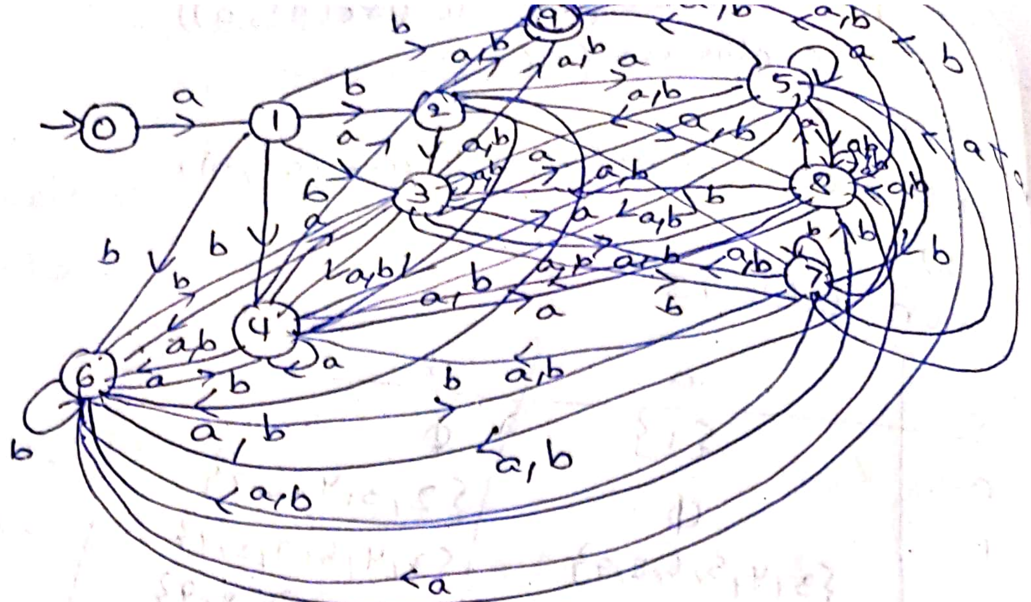


30/7/22

* Minimization of DFA :-

- Also called reduction of DFA.

* Minimize the following DFA. The DFA transition table is given below.

δ	0	1		
→ q_0	q_1	q_3	I I	II III
q_1	q_2	q_4	I II	II III
q_2	q_1	q_4	I II	III III
q_3	q_2	q_4	I II	
* q_4	q_4	q_4	~~I~~ ~~II~~	

Ans The given DFA $M = (Q, \Sigma, \delta, q_0, F)$

where $Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$q_0 = \{q_0\}$; $F = \{q_4\}$

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

~~Exam~~ Minimise the given DFA as

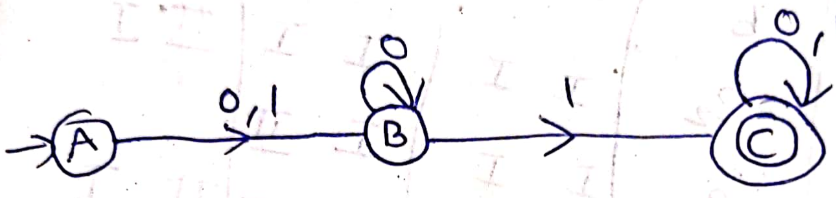
$$\Pi_0 = (q_0, q_1, q_2, q_3, q_4)$$

$$\Pi_1 = \underset{\text{I}}{(q_0, q_1, q_2, q_3)}\underset{\text{II}}{(q_4)}$$

$$\Pi_2 = \underset{\text{I}}{(q_0)}\underset{\text{II}}{(q_1, q_2, q_3)}\underset{\text{III}}{(q_4)}$$

$$\Pi_3 = \underset{\substack{\text{I}\\ A}}{(q_0)}\underset{\substack{\text{II}\\ B}}{(q_1, q_2, q_3)}\underset{\substack{\text{III}\\ C}}{(q_4)}$$

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



The minimized DFA

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

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

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

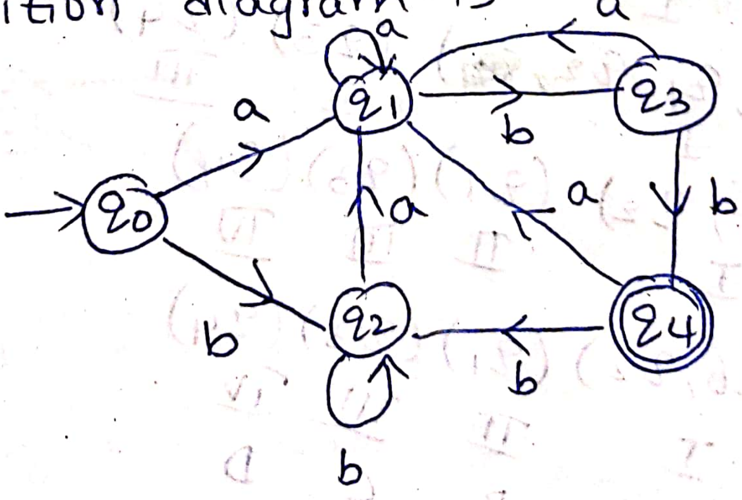
$q_0 = \{A\}$

$F = \{C\}$

$\delta = Q \times \Sigma \to Q$

2/8/22

* Minimize the following DFA. The DFA transition diagram is



The given DFA $M = (Q, \Sigma, \delta, q_0, F)$.

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

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

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

$q_0 = \{q_0\}$; $F = \{q_4\}$

Transition table is

δ	a	b						
$\rightarrow q_0$	q_1	q_2	I	I	I	I	II	I
q_1	q_1	q_3	I	I	I	III		
q_2	q_1	q_2	I	I	I	I	II	I
q_3	q_1	q_4	I	II				
$* q_4$	q_1	q_2						

The Minimization of DFA is

$\Pi_0 = (q_0, q_1, q_2, q_3, q_4)$

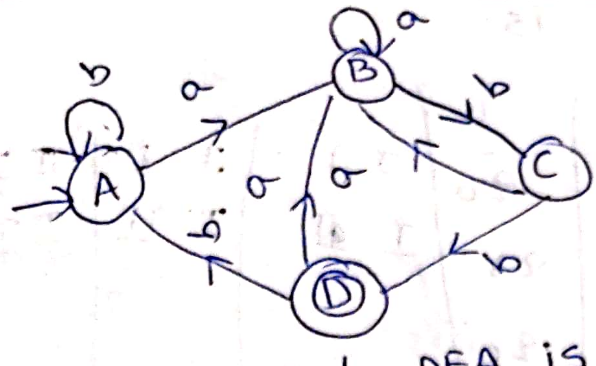
$\Pi_1 = \underbrace{(q_0, q_1, q_2, q_3)}_{\mathrm{I}} \underbrace{(q_4)}_{\mathrm{II}}$

$\Pi_2 = \underbrace{(q_0, q_1, q_2)}_{\mathrm{I}} \underbrace{(q_3)}_{\mathrm{II}} \underbrace{(q_4)}_{\mathrm{III}}$

$\Pi_3 = \underbrace{(q_0, q_2)}_{\mathrm{I}} \underbrace{(q_1)}_{\mathrm{II}} \underbrace{(q_3)}_{\mathrm{III}} \underbrace{(q_4)}_{\mathrm{IV}}$

$\Pi_4 = \underbrace{(q_0, q_2)}_{\mathrm{I}} \underbrace{(q_1)}_{\mathrm{II}} \underbrace{(q_3)}_{\mathrm{III}} \underbrace{(q_4)}_{\mathrm{IV}}$

A B C D



The minimized DFA is

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

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

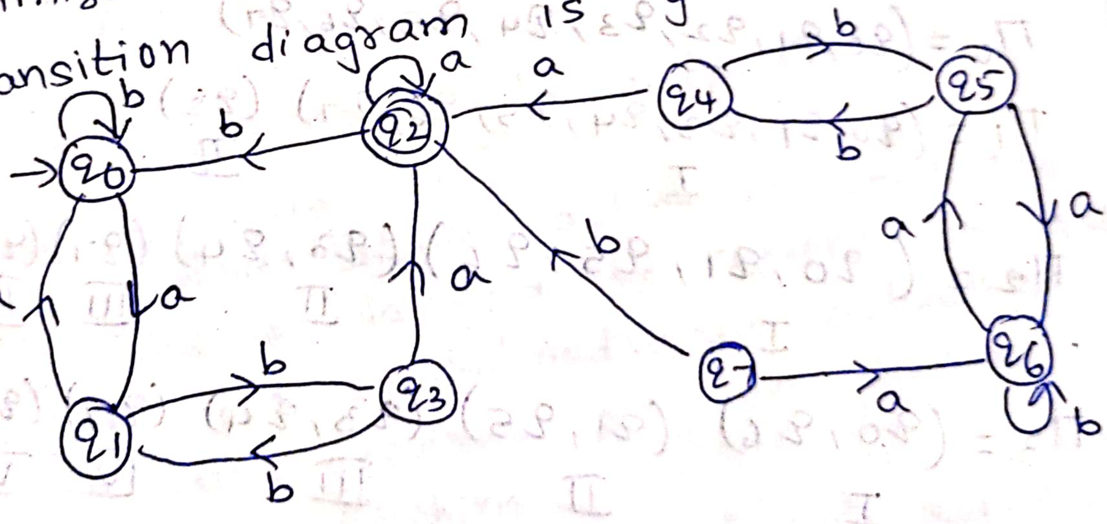
$\Sigma = \{a, b\}$

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

$q_0 = \{A\}$

$F = \{D\}$

*Minimize the following DFA. The DFA transition diagram is given below.



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

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

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

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

$q_0 = \{q_0\}$

$F = \{q_2\}$

Transition table is

δ	a	b			
→ q_0	q_1	q_0	I I	I I	II I
q_1	q_0	q_3	I II	I II	I III
* q_2	q_2	q_0			
q_3	q_2	q_1	II I	IV I	V II
q_4	q_2	q_5	II I	IV I	V II
q_5	q_6	q_4	I I	I II	I III
q_6	q_5	q_6	I I	I I	II I
q_7	q_6	q_2	I II		D

the Minimization of DFA A is

$\Pi_0 = (q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7)$

$\Pi_1 = \underbrace{(q_0, q_1, q_3, q_4, q_5, q_6, q_7)}_{\text{I}} \underbrace{(q_2)}_{\text{II}}$

$\Pi_2 = \underbrace{(q_0, q_1, q_5, q_6)}_{\text{I}} \underbrace{(q_3, q_4)}_{\text{II}} \underbrace{(q_7)}_{\text{III}} \underbrace{(q_2)}_{\text{IV}}$

$\Pi_3 = \underbrace{(q_0, q_6)}_{\text{I}} \underbrace{(q_1, q_5)}_{\text{II}} \underbrace{(q_3, q_4)}_{\text{III}} \underbrace{(q_7)}_{\text{IV}} \underbrace{(q_2)}_{\text{V}}$

$\Pi_4 = \underbrace{(q_0, q_6)}_{\text{I}} \underbrace{(q_1, q_5)}_{\text{II}} \underbrace{(q_3, q_4)}_{\text{III}} \underbrace{(q_7)}_{\text{IV}} \underbrace{(q_2)}_{\text{V}}$

A B C D E

the

3/8/22

*Finite

-Moore

-It is

next

and t

a give

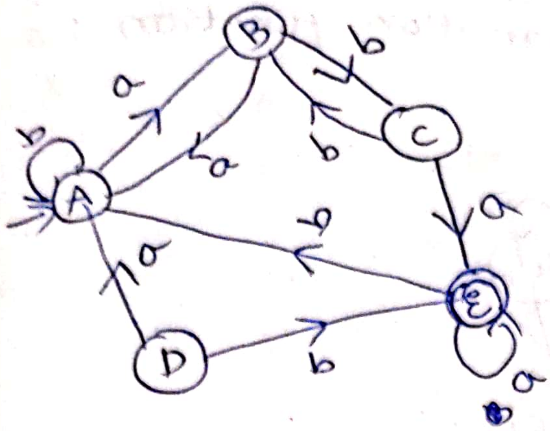
state

-The M

value

Mo

where



the minimized DFA is

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

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

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

$\delta = Q \times \Sigma \to Q$

$q_0 = \{A\}$

$F = \{E\}$

3/8/22

*Finite automata with output:-

- Moore machine:-
- It is a finite state machine in which the next state is decided by current state of ~~and~~ the input symbol and the output at a given time depends only on present state of the machine.
- The Moore machine is defined by 6 tuple values.

