Introduction To Formal Languages and Theory Of Automata

Automata

Automata something that works automatically.

Input

Alphabets = set of input symbols. they are denoted by Σ

Formal	Informal
alphabets	alphabets
rules	no rules
Regular Lang.	
Regular Expression	

Automata is the representation of:

1. Regular Expression (RE)
2. Finite Automata (FA) → (i) DFA (ii) NFA
3. Transition Graph (TG)
   - i) Mealy machine
   - ii) Moorey machine
4. (CFG) Context-Free Grammar
5. Push Down Automata (PDA)
6. Turing machine (TM)

Regular Expression: (R.E)

Language over $\Sigma = \{a, b\}$ which accepts 'a' generates

$RE = a$

Language over $\Sigma = \{a, b\}$ which accepts 'b'

$R.E. = b$

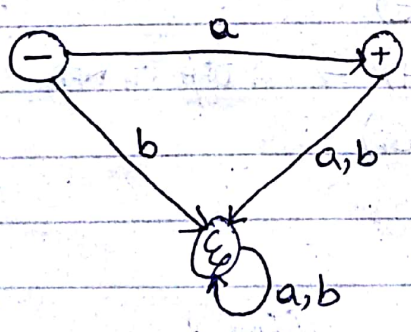
Finite Automata: (F.A)

States:

Basic & fundamental	Initial State	⊖	Final State	⊕
	Starting "	→○	Accepting "	◎
~~Intermediate states~~ optional as needed	Intermediat state	○	Error state (or) dump "	⊗

Sol.

① $R.E = a$



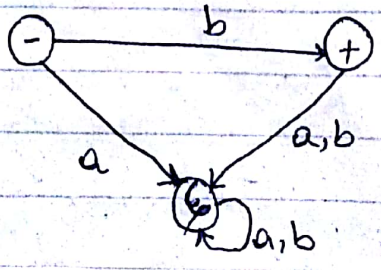
Operators of R.E

i) . and then

ii) + or

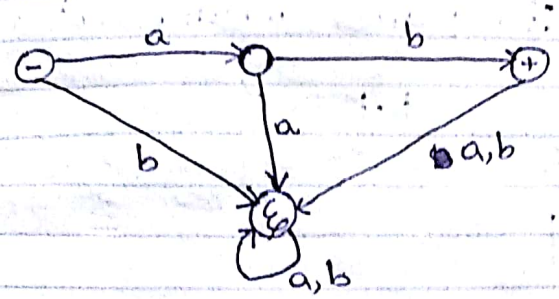
Sol.

② $R.E = b$



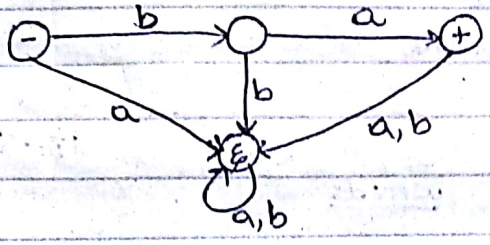
③ Language over $\Sigma = \{a, b\}$ which accepts a then b.
↑ and

Sol. $R.E = a \cdot b$



Sol.

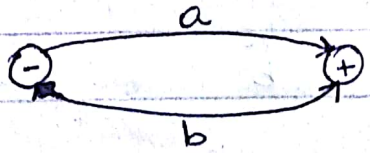
④ $R.E = b \cdot a$

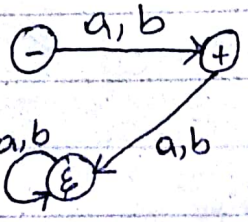


⑤ Language over $\Sigma = \{a, b\}$ which accepts a or b

Sol

$R.E = a + b$

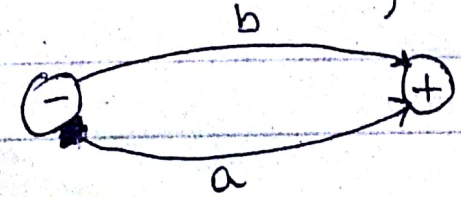




Same

Sol.

⑥ $R.E = b + a$



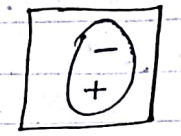
R.E: $a+b = b+a$

R.E : $a.b \neq b.a$

⑦ Language over $\Sigma = \{a, b\}$ which accept nothing (null)

Sol.

$R.E = \wedge$ $\qquad$ $R.E = \emptyset$



→ a state can be simultaneously initial state or final state

13-9-18

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

① $R.E = a$

the lang. over $\Sigma = \{a, b\}$ that accepts a

② $R.E = b$

③ $R.E = a+b$

④ $R.E = a.b$

that accepts a then b

b after a

a and b

⑤ $R.E = b.a$

⑥ $R.E = \wedge$

words

a

b

a

b

ab

ba

$\wedge$ (null)

Operator on R·E

① Kleenie Plus +
② Kleen Star *

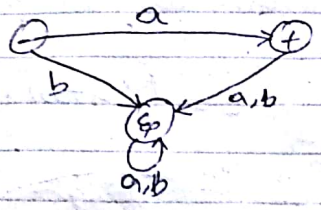
if we apply operators on R·E that n also a R·E

R.E	Words	English Sentence.
$R \cdot E = a^+$	a aa aaa aaaa aaaaa.....	the language over $\Sigma=\{a,b\}$ that accepts one or more a's.
$R \cdot E = a^*$	^ a aa aaa aaaa aaaaa.....	the language over $\Sigma=\{a,$ accepts zero or more a's
$R \cdot E = (a \cdot b)^+$	ab ab ab ab ab ab ..... ab ab ab ab ab ab ab ab ab----	the language over $\Sigma=\{a,b\}$ accepts one or more ab's ↓ pairs of ab chunks of ab clumps of ab
$R \cdot E = (a \cdot b)^*$	^ ab ab ab ab ab ab	the language over $\Sigma=\{a,b\}$ accepts zero or more pairs of ab
$R \cdot E = a^* \cdot b^*$	^ a b aa bb aaa bbbb a b aaaa----	the language over $\Sigma=\{a,b\}$ that accepts zero or more a's then zero or more b's

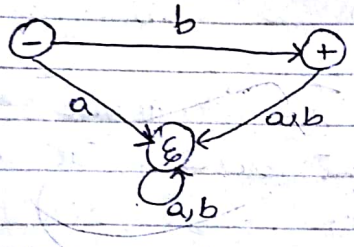
R.E	words	English Sentence
$R.E = (a+b)^+$	a b ..... aa ab ba bb ..... aaa aab aba abb baa bab bba bbb	the language over $\Sigma = \{a, b\}$ ~~accepts one or more a's at a time or one b at a time.~~ that accepts / generates anything except null
$R.E = (a+b)^*$	^ a b ..... aa ab ba bb	the language over $\Sigma = \{a, b\}$ that accepts any thing
$R.E = a^* + b^*$	^ a aa bbbb aaaaa..... bb	the language over $\Sigma = \{a, b\}$ that accepts Zero or more a's OR Zero or more b's.

RE to Finite Automata

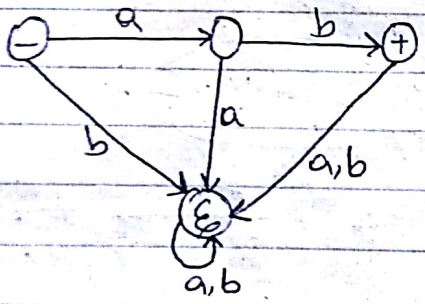
① RE = a



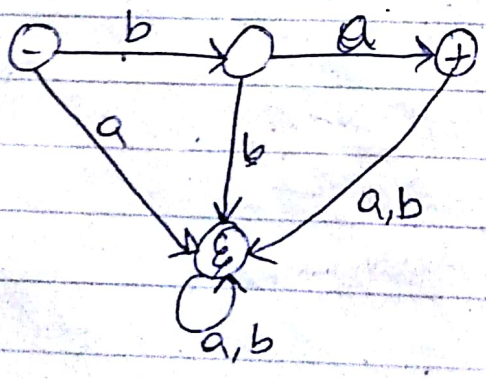
② R.E = b



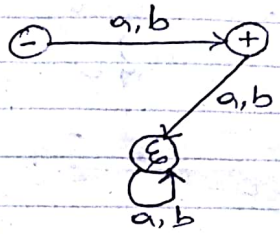
③ R.E = a.b



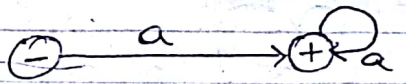
④ R.E = b.a



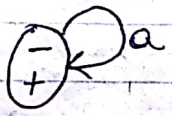
⑤ R.E = a+b



⑥ R.E = a^+

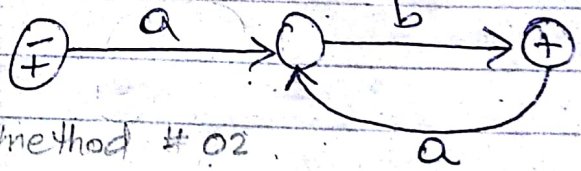


⑦ R.E = a^*

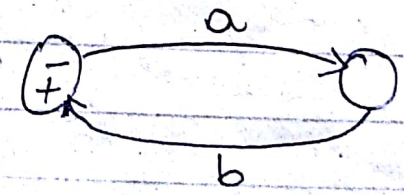


⑧ R.E = $(a.b)^*$

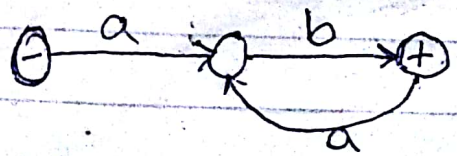
method # 01



method # 02

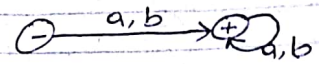


⑨ R.E = $(a.b)^+$



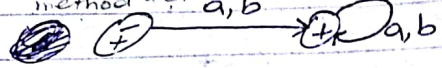
a^*, b^*

(10) $R.E = (a+b)^+$



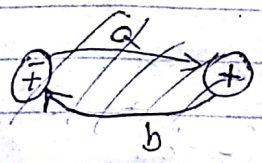
(11) $R.E = (a+b)^*$

method #01



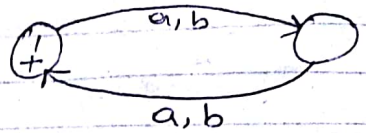
method #02

(12) $a^* \cdot b^*$



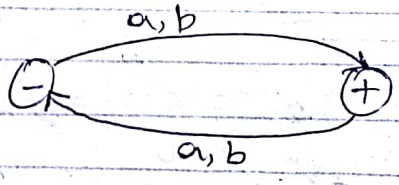
(i) The language over $\Sigma = \{a, b\}$ that accepts all words even length

$R.E = ((a+b) \cdot (a+b))^*$



(ii) The language over $\Sigma = \{a, b\}$ that accepts all words of odd length.

$RE = (a+b) \cdot ((a+b) \cdot (a+b))^*$



Initia
final

Algorithm

* Do not
* Do not
* Make
* Make

Q) The l

i) Anything

ii) Nothin

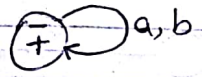
Initial state (exactly one)
Final state (zero or more)

Algorithm of inverting of an Finite automata (F.A):-

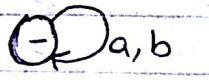
1. Do not change any state or transition
2. Do not change initial state
3. Make each final state as non-final state.
4. Make each Non-final state as final state.

