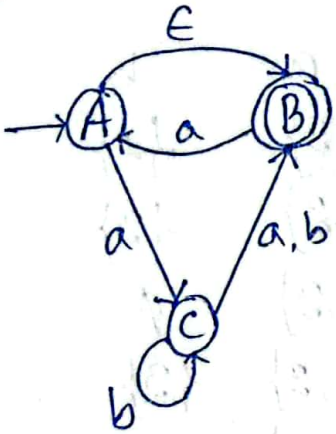
rules:-

① S.T.T (state Transition table) for ε-NFA

② Find ε-closure for all states.

③ Make S.T.T for DFA

Ex-1:-



ε-NFA S.T.T

Ps	a	b
→A	C	ϕ
*B	A	ϕ
C	B	BC

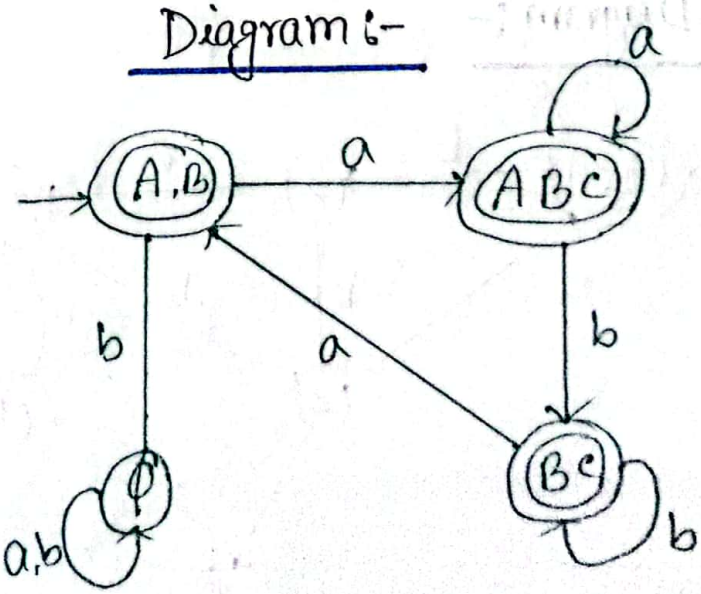
ε-closur all state

ε-closure(A) → {A,B}
ε-closure(B) → {B}
ε-closure(C) → {C}

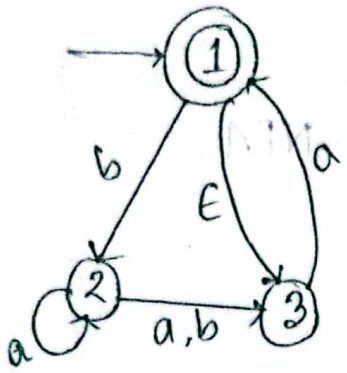
DFA S.T.T

Ps	a	b
→*{A,B}	{A,B,C}	ϕ
*{A,B,C}	{A,B,C}	{BC}
*{BC}	{A,B}	{BC}

Diagram:-



Ex-2 :-



ϵ-NFA S.T.T

Ps	a	b
→*{1}	∅	{2}
{2}	{2,3}	{3}
{3}	{1}	∅

ϵ-closure all state

ϵ-closure (1) → {1,3}

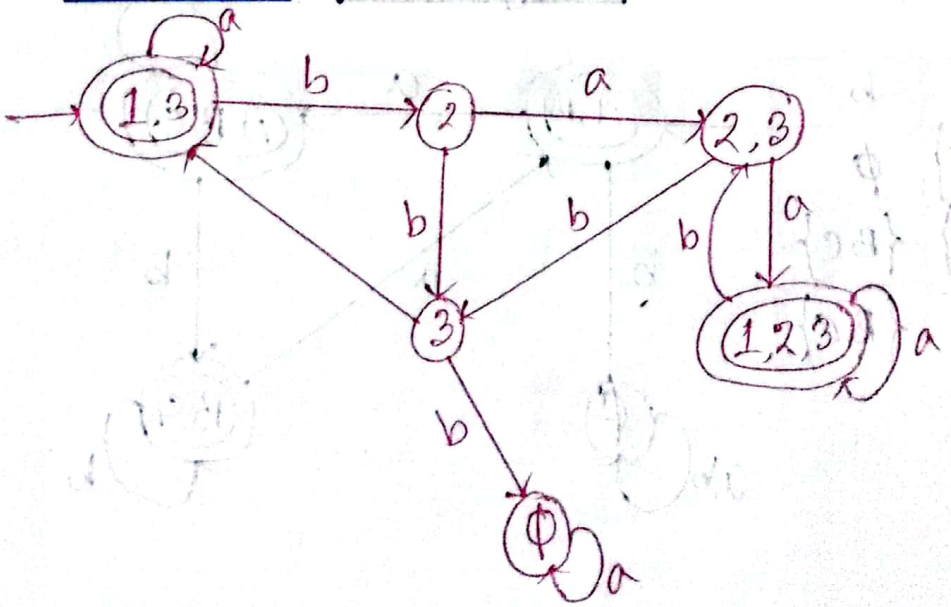
ϵ-closure (2) → {2}

ϵ-closure (3) → {3}

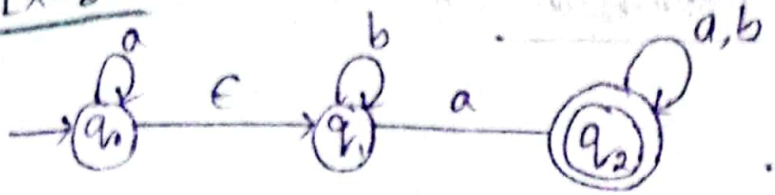
DFA-S.T.T

Ps	a	b.
→*{1,3}	{1,3}	{2}
{2}	{2,3}	{3}
{3}	{1,3}	∅
{2,3}	{1,2,3}	{3}
*{1,2,3}	{1,2,3}	{2,3}

Diagram :-



Ex-3:-



step:- (i) first draw the S.T.T for ε-NFA

(ii) find ε-closure for all state.

(iii) Make S.T.T for DFA

ε-NFA S.T.T

PS	a	b
→ q_0	q_0	ϕ
q_1	q_2	q_1
* q_2	q_2	q_2

ε-closure

ε-closure $(q_0) \longrightarrow \{q_0 q_1\}$

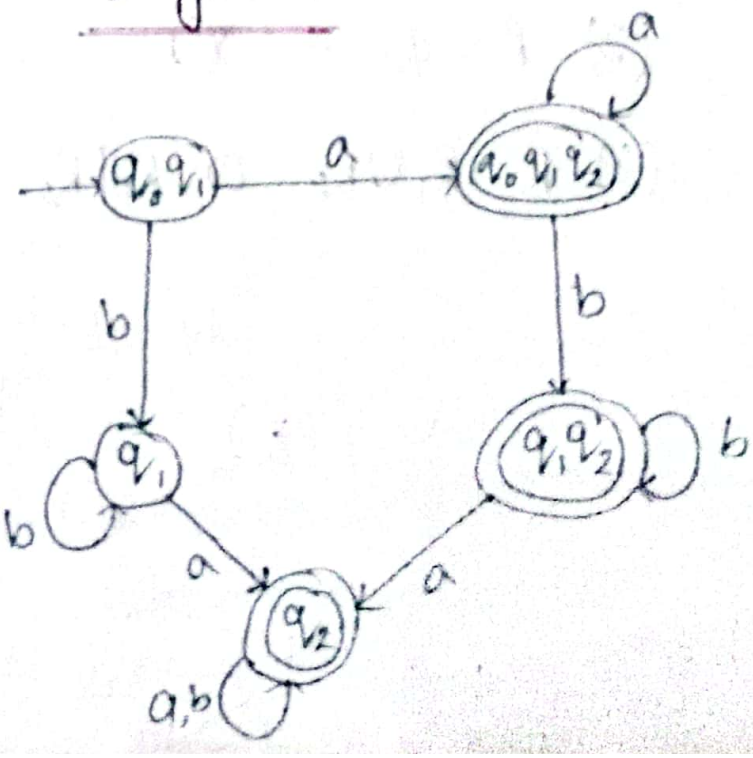
ε-closure $(q_1) \longrightarrow \{q_1\}$

ε-closure $(q_2) \longrightarrow \{q_2\}$

DFA-S.T.T

PS	a	b
→ $q_0 q_1$	$\{q_0 q_1 q_2\}$	$\{q_1\}$
* $q_0 q_1 q_2$	$\{q_0 q_1 q_2\}$	$\{q_1 q_2\}$
q_1	$\{q_2\}$	$\{q_1\}$
* $q_1 q_2$	$\{q_2\}$	$\{q_1 q_2\}$
* q_2	$\{q_2\}$	$\{q_2\}$

Diagram :-



Chapter-3:- Regular Expression

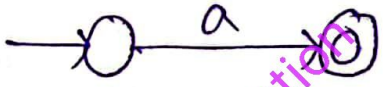
Date:- 25 April, 24

#Regular Expression:-

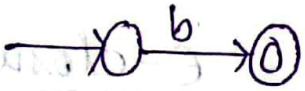
Regular expression are used for representing certain sets of strings in an algebraic fashion.