$M_0 = (Q, \Sigma, \delta, \Delta, \lambda, q_0)$

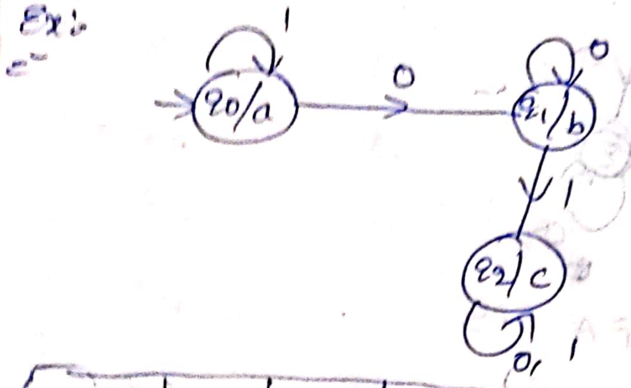
where Q is finite no. of states
Σ is input alphabet
Δ is output alphabets (or) symbols
q_0 is initial state
δ is input transition function

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

λ is output transition function

$\lambda : Q \rightarrow \Delta$

Ex:



δ	0	1	λ
$\rightarrow q_0$	q_1	q_0	a
q_1	q_1	q_2	b
q_2	q_2	q_2	c

*Find out the output for the given string 0100

Ans $q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_2 \xrightarrow{0} q_2$

a b c c c

- In Moore machine, if input string length is n then output length is n+1

- Melay machine :-

- It is a finite state machine in which the output symbols depends upon the input symbols of current state.

- Melay machine is defined by

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

where Q is finite no. of states

Σ is input alphabet

δ is input transition function

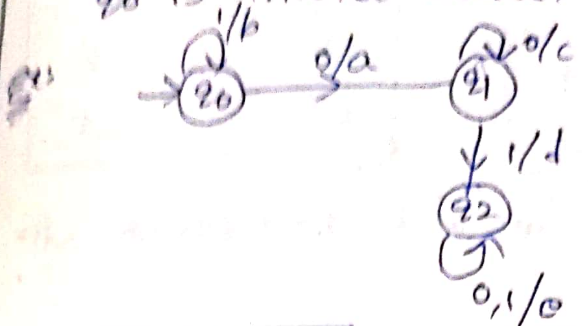
$\delta : Q \times \Sigma \rightarrow Q$

Δ is the output alphabet

λ is output transition function

$\lambda : Q \rightarrow \Delta$

q_0 is initial state.



δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

λ	0	1
q_0	a	b
q_1	c	d
q_2	e	e

*Findout the output for the string 0100

$$q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_2 \xrightarrow{0} q_2$$

a d e e

Length of input string = length of output

*Moore machine to Melay machine conversion:-

$$\lambda'(q,a) = \lambda(\delta(q,a))$$

*Convert the moore machine to melay machine.

The M_0 transition table is given below

δ	0	1	λ
$\rightarrow q_0$	q_1	q_2	1
q_1	q_3	q_2	1
q_2	q_2	q_1	0
q_3	q_0	q_3	1

Ans Moore Machine is given as

$M_0 = (Q, \Sigma, \delta, \Delta, q_0, \lambda)$

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

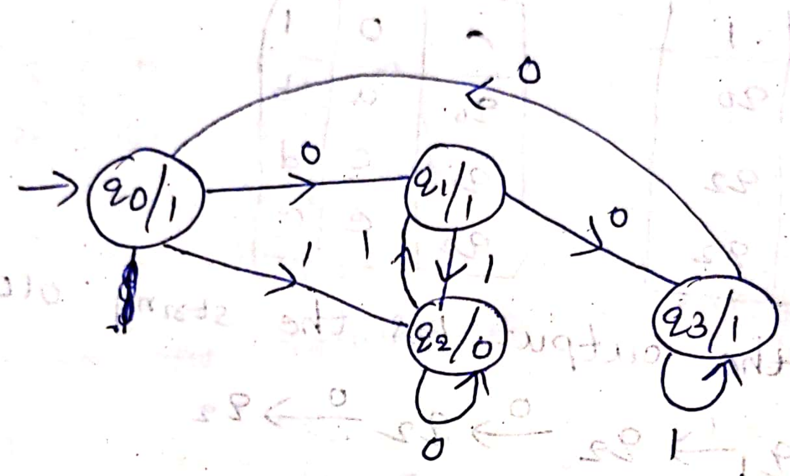
$\Sigma = \{0, 1\}$

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

$q_0 = \{q_0\}$

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

$\lambda : Q \to \Delta$



Converting from Moore machine to Melay machine

$M_e = (\Delta, Q, \Sigma, \delta, \lambda', q_0)$

$\lambda'(q, a) = \lambda(\delta(q, a))$

$\lambda'(q_0, 0) = \lambda(\delta(q_0, 0)) = \lambda(q_1) = 1$

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1)) = \lambda(q_2) = 0$

$\lambda'(q_1, 0) = \lambda(\delta(q_1, 0)) = \lambda(q_3) = 1$

$\lambda'(q_1, 1) = \lambda(\delta(q_1, 1)) = \lambda(q_2) = 0$

$\lambda'(q_2, 0) = \lambda(\delta(q_2, 0)) = \lambda(q_2) = 0$

$\lambda'(q_2, 1) = \lambda(\delta(q_2, 1)) = \lambda(q_1) = 1$

$\lambda'(q_3, 0) = \lambda(\delta(q_3, 0)) = \lambda(q_0) = 1$

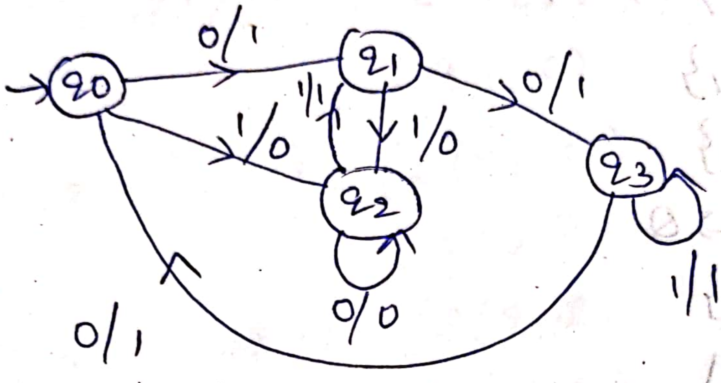
$\lambda'(q_3, 1) = \lambda(\delta(q_3, 1)) = \lambda(q_3) = 1$

The transition table of Melay machine is

δ	0	1
→ q_0	q_1	q_2
q_1	q_3	q_2
q_2	q_2	q_1
q_3	q_0	q_3

λ	0	1
q_0	1	0
q_1	1	0
q_2	0	1
q_3	1	1

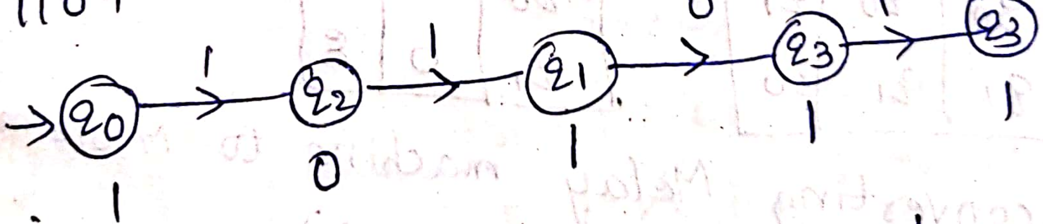
The transition diagram is



~~Me to Mo conversion :-~~

*Find out the output for string 1101 for both Mo & Me for above problem.

1101



Mo :- 1

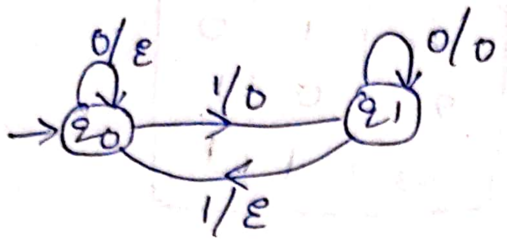
Me :- 0

Moore machine :- 1 0 1 1 1

Melay machine :- 0 1 1 1

Me to Mo conversion: $\delta'((q,b),a)) = \delta(q,a), \lambda(q,a)$

* Convert given Melay machine to Moore machine. Me transition diagram is



Given Melay Machine is

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

$Q = \{q_0, q_1\}$

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

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

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

$\lambda : Q \rightarrow \Delta$

$q_0 = \{q_0\}$

The transition table is

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_1	q_0

λ	0	1
$\rightarrow q_0$	ε	0
q_1	0	ε

Now, converting Melay machine to Moore machine

$M_o : (\underbrace{Q \times \Delta}_{Q'}, \Sigma, \Delta, \delta', \lambda', \underbrace{(q_0, \varepsilon)}_{q_0})$

$Q' = Q \times \Delta$

$= (q_0, q_1) \times (0, \varepsilon)$

$Q' = \{(q_0, 0), (q_0, \varepsilon), (q_1, 0), (q_1, \varepsilon)\}$

$$\delta'((q_0,0),0)) = (\delta(q_0,0), \lambda(q_0,0))$$
$$= (q_0, \varepsilon)$$

$$\delta'((q_0,0),1)) = (\delta(q_0,1), \lambda(q_0,1))$$
$$= (q_1, 0)$$

$$\delta'((q_0,\varepsilon),0)) = (\delta(q_0,0), \lambda(q_0,0))$$
$$= (q_0, \varepsilon)$$

$$\delta'((q_0,\varepsilon),1)) = (\delta(q_0,1), \lambda(q_0,1))$$
$$= (q_1, 0)$$

$$\delta'((q_1,0),0)) = (\delta(q_1,0), \lambda(q_1,0))$$
$$= (q_1, 0)$$

$$\delta'((q_1,0),1)) = (\delta(q_1,1), \lambda(q_1,1))$$
$$= (q_0, \varepsilon)$$

$$\delta'((q_1,\varepsilon),0)) = (\delta(q_1,0), \lambda(q_1,0))$$
$$= (q_1, 0)$$

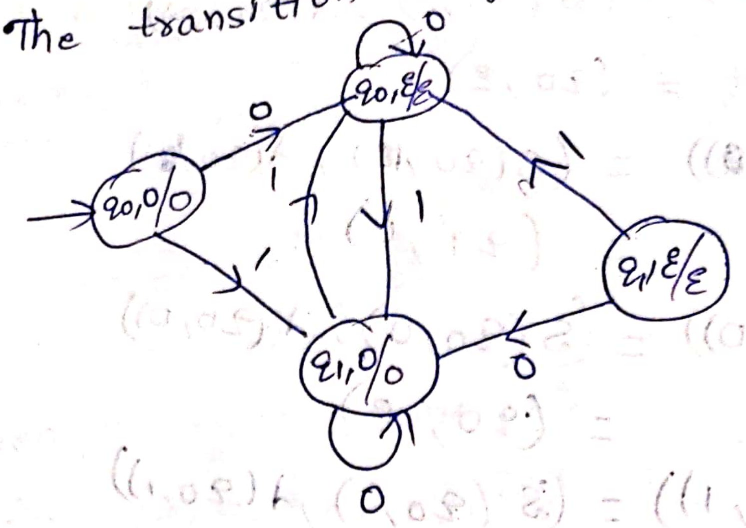
$$\delta'((q_1,\varepsilon),1) = (\delta(q_1,1), \lambda(q_1,1))$$
$$= (q_0, \varepsilon)$$

Moore Machine transition table is

$\lambda'(q_0,0) = 0$

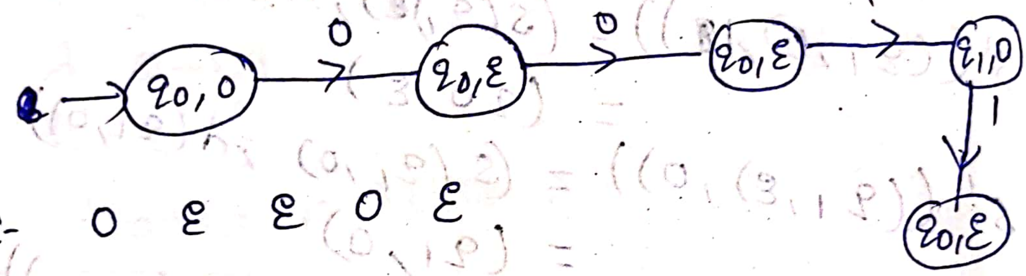
δ	0	1	λ
$\rightarrow (q_0,0)$	(q_0,ε)	$(q_1,0)$	0
(q_0,ε)	(q_0,ε)	$(q_1,0)$	ε
$(q_1,0)$	$(q_1,0)$	(q_0,ε)	0
(q_1,ε)	$(q_1,0)$	(q_1,ε)	ε

The transition diagram is



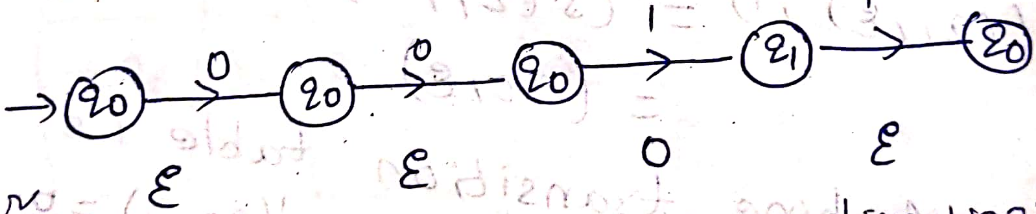
- check the string 0011 for Moore & Melay machines.