Q) The language over $\Sigma = \{a, b\}$ that accepts:

i) Anything



ii) Nothing



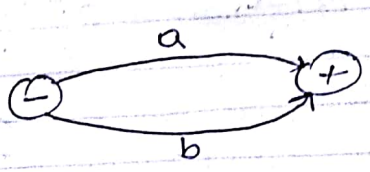
(∀ DFA is NFA) ∧ (∀ NFA is not DFA)

Non-Deterministics Finite Automata :- (NFA)

(∀ DFA is NFA) ∧ (∀ NFA is not DFA)

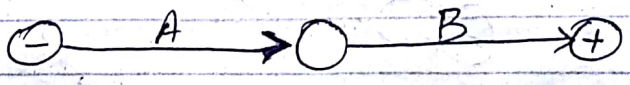
R.E. to NFA

① A + B



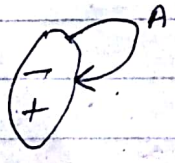
NFA / DFA

② A · B

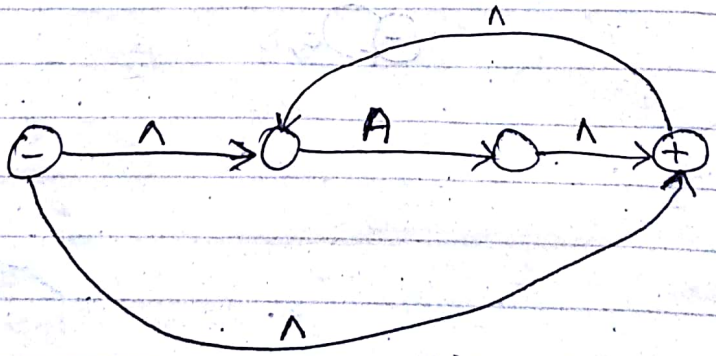


NFA / DFA

③ A*



DFA



NFA

- More than one transition of same input
- Can Skip an Input
- Can Have null transition.

is not DFA)

:- (NFA).

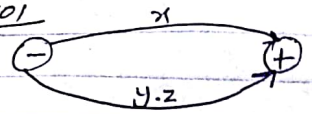
ot DFA)

Example #01

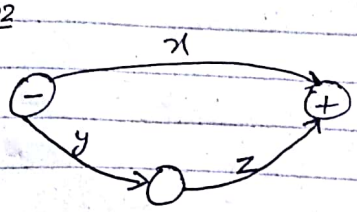
$x + yz$

This is the form of A+B

Step#01



Step#02

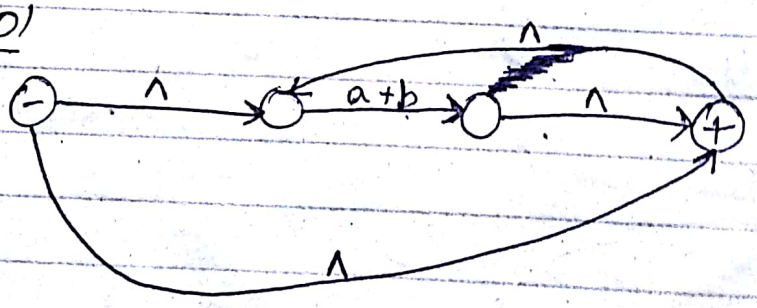


Example #02 :