There are 6 cases of Regular Expressions —

(1) $R = a$; Where $a \in \Sigma$



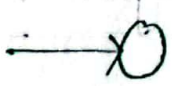
$R = b$



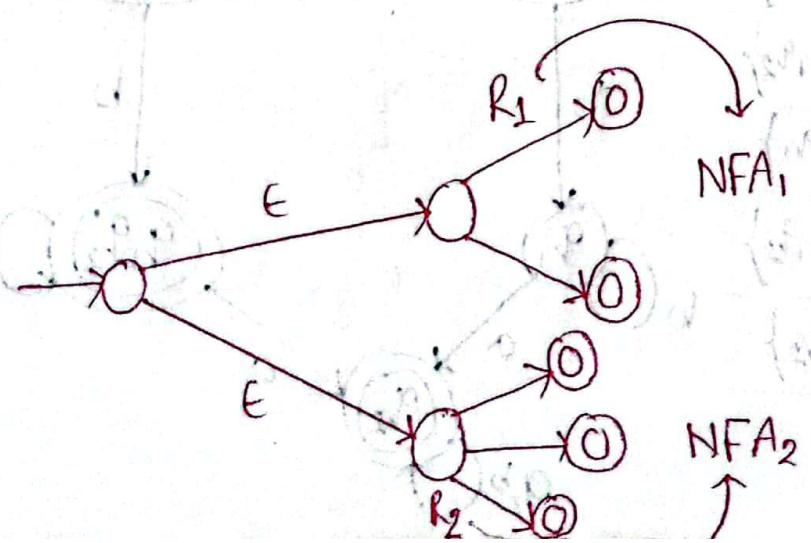
(2) $R = \epsilon$

initial state-la
final state 2ta

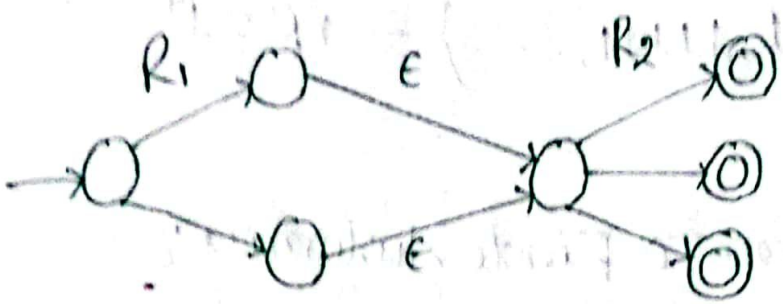
(3) $R = \phi$



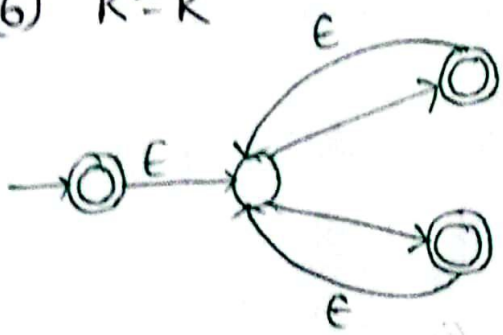
(4) $R = R_1 \cup R_2$ or, $R_1 | R_2$



(5) $R = R_1 + R_2 \mid R_1 R_2 \mid R_1 \cdot R_2$ → Concatenation



(6) $R = R^*$



Describe the following set as regular expression :-

1. $\{0,1,2\} \longrightarrow R = 0 \cup 1 \cup 2$
   $= 0+1+2$

2. $\{\epsilon, ab\} \longrightarrow R = \epsilon \cup ab$
   $= \epsilon + ab$
   $= \epsilon ab$

3. $\{abb, a, b, bb, bba\} \longrightarrow R = abb \cup a \cup b \cup bb \cup bba$
   $= abb + a + b + bb + bba$

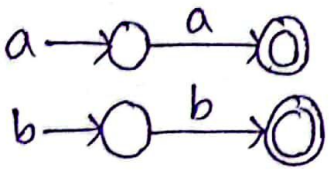
4. $\{\epsilon, 0, 00, 000, 0000, \ldots\} \rightarrow R = 0^*$