0 0 1 1



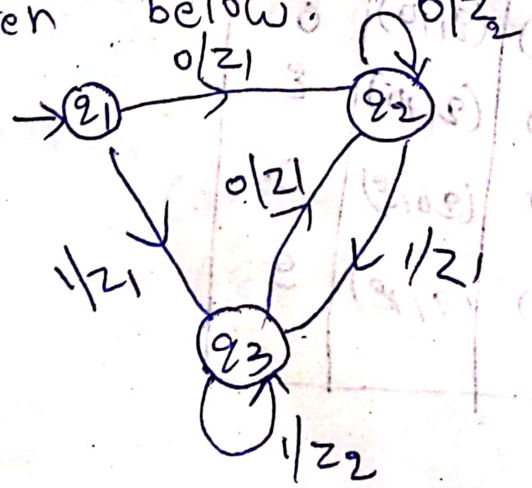
Mo :- 0 ε ε 0 ε

Me :- ε ε 0 ε



6/8/22

*Convert Me to Mo. The Me transition diagram is given below:



Given Melay machine is

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

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

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

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

$\lambda : Q \to \Delta$

$\Delta = \{z_1, z_2\}$

$q_0 = \{q_1\}$

The transition table is

δ	0	1
q_1	q_2	q_3
q_2	q_2	q_3
q_3	q_2	q_3

λ	0	1
q_1	z_1	z_1
q_2	z_2	z_1
q_3	z_1	z_2

Now converting Melay machine to Moore machine.

$M_o = (Q', \Sigma, \delta', \Delta, \lambda', q_0')$ $\quad q_0' = \{(q_1, z_1)\}$

$Q' = Q \times \Delta$

$= \{q_1, q_2, q_3\} \times \{z_1, z_2\}$

$Q' = \{(q_1, z_1), (q_1, z_2), (q_2, z_1), (q_2, z_2), (q_3, z_1), (q_3, z_2)\}$

$\delta'((q_1, z_1), 0) = (\delta(q_1, 0), \lambda(q_1, 0))$

$= (q_2, z_1)$

$\delta'((q_1, z_1), 1) = (\delta(q_1, 1), \lambda(q_1, 1))$

$= (q_3, z_1)$

$\delta'((q_2, z_1), 0) = (\delta(q_2, 0), \lambda(q_2, 0))$

$= (q_2, z_2)$

$\delta'((q_2,z_1),1) = (\delta(q_2,1), \lambda(q_2,1))$
$= (q_3,z_1)$

$\delta'((q_1,z_2),0) = (\delta(q_1,0), \lambda(q_1,0))$
$= (q_2,z_1)$

$\delta'((q_1,z_2),1) = (\delta(q_1,1), \lambda(q_1,1))$
$= (q_3,z_1)$

$\delta'((q_2,z_2),0) = (\delta(q_2,0), \lambda(q_2,0))$
$= (q_2,z_2)$

$\delta'((q_2,z_2),1) = (\delta(q_2,1), \lambda(q_2,1))$
$= (q_3,z_1)$

$\delta'((q_3,z_1),0) = (\delta(q_3,0), \lambda(q_3,0))$
$= (q_2,z_1)$

$\delta'((q_3,z_1),1) = (\delta(q_3,1), \lambda(q_3,1))$
$= (q_3,z_2)$

$\delta'((q_3,z_2),0) = (\delta(q_3,0), \lambda(q_3,0))$
$= (q_2,z_1)$

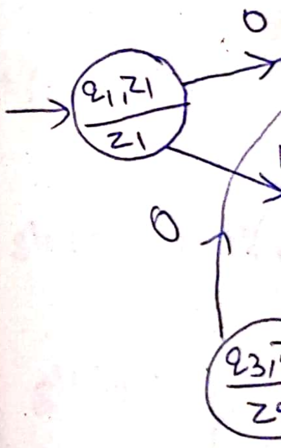
68 $\delta'((q_3,z_2),1) = (\delta(q_3,1), \lambda(q_3,1))$
* $= (q_3,z_2)$

$\lambda'((q_1,z_1)) = z_1$; $\lambda'((q_1,z_2)) = z_2$
$\lambda'((q_2,z_1)) = z_1$; $\lambda'((q_2,z_2)) = z_2$
$\lambda'((q_3,z_1)) = z_1$; $\lambda'((q_3,z_2)) = z_2$

Moore machine tr

δ	0
$\rightarrow (q_1,z_1)$	(q_2,z_1)
(q_1,z_2)	(q_2,z_1)
(q_2,z_1)	(q_2,z_2)
(q_2,z_2)	(q_2,z_2)
(q_3,z_1)	(q_2,z_1)
(q_3,z_2)	(q_2,z_1)

The Moore mo

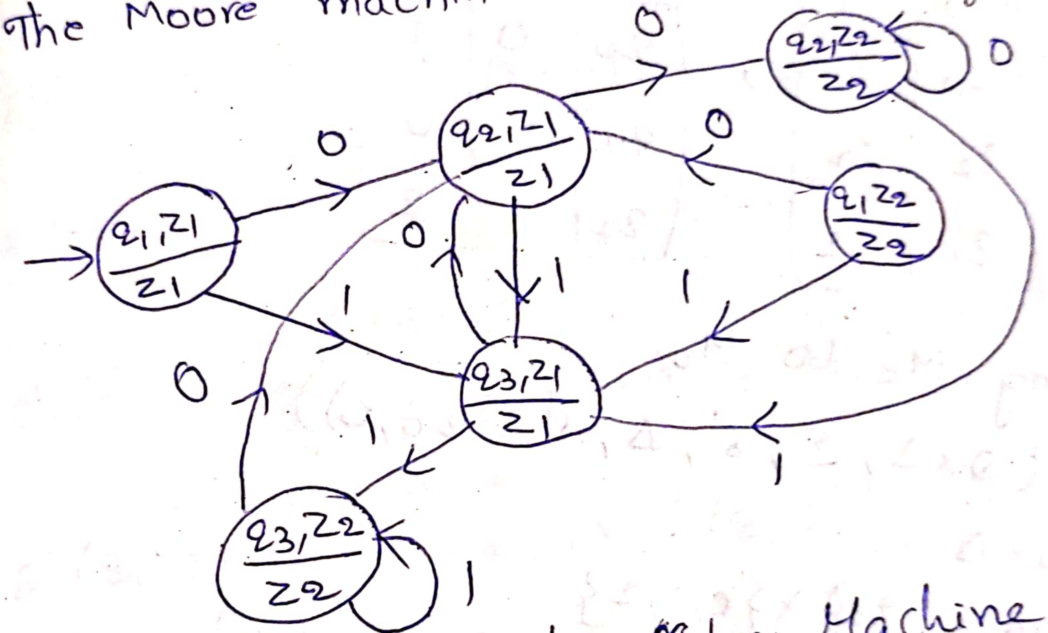


* Convert M
transitio

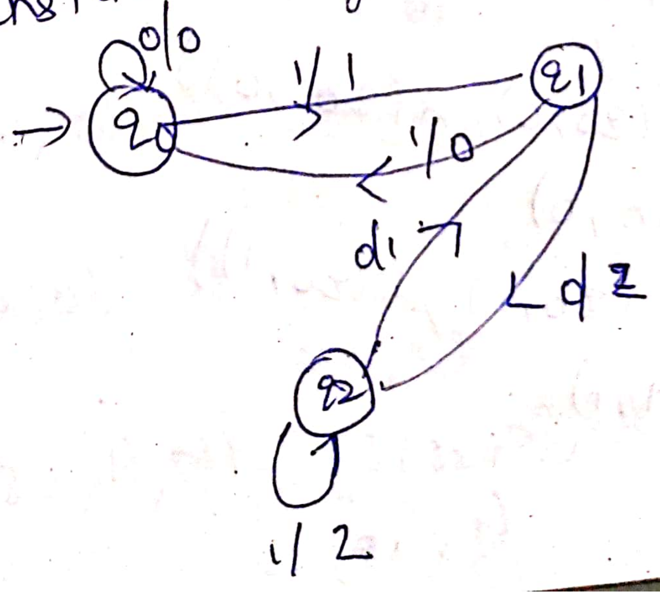
Moore machine transition table is

S	0	1	λ
$\rightarrow(q_1, z_1)$	(q_2, z_1)	(q_3, z_1)	z_1
(q_1, z_2)	(q_2, z_1)	(q_3, z_1)	z_2
(q_2, z_1)	(q_2, z_2)	(q_3, z_1)	z_1
(q_2, z_2)	(q_2, z_2)	(q_3, z_1)	z_2
(q_3, z_1)	(q_2, z_1)	(q_3, z_2)	z_1
(q_3, z_2)	(q_2, z_1)	(q_3, z_2)	z_2

The Moore machine transition diagram is



* Convert Me to Mo. The Mealy Machine transition diagram is given below.



The given Melay machine is

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

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

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

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

$\lambda : Q \rightarrow \Delta$

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

$q_0 = \{q_0\}$

The transition table is

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

λ	0	1
q_0	0	1
q_1	2	0
q_2	1	2

Converting Me to Mo.

$M_o = (Q \times \Delta, \Sigma, \delta', \Delta, \lambda', (q_0, 0))$

$Q' = Q \times \Delta$

$= \{q_0, q_1, q_2\} \times \{0, 1, 2\}$

$= \{(q_0,0), (q_0,1), (q_0,2), (q_1,0), (q_1,1), (q_1,2), (q_2,0), (q_2,1), (q_2,2)\}$

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

$= (q_0, 0)$

$\delta'((q_0,0),1) = (\delta(q_0,1), \lambda(q_0,1))$

$= (q_1, 1)$

$\delta'((q_0,1),0) = (\delta(q_0,0), \lambda(q_0,0))$
$= (q_0, 0)$
$\delta'((q_0,1),1) = (\delta(q_0,1), \lambda(q_0,1))$
$= (q_1, 1)$
$\delta'((q_0,2),0) = (\delta(q_0,0), \lambda(q_0,0))$
$= (q_0, 0)$
$\delta'((q_0,2),1)) = (\delta(q_0,1), \lambda(q_0,1))$
$= (q_1, 1)$
$\delta'((q_1,0),0)) = (\delta(q_1,0), \lambda(q_1,0))$
$= (q_2, 2)$
$\delta'((q_1,0),1)) = (\delta(q_1,1), \lambda(q_1,1))$
$= (q_0, 0)$
$\delta'((q_1,1),0)) = (\delta(q_1,0), \lambda(q_1,0))$
$= (q_2, 2)$
$\delta'((q_1,1),1)) = (\delta(q_1,1), \lambda(q_1,1))$
$= (q_0, 0)$
$\delta'((q_1,2),0)) = (\delta(q_1,0), \lambda(q_1,0))$
$= (q_2, 2)$
$\delta'((q_1,2),1)) = (\delta(q_1,1), \lambda(q_1,1))$
$= (q_0, 0)$
$\delta'((q_2,0),0)) = (\delta(q_2,0), \lambda(q_2,0))$
$= (q_1, 1)$
$\delta'((q_2,0),1)) = (\delta(q_2,1), \lambda(q_2,1))$
$= (q_2, 2)$
$\delta'((q_2,1),0)) = (\delta(q_2,0), \lambda(q_2,0))$
$= (q_1, 1)$

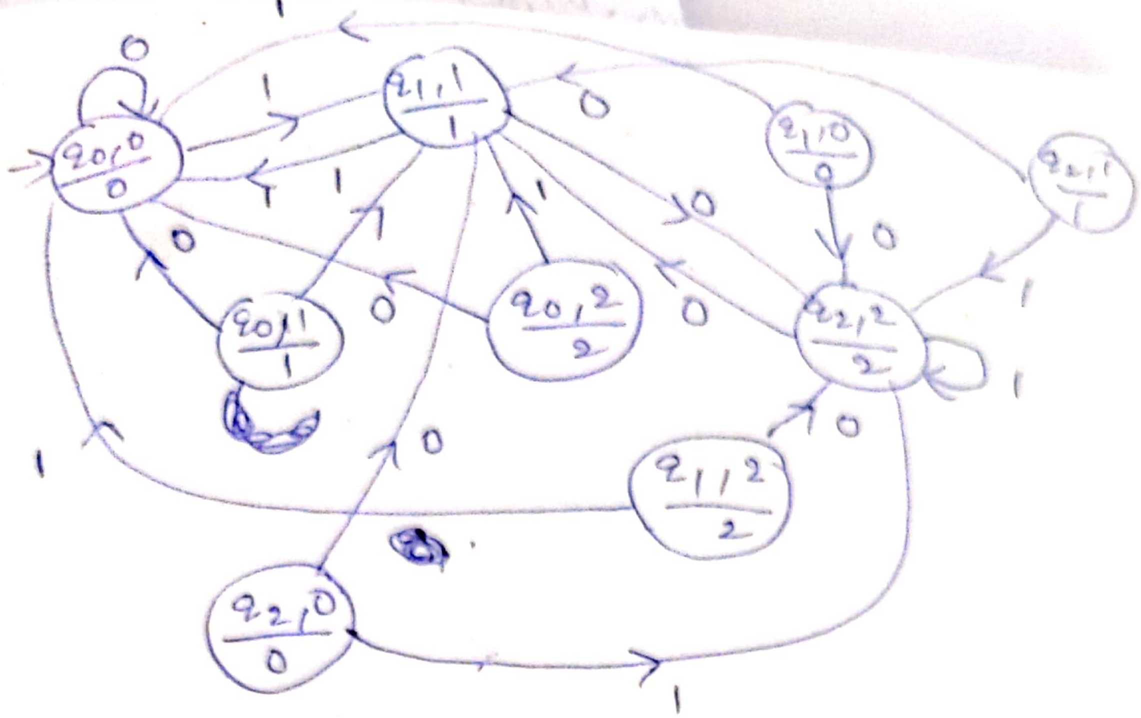
$\delta'((q_2,1),1) = (\delta(q_2,1), \lambda(q_2,1))$
$= (q_2,2)$
$\delta'((q_2,2),0)) = (\delta(q_2,0), \lambda(q_2,0))$
$= (q_1,1)$
$\delta'((q_2,2),1)) = (\delta(q_2,1), \lambda(q_2,1))$
$= (q_2,2)$