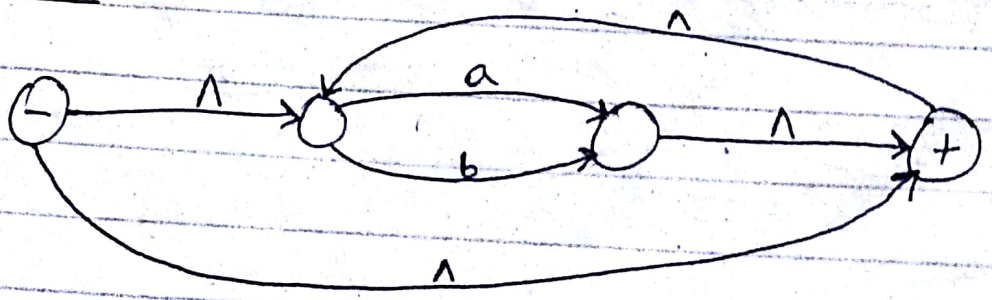
$(a+b)^*$

this is the form A^*

Step#01



Step#02

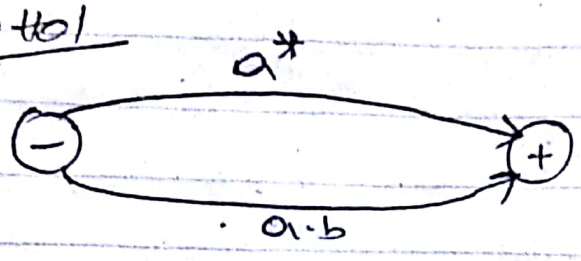


Example #03

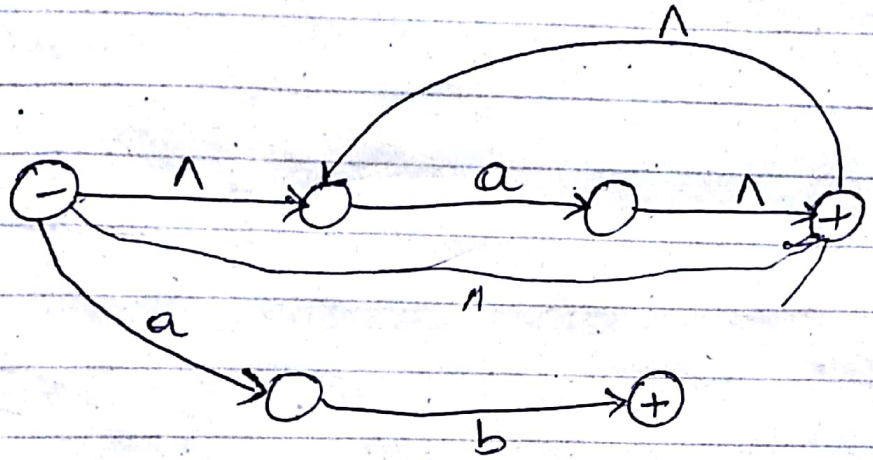
$a^* + a.b$

This the form of $A + B$

Step #01



Step #02



Example #04

① R.E = $(a \cdot b)^*$

② R.E = $a^* \cdot b^*$

③ R.E = $a^* + b^*$

④ RE = $(a+b)^*$

⑤ RE = $(a+ab)^*$

⑥ RE = $(ab+ba)^*$

⑦ RE = $a^* + ab$

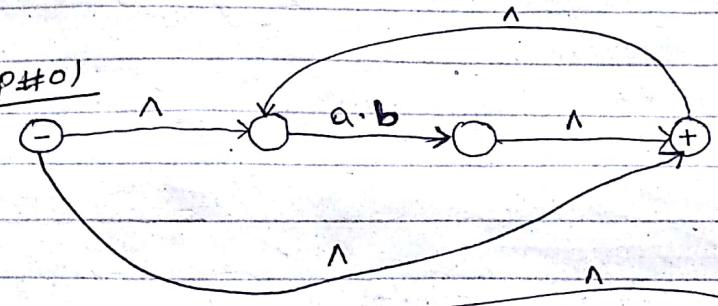
⑧ RE = $a^* \cdot b$

⑨ RE = $a^* + b$

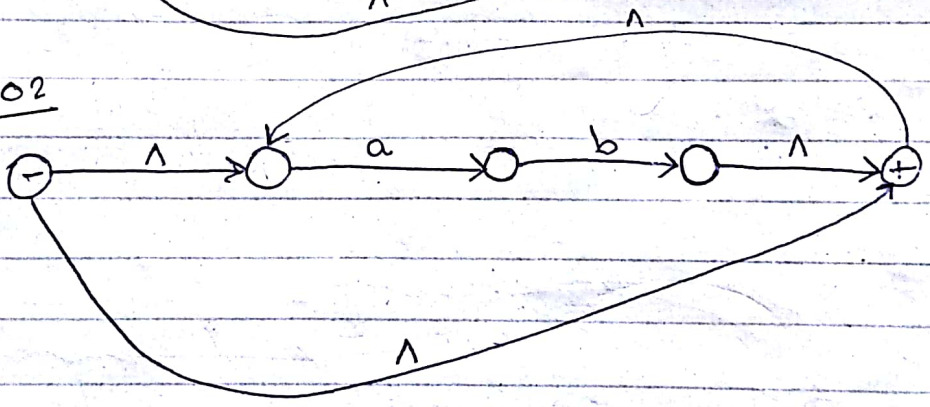
⑩ RE = $(a^*+b^*)^*$

④

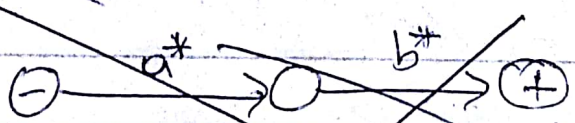
Step#01



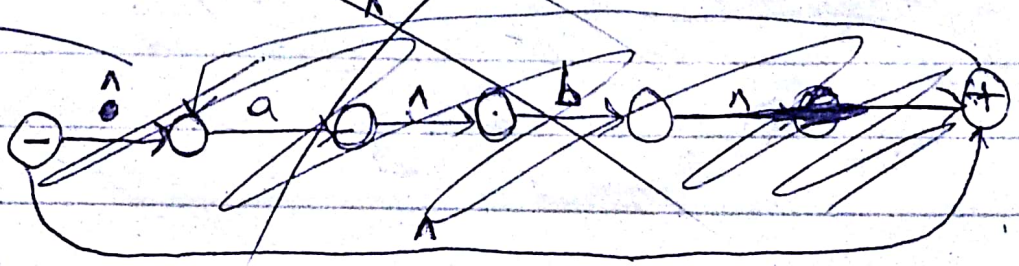
Step#02



② ~~Step #01~~

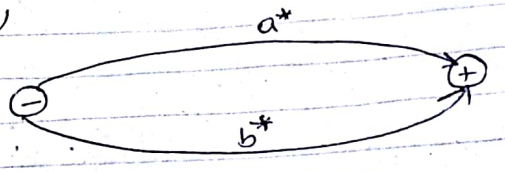


~~Step#02~~

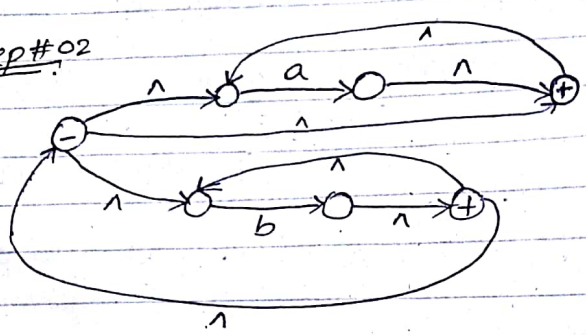


③ R.E = $a^* + b^*$

Step #01

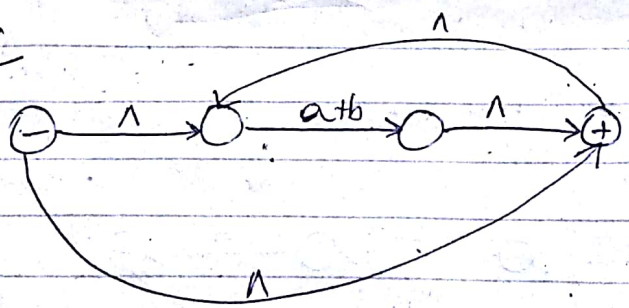


Step #02

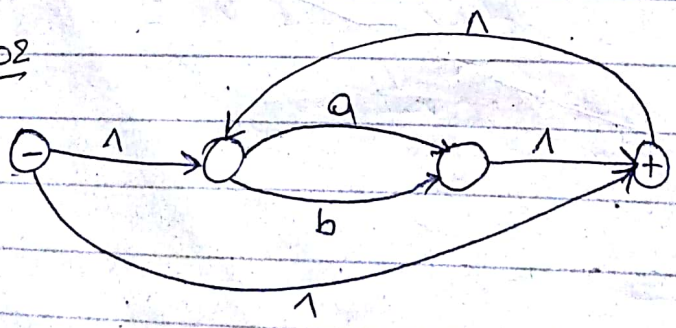


④ R.E = $(a+b)^*$

Step #01

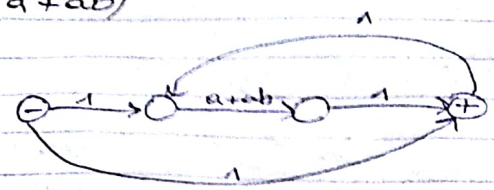


Step #02

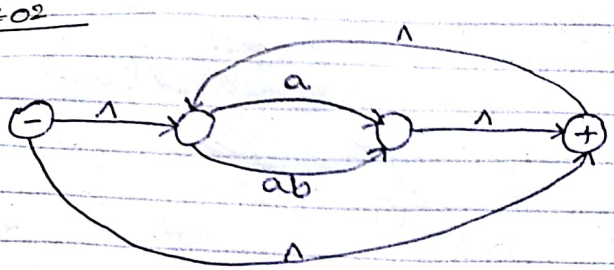


⑤ RE= $(a+ab)^*$

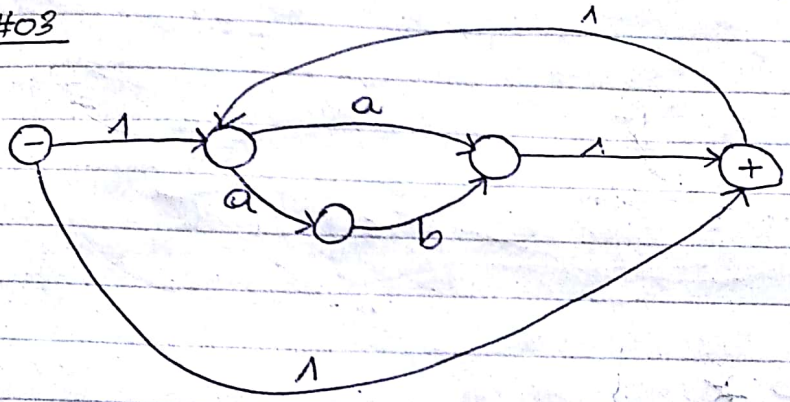
Step#01



Step#02

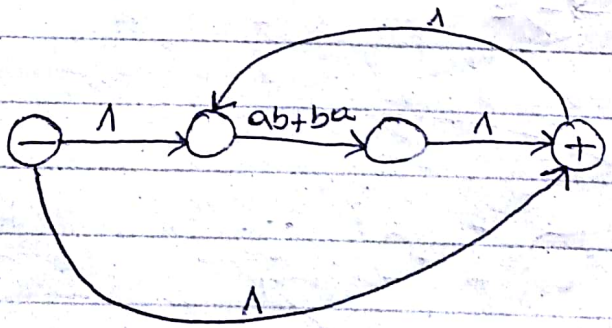


Step#03

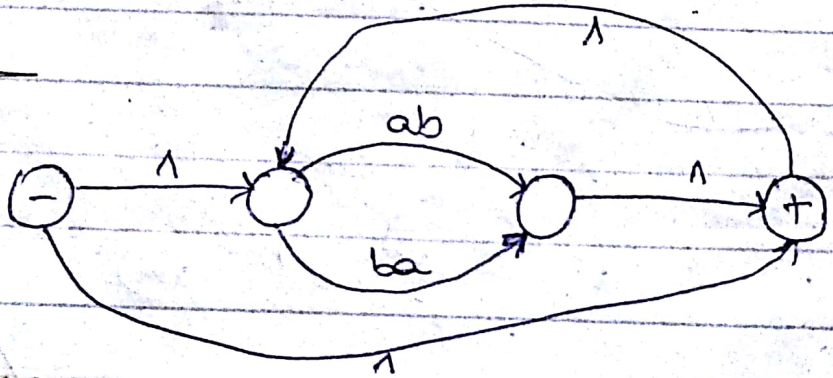


⑥ RE = $^{\$}(ab + ba)^*$

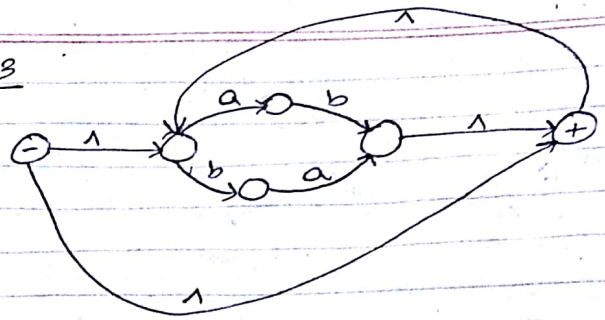
Step#01



Step#02

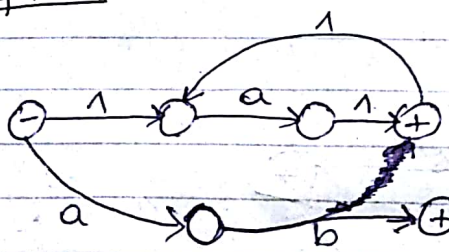


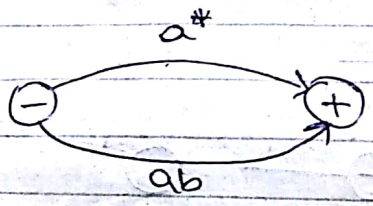
Step#03



⑦ RE = $a^* + ab$

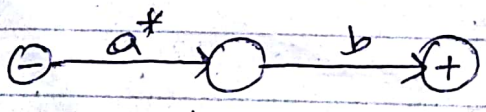
Step#01



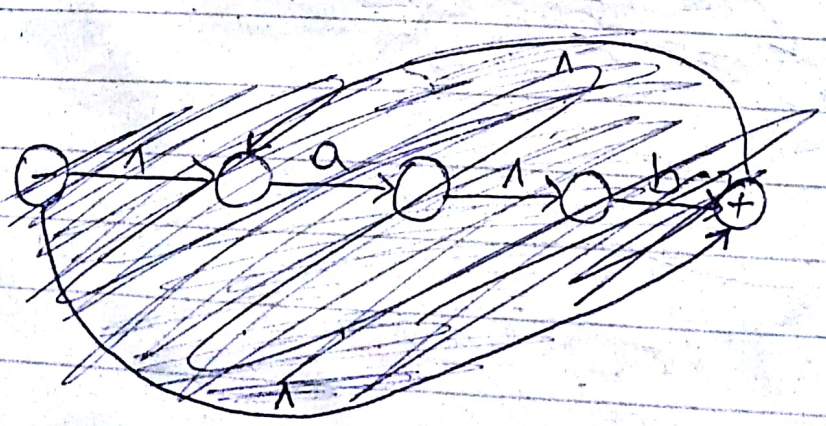


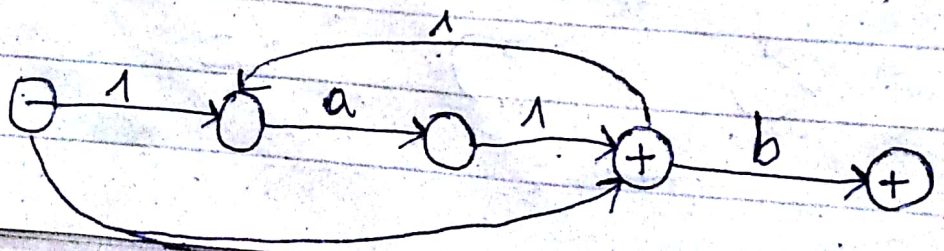
⑧ RE = $a^* . b$ ★

Step#01



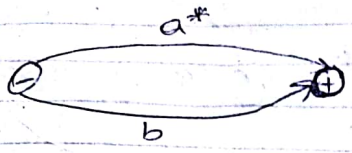
Step #02



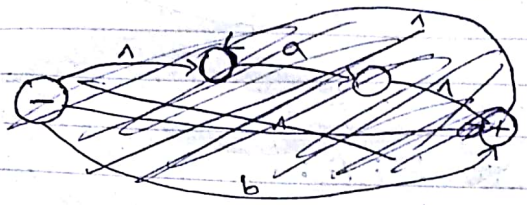


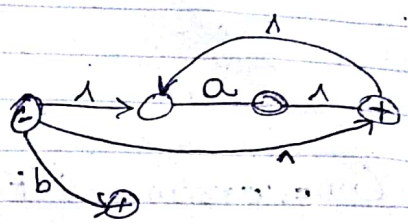
⑨ $a^* + b$

Step#01

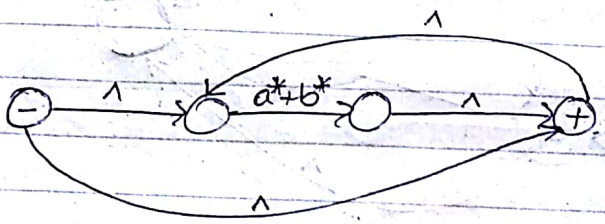


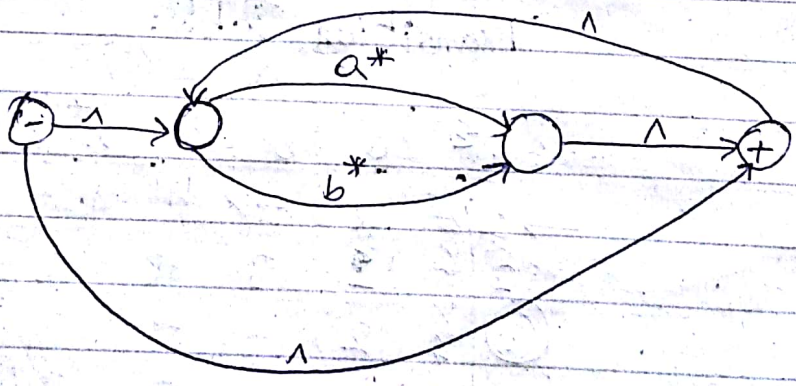
Step#02

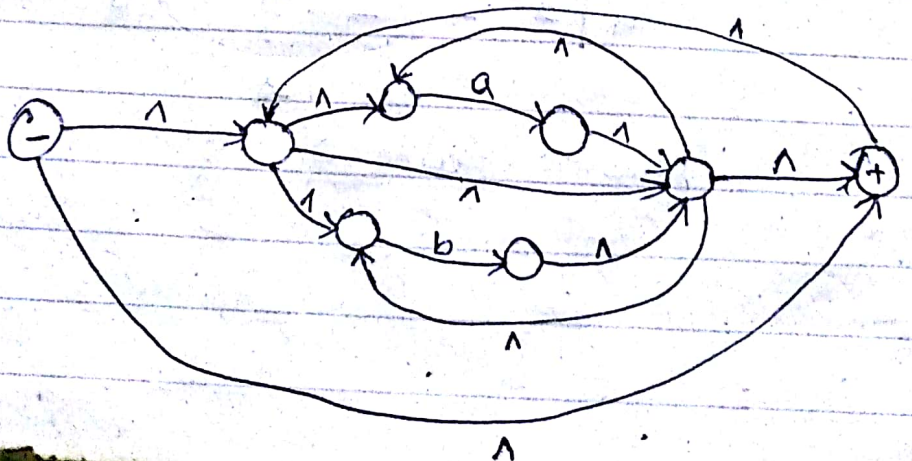




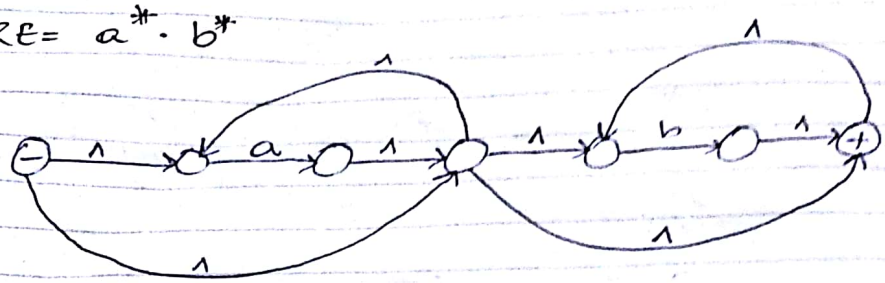
⑩ $(a^* + b^*)^*$



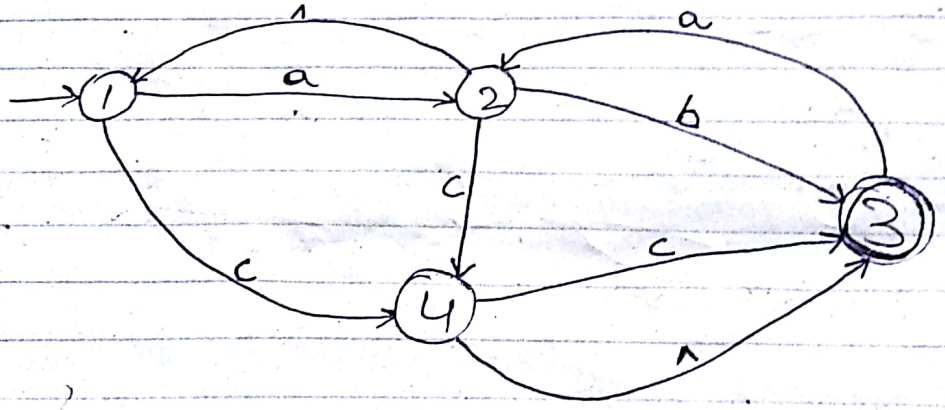




② RE = $a^* \cdot b^*$



Conversion NFA to DFA:



R.W

Words	
Yes	No
ab	abc
abab	a
cc	b
c	ccb
ccab	caa
ccac	
ccac	
ccac	

Transition Table:

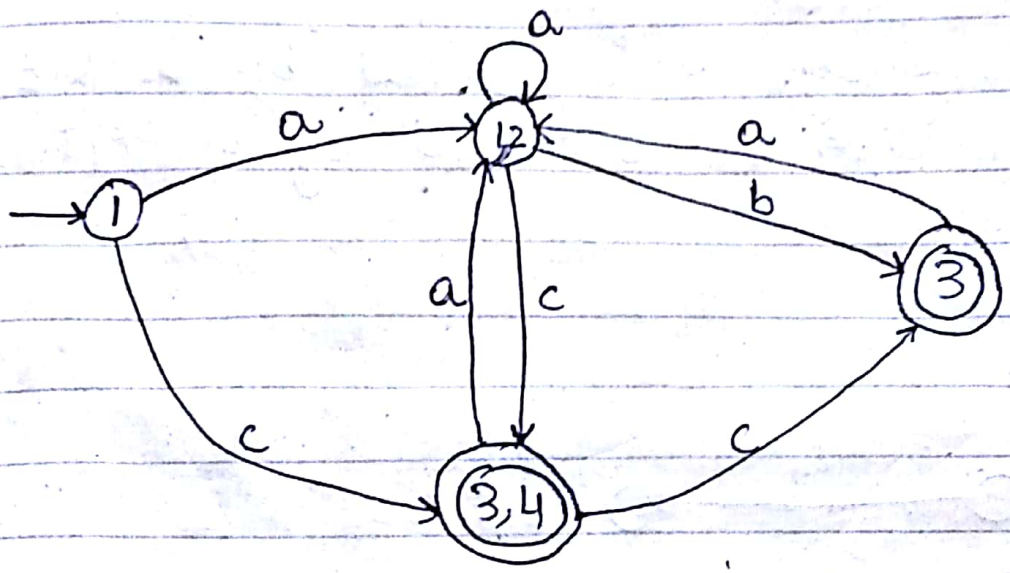
States	ε	a	b	c	^*
→ 1		2	–	4	1
2		–	3	4	1,2
(3)		2	–	–	3
4		–	–	3	3,4

1,2 means 1 or 2

Transition table of given NFA

states \ Σ	$a\Lambda^*$	$b\Lambda^*$	$c\Lambda^*$
→1	1, 2	–	3, 4
1, 2	1, 2	3	3, 4
(3, 4)	1, 2	–	3
(3)	1, 2	–	–

Required DFA :-



Kleene's Theorem:-

Part-I: If a language is accepted / expressed by a FA, it can also be accepted by a TG.

Part-II: If a language is accepted by a FA/TG, it can also be accepted by a R.E.

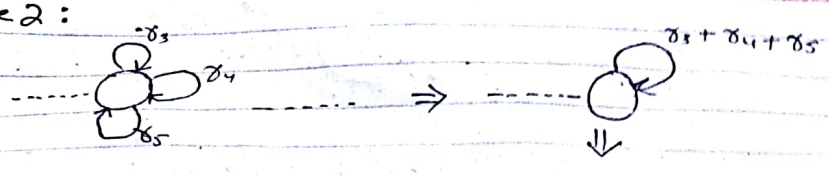
Part-III: If a language is accepted by a R.E, it can also be accepted by a FA/TG.

Part - II

$r = a/a^*/r.e/$ sub r.

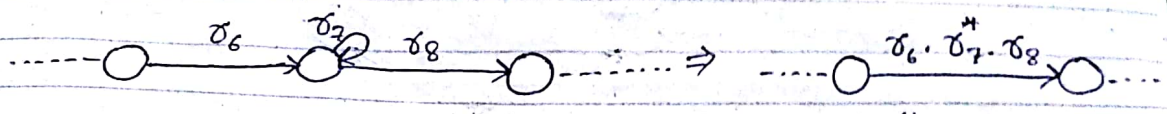
Case 0:

Case 2:



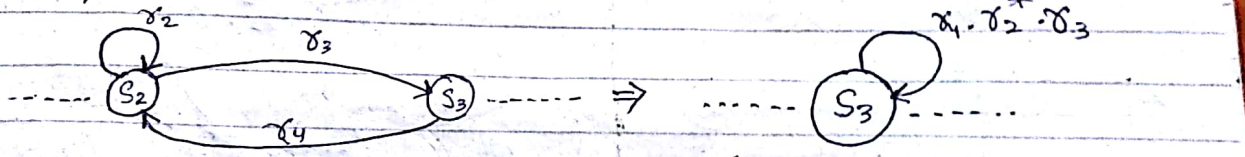
$$RE = \gamma_3 + \gamma_4 + \gamma_5$$

Case 3:



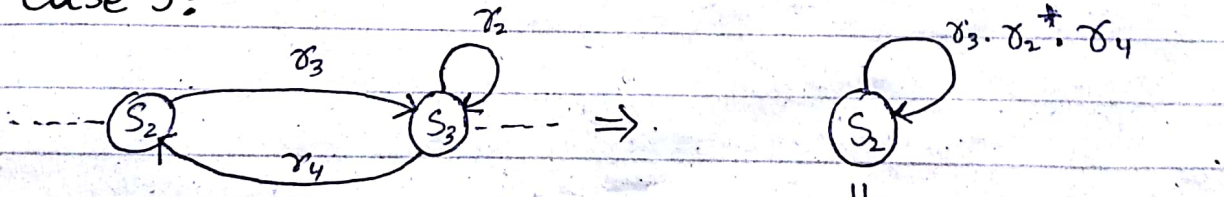
$$RE = \gamma_6 \cdot \gamma_7^* \cdot \gamma_8$$

Case 4:



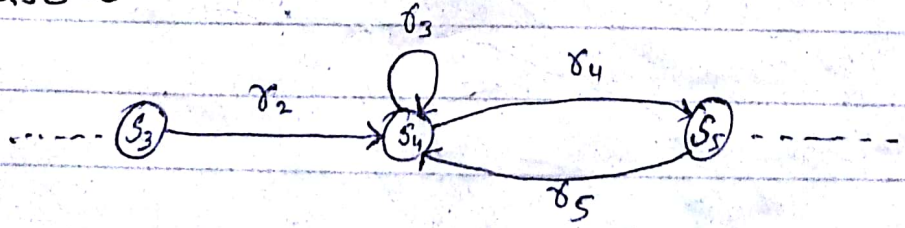
$$R.E = \gamma_4 \cdot \gamma_2^* \cdot \gamma_3$$

Case 5:



$$R.E = \gamma_3 \cdot \gamma_2^* \cdot \gamma_4$$

Case 6:



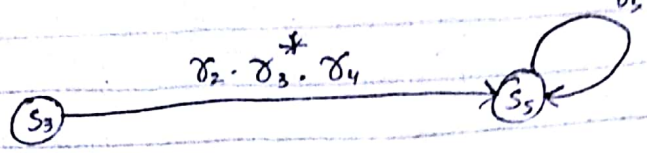
See Case 7 for its RE both are same.

Dec —

Q4(a)

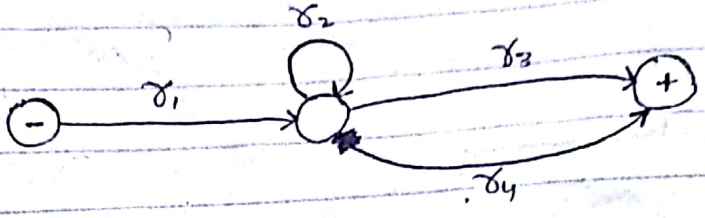
↳ RE

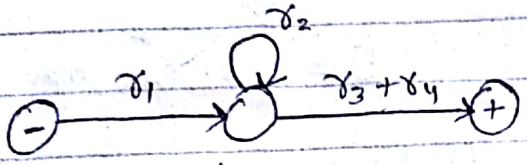
Case 7:



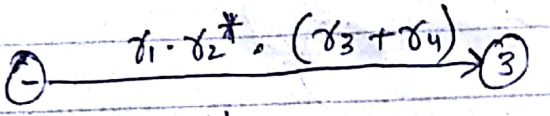
$$R.E = (\gamma_2 \cdot \gamma_3^* \cdot \gamma_4) \cdot (\gamma_5 \cdot \gamma_3^* \cdot \gamma_4)^*$$

Case 8:



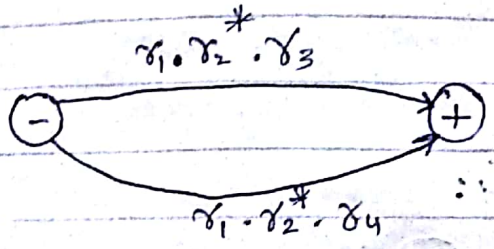


⇓



⇓

$$R.E = \gamma_1 \cdot \gamma_2^* (\gamma_3 + \gamma_4)$$



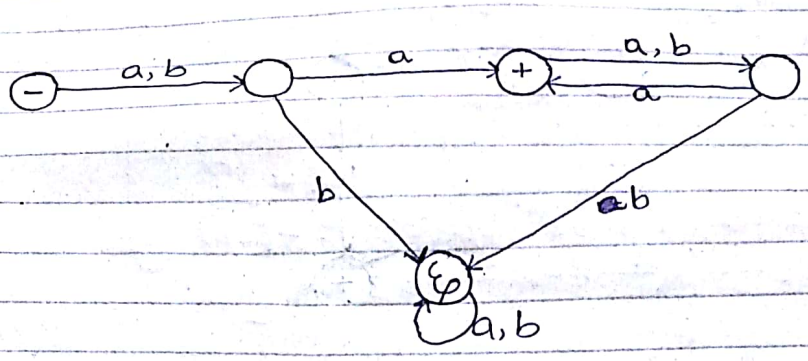
⇓

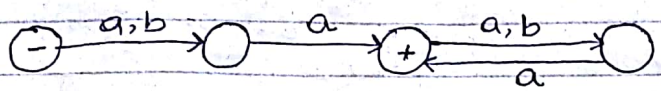
$$R.E = (\gamma_1 \cdot \gamma_2^* \cdot \gamma_3) + (\gamma_1 \cdot \gamma_2^* \cdot \gamma_4)$$