5. $\{1, 11, 111, 1111, 11111, \ldots\} \rightarrow R = 1^+$

Regular Expression TO Finite Automata :-

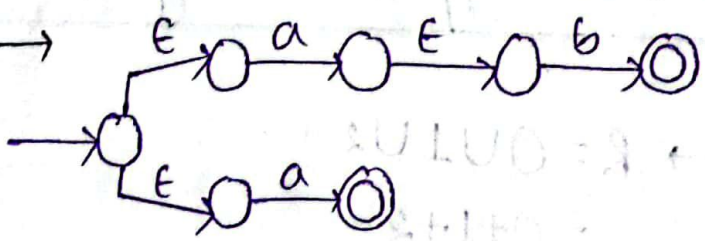
Ex-1 :- $(ab \cup a)^*$

Soln :-

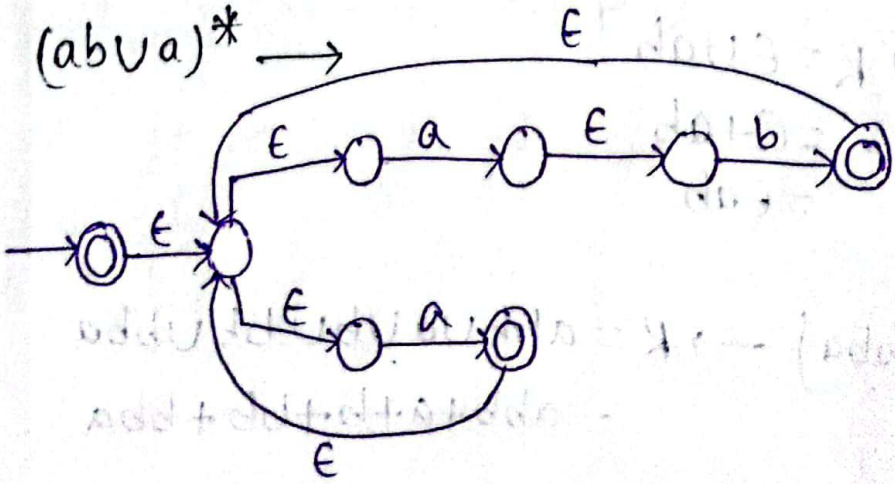


ab →

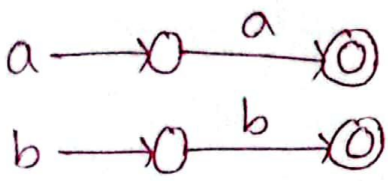
$(ab \cup a)$ →



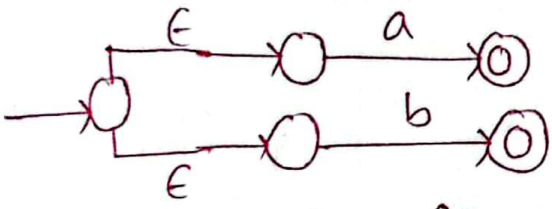
$(ab \cup a)^*$ →



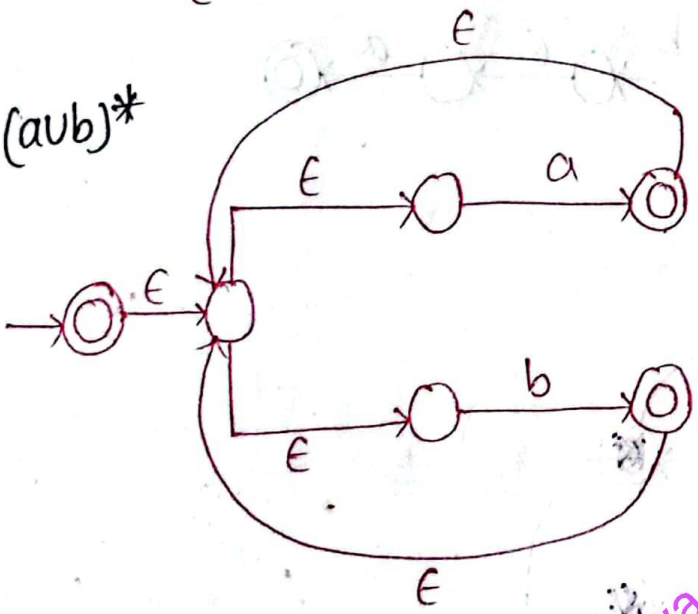
Ex-2:- (a∪b)*



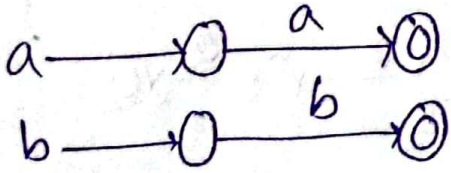
a∪b



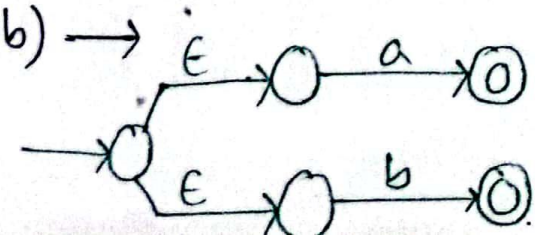
(a∪b)*



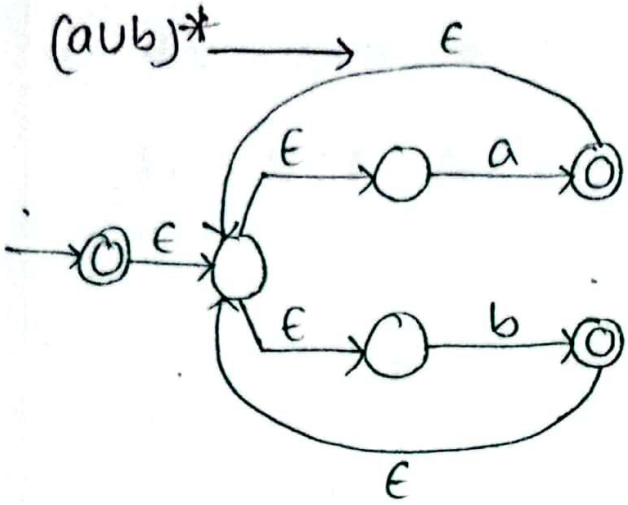
Ex-3:- (a∪b)*aba



(a∪b) →

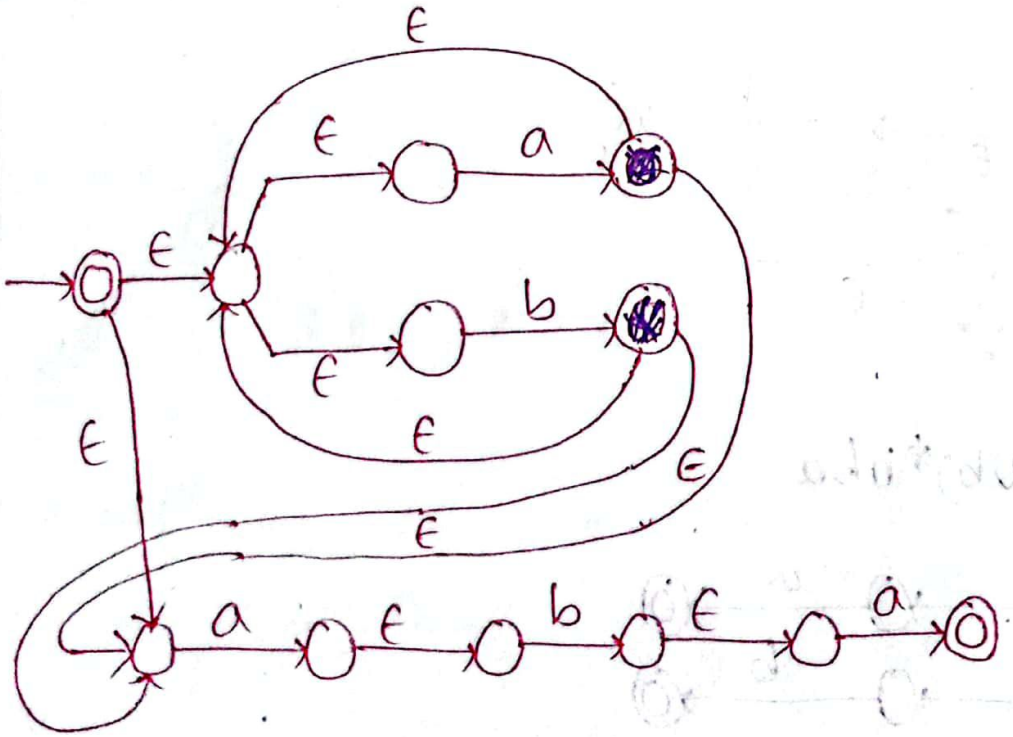


(a∪b)*

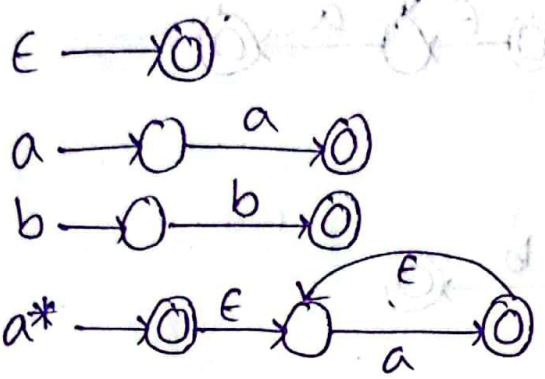


aba

(a∪b)* aba

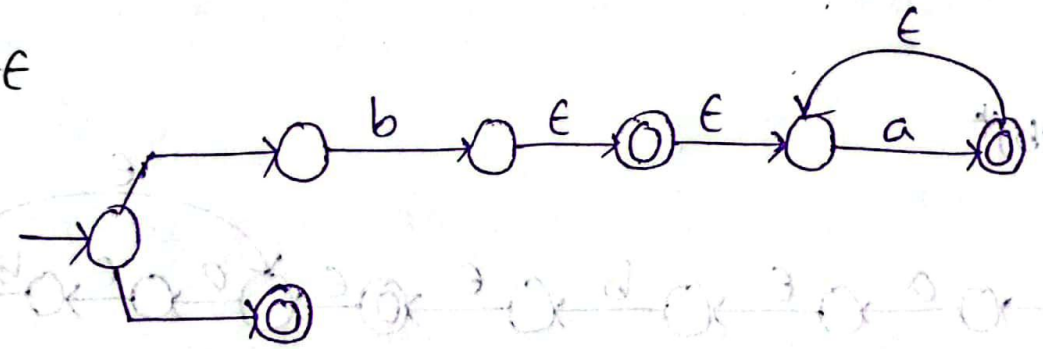


Ex-4:- $ba^* + \epsilon$



ba^*

$ba^* + \epsilon$

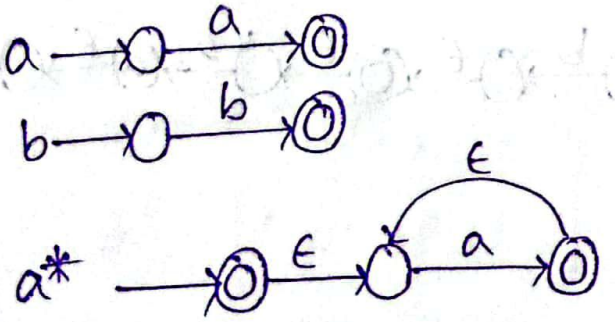


Ex-5:- $a^+ \cup (ab)^+$

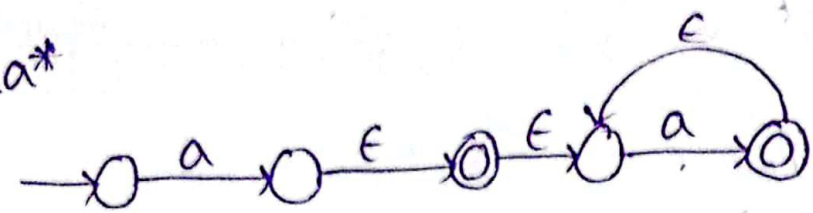
$= aa^* \cup (ab(ab)^*$

a^+ = 1 or more

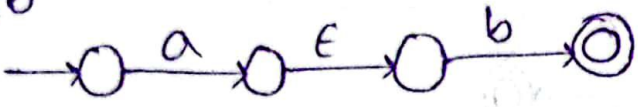
$= \{a, aa, \ldots\}$



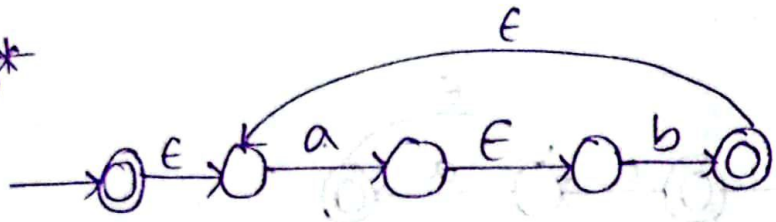
aa*



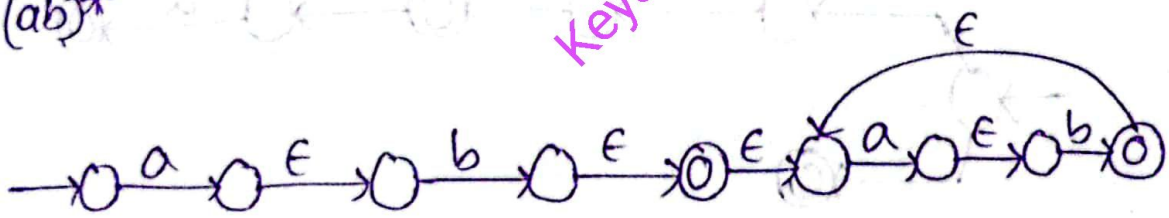
ab



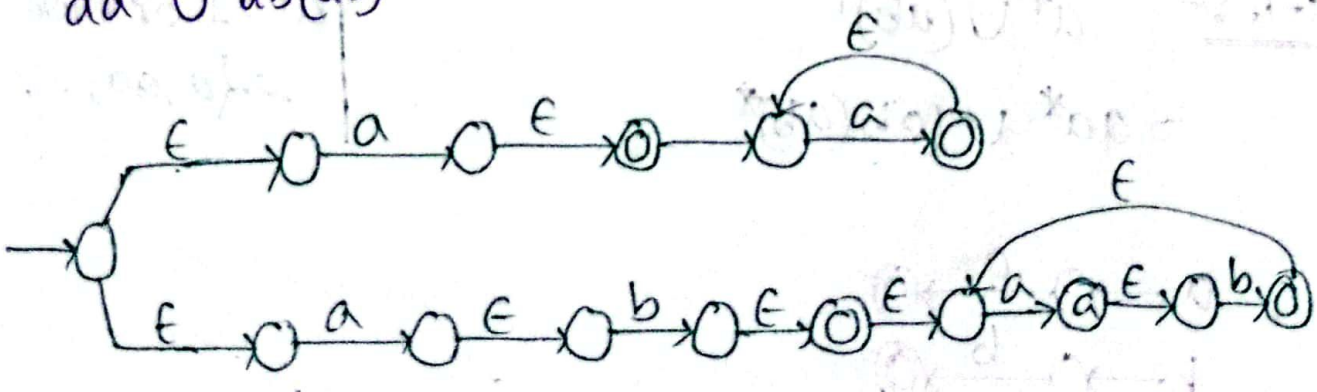
ab*



ab(ab)*



aa* ∪ ab(ab)*



Designing regular expressions- examples:-

L_1 = {Language accepting strings of length exactly 2}

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

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

$R = aa+ab+ba+bb$

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

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

2 or more than 2

L_2 = {Language accepting strings of length at least 2}

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

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

$R = aa \cup ab \cup ba \cup bb \cup aaa \cup bbb \ldots$

$= aa+ab+ba+bb+aaa+bbb+aba+\ldots$

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

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

Ans:

L_3 = {Language accepting strings of length at most 2}

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

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

$R = \epsilon + a + b + aa + ab + ba + bb$

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

Ans:

Identities of Regular Expression:-

1. $\phi + R = R$
2. $\phi R + R\phi = \phi$
3. $\epsilon R = R\epsilon = R$
4. $\epsilon^* = \epsilon$
5. $(R^*)^* = R^*$
6. $(P+Q)R = PR + QR$
7. $R(P+Q) = RP + RQ$

Arden's theorem:-

If P and Q are two Regular expressions over Σ, and if P doesn't contain ϵ, then $R = Q + RP \rightarrow R = QP^*$

Σ → Sigma

$R = Q + RP$ ——— ①

$= Q + QP^*P$ $[\because R = QP^*]$

$= Q(\epsilon + P^*P)$ $[\epsilon + R^*R = R^*]$

$= QP^*$ [proved]

$R = Q + RP$

$= Q + (Q+RP)P \quad [\because R = Q+RP]$

$= Q + QP + RP^2$

$= Q + QP + (Q+RP)P^2$

$= Q + QP + QP^2 + RP^3$

.

.

.