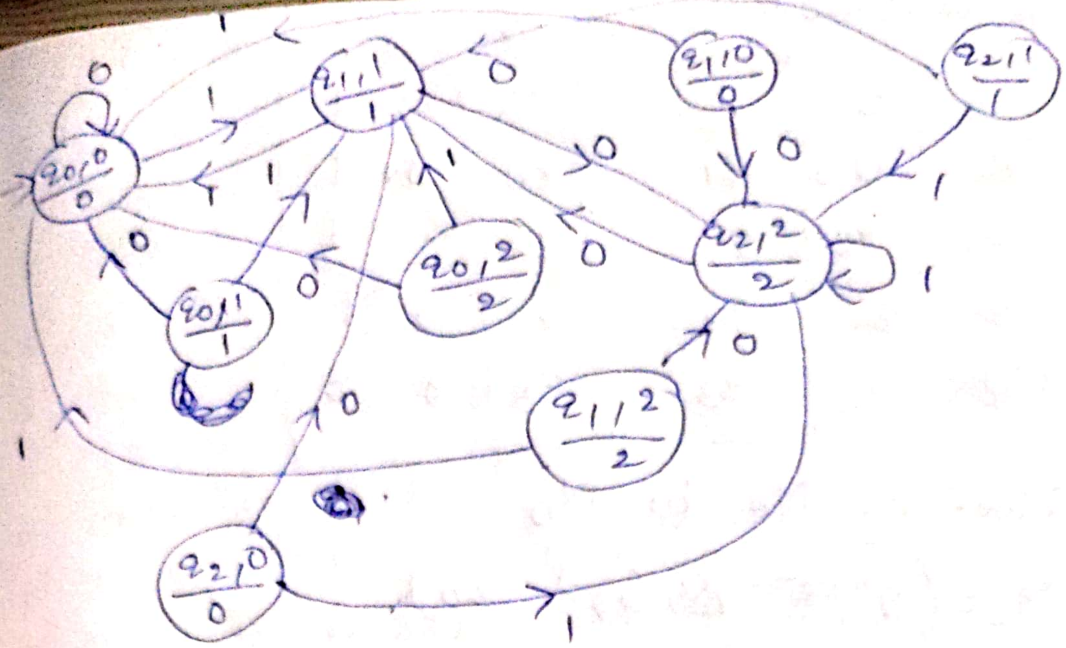
$\lambda'(q_0,0) = 0$; $\lambda'(q_0,1) = 1$; $\lambda'(q_0,2) = 2$
$\lambda'(q_1,0) = 0$; $\lambda'(q_1,1) = 1$; $\lambda'(q_1,2) = 2$
$\lambda'(q_2,0) = 0$; $\lambda'(q_2,1) = 1$; $\lambda'(q_2,2) = 2$

The Moore machine transition table is

δ	0	1	λ'
$\rightarrow (q_0,0)$	$(q_0,0)$	$(q_1,1)$	0
$(q_0,1)$	$(q_0,0)$	$(q_1,1)$	1
$(q_0,2)$	$(q_0,0)$	$(q_1,1)$	2
$(q_1,0)$	$(q_2,2)$	$(q_0,0)$	0
$(q_1,1)$	$(q_2,2)$	$(q_0,0)$	1
$(q_1,2)$	$(q_2,2)$	$(q_0,0)$	2
$(q_2,0)$	$(q_1,1)$	$(q_2,2)$	0
$(q_2,1)$	$(q_1,1)$	$(q_2,2)$	1
$(q_2,2)$	$(q_1,1)$	$(q_2,2)$	2

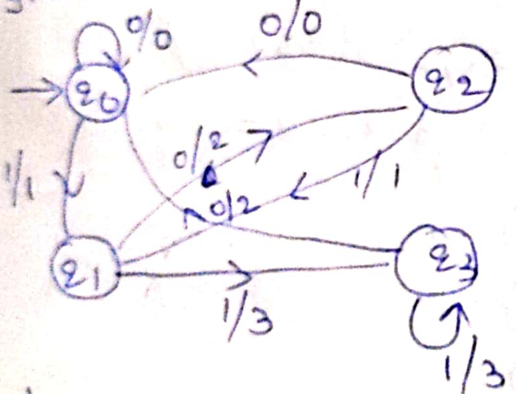
Scanned with CamScanner


q0,0 / 0
q1,1 / 1
q1,0 / 0
q2,1 / 1
q0,1 / 1
q0,2 / 2
q2,2 / 2
q1,2 / 2
q2,0 / 0



12/8/22

* Convert Melay machine to moore machine. The melay machine transition diagram is given below :



Ans: The given Melay machine is

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

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

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

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

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

$\lambda : Q \rightarrow \Delta$

$q_0 = \{q_0\}$

The transition table is

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_0	q_3

λ	0	1
q_0	0	1
q_1	2	3
q_2	0	1
q_3	2	3

Converting M_e to M_0

$M_0 = (Q', \Sigma, \Delta, \delta', \lambda', q_0)$

$Q' = Q \times \Delta$

$= \{q_0, q_1, q_2, q_3\} \times \{0, 1, 2, 3\}$

$= \{(q_0,0), (q_0,1), (q_0,2), (q_0,3), (q_1,0), (q_1,1), (q_1,2), (q_1,3), (q_2,0), (q_2,1), (q_2,2), (q_2,3), (q_3,0), (q_3,1), (q_3,2), (q_3,3)\}$

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

$\delta'((q_0,0),1) = (q_1,1)$

$\delta'((q_0,1),0)) = (q_0,0)$

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

$\delta'((q_0,2),0)) = (q_0,0)$

$\delta'((q_0,2),1)) = (q_1,1)$

$\delta'((q_0,3),0)) = (q_0,0)$

$\delta'((q_0,3),1)) = (q_1,1)$

$\delta'((q_1,0),0)) = (q_2,2)$

$\delta'((q_1,0),1) = (q_3,3)$

$\delta'((q_1,1),0)) = (q_2,2)$

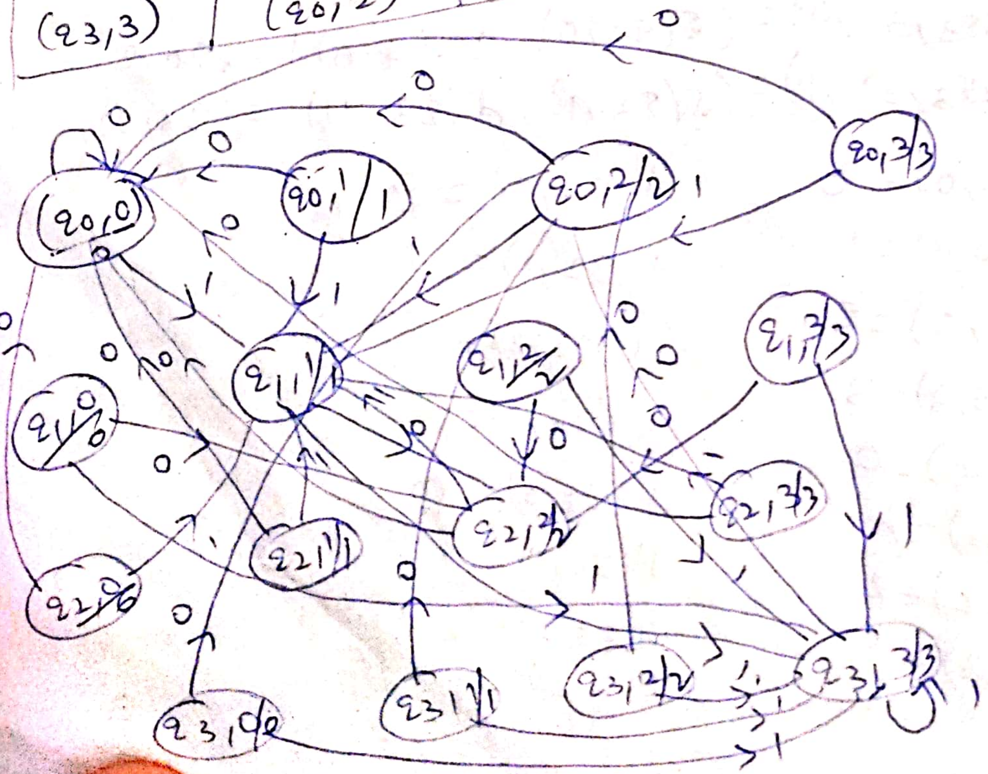
$\delta'((q_1,1),1)) = (q_3,3)$

$$\delta'((q_1,2),0) = (q_2,2)$$
$$\delta'((q_1,2),1) = (q_3,3)$$
$$\delta'((q_1,3),0) = (q_2,2)$$
$$\delta'((q_1,3),1) = (q_3,3)$$
$$\delta'((q_2,0),0) = (q_0,0)$$
$$\delta'((q_2,0),1) = (q_1,1)$$
$$\delta'((q_2,1),0) = (q_0,0)$$
$$\delta'((q_2,1),1) = (q_1,1)$$
$$\delta'((q_2,2),0) = (\delta(q_2,0),\lambda(q_2,0)) = (q_0,0)$$
$$\delta'((q_2,2),1) = (\delta(q_2,1),\lambda(q_2,1)) = (q_1,1)$$
$$\delta'((q_2,3),0) = (\delta(q_2,0),\lambda(q_2,0)) = (q_0,0)$$
$$\delta'((q_2,3),1) = (\delta(q_2,1),\lambda(q_2,1)) = (q_1,1)$$
$$\delta'((q_3,0),0) = (\delta(q_3,0),\lambda(q_3,0)) = (q_0,2)$$
$$\delta'((q_3,0),1) = (\delta(q_3,1),\lambda(q_3,1)) = (q_3,3)$$
$$\delta'((q_3,1),0) = (\delta(q_3,0),\lambda(q_3,0)) = (q_0,2)$$
$$\delta'((q_3,1),1) = (\delta(q_3,1),\lambda(q_3,1)) = (q_3,3)$$
$$\delta'((q_3,2),0) = (\delta(q_3,0),\lambda(q_3,0)) = (q_0,2)$$
$$\delta'((q_3,2),1) = (\delta(q_3,1),\lambda(q_3,1)) = (q_3,3)$$
$$\delta'((q_3,3),0) = (\delta(q_3,0),\lambda(q_3,0)) = (q_0,2)$$
$$\delta'((q_3,3),1) = (\delta(q_3,1),\lambda(q_3,1)) = (q_3,3)$$

$\lambda'(q_0,0) = 0$	$\lambda'(q_2,0) = 0$
$\lambda'(q_0,1) = 1$	$\lambda'(q_2,1) = 1$
$\lambda'(q_0,2) = 2$	$\lambda'(q_2,2) = 2$
$\lambda'(q_0,3) = 3$	$\lambda'(q_2,3) = 3$
$\lambda'(q_1,0) = 0$	$\lambda'(q_3,0) = 0$
$\lambda'(q_1,1) = 1$	$\lambda'(q_3,1) = 1$
$\lambda'(q_1,2) = 2$	$\lambda'(q_3,2) = 2$
$\lambda'(q_1,3) = 3$	$\lambda'(q_3,3) = 3$