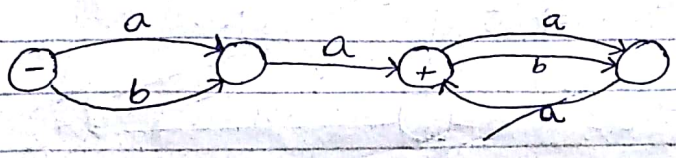
JULY – 2015

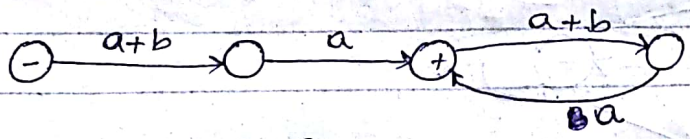
QUESTION: 2 (c)

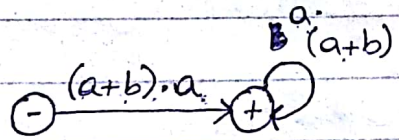
FA to RE

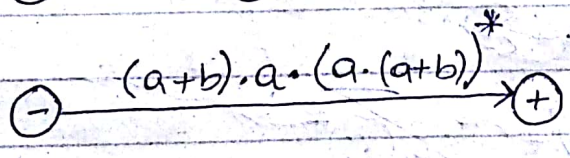










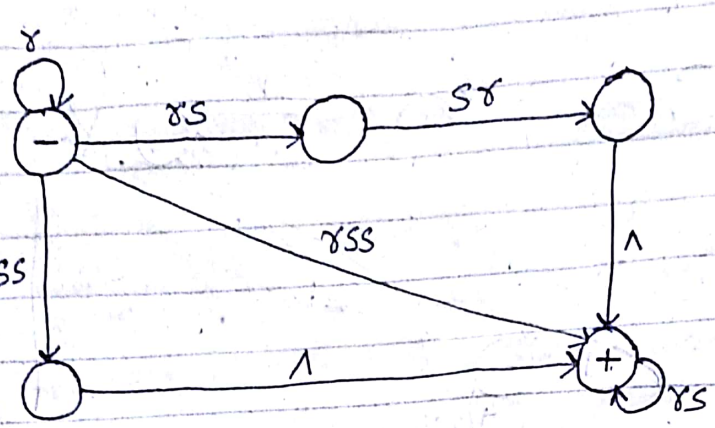


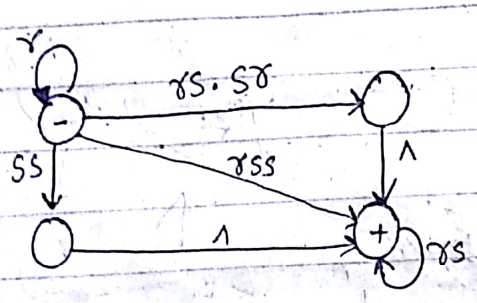
$$RE = (a+b) \cdot a \cdot (a \cdot (a+b))^*$$

JULY - 2015

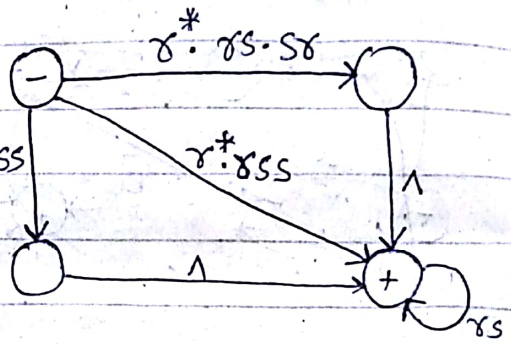
QUESTION: 3(a)

TG to RE

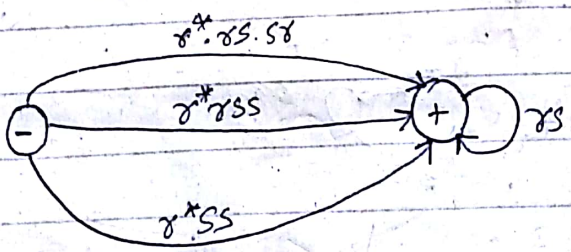


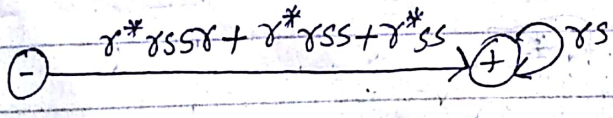


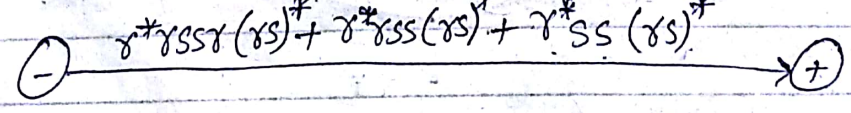
⇒ $\gamma^*.SS$



⇒







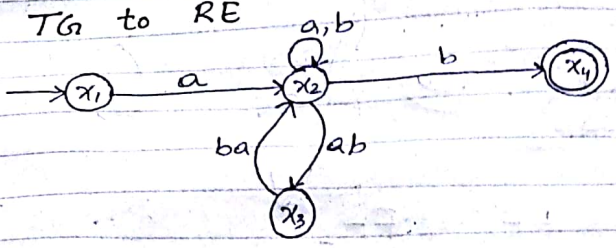
$$RE = \gamma^*\gamma SS\gamma(\gamma S)^* + \gamma^*(\gamma S)^*\gamma SS + \gamma^*(\gamma S)^*SS$$

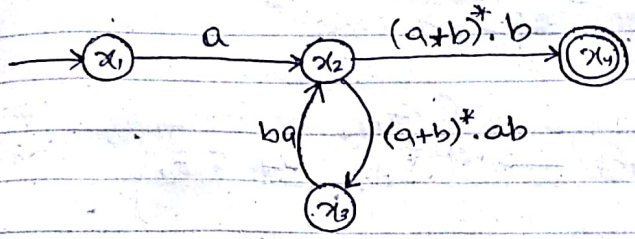
$$RE = \gamma^*.(\gamma SS\gamma + \gamma SS + SS).(\gamma S)^*$$

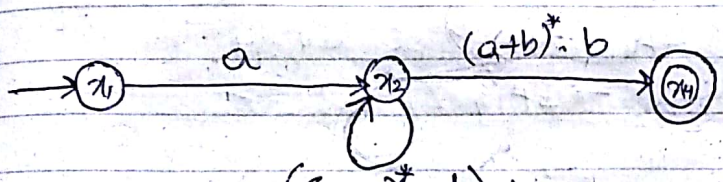
DECEMBER – 2014

QUESTION : 4 B (a)

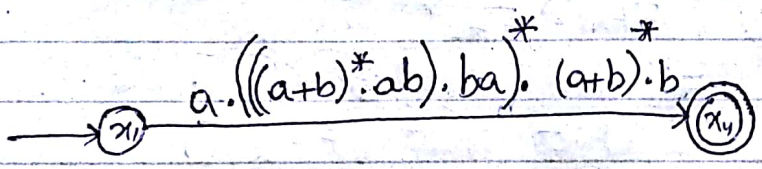
TG to RE







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



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

Combination of FAs:

① Union → { $FA_1 + FA_2$, $FA_2 + FA_1$, $FA_1 \cup FA_2$, $FA_2 \cup FA_1$ }

② Intersection → { $FA_1 \cap FA_2$, $FA_2 \cap FA_1$ }

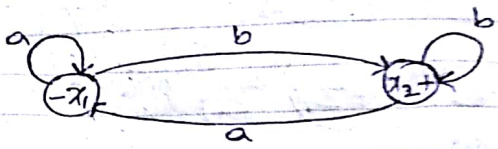
③ Symmetrical Difference → { $FA_1 \oplus FA_2$, $FA_2 \oplus FA_1$ } ⊕ (XOR)

④ Difference → { $\{FA_1 - FA_2\}$, $\{FA_2 - FA_1\}$ }

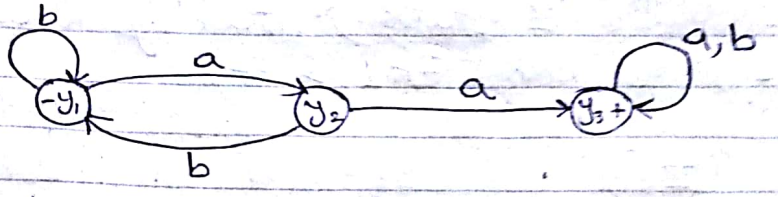
⑤ Concatenation → { $FA_1 \cdot FA_2$ }

FA, ∪ FA₂

FA₁ (DFA):



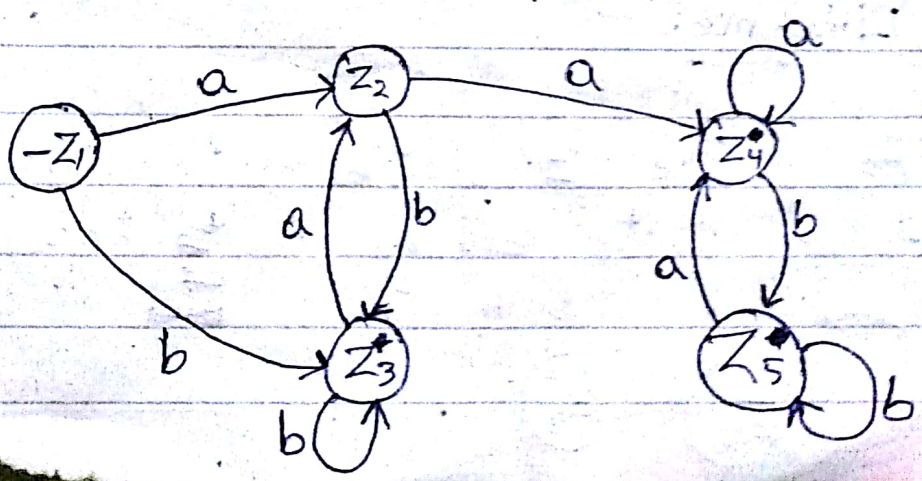
FA₂ (DFA):



(*) (XOR)

Transition Table:

New State	a	b
$-Z_1 \cong (-x_1, -y_1)$	$Z_2 \cong (x_1, y_2)$	$Z_3 \cong (x_2, y_1)$
$Z_2 \cong (x_1, y_2)$	$Z_4 \cong (x_1, y_3)$	$Z_3 \cong (x_2, y_1)$
$Z_3 \cong (x_2, y_1)$	$Z_2 \cong (x_1, y_2)$	$Z_3 \cong (x_2, y_1)$
$Z_4 \cong (x_1, y_3)$	$Z_4 \cong (x_1, y_3)$	$Z_5 \cong (x_2, y_3)$
$Z_5 \cong (x_2, y_3)$	$Z_4 \cong (x_1, y_3)$	$Z_5 \cong (x_2, y_3)$



Union:

The Z states containing final states of either FA1 or FA2 or both.