$= Q + QP + QP^2 + \ldots + QP^n + RP^{n+1} \quad [\because R = QP^*]$

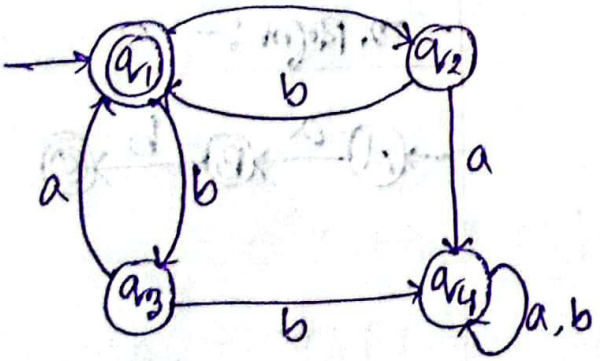
$= Q + QP + QP^2 + \ldots QP^n + QP^*P^{n+1}$

$= Q[\epsilon + P + P^2 + \ldots P^n + P^*P^{n+1}]$

$\therefore R = QP^*$

Date – 02 May, 24

DFA TO Regular Expression:-



$q_1 = \epsilon + q_2 b + q_3 a$ ——— (I)

$q_2 = q_1 a$ ——— (II)

$q_3 = q_1 b$ ——— (III)

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b$ ——— (IV)

From (1):

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

$\Rightarrow q_1 = \epsilon + (q_1 a) b + (q_1 b) a$

$\Rightarrow q_1 = \epsilon + q_1 ab + q_1 ba$

$\Rightarrow q_1 = \epsilon + q_1 (ab + ba)$ $[\because R = Q + RP]$

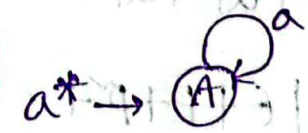
$R = QP^*$

$\Rightarrow q_1 = \epsilon + q_1 (ab + ba)^*$

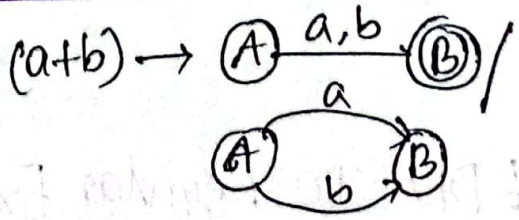
$\therefore q_1 = (ab + ba)^*$ $[\epsilon . R = R]$

p এর মান epsilon হলে পারবে না

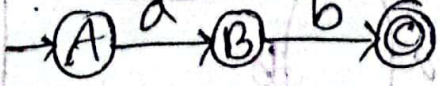
$\therefore$ Regex:- $(ab + ba)^*$



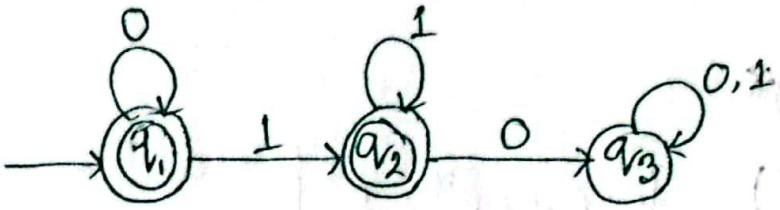
Union এর জন্য short technique :-



Concatenation এর জন্য :-



DFA TO Regular Expression (When there are multiple final state)



$q_1 \longrightarrow \epsilon + q_1 0$ ——— (I)

$q_2 \longrightarrow q_1 1 + q_2 1$ ——— (II)

$q_3 \longrightarrow q_2 0 + q_3 0 + q_3 1$ ——— (III)

from (I);

final state q_1;

$q_1 = \epsilon + q_1 0$ $\left[\because R = Q + RP \quad R = QP^*\ \text{Arden's theorem}\right]$

$\Rightarrow q_1 = \epsilon 0^*$ $[\because \epsilon \cdot R = R]$

$\therefore q_1 = 0^*$ ——— (IV)

from (II);

final state q_2,

$q_2 = q_1 1 + q_2 1$

$\Rightarrow q_2 = 0^* 1 + q_2 1$ $[\because q_1 = 0^*$ ——— (IV)$]$

$\Rightarrow q_2 = 0^* 1 (1)^*$ $[R = Q + RP = QP^*]$

R = Union of both final states

$= q_1 + q_2$

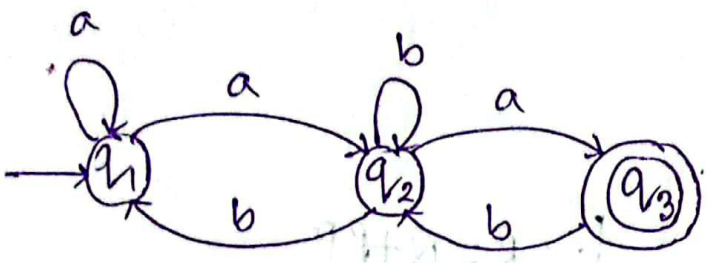
$= 0^* + 0^* 1 (1)^*$

$= 0^* (\epsilon + 1(1)^*)$ $\quad [\because \epsilon + RR^* = R^*]$

$= 0^* 1^*$

Regex :- $0^* 1^*$

NFA TO Regular Expression :-



$q_3 = q_2 a$ ——— (I)

$q_2 = q_1 a + q_2 b + q_3 b$ ——— (II)

$q_1 = q_1 a + q_2 b + \epsilon$ ——— (III)

from (I) :

$q_3 = q_2 a$

$\Rightarrow q_3 = (q_1 a + q_2 b + q_3 b) a$

$\Rightarrow q_3 = q_1 aa + q_2 ba + q_3 ba$ ——— (IV)

from (II);

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

$$= q_1 a + q_2 b + (q_2 a) b \quad [\because \text{putting value of } q_3]$$

$$= q_1 a + q_2 b + q_2 a b$$

$$= q_1 a + q_2 (b + ab) \quad [\because R = Q + RP \rightarrow R = QP^*]$$

$$\therefore q_2 = q_1 a (b+ab)^* \quad \text{——— (V)}$$

from (III);

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

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

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

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

$$\therefore q_1 = (a + a(b+ab)^*) b^* \quad \text{——— (VI)}$$

final state q_3;

$$q_3 = q_2 a$$

$$\Rightarrow q_3 = q_1 a (b+ab)^* a \quad [\text{putting value of } q_2]$$

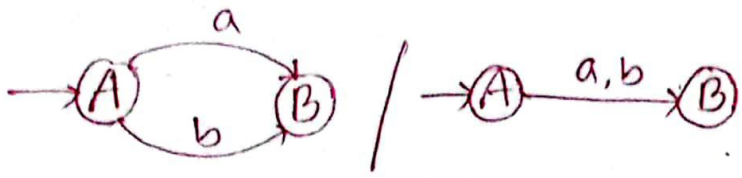
$$\Rightarrow q_3 = ((a+a)(b+ab)^* b)^* a (b+ab)^* a \quad [\text{putting value of } q_1]$$

$$\therefore q_3 = (a+a)(b+ab)^* b)^* a (b+ab)^* a \ldots$$

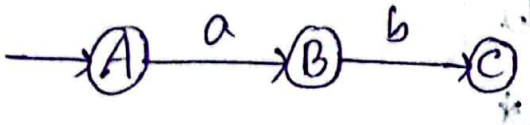
∴ Required Regular Expression for the given NFA.

Regular Expression To Finite Automata :-

Union : 1|+|U a+b|aUb|a|b



concatenation :- . a.b|ab



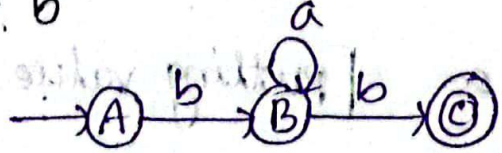
closure of any symbol :- a* → 

Star একটা inner loop হবে.

Convert the following RE to FA :-

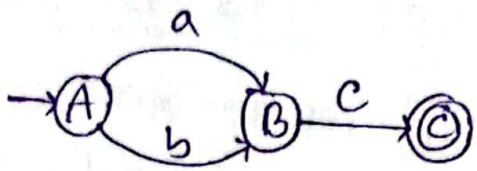
① ba* b

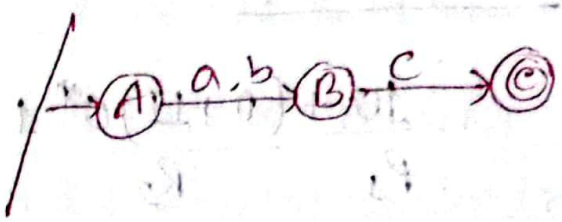
Soln :-



② (a+b) c

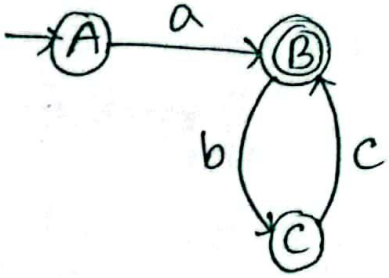
Soln:-

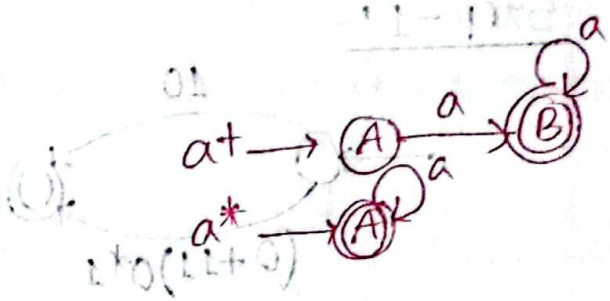




③ a(bc)*

Soln:-

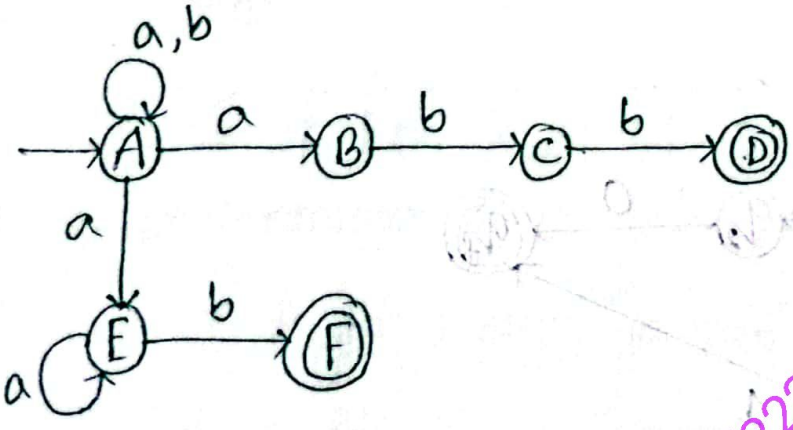




example :-1

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

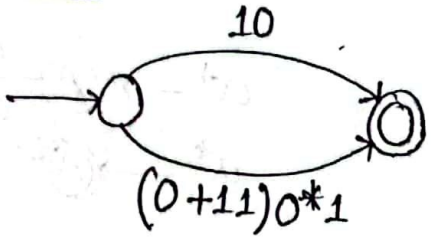
$= \{(a+b)^* (abb)\} + \{(a+b)^* (a^+b)\}$