Moore Machine transition tables is

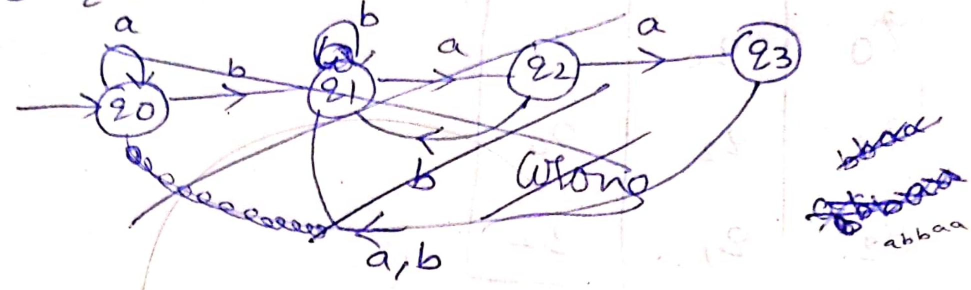
δ	0	1	λ
$(q_0,0)$	$(q_0,0)$	$(q_1,1)$	0
$(q_0,1)$	$(q_0,0)$	$(q_1,1)$	1
$(q_0,2)$	$(q_0,0)$	$(q_1,1)$	2
$(q_0,3)$	$(q_0,0)$	$(q_1,1)$	3
$(q_1,0)$	$(q_2,2)$	$(q_3,3)$	0
$(q_1,1)$	$(q_2,2)$	$(q_3,3)$	1
$(q_1,2)$	$(q_2,2)$	$(q_3,3)$	2
$(q_1,3)$	$(q_2,2)$	$(q_3,3)$	3
$(q_2,0)$	$(q_0,0)$	$(q_1,1)$	0
$(q_2,1)$	$(q_0,0)$	$(q_1,1)$	1
$(q_2,2)$	$(q_0,0)$	$(q_1,1)$	2
$(q_2,3)$	$(q_0,0)$	$(q_1,1)$	3
$(q_3,0)$	$(q_0,2)$	$(q_3,3)$	0
$(q_3,1)$	$(q_0,2)$	$(q_3,3)$	1
$(q_3,2)$	$(q_0,2)$	$(q_3,3)$	2
$(q_3,3)$	$(q_0,2)$	$(q_3,3)$	3



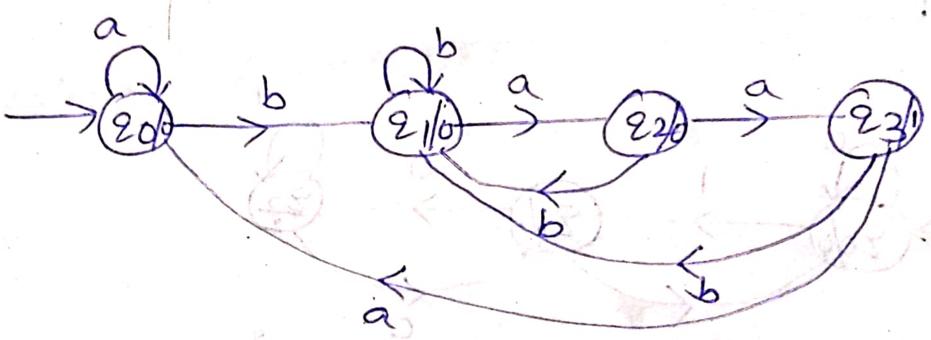
*Design of moore & melay machine:-

*Construct moore machine for set of all strings over $\Sigma=\{a,b\}$ where all strings are ending with 'baa', It can print 1. otherwise 0 as the output.

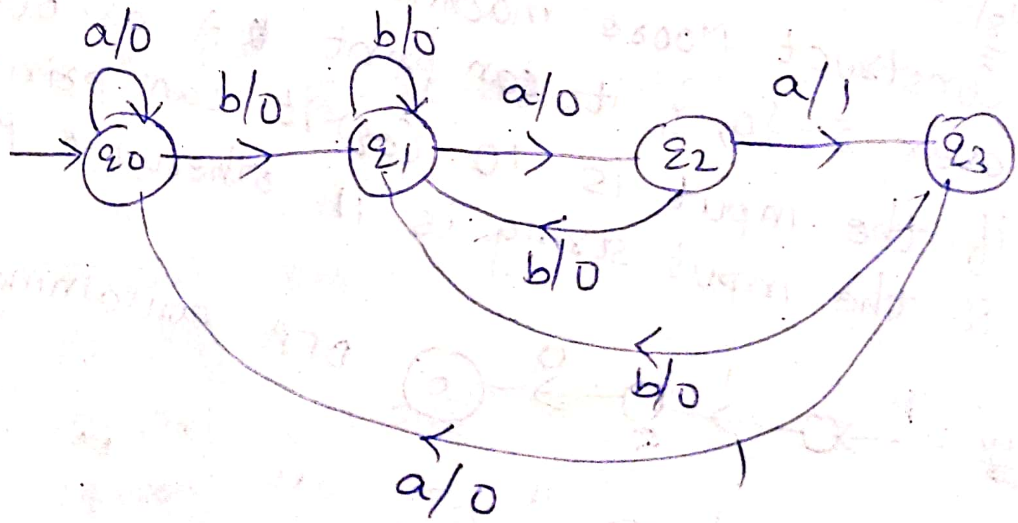
Ans $L=\{baa, abaa, bbaa, aabaa, \text{------}\}$



Mo



Me

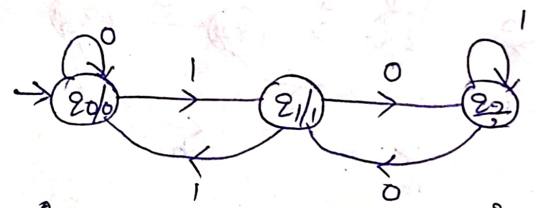


16/8/22

*Construct Moore machine that takes binary number as input and produce mod 3 remainder as a output. $\Sigma = \{0,1\}$

Ans

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2



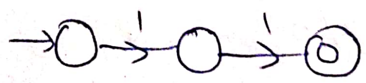
0   1   2

17/8/22

*Construct Moore machine for set of all strings over $\Sigma = \{0,1\}$ it can print A as output if the input is 10 (or) it can print B if the input string is 11 otherwise print C.

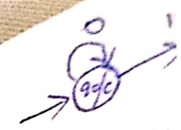
Ans

DFA containing 10



"   "   11

Now combining both.



*Construct M... number a... of number

Ans

0/1

q_0

*Equivalence

*Check wheth...

c

q_1

d

d

c

q_2

- Input alph...

$\Sigma = \{c, d$

state

(q_1, q_4)

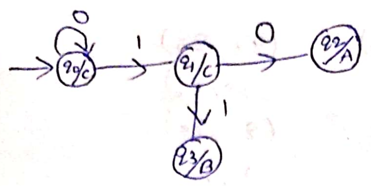
(q_2, q_5)

(q_3, q_7)

(q_1, q_6)

(q_2, q_4)

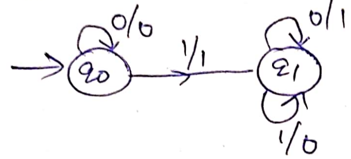
(q_3, q_5)



* Construct Mealy machine that takes binary number as input and produce 2's comp. of number as output.

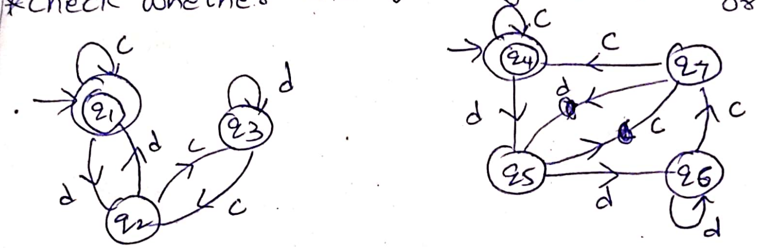
0010
11 10

Ans



* Equivalence of NFA, DFA :-

* Check whether the following FA's are equivalent or not.



- Input alphabets : $\Sigma_1 = \{c, d\}$ $\Sigma_2 = \{c, d\}$ $\Rightarrow$ equal.

state	c	d
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)
(q_3, q_7)	(q_2, q_4)	(q_3, q_5)
(q_1, q_6)	(q_1, q_7)	(q_2, q_6)
(q_2, q_4)	(q_3, q_4)	(q_1, q_5)
(q_3, q_5)	(q_2, q_7)	(q_3, q_6)
(q_1, q_7)	(q_1, q_4)	(q_2, q_5)
(q_2, q_6)	(q_3, q_7)	(q_1, q_6)

(q_3, q_4)	(q_2, q_4)	(q_3, q_5)
(q_1, q_5)	(q_1, q_7)	(q_2, q_6)
(q_2, q_7)	(q_3, q_4)	(q_1, q_5)
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
	NF NF	NF NF

NF - Not final state.

Hence, the given two FA's are not equivalent.

UNIT-2

***Regular expressions:-**

- Regular set : Let Σ be an input alphabet, then a class of set of all strings over Σ called Regular sets.

Ex:- ϕ is a regular set over Σ
ϵ is a regular set over Σ.

- If R, S two regular sets over Σ then
  - i) $R \cup S$ is a regular set over Σ
  - ii) RS or $R \circ S$ is also regular set over Σ
  - iii) R^* is also regular set over Σ

- Regular expressions : Let Σ is an input alphabet, then the regular expressions over Σ denoting the regular sets

Ex:- $\{\phi\}$ is a regular set, then the regular expression is denoting ϕ.
ϵ is a regular expression denoting the regular set $\{\epsilon\}$

- If R, S are two regular sets then regular expression denoting
  - i) R+S is a regular set $\{R \cup S\}$ (or) $\{R, S\}$ the regular expression denoting
  - (ii) RS is the regular expression denoting the regular set $\{RS\}$ (or) $\{R \circ S\}$
  - (iii) R^* is a regular expression denoting the regular set $\{R^*\}$

- a is the regular expression denoting the regular set $\{a\}$.

Examples

Regular sets
1) $\{101\}$
2) $\{10, 11\}$
3) $\{a, b, c\}$
4) $\{\epsilon, 0, 10, 00, \ldots\}$
5) $\{1, 11, 111, \ldots\}$

*Write the regular ... all
1) the set of ...
2) "
3) "
4) "
5) "
6) "
7) " atleast 2 ... two occurence ...
8) "

Ans
1) $\Sigma = \{0, 1\}$
R.E $= \Sigma^* 0 ...$
$= (0+1) ...$

3) $\Sigma = \{0, 1\}$
R.E $= 0^* 1 ...$

5) $\Sigma = \{0, 1\}$
R.E $= (0+1)^*$

Regular sets	Regular expressions
1) $\{101\}$	101
2) $\{10, 11\}$	$10+11$
3) $\{a, b, c\}$	$a+b+c$
4) $\{\epsilon, 0, 00, \dots\}$	0^* $\cancel{\epsilon+0+00+\dots}$
5) $\{1, 11, 111, \dots\}$	1^+ $\cancel{1+11+111+\dots}$

* Write the regular expressions for the following:-

① The set of all strings over $\Sigma=\{0,1\}$ ending with 00

② " " " " " starting with 0, ending with 1

③ " " " " " containing single 1

④ " " " " " has atleast one 1.

⑤ " " " " where 10^{th} symbol from right end is 1.

⑥ " " " " " having even no. of zeroes & odd no. of ones

⑦ " " " $\Sigma=\{a,b\}$ where there exists atleast 2 occurences of b's in between two occurences of a's.

⑧ " " $\Sigma=\{0,1\}$ the no. of zeroes are divisible by 3.

Ans

① $\Sigma=\{0,1\}$

$R.E = \Sigma^* 00$

$= (0+1)^* 00$

② $\Sigma=\{0,1\}$

$R.E = 0\Sigma^* 1$

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

③ $\Sigma=\{0,1\}$

$R.E = 0^* 1 0^*$

④ $\Sigma=\{0,1\}$

$R.E = \Sigma^* 1 \Sigma^*$

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

⑤ $\Sigma=\{0,1\}$

$R.E = (0+1)^* 1 (0+1)^9$

⑥ $\Sigma = \{0,1\}$

~~$RE = 00^{*} + 1^{*} 1^{*}$~~ $RE = 00^{+} \mid 11^{+}$

⑦ $\Sigma = \{a, b\}$

$RE = abb(\cancel{a}b)^{*} a$ $RE = abbb^{*} a$

⑧ $\Sigma = \{0, 1\}$

$RE = 000^{*} \cancel{\phi} . 1^{*}$

Identity rules

① $r_1 + \phi = r_1$

② $r_1 \phi = \phi r_1 = \phi$

*③ $\epsilon r_1 = r_1 \epsilon = r_1$

④ $r_1 + r_2 = r_2 + r_1$

⑤ $r_1 + r_1 = r_1$

⑥ $r_1(r_2 + r_3) = (r_1 + r_2) + r_3$

*⑦ $r^{*} r^{*} = r^{*}$

*⑧ $r r^{*} = r^{*} r$

*⑨ $(r^{*})^{*} = r^{*}$

*⑩ $(r_1 + r_2)^{*} = (r_1 + r_2^{*})^{*}$

*⑪ $\epsilon + (rr)^{*} = r^{*} = \epsilon + rr^{*}$

⑫ $(r_1 + r_2) r_3 = r_1 r_3 + r_2 r_3$

⑬ $(r_1 r_2)^{*} r_3 = r_1 (r_2 r_3)^{*}$

*⑭ $(r_1 + \epsilon)^{*} = r_1^{*}$

* Arden's

$R = RP + Q$

$R = QP^{*}$

*Find the

$q_1 = \epsilon + q$

$q_2 = q_1 0$

$q_3 = q_1 1$

$q_4 = q_2 0$

④ ⇒ $q_4 = q_4($

① ⇒ $q_1 = \epsilon +$

$q_1 = q_1$

$\frac{q_1}{R} = \frac{q_1}{R}$

⑤ ⇒ $q_4 = q_4$

$q_4 = q_4$

$\frac{q_4}{R} = \frac{q}{R}$

$\boxed{q_4 = (0}$

$q_2 = (01$

11x

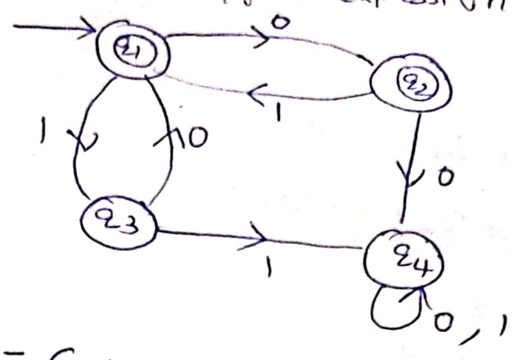
$e = abbb^*a$

* arden's formula

$R = RP + Q$

$R = QP^*$

* Find the regular expression for below FA :-



$q_1 = \epsilon + q_2 1 + q_3 0 \rightarrow (1)$

$q_2 = q_1 0 \rightarrow (2)$

$q_3 = q_1 1 \rightarrow (3)$

$q_4 = q_2 0 + q_3 1 + q_4 0 + q_4 1 \rightarrow (4)$

$(4) \Rightarrow q_4 = q_4(0+1) + q_3 1 + q_2 0 \rightarrow (5)$

$(1) \Rightarrow q_1 = \epsilon + q_2 1 + q_3 0$

$q_1 = q_1 01 + q_1 10 + \epsilon$ (from (2) & (3))