FA1 ∪ FA2 } FA1 + FA2 } $+Z_3$ $+Z_4$ $+Z_5$
FA2 ∪ FA1 } FA2 + FA1 }

Intersection:

FA1 ∩ FA2 } The Z states containing final states
FA2 ∩ FA1 } of both FA1 & FA2

$+Z_5$

Difference:

FA1 − FA2 → the Z states containing final state of FA1 but not FA2

&

$+Z_3$

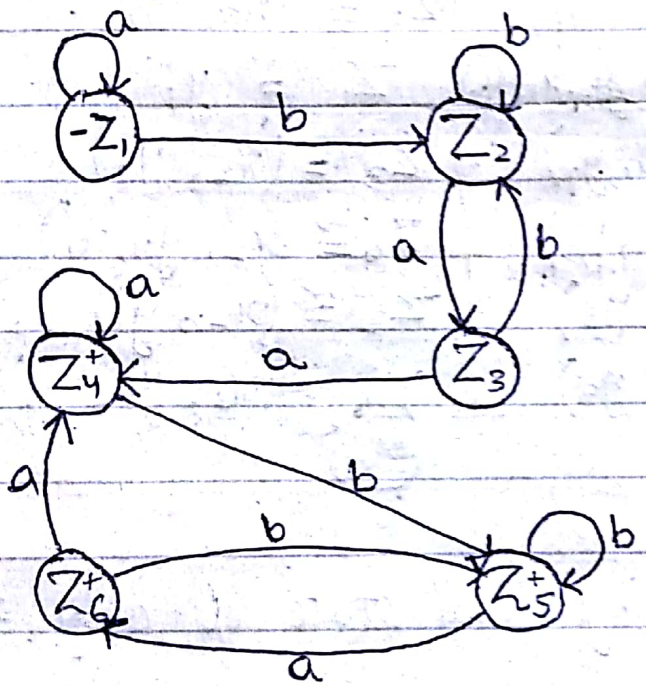
FA2 − FA1 → the Z states containing final state of FA2 but not FA1

Concatenation:

The moment a final state of first FA is entered, the possibility of the initial state of the second FA will be included as well.

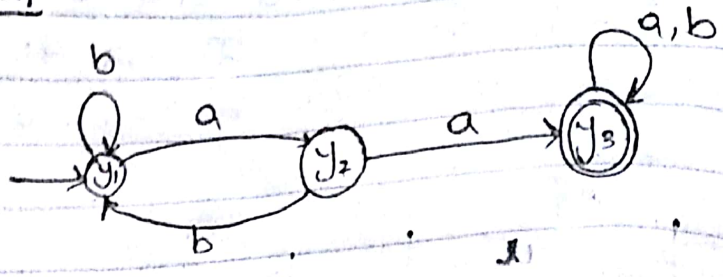
$FA_1 \cdot FA_2$

states	a	b
$-Z_1 \cong x_1$	$x_1 \cong Z_1$	$(x_2, y_1) \cong Z_2$
$Z_2 \cong (x_2, y_1)$	$(x_1, y_2) \cong Z_3$	$(x_2, y_1) \cong Z_2$
$Z_3 \cong (x_1, y_2)$	$(x_1, y_3) \cong Z_4$	$(x_2, y_1) \cong Z_2$
$Z_4 \cong (x_1, y_3)$	$(x_1, y_3) \cong Z_4$	$(x_2, y_1, y_3) \cong Z_5$
$Z_5 \cong (x_2, y_1, y_3)$	$(x_1, y_2, y_3) \cong Z_6$	$(x_2, y_1, y_3) \cong Z_5$
$Z_6 \cong (x_1, y_2, y_3)$	$(x_1, y_3) \cong Z_4$	$(x_2, y_1, y_3) \cong Z_5$

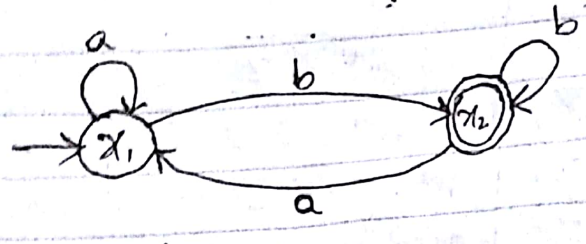


Solve $FA_2 \cdot FA_1$

$FA_2 \Rightarrow$



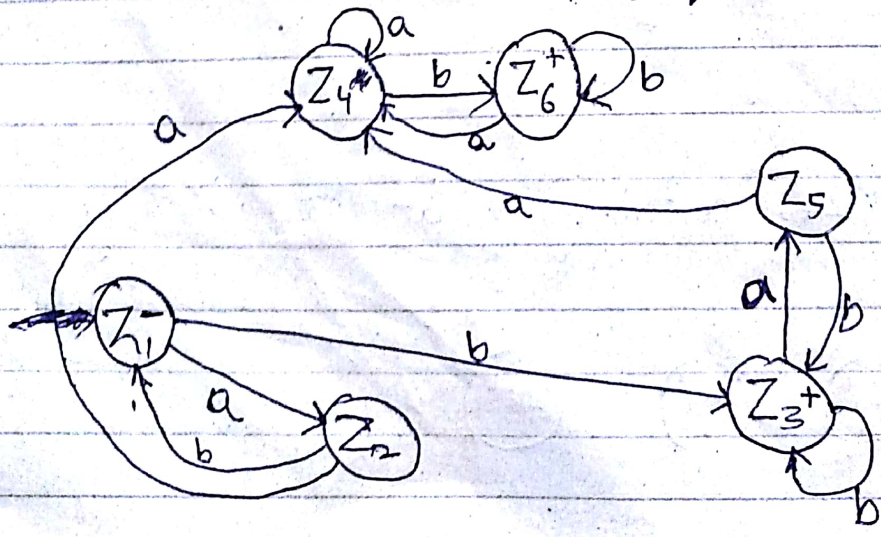
$FA_1 \Rightarrow$



Sol

States	a	b
$\rightarrow Z_1 \cong y_1$	$Z_2 \cong y_2$	$Z_3 \cong (x_2, y_1)$
$Z_2 \cong y_2$	$Z_4 \cong (x_1, y_3)$	$Z_1 \cong y_1$
$+ Z_3 \cong (x_2, y_1)$	$Z_5 \cong (x_1, y_2)$	$Z_3 \cong (x_2, y_1)$
$Z_4 \cong (x_1, y_3)$	$Z_4 \cong (x_1, y_3)$	$Z_6 \cong (x_2, y_3, x_1)$
$Z_5 \cong (x_1, y_2)$	$Z_4 \cong (x_1, y_3)$	$Z_3 \cong (x_2, y_1)$
$+ Z_6 \cong (x_1, x_2, y_3)$	$Z_4 \cong (x_1, y_3)$	$Z_6 \cong (x_2, y_3, x_1)$

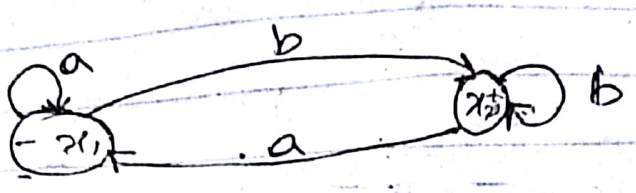
The Z states containing final state of only FA_2.



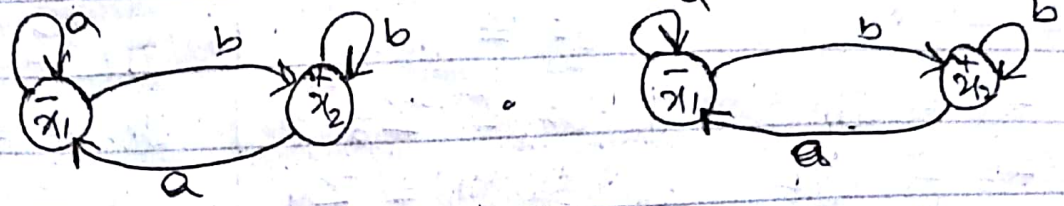
Clousure of an FA

$(FA_1)^* = FA_1 \cdot FA_1$

$FA_1 \Rightarrow$



$(FA_1)^* \Rightarrow$

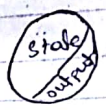


States	a	b
	this x_1 is Non-Final	
$\pm$ $Z_1 \cong x_1$	$x_1 \cong Z_2$	$(x_2, x_1) \cong Z_3$
$Z_2 \cong x_1$	$x_1 \cong Z_2$	$(x_2, x_1) \cong Z_3$
$+$ $Z_3 \cong (x_2, x_1)$	$x_1 \cong Z_2$	$(x_2, x_1) \cong Z_3$

FSM with Output (Finit State machine):

① Edward. F. Moore machine
② George. H. Mealy machine

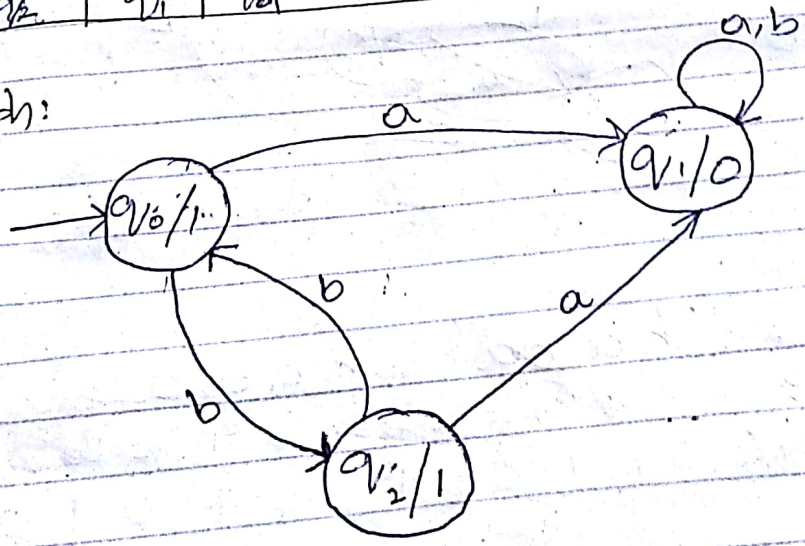
Moore Machine:



T.Table

States	a	b	Output
→ q_0	q_1	q_2	1
q_1	q_1	q_1	0
q_2	q_1	q_0	1

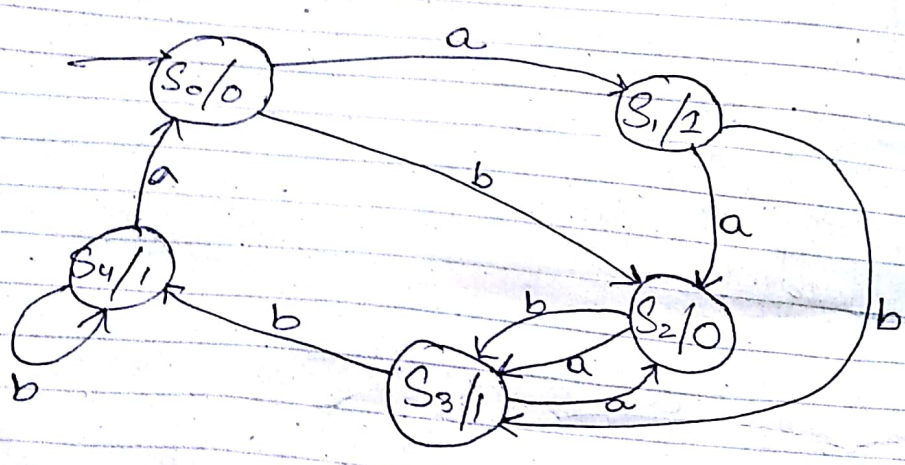
Graph:



Input String : b b a b a a a b b a b a b b
States : q_0 q_2 q_0 q_1 q_1 q_1 q_1 q_1 q_1 q_1 q_1 q_1 q_1 q_1
Output : 1 1 1 0 0 0 0 0 0 0 0 0 0 0

Example 2 :

States	a	b	output
S_0	S_1	S_2	0
S_1	S_2	S_3	1
S_2	S_3	S_3	0
S_3	S_2	S_4	1
S_4	S_0	S_4	1



Conversion of Moore to Mealy machine

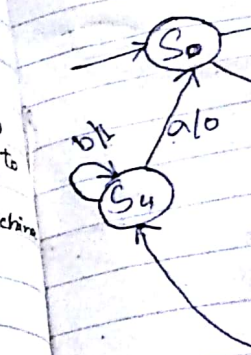
Input String : b a b a b a b a a a b a

States : S_0 S_2 S_3 S_4 S_0 S_2 S_3 S_4 S_0 S_1 S_2 S_3 S_2

Output : 0 0 1 1 0 0 1 1 0 1 0 1 0

Mealy Mach

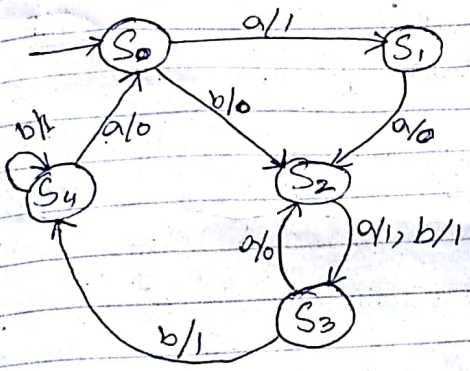
States	I
→ S_0	S_1
S_1	S_2
S_2	S_3
S_3	S_4
S_4	