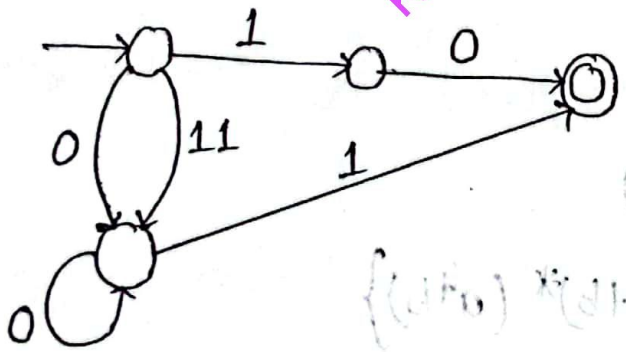
example-02:-

$$\underbrace{10}_{R_1} + \underbrace{(0+11)0^*1}_{R_2}$$

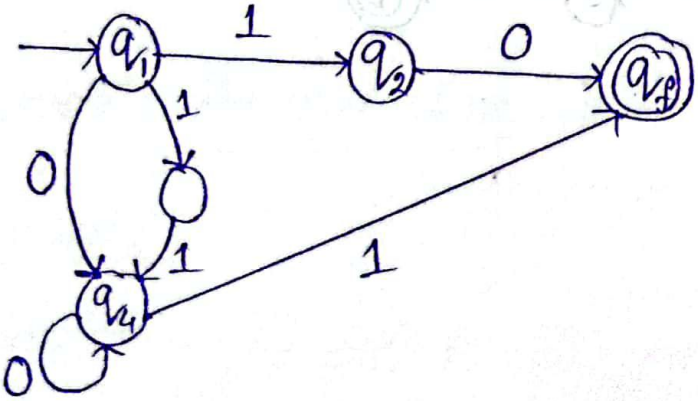
Step-1:-



Step-2:-



Step-3:-



Regular Grammar

Date - 08 May. 24

Noam Chomsky gave a Mathematical model of grammar which is effective for writing computer languages.

The four types of Grammar according to Noam Chomsky are —

Grammar type	Grammar accepted	Language accepted	Automation
Type-0	Unrestricted grammar	Recursively Enumerable Language	Turing Machine
Type-1	Context Sensitive Grammar	Context sensitive Language	Linear Bounded Automation
Type-2	Context free grammar	context free Language	pushdown Automata
Type-3	Regular Grammar	Regular Language	Finite state Automation.

Regular Grammar can be divided into 2 types :-

(i) Right linear Grammar.

(ii) Left linear Grammar.

RLG (Right linear Grammar)	LLG (Left linear)
A grammar is said to be Right linear if all productions are of the form- $A \to xB$, $A \to x$ Where, $A, B \in V$ and $x \in T$	A grammar is said to be left linear if all productions are of the form- $A \to Bx$, $A \to x$ Where, $A, B \in V$ and $x \in T$

Example :-

$S \to abS \mid b$

Non-terminal Symbol.

terminal Symbol

Right linear Grammar

We see that, S which is a non-terminal symbol lies to the right of a and B which is the terminal symbol. so, this is a Right linear grammar.

example :-

$S \to Sbb \mid b$ ⟶ Left linear Grammar.

Non-terminal

terminal

⊛ the Non-terminal symbol S lies to the left of the terminal symbol B so this is the left linear grammar

Derivations From a Grammar

The set of all strings that can be derived from a Grammar is said to be the LANGUAGE generated from that Grammar.

example-01 Consider the Grammar,

$$G_1 = (\{S, A\}, \{a, b\}, S, \{S \to aAb, aA \to aaAb, A \to \epsilon\})$$

Soln:-

$S \to aAb$ [by $S \to aAb$]

$\to aaAbb$ [by $aA \to aaAb$]

$\to aaaAbbb$ [by $aA \to aaAb$]

$\to aaabbb$ [by $A \to \epsilon$]

example-02:- $G_2 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

Soln:-

$S \to AB$

$\to aB$

$\to ab$

$\therefore L(G_2) = \{ab\}$

example-03:- $G_3 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to aA|a ; B \to bB|b\})$

Soln:-

$S \to AB$
$\to ab$

$S \to AB$
$\to aAbB$
$\to aabb$

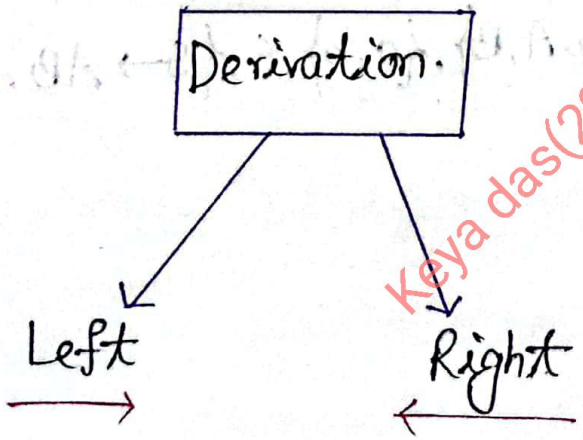
$S \to AB$
$\to aAb$
$\to aab$

$S \to AB$
$\to abB$
$\to abb$

$L(G_3) = \{ab, a^2b^2, a^2b, ab^2, \ldots\}$

$= \{a^m b^n \mid m \geq 0 \text{ and } n \geq 0\}$

#



1. Leftmost Derivation :-

example:-1 PR :- $S \rightarrow S+S/S*S/a/b$

Production rule

String:- a+a*b

Soln:-

$S = S+S$

$= a+S$

$= a+S*S$

$= a+a*S$

$= a+a*b$

∴ String:- a+a*b

example-02:- PR:- $E \rightarrow E+E/E-E/a/b$

String:- a-b+a

Soln:-

$E = E+E$

$= E-E+E$

$= a-E+E$

$= a-b+E$

$= a-b+a$

∴ String:- a-b+a

2. Rightmost Derivation:-

example-01:- PR:- $S \rightarrow S+S/S*S/a/b$

String:- a+a*b

Soln:-

$S = S+S$

$= S+S*S$

$= S+S*b$

$= S+a*b$

$= a+a*b$

∴ String:- a+a*b

example-2:- PR = $E \rightarrow E+E \mid E-E \mid a \mid b$

String: a-b+a

Soln:-

$E = E-E$

$= E-E+E$

$= E-E+a$

$= E-b+a$

$= a-b+a$

∴ String: a-b+a

[কোনো প্রকার language কে describe করতে help করে]

Topic:- Context Free Grammar(CFG).

CFG:- A CFG(Context Free Grammar) is a set of rules/Production used to generate patterns of strings.

CFG are a 4 tuples. They are —

(V/N, T/Σ, P/R, S) Where,

V/N → Set of variables or Non-terminals symbol.

α-তে জিরো এর বেশি variable এবং terminal এর combination α ∈ (V∪T)

T/Σ ⟶ set of Terminals symbol.

Production → A→α (বাম পাশে থাকবে)

P/R ⟶ Production Rules.

S ⟶ start variables. ⟶ Start Variable হবে Variable-এর থেকেই একটা

S∈V

⊛ Context Free Grammar has Production rule of the form —

$$A \rightarrow \alpha$$ ⟶ [redacted]

Where, $\alpha = (V \cup \Sigma)^*$ and

$A \in V$

example-1:- CFG for $\{0^n 1^n \mid n \geq 1\}$

[redacted]

$= \{0^1 1^n\} = \{0 1\}$

$= \{0^2 1^2\} = \{00\ 11\}$

CFG,

S → 01/0S1 (Production)

L.H.S R.H.S

[redacted]

$V = \{S\}$ $T = \{0,1\}$

or,

S ⟶ 01
S ⟶ 0S1 ⟶ P

$S \Rightarrow \{S\}$

[redacted]

Identify the terminals, non-terminals, start variable from the following grammar :-

① $E \to E+T/T$

$T \to T*F/F$

$F \to (E)/id$

Soln:-

$V=\{E, T, F\}$

$T=\{"+", "*", "(", ")", id\}$

$S=\{E\}$

② $S \to (L)/a$

$L \to L,S/S$

Soln:

$V=\{S, L\}$

$T=\{"(", ")", ",", "a"\}$

$S=\{S\}$

Example-1 :- [n এর মান 0 বা তার থেকে বেশি হতে পারে]

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

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

a^*

CFG :- $A \to aA/\epsilon$

DFA :-

example-2 :-

$L=\{a^n / n \geq 1\}$

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

$=a^+ \longrightarrow$ 1 or more

CFG :- $A \to aA/a$

example-3:-

L = {set of all strings over a, b}

= {ε, a, b, aa, ab, ba, bb, ...}

= (a+b)*

CFG:- A → aA / bA / ε

example-4:- CFG for set of all strings which length at least 2

Soln:- L = {aa, ab, ba, bb, aaa, ...}

= (a+b) (a+b) (a+b)*

A A B

CFG:- S → AAB

A → a/b

B → aB/bB/ε

example-5:- CFG for set of all strings of at least 3 0's.

E বলতে যেকোনো স্ট্রিং বোঝানো হচ্ছে যার সংখ্যক (0+1)* এর combination যেকোনো হতে পারে।

E0E0E0E

S → E0E0E0E

E → 0E/1E/ε

example-6:- Set of all strings which at most 2

Soln:- $L = \{\epsilon, a, b, aa, ab, ba, bb\}$.

$$RE = \underbrace{(a+b+\epsilon)}_{A}\ \underbrace{(a+b+\epsilon)}_{A}$$

CFG :-

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

example-7:- Starts with a and ends with b

Soln:- $a\ \underbrace{(a+b)^*}_{A} b$

CFG:- $S \rightarrow aAb$
$A \rightarrow aA/bA/\epsilon$

example-8:- Starts and ends with different symbol.

Soln:- $a(a+b)^*b \mid b(a+b)^*a$

$$= a\ \underbrace{(a+b)^*}_{A} b + b\ \underbrace{(a+b)^*}_{A} a$$

CFG-

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

ex-09:- starts and ends with same symbol-

Soln:- $a\underbrace{(a+b)^*}_{A}a \mid b\underbrace{(a+b)^*}_{A}b \mid \epsilon \mid a \mid b$

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

CFG:-

$S \rightarrow aAa \mid bAb \mid \epsilon \mid a \mid b$

$A \rightarrow aA \mid bA \mid \epsilon$

ex-10:- Even length string.