$\underbrace{q_1}_{R} = \underbrace{q_1}_{R}\underbrace{(01+10)}_{P} + \underbrace{\epsilon}_{Q} \Rightarrow \boxed{q_1 = (01+10)^*}$

$(5) \Rightarrow q_4 = q_4(0+1) + q_1 11 + q_1 00$

$q_4 = q_4(0+1) + q_1(11+00)$

$\underbrace{q_4}_{R} = \underbrace{q_4}_{R}\underbrace{(0+1)}_{P} + \underbrace{(01+10)^*(11+00)}_{Q}$

$$\boxed{q_4 = (01+10)^*(11+00)(0+1)^*}$$

$q_2 = (01+10)^* 0$

$q_3 = (01+10)^* 1$

*Find regular expression for given FA

$\boxed{q_0 = \epsilon\ \cancel{\phi}} \rightarrow ①$

$q_1 = q_0 0 + q_0 1 + q_1 0 \rightarrow ②$

$q_2 = q_1 0 + q_1 1 + q_2 1 \rightarrow ③$

$② \Rightarrow q_1 = q_0(0+1) + q_1 0$

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

$\underset{R}{q_1} = \underset{R}{q_1}\,\underset{P}{0} + \underset{Q}{(0+1)}$

$\boxed{q_1 = (0+1)\,0^*}$

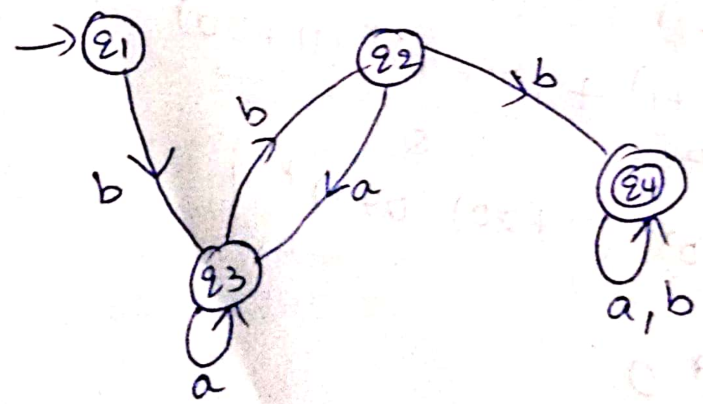
$q_2 = (0+1)0^*\,0 + (0+1)0^*\,1 + q_2 1$

$\underset{R}{q_2} = \underset{R}{q_2}\,\underset{P}{1} + \underbrace{((0+1)\,0^*)(0+1)}_{Q}$

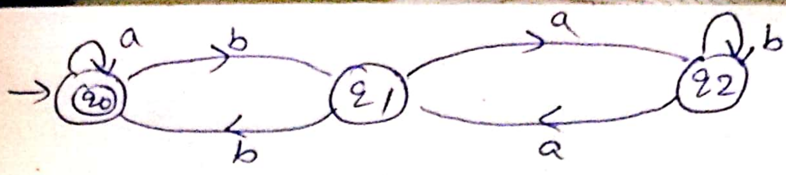
$\boxed{q_2 = (0+1)\,0^*(0+1)\,1^*}$

Find regular expression for FA's below:

①



(2)



(1) Sol $\boxed{q_1 = \epsilon} \rightarrow (1)$

$q_2 = q_3 b \rightarrow (2)$

$q_3 = q_1 b + q_3 a + q_2 a \rightarrow (3)$

$q_4 = q_2 b + q_4 a + q_4 b \rightarrow (4)$.

$\underbrace{q_4}_{R} = \underbrace{q_4}_{R}\underbrace{(a+b)}_{P} + \underbrace{q_2 b}_{Q}$

~~q_4~~.

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

$q_3 = q_3(a+ba) + q_1 b$

$q_3 = q_3(a+ba) + b$

$\boxed{q_3 = b(a+ba)^*}$

$\boxed{q_2 = b(a+ba)^* b}$

$q_4 = q_4(a+b) + b(a+ba)^* b b$

$\boxed{q_4 = b(a+ba)^* bb (a+b)^*}$

(2) Sol

$q_0 = \epsilon + q_0 a + q_1 b \rightarrow (1)$

$q_1 = q_0 b + q_2 a \rightarrow (2)$

$q_2 = q_1 a + q_2 b \rightarrow (3)$

$q_0 = \epsilon + q_0 a + q_1 b$

given FA

$q_0 1 + q_1 0$

$q_1 1 + q_2 1 \rightarrow ②$

$1) + q_1 0 \rightarrow ③$

$1) + q_1 0$

$(0+1)$

ϕ

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

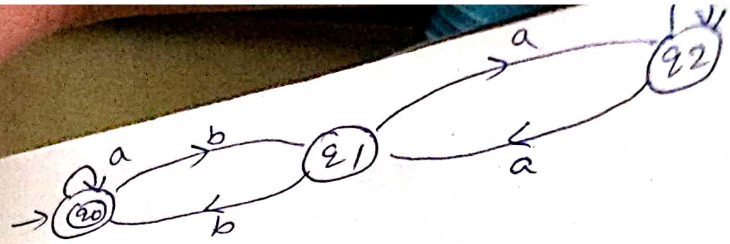
$1) 0^*)(0+1)$

ϕ

$) 1^*$

ion for FA's below:

②



① Sol

$\boxed{q_1 = \epsilon} \rightarrow ①$

$q_2 = q_3 b \rightarrow ②$

$q_3 = q_1 b + q_3 a + q_2 a \rightarrow ③$

$q_4 = q_2 b + q_4 a + q_4 b \rightarrow ④$

$\underbrace{q_4}_{R} = \underbrace{q_4}_{R}\underbrace{(a+b)}_{P} + \underbrace{q_2 b}_{Q}$

Sub ④

$q_3 = q_1 b + q_3 a + q_3 ba$

$q_3 = q_3(a+ba) + q_1 b$

$q_3 = q_3(a+ba) + b$

$\boxed{q_3 = b(a+ba)^*}$

$\boxed{q_2 = b(a+ba)^* b}$

$q_4 = q_4(a+b) + b(a+ba)^* bb$

$\boxed{q_4 = b(a+ba)^* bb (a+b)^*}$

② Sol

$q_0 = \epsilon + q_0 a + q_1 b \rightarrow ①$

$q_1 = q_0 b + q_2 a \rightarrow ②$

$q_2 = q_1 a + q_2 b \rightarrow ③$

$q_0 = \epsilon + q_0 a + q_1 b$

Scanned with CamScanner

$$\underset{R}{q_2} = \underset{R}{q_2}\underset{P}{b} + \underset{Q}{q_1 a}$$

$$q_2 = q_1 ab^* \rightarrow (4)$$

$$\underset{R}{q_1} = \underset{Q}{q_0 b} + \underset{R}{q_1}\underset{P}{ab^* a}$$

$$q_1 = q_0 b (ab^* a)^* \rightarrow (5)$$

$$q_0 = \epsilon + q_0 a + q_0 b (ab^* a)^* b$$

$$\underset{R}{q_0} = \underset{R}{q_0}\underset{P}{(a + b(ab^* a)^* b)} + \underset{Q}{\epsilon}$$

$$\boxed{q_0 = \epsilon (a + b(ab^* a)^* b)^*}$$

$$\Rightarrow \boxed{q_0 = (a + b(ab^* a)^* b)^*}$$

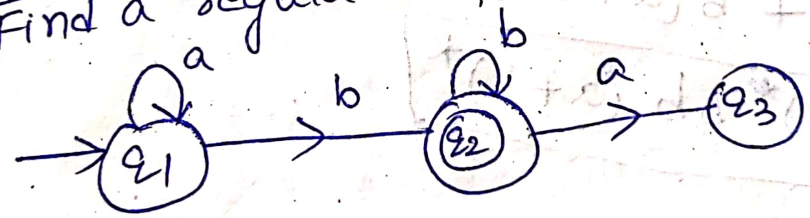
$$\boxed{q_1 = (a + b(ab^* a)^* b)^* b (ab^* a)^*}$$

$$\boxed{q_2 = (a + b(ab^* a)^* b)^* b (ab^* a)^* ab^*}$$

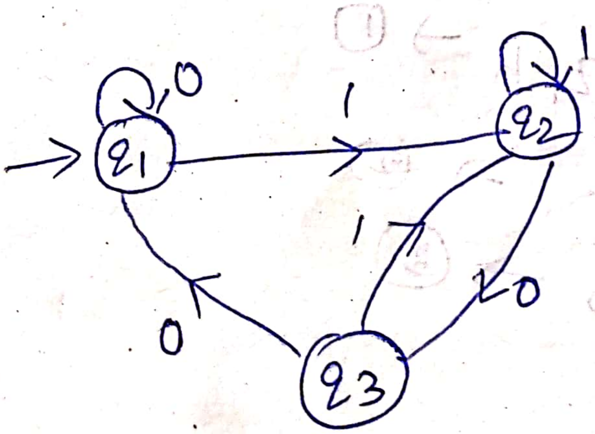
18/8/22

* Find a regular expression for the given FA

(1)



(2)



(1)Sol $q_1 = \epsilon$

$q_2 =$

$q_3 =$

$\underset{R}{q_1} = \underset{R}{q_1}$

$q_1 = \epsilon$

$\underset{R}{q_2} =$

q_3

(2)Sol q_1

q_2

q

* Con

car

conve

(1) Sol

$$q_1 = \epsilon + q_1 a \rightarrow (1)$$
$$q_2 = q_1 b + q_2 b \rightarrow (2)$$
$$q_3 = q_2 a \rightarrow (3)$$

$$\underset{R}{q_1} = \underset{R}{q_1}\,\underset{P}{a} + \underset{Q}{\epsilon}$$

$$q_1 = \epsilon a^* \Rightarrow \boxed{q_1 = a^*}$$

$$\underset{R}{q_2} = \underset{R}{q_2}\,\underset{P}{b} + \underset{Q}{a^* b} \Rightarrow \boxed{q_2 = a^* b b^*}$$

$$\boxed{q_3 = a^* b b^* a}$$

(2) Sol

$$q_1 = \epsilon + q_1 0 + q_3 0 \text{ — } (1)$$
$$q_2 = q_1 1 + q_2 1 + q_3 1 \text{ — } (2)$$
$$q_3 = q_2 0 \text{ — } (3)$$

$$q_2 = q_2 1 + q_2 01 + q_1 1$$
$$q_2 = q_2 (1+01) + q_1 1$$
$$q_2 = q_1 1 (1+01)^*$$

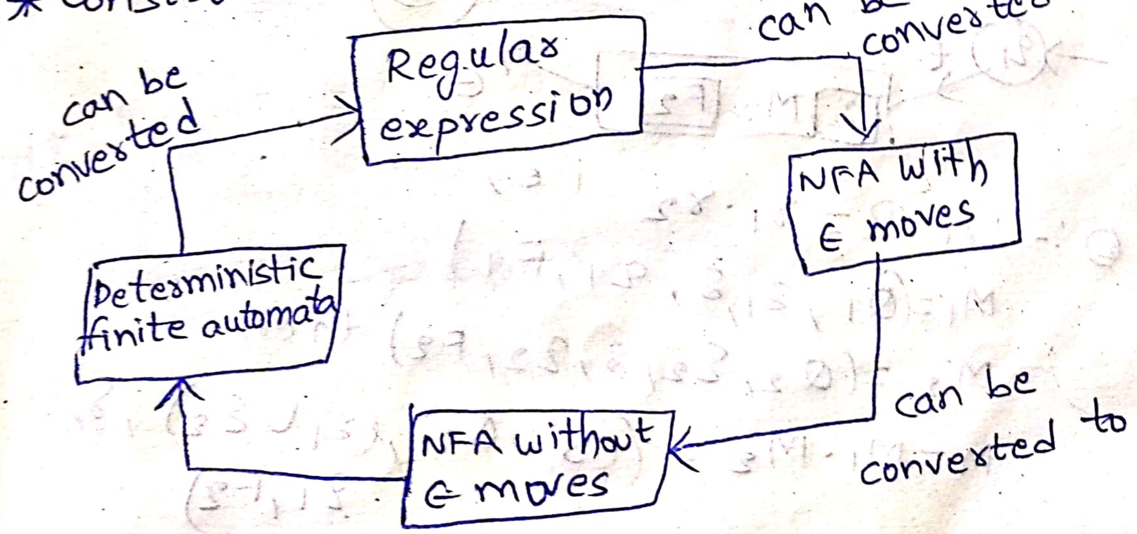
$$q_3 = q_1 1 (1+01)^* 0$$

$$q_1 = \epsilon + q_1 0 + q_1 1 (1+01)^* 00$$
$$q_1 = q_1 (0 + 1(1+01)^* 00) + \epsilon$$
$$q_1 = \epsilon (0 + 1(1+01)^* 00)^* \Rightarrow \boxed{q_1 = (0+1(1+01)^*00)^*}$$

$$\boxed{q_2 = (0+1((1+01)^*00)^* 1(1+01)^*}$$

$$\boxed{q_3 = (0+1(1+01)^*00)^* 1(1+01)^* 0}$$

* Construction of regular expression for FA



- Construction of FA from regular expression:-

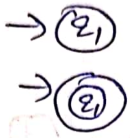
Theorem:-

Let R be a regular expression then there exists a NFA with ϵ transitions or NFA that accept $L(R)$.