Mealy Machine:

States	Input		Output	
	a	b	a	b
→ S_0	S_1	S_2	1	0
S_1	S_2	S_3	0	1
S_2	S_3	S_3	1	1
S_3	S_2	S_4	0	1
S_4	S_0	S_4	0	1

Conversion of moore to Mealy machine



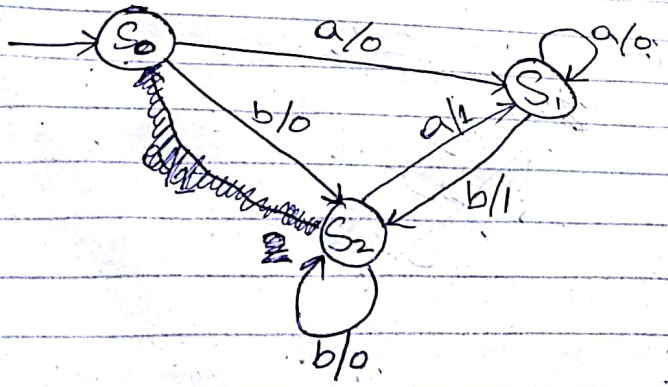
Mealy to Moore Conversion:

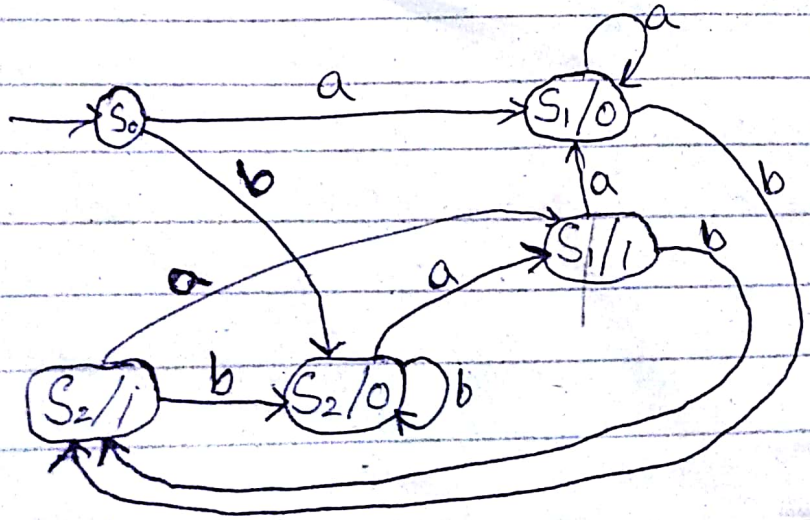
There are two cases, to convert mealy to moore.

(i) If incoming transitions have same output then put the output into the state.

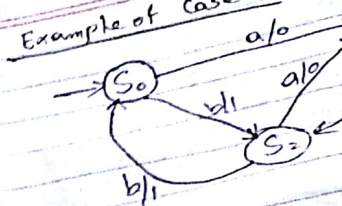
(ii) If incoming transitions have opposite output then break the state with two seperate outputs.

State	Input		Output	
	a	b	a	b
S_0	S_1	S_2	0	0
S_1	S_1	S_2	0	1
S_2	S_1	S_2	1	0

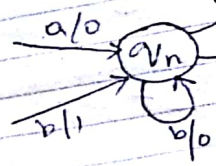




Example of Case 1:

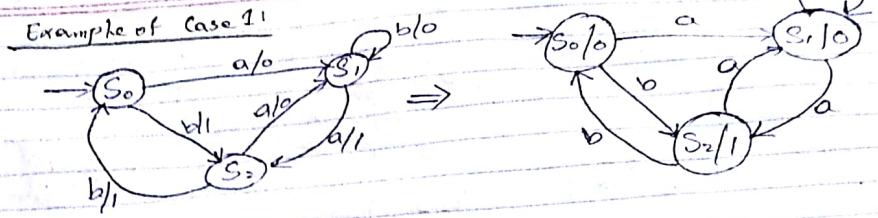


Case 2 Explan

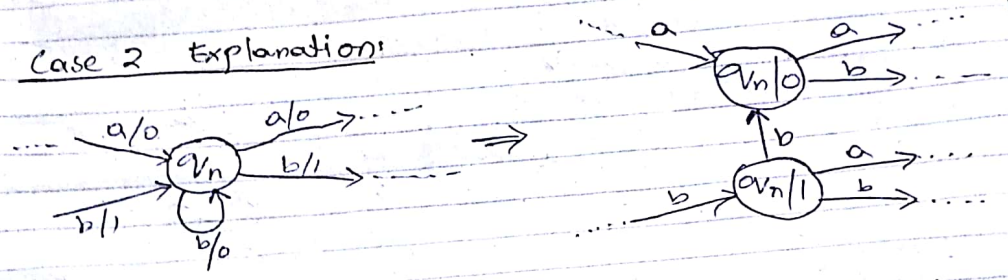


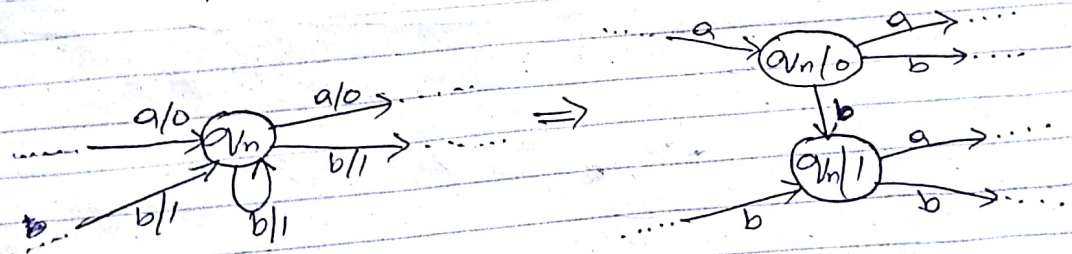
to moore
output then
put then
te
outputs:

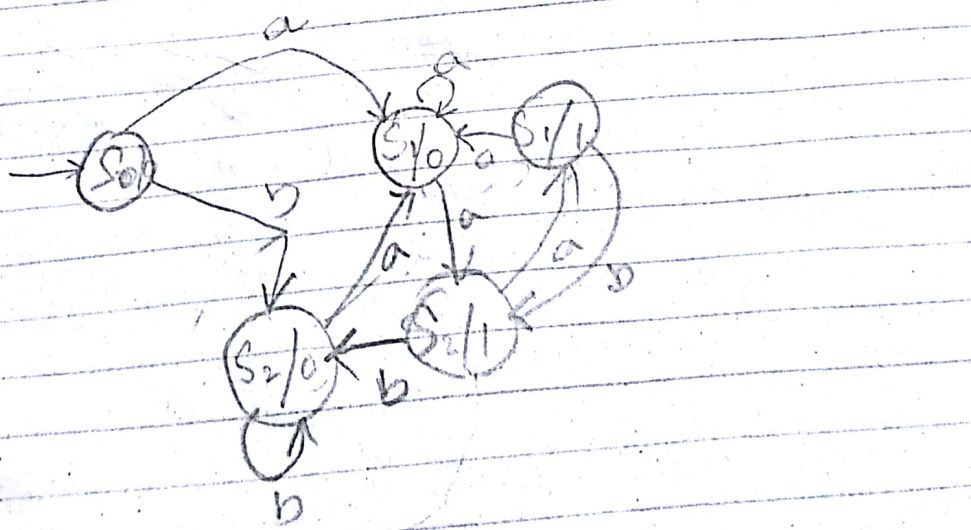
Example of Case 1:



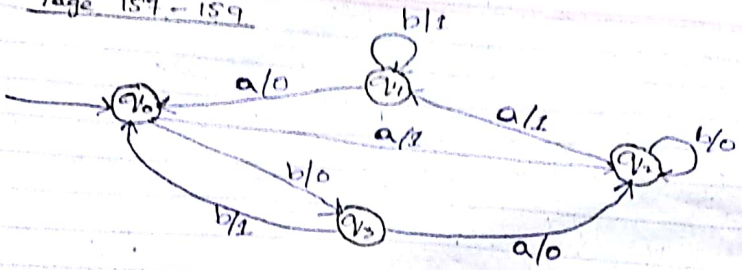
Case 2 Explanation:

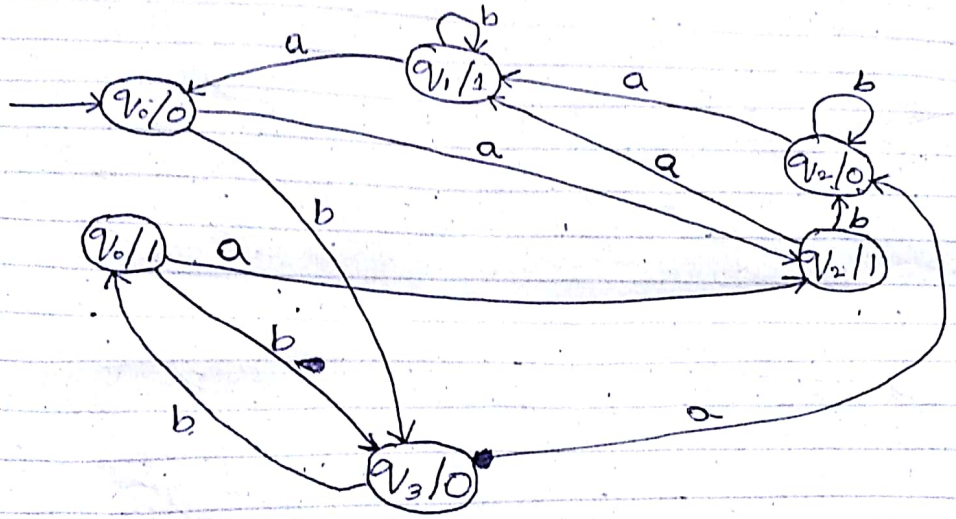






Page 157 - 159





Moore Machines

T.T to FA

i)

State	a	b	O
q_0	q_1	q_2	
q_1	q_1	q_1	
q_2	q_1	q_0	

ii)

States	a
q_0	q_0
q_1	q_1
q_2	q_2

iii)

states	a
q_0	q_0
q_1	$q_$
q_2	q

iv) State

q_0

q_1

q

q

Context Free Grammar: (CFG)

A CFG is a collection of three things:

1. a set of alphabets Σ i.e. letters, called Terminals (Terminal symbol) and cannot produce and usually written in small letters; from which we are going to make words of the string.

2. a set of symbols called Non-Terminals, one of which is a starting symbol, denoted by S. Usually written in capital letters / uppercase → <S> or S. Usually the non-terminal of the CFG, and it can produce.

3. a finite set of productions.