Soln:- $\underbrace{(a+b)}_{A}\ \underbrace{(a+b)}_{A}$

CFG:- $S \rightarrow AA$

$A \rightarrow a \mid b$

Or, $\underbrace{((a+b)\ (a+b))^*}_{B}$

$S \rightarrow BS \mid \epsilon$

$B \rightarrow AA$

$A \rightarrow aA \mid bA \mid \epsilon$

ex-: For generating a language that generates equal number of a 'a' s and 'b' s in the form $a^n b^n$, the CFG will be defined as,

$$G = \{(S, A), (a, b), (S \to aAb,\ A \to aAb/\epsilon)\}$$

Soln:-

$S \to aAb$

$\to aaAbb$ [by $A \to aAb$]

$\to aaaAbbb$ [by $A \to aAb$]

$\to aaabbb$ [by $A \to \epsilon$]

$\to a^3 b^3 \longrightarrow a^n b^n$

ex-:- $L = a^n b^{2n}$ Where $n \geq 1$

$= \{abb, aabbbb, aaabbbbbb, \ldots\}$

CFG- $S \to aSbb / abb$

Date- 09 May, 24

Topic :- Derivation Tree

A derivation Tree or Parse Tree is an ordered rooted tree that graphically represents the semantic information of strings derived from a CFG.

Here,

V = Variables / Non-terminal Symbol

T = Terminal Symbol.

P = Production rules.

S = Start Symbol.

There are 2 types of derivation tree. they are-

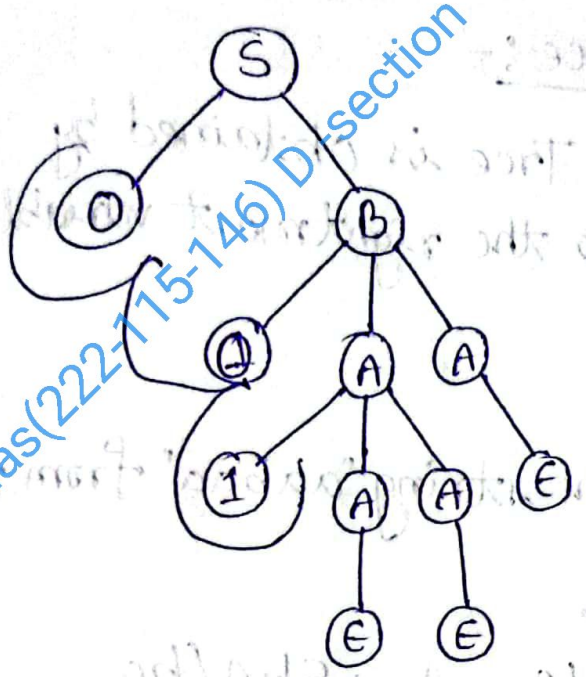
(i) Left derivative tree

(ii) Right derivative tree.

example- For the Grammar,

G = {V, T, P, S} Where

$S \rightarrow 0B$; $A \rightarrow 1AA \mid \epsilon$, $B \rightarrow 0AA$



Root vertex - Must be labeled by the start symbol.

Vertex - Labelled by Non-Terminal Symbols

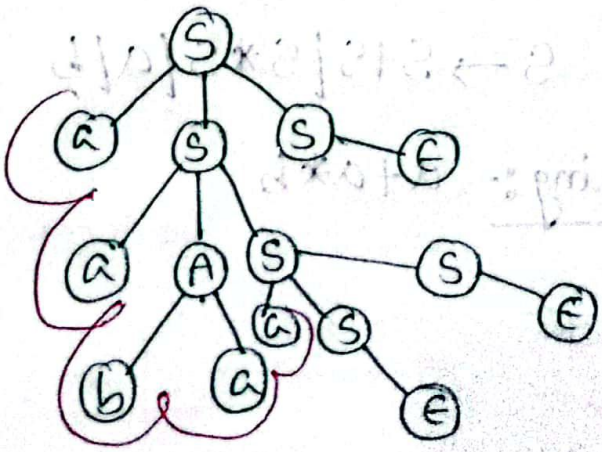
Leaves - Labelled by Terminal symbols or ϵ

String:- 001

Left Derivative Tree:

A left derivation tree is obtained by applying production to the left most variable in each step.

ex-: For generating the string "aabaa" from the grammar, S → aAS/aSS/ϵ, A → SbA/ba



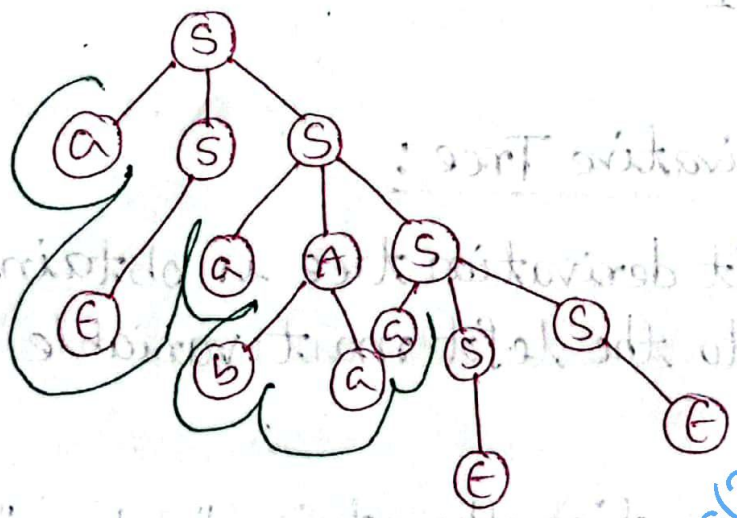
String:- aabaa

#Right Derivative Tree :-

A Right derivation Tree is obtained by applying production to the rightmost variable in each step.