Proof:-

- zero operators means the R ~~first~~ must accept ϵ, ϕ. For this regular expression $R=\phi$ or $R=\epsilon$ i.e., $R=\phi$ $\rightarrow q_1$

$R=\epsilon$ $\rightarrow q_1$



- One or more operators

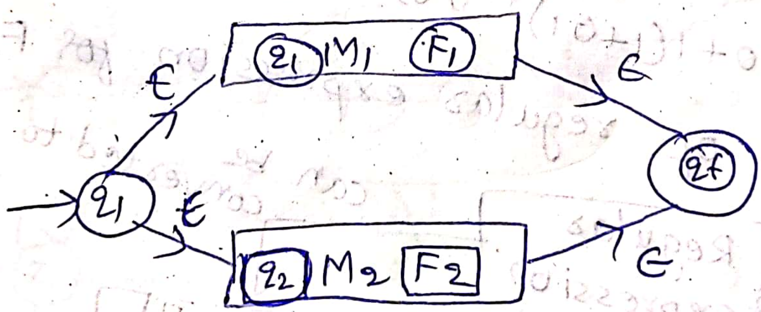
Assume that the theorem is true for $n \geq 1$. for any given regular expression, there are 3 cases Those are:-

case ①:- $R = r_1 + r_2$ where r_1 and r_2 are two regular expressions i.e., $M_1 = (Q_1, \Sigma_1, \delta, q_1, F_1)$

$M_2 = (Q_2, \Sigma_2, \delta, q_2, F_2)$ then

$M_1 \cup M_2 = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1, q_f)$

i.e.,

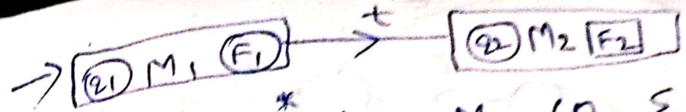


case ②:- Let $R = r_1 \cdot r_2$ i.e.,

$M_1 = (Q_1, \Sigma_1, \delta, q_1, F_1)$

$M_2 = (Q_2, \Sigma_2, \delta, q_2, F_2)$ then

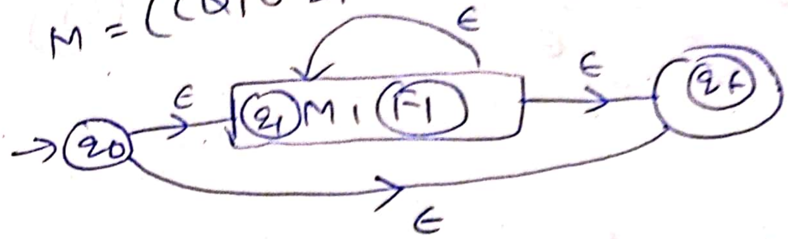
$M = M_1 \cdot M_2 = ((Q_1 \cup Q_2), (\Sigma_1 \cup \Sigma_2), \delta, q_1, F_2)$



case ③ :- $R = r_1^*$ i.e., $M_1 = (Q_1, \Sigma, \delta, q_1, F_1)$

$M = M_1^*$

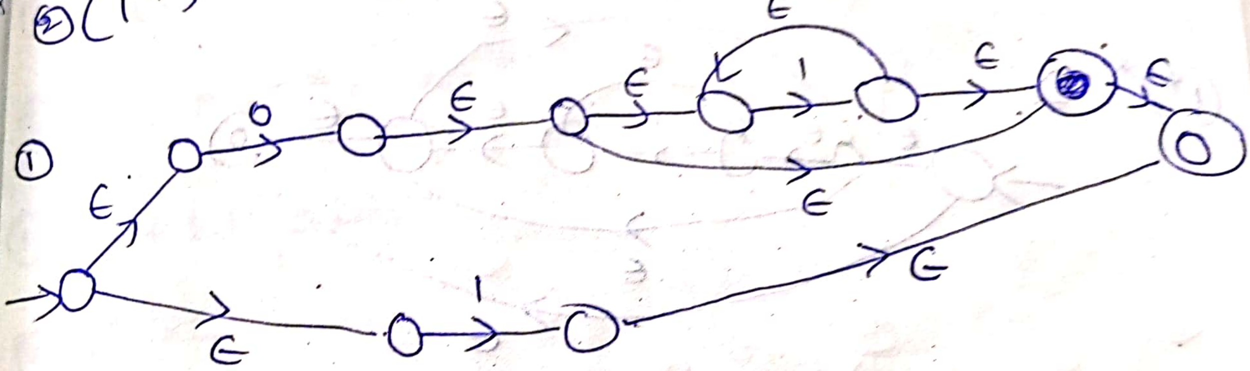
$M = ((Q_1 \cup q_1 \cup F_1), \Sigma, \delta, q_0, q_f)$



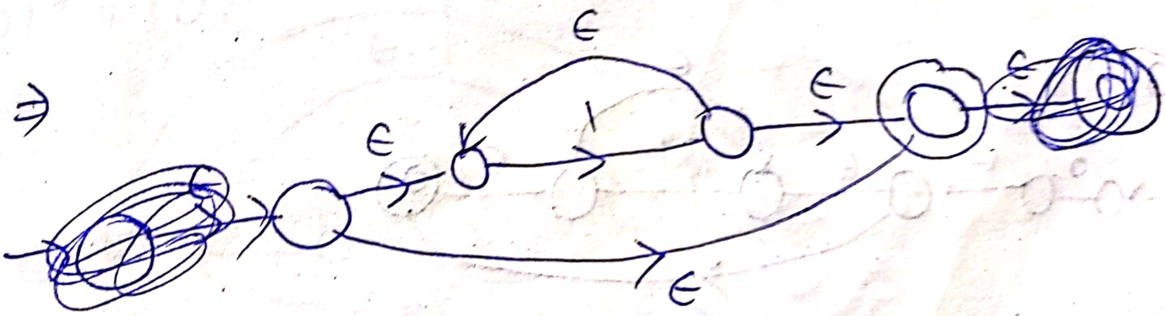
*Construct FA for the given regular expressions