Non-Terminal $\xrightarrow[\text{produces}]{\text{gives}}$ finite set of terminals and/or non-terminals

$N \to T$
$N \to TT$
$N \to TT \ldots T$
$N \to N$
$N \to NN$
$N \to N, N \ldots N$
$N \to TN$
$N \to NT$
$N \to T N \ldots N$
$N \to N \ldots N T$

$N \to T \ldots T \quad N \ldots N$
$N \to T$ and/or N

i) RE = a

CFG = $S \longrightarrow a$

ii) R.E = $\wedge$

CFG = $S \longrightarrow \wedge$

iii) RE = b

CFG = $S \longrightarrow b$

iv) RE = a.b

CFG = $S \longrightarrow ab$

(OR)

CFG = $S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow b$

v) R.E = a+b

CFG = $S \longrightarrow a \mid b$

Ptact

Q1(a) Draw

i) $(a+b)^*$

ii) $(a.b)^+$

iii) (a

Q1(b)

i)

TG

① R.E = a^*

CFG $S \longrightarrow \underset{\text{Production 1/Rule 1}}{aS} \mid \underset{\text{Production 2/Rule 2}}{\wedge}$

this can be written as:

$S \longrightarrow aS$ Production/Rule 1

$S \longrightarrow \wedge$ Production/Rule 2

Q) Derive the string aaaaa from above CFG

$S \longrightarrow aS$ (Production 1)

$S \longrightarrow aaS$ "

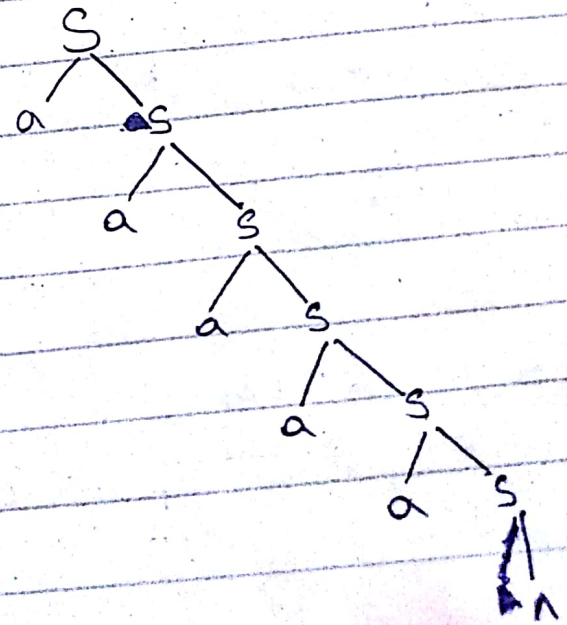
$S \longrightarrow aaaS$ "

$S \longrightarrow aaaaS$ "

$S \longrightarrow aaaaa\wedge$ (Production 2)

$S \longrightarrow aaaaa$

Its Derivation Tree | Parse Tree :



② R.E = ~~a^+~~

~~$S \longrightarrow aS \mid a$~~

Derive the

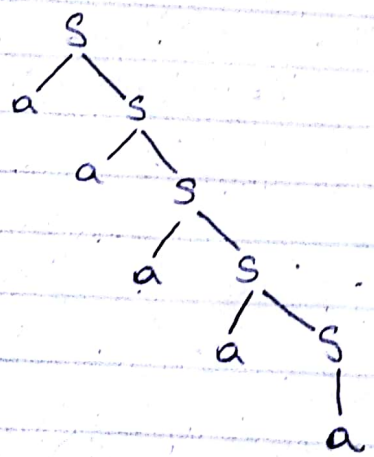
②RE = a^+

$S \rightarrow aS/a$

Derive the string aaaaa:

$S \rightarrow aS$ (Prod 1)
$S \rightarrow aaS$ "
$S \rightarrow aaaS$ "
$S \rightarrow aaaaS$ "
$S \rightarrow aaaaa$ (Product 2)

Derivation Tree:



③R.E = (a+b)

S —

Derive

③ R.E = $(a+b)^*$

$S \longrightarrow aS|bS|\wedge$ (or) $S \longrightarrow aS|bS|a|b|\wedge$

Derive the string baabbbab

$S \longrightarrow bS$ P2
$S \longrightarrow baS$ P1
$S \longrightarrow baaS$ P1
$S \longrightarrow baabS$ P2
$S \longrightarrow babbbS$ P2
$S \longrightarrow baabbbS$ P2
$S \longrightarrow baabbbaS$ P1
$S \longrightarrow baabbbabS$ P2
$S \longrightarrow baabbbab\wedge$ P3
$S \longrightarrow baabbbab$

④ $S \longrightarrow aS|a|b$

R.E = $(a+b)^+$

⑤ $S \longrightarrow XaaX$
$X \longrightarrow aX$
$X \longrightarrow bX$
$X \longrightarrow \wedge$

$X \longrightarrow aX|bX|\wedge$

R.E = $(a+b)^*.aa.(a+b)^*$

⑥ R·E = $a^* \cdot b^*$

$S \longrightarrow XY$
$X \longrightarrow aS \mid \wedge$
$Y \longrightarrow bS \mid \wedge$

⑦ R·E = $a^* + b^*$

$S \longrightarrow X \mid Y$
$X \longrightarrow aS \mid \wedge$
$Y \longrightarrow bS \mid \wedge$

Home Work

① R·E = $a^+ \cdot b^+$

② R·E = $(a \cdot b)^+$

③ R·E = $(a^* \cdot b^*)^*$

④ R·E = $(a^* + b^*)^*$

⑤ R·E = $a \cdot (a+b)^* \cdot b$

⑥ R·E = $a \cdot (a+b)^* \cdot b + b \cdot (a+b)^* \cdot b$

⑦ CFL = $a^n b^n$; [n >= 0]

⑧ Palindrome string

⑨ Even – Even

⑩ Odd – odd

Home Work

① $a^+ \cdot b^+$

$S \longrightarrow XY$
$X \longrightarrow aXS \mid a$
$Y \longrightarrow bSY \mid b$

② R·E = $(a \cdot b)^+$

$S \longrightarrow abS \mid ab$

③ R·E = $(a^* \cdot b^*)^*$

$S \longrightarrow XY \mid \wedge$
$X \longrightarrow aSX \mid \wedge$
$Y \longrightarrow bSY \mid \wedge$

⑤ R·E = $a \cdot (a+b)^* \cdot b$

$S \longrightarrow aXb$
$X \longrightarrow aSX \mid bSX \mid \wedge$

⑥ $a \cdot (a+b)^* \cdot b + b \cdot (a+b)^* \cdot b$

$S \longrightarrow a \cdot X \cdot b \mid b \cdot X \cdot b$

$X \longrightarrow aX \mid bX \mid \wedge$

④ R.E = $(a^* + b^*)^*$

$S \longrightarrow x \mid y \mid \wedge$

$X \longrightarrow aX \mid \wedge$

$Y \longrightarrow bY \mid \wedge$

⑧ Palindrome

$S \longrightarrow asa \mid bsb \mid a \mid b$ (odd) aababaa (odd)

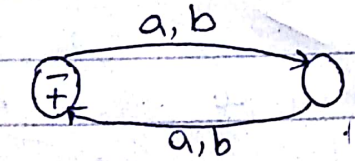
$S \longrightarrow aSa \mid bSb \mid \wedge$ (even) bbaabb (even)

$S \longrightarrow aSa \mid bSb \mid a \mid b \mid \wedge$ (general palindrome)

⑨ Even – Even

aa bb

aaaa bbbb



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

CFG : $S \longrightarrow XX \mid \wedge$

$X \longrightarrow aa \mid bb$

Context Free Language

① CFL : $a^n b^n$; $n \geq 0$

$S \rightarrow aSb \mid \Lambda$

same no. of a's and b's

② CFL : $a^n b^{2n}$; $n \geq 0$

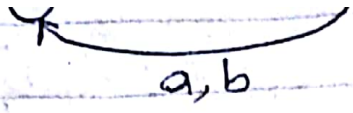
$S \rightarrow aSbb \mid \Lambda$

b occurs twice then b

③ $a^m b^n$; $m \geq 0$ and $n = m+1$

$S \rightarrow aSb \mid b$

in which b occurs one more than a



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

[redacted]

$S \rightarrow X \mid XY$

$X \rightarrow a \mid b$

[redacted]

$Y \rightarrow XXY \mid \Lambda$

Chomskey Normal Form (CNF):

If L is a language generated by some ↓ CFG, then there is another CFG that generates all the non-Λ words of L, all of whose productions are of one of two basic forms.

N.T $\longrightarrow$ one or more N.Ts

N.T $\longrightarrow$ only one Terminal.

① R.E = a.b

(CFG) $S \longrightarrow ab$ (CFG not in CNF)

— or —

$S \longrightarrow AB$
$A \longrightarrow a$
$B \longrightarrow b$ (CFG in CNF)

② R.E = a

$S \longrightarrow a$ (CFG in CNF)

③ R.E = a+b

$S \longrightarrow a \mid b$ (CFG in CNF)

(4) $S \rightarrow XY$
$X \rightarrow XX \mid a$
$Y \rightarrow YY \mid b$

$S \rightarrow XY$
$X \rightarrow XX$
$X \rightarrow a$
$Y \rightarrow YY$
$Y \rightarrow b$

(5) $S \rightarrow XY$
$X \rightarrow aX \mid \Lambda$
$Y \rightarrow bY \mid \Lambda$

CNF

$S \rightarrow XY$
$X \rightarrow AaX \mid \Lambda$
$A \rightarrow a$
$Y \rightarrow BY \mid \Lambda$
$B \rightarrow b$

⑥ $S \longrightarrow X_1 \mid X_2 a X_2 \mid aSb \mid b$

$X_1 \rightarrow X_2 X_2 \mid b$

$X_2 \rightarrow aX_2 \mid aaX_1$

$S \longrightarrow X_1 \mid X_2 A X_2 \mid AS \mid B \mid b$

$X_1 \longrightarrow X_2 X_2 \mid b$

$X_2 \longrightarrow AX_2 \mid AAX_1$

~~$A \longrightarrow aA \mid a$~~ (F = flag)

$B \longrightarrow b$

$A \longrightarrow a$

⑦ R.E = anything = $(a+b)^*$

CFG ⇒ $S \longrightarrow aS \mid bS \mid \Lambda$

CNF ⇒ $S \longrightarrow AS \mid BS \mid \Lambda$

$A \longrightarrow a$

⑨ S —
A —
B

S
S
A
A
A
B

③ S → bA | aB
A → bAA | aS | a
B → aBB | bS | b

S → YA
S → XB
A → YAA
A → XS
A → a
B → XBB
B → YS
B → b
X → a
Y → b

Q.) Write Down CFG for any five

i) $1^{2n} \cdot 0^n$; where $n >= 0$

$S \rightarrow 11\,S\,0 \mid \wedge$

——— OR ———

$S \rightarrow 11\,S\,0$

$S \rightarrow \wedge$

ii) $(0+1)^n + (11+00)^n$; where $n >= 0$

$S \rightarrow XS \mid YS \mid \wedge$

$X \rightarrow 0 \mid 1$

$Y \rightarrow 11 \mid 00$

iii) Palindrome Strings

$S \rightarrow asa$

$S \rightarrow bSb$

$S \rightarrow a$

$S \rightarrow b$

$S \rightarrow \wedge$

iv) language over $\Sigma = \{a, b\}$ having strings in which a is exactly divisible by 2.

$S \rightarrow XSb \mid \wedge$

$X \rightarrow aa$

v) language over $\Sigma = \{a, b\}$ having strings which accept strings having same no. a's and b's

$S \rightarrow asb \mid \lambda$

Q.) Write (any two of) given CFGs in CNF

i) CFG

$S \rightarrow bA \mid aB$
$A \rightarrow bAA \mid Ba \mid a$
$B \rightarrow aBB \mid Ab \mid b$

CNF

$S \rightarrow XA \mid WB$
$A \rightarrow XAA \mid BW \mid a$
$B \rightarrow WBB \mid AX \mid b$
$X \rightarrow b$
$W \rightarrow a$

ii) CFG

$S \rightarrow AACD$
$A \rightarrow aAb$
$C \rightarrow aC \mid a$
$D \rightarrow aDa \mid bDb$

CNF

$S \rightarrow AACD$
$A \rightarrow XAY$
$C \rightarrow XC \mid a$
$D \rightarrow XDX \mid YDY$
$X \rightarrow a$
$Y \rightarrow b$