ex- For generating the string "aabaa" from the Grammar,

$S \rightarrow aAS \mid aSS \mid \epsilon, \quad A \rightarrow SbA \mid ba$

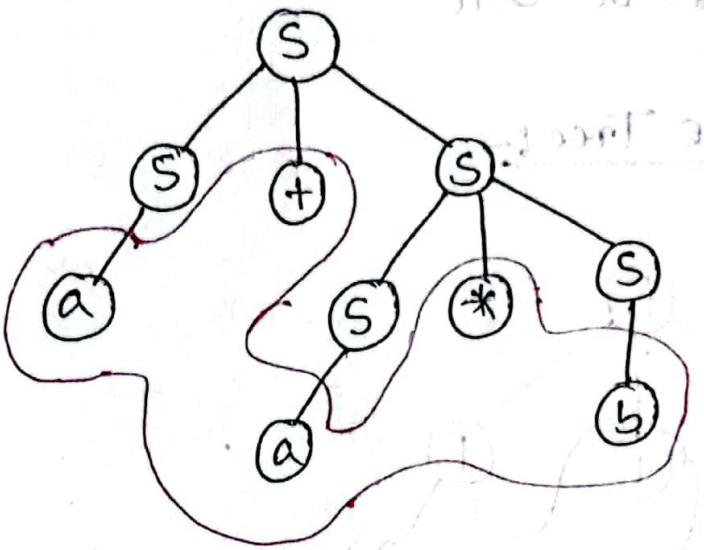


String :- aabaa

example-1 :- $S \rightarrow S+S \mid S*S \mid a \mid b$

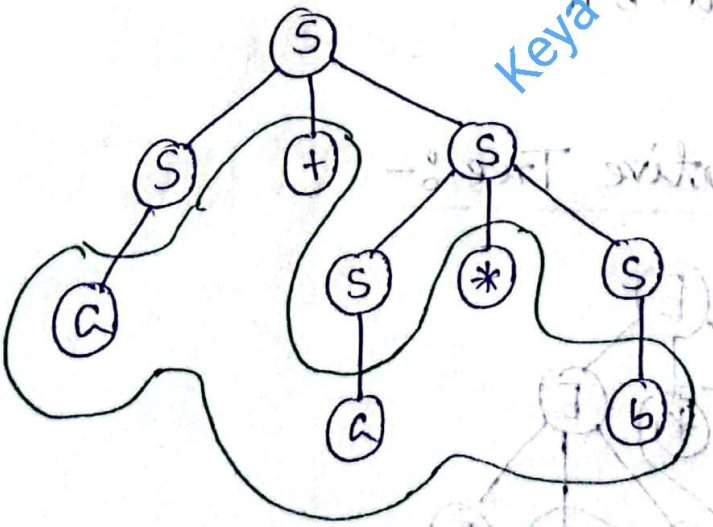
String :- a+a*b

Left Derivative Tree:-



String:- a+a*b

Right Derivative Tree:-

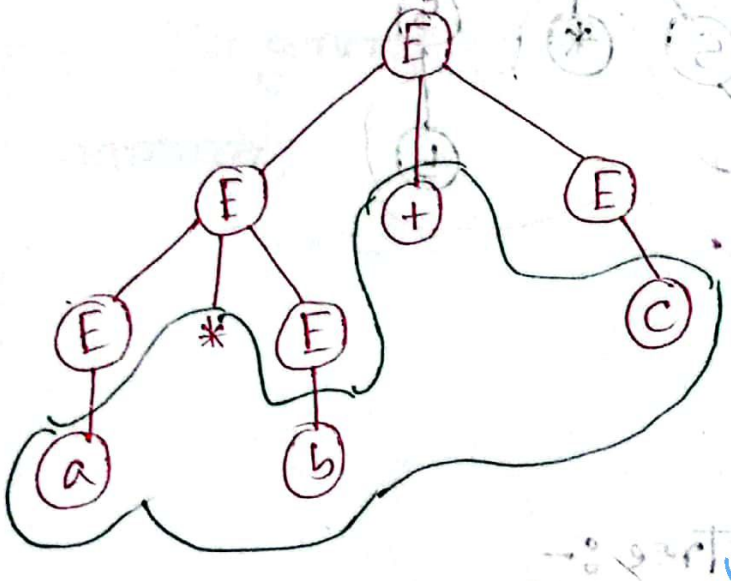


String:- a+a*b

ex-02 :- E = E+E / E*E / a / b / c

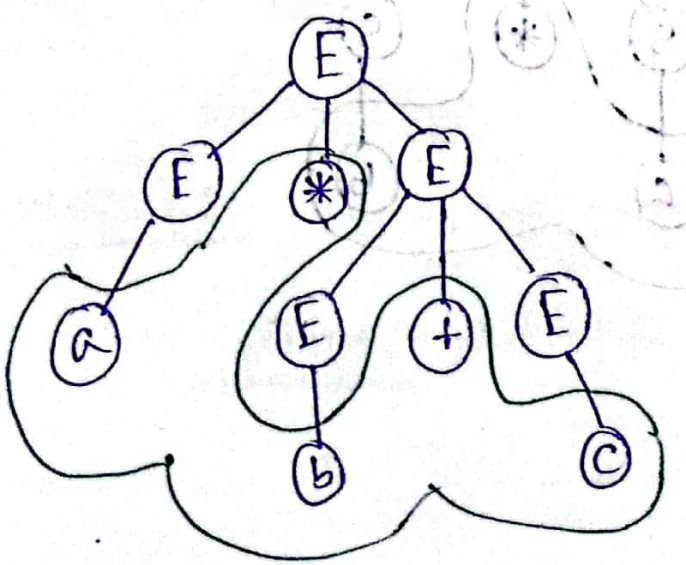
String : a * b + c

Left derivate Tree :-



String :- a * b + c

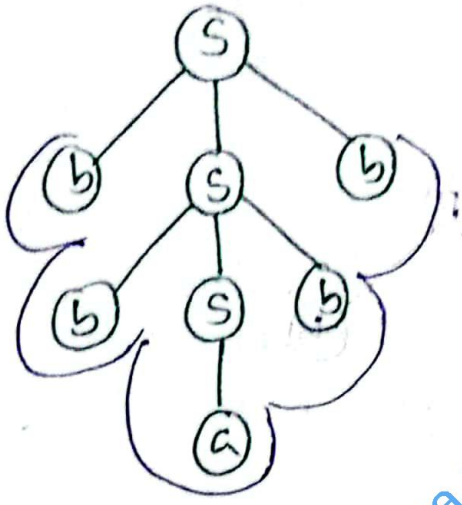
Right Derivative Tree :-



String :- a * b + c

ex-03:- $S \rightarrow bSb \mid a \mid b$

String- 'bab' "bbabb"



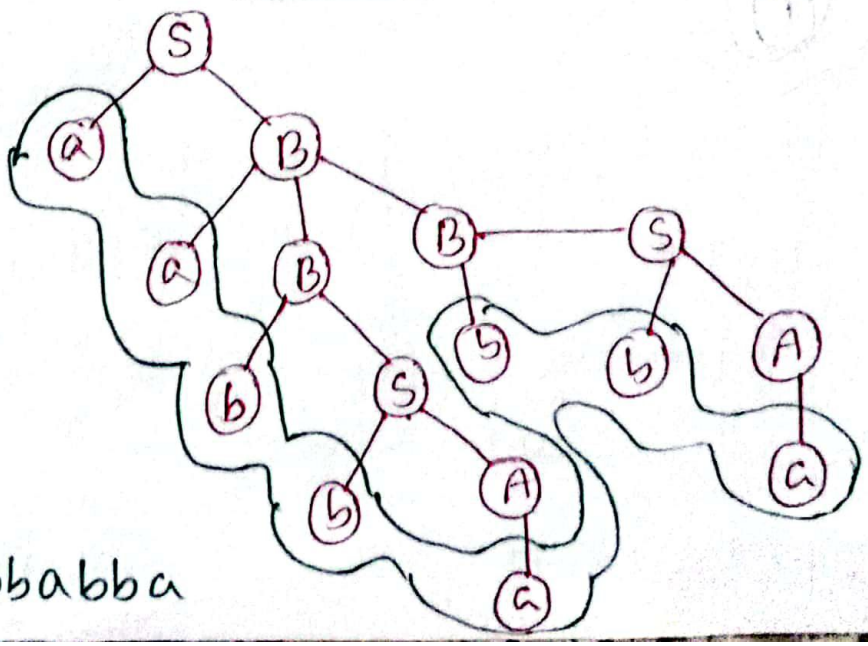
String:- bbabb.

ex-04:- $S \rightarrow aB \mid bA$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

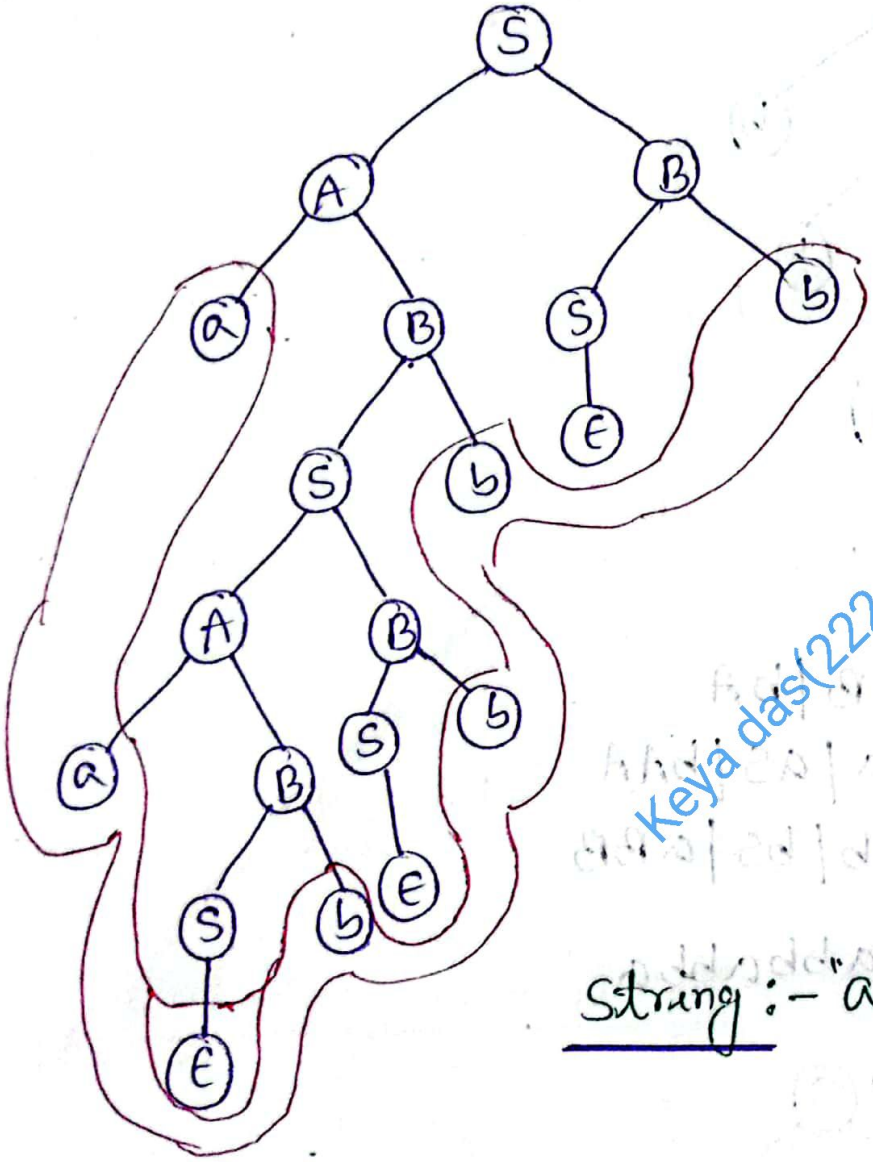
String:- aabbabba



String:- aabbabba

ex-05 :- $S \rightarrow AB/\epsilon$, $A \rightarrow aB$, $B \rightarrow Sb$

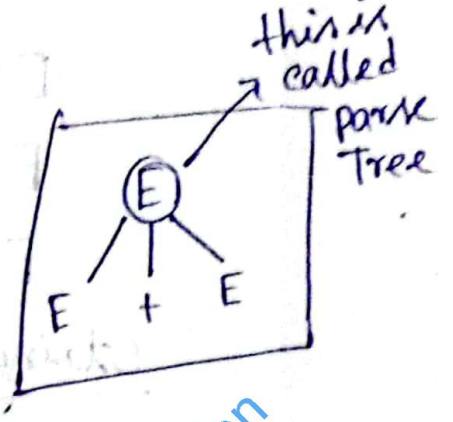
String : "aabbbb"



String :- "aa bbbb"

Ambiguity in Grammar :-

যেকোনো Grammar-এ যদি two or more derivation tree (either left/Right) পাওয়া যায়, তাকে ambiguous grammar বলে।



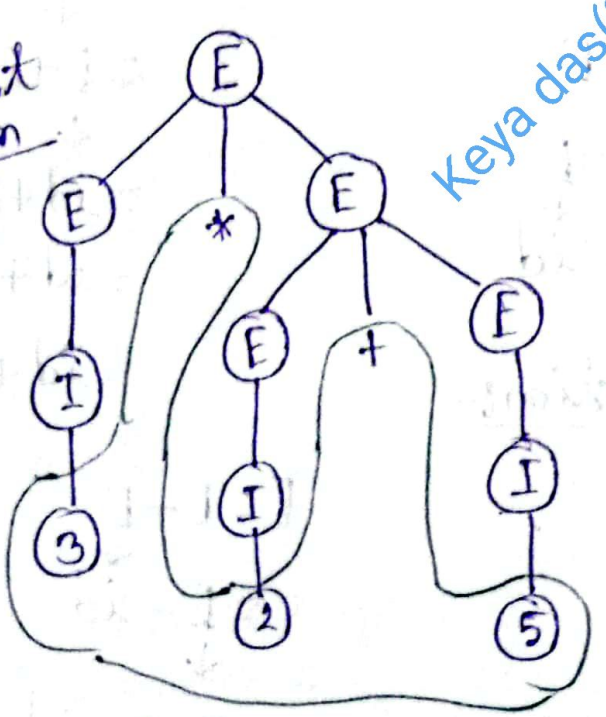
example-1 :-

E = I
E = E+E
E = E*E
E = (E)
I = ϵ | 0 | 1 | 2 | . . . | 9

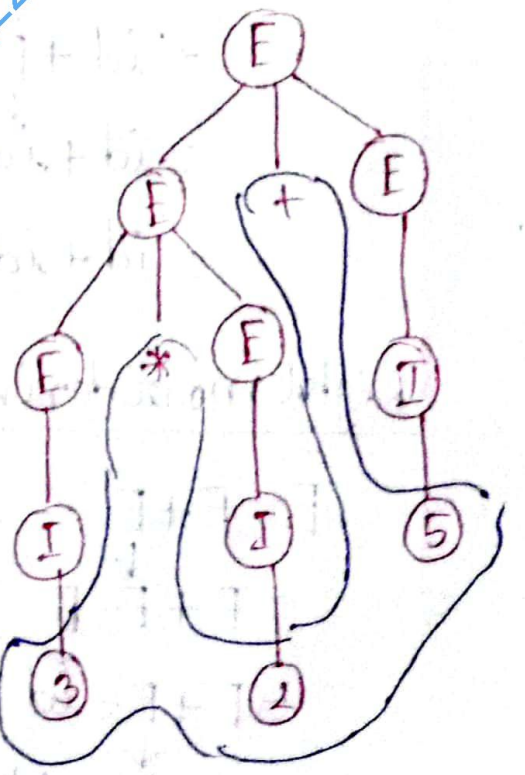
String: 3*2+5

Soln:

Left most derivation



String: 3*2+5



String: 3*2+5

∴ The Grammar is ambiguous.

Ex-2:- Check the grammar ambiguous or not.

$E \rightarrow E+E$

$E \rightarrow E-E$

$E \rightarrow id$

String: id+id-id

Soln:- Left most derivation:-

$E = E+E$
$= id+E$
$= id+E-E$
$= id+id-E$
$= id+id-id$

Left:-

$E = E-E$
$= E+E-E$
$= id+E-E$
$= id+id-E$
$= id+id-id$

Right most derivation:-

$E = E+E$
$= E+E-E$
$= E+E-id$
$= E+id-id$
$= id+id-id$

$E = E-E$
$= E-id$
$= E+E-id$
$= E+id-id$
$= id+id-id$

∴ The grammar is a ambiguous.

Ex-3:- Check the grammar ambiguous or not.

$S \rightarrow asb \mid ss$

$S \rightarrow \epsilon$

string – "aabb"

Soln:-

$S \rightarrow asb$

$\rightarrow aasbb$ [by $S \rightarrow asb$]

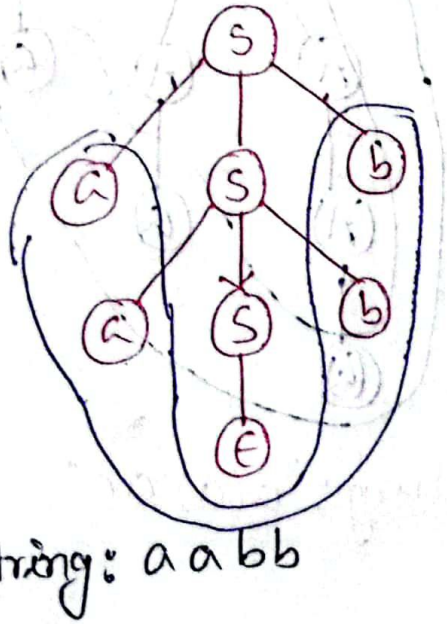
$\rightarrow aabb$ [by $S \rightarrow \epsilon$]

$S \rightarrow SS$

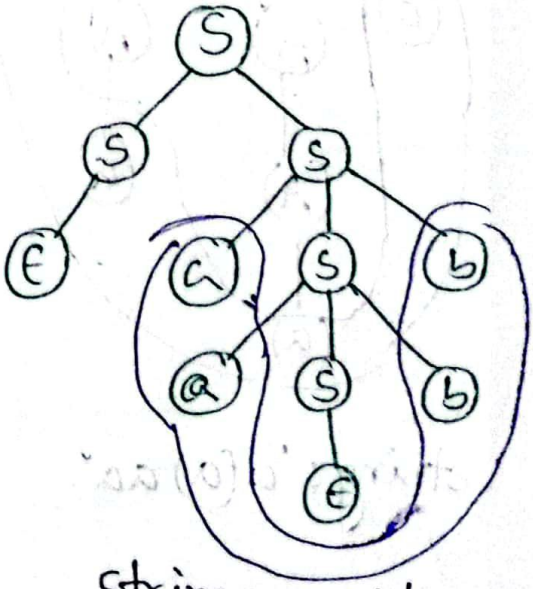
$\rightarrow S asb$

$\rightarrow S aasbb$

$\rightarrow aa bb$



∴ String: aabb



String: aabb

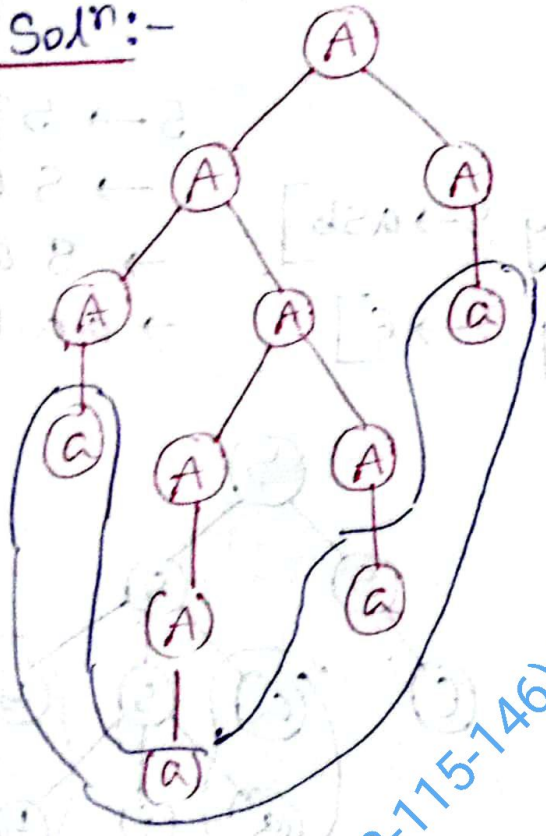
∴ The grammar is a ambiguous.

Ex-4:- Check whether the grammar G is ambiguous or not.

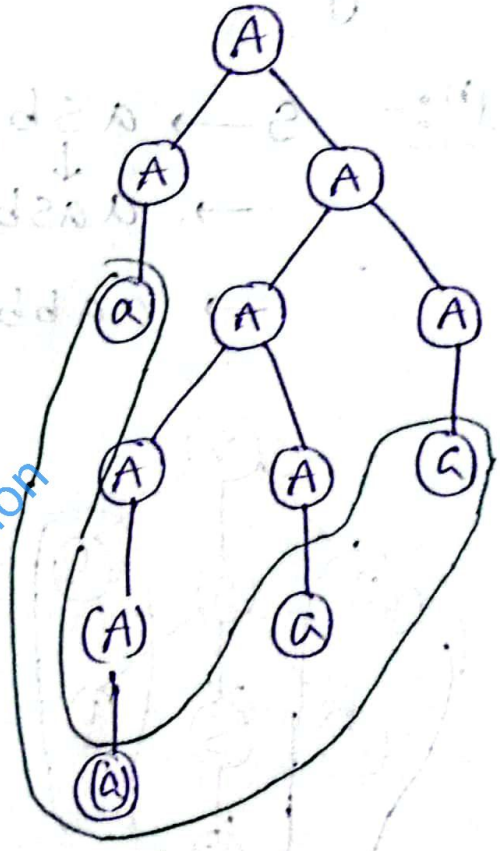
$A \rightarrow AA$
$A \rightarrow (A)$
$A \rightarrow a$

String: "a(a)aa"

Soln:-



String: "a(a)aa"



String: "a(a)aa"

∴ this grammar is a ambiguous.

Date - 13 May, 24

Check the grammar ambiguous or not.

$E = E+E$

$E = E-E$

$E = x/y/z$

String :- $x-y+z$

Soln :- Leftmost Derivation :-

$E = E+E$

$= E-E+E$

$= x-E+E$

$= x-y+E$

$= x-y+z$

$E = E-E$

$= x-E$

$= x-E+E$

$= x-y+E$

$= x-y+z$

Rightmost Derivation :-

$E = E+E$

$= E+E-E\,z$

$= E-E+z$

$= E-y+z$

$= x-y+z$

$E = E-E$

$= E-E+E$

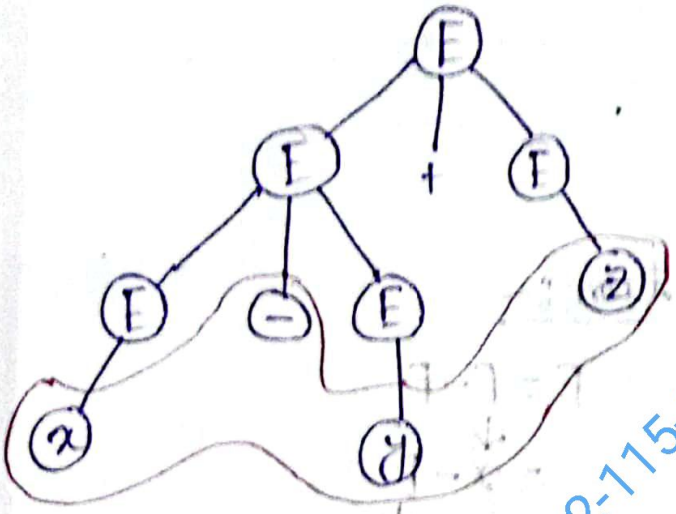
$= E-E+z$

$= E-y+z$

$= x-y+z$

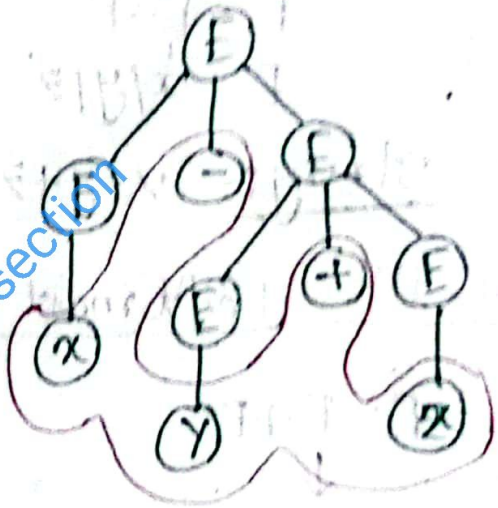
∴ This grammar is a ambiguous.

Tree :- Left / Parse Tree :-



String :- x-y+z

Right / Parse Tree



String : x-y+z

Topic :- Push Down Automata (PDA) :-