① $01^* + 1$ ② $(1^*)^*$ ③ 1^* ④ 01^* ⑤ $1+0^*$ ⑥ 1^*+10

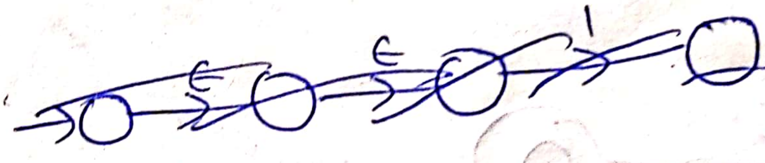
①

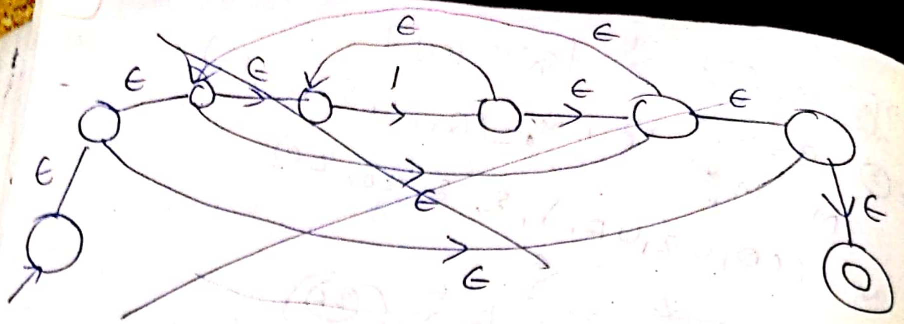


$1^* \Rightarrow$

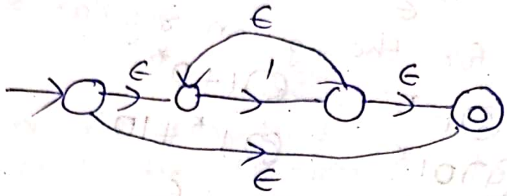


$(1^*)^* \Rightarrow$

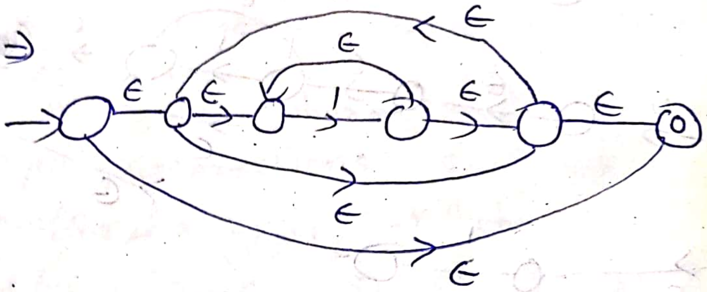




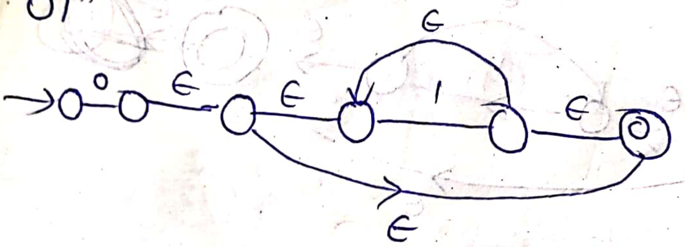
1* ⇒



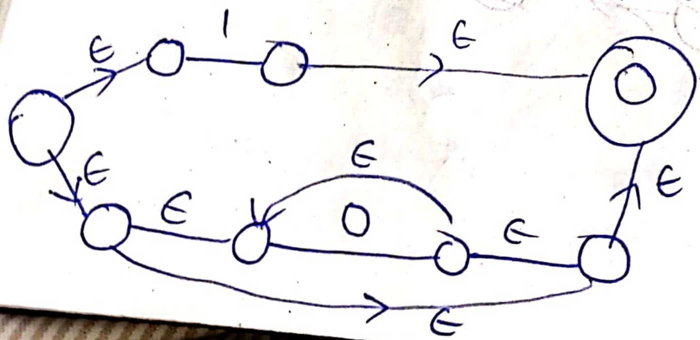
(1*)* ⇒



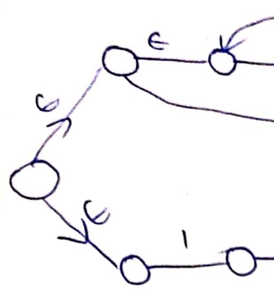
01*



1+0*



1* +10



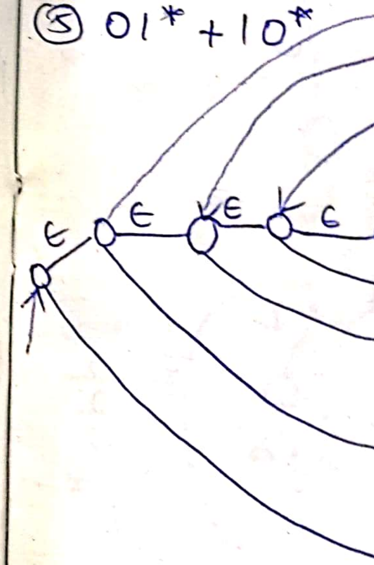
① (((0*)*)*)*

② ((1*0)*01

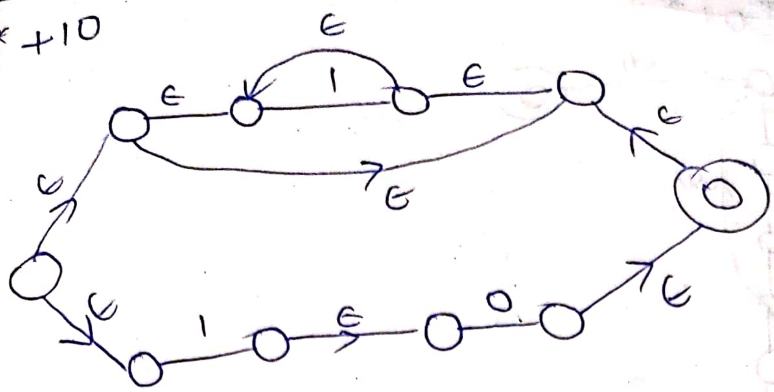
③ (a+b)*abb

④ a*bc

⑤ 01*+10*



$1^* + 10$



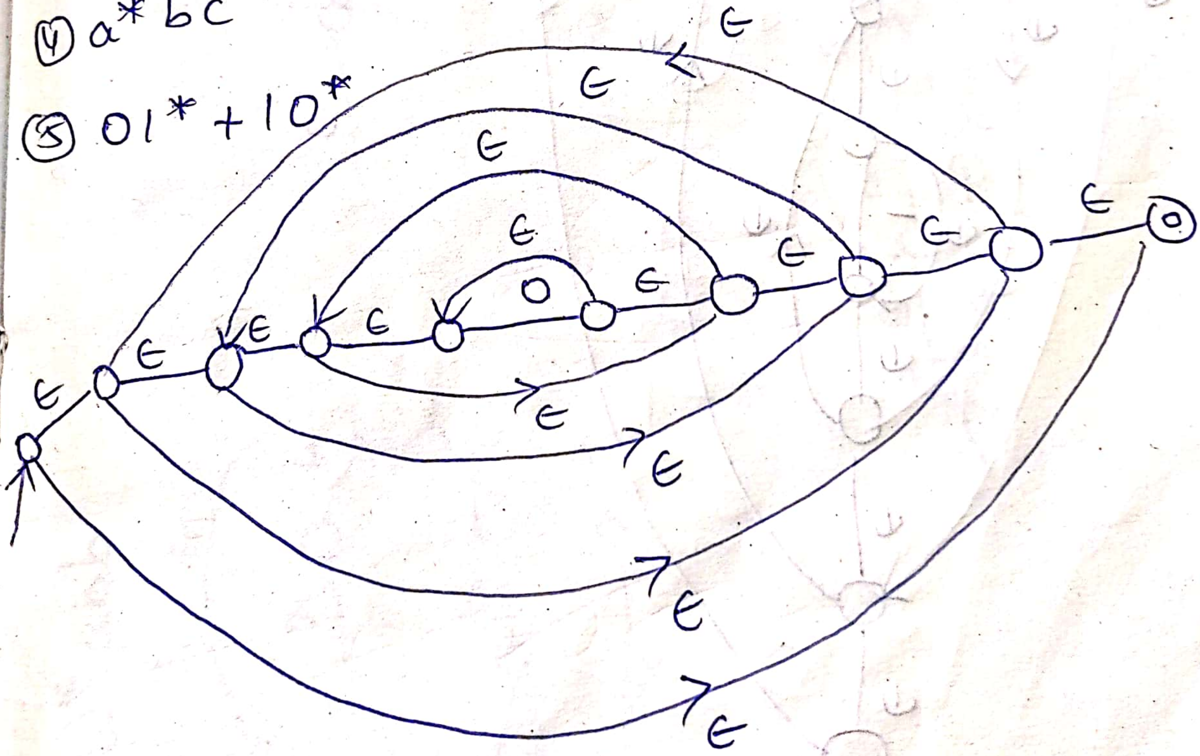
① $(((0^*)^*)^*)^*$

② $((1^* 0)^* 01^*)^*$

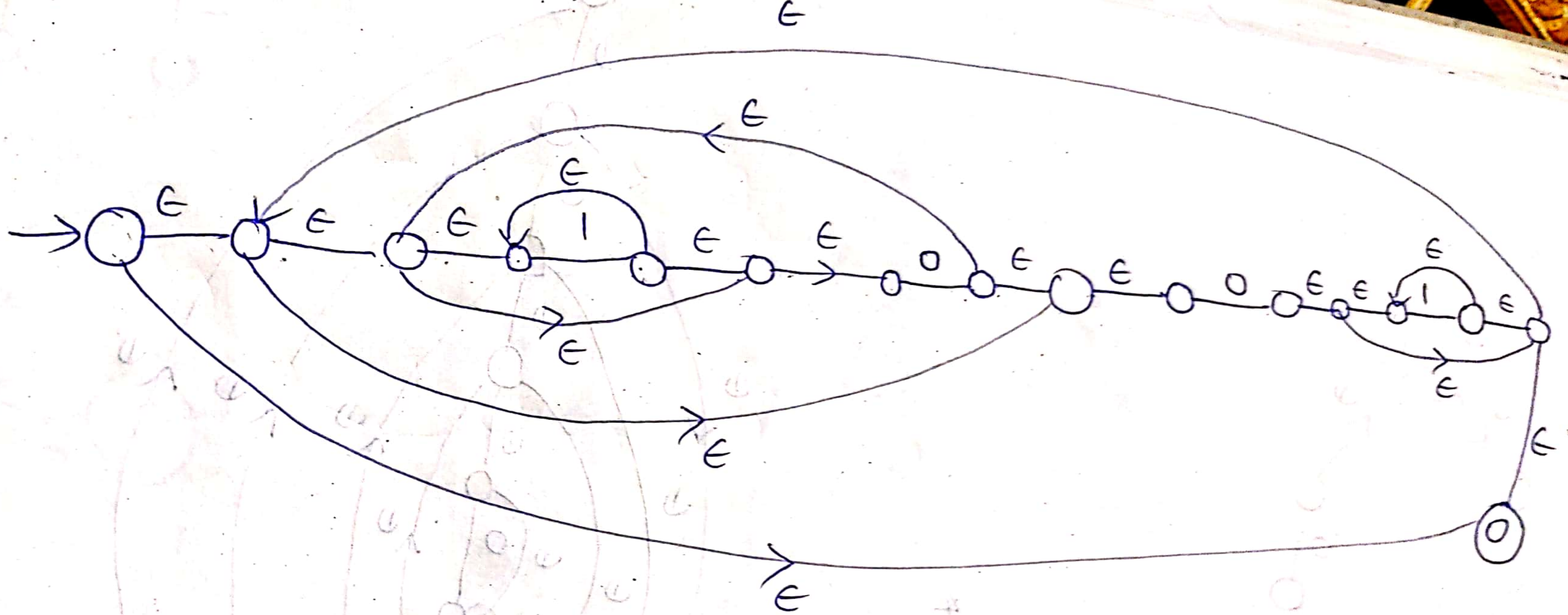
③ $(a+b)^* abb$

④ $a^* bc$

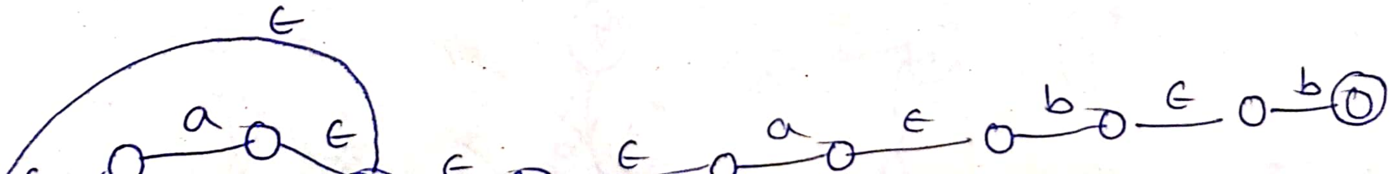
⑤ $01^* + 10^*$



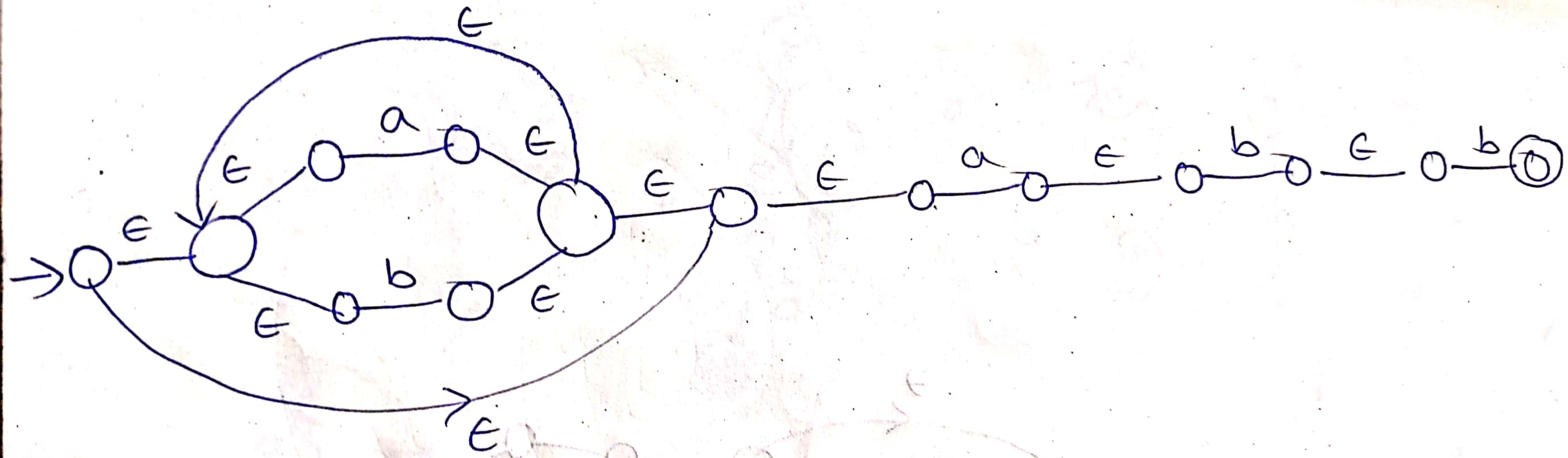
② $((1^*0)^*01^*)^*$



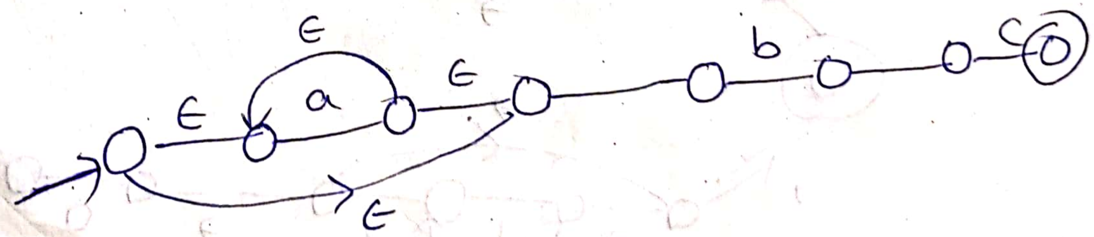
③ $(a+b)^*abb$



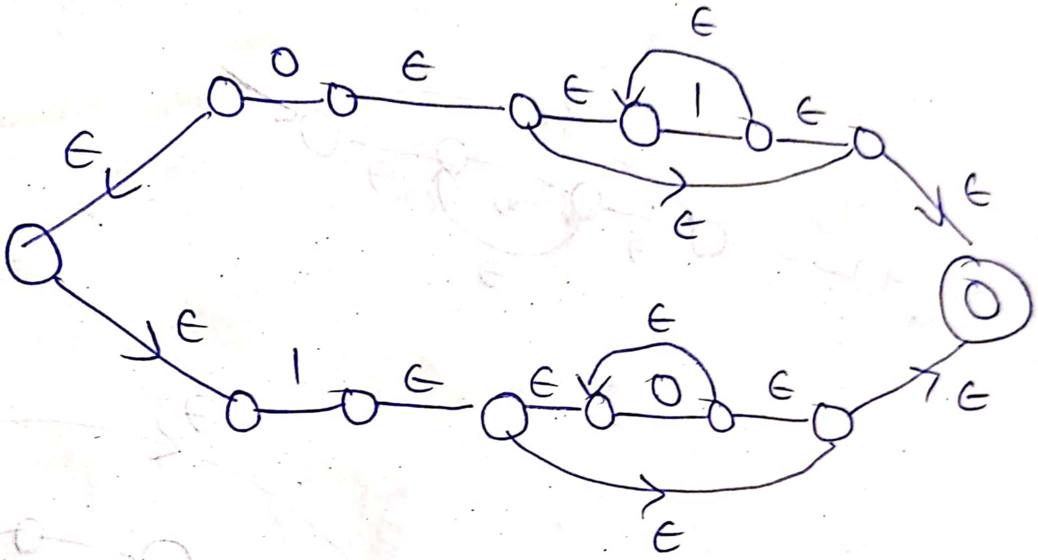
(3) $(a+b)^* abb$



(4) $a^* bc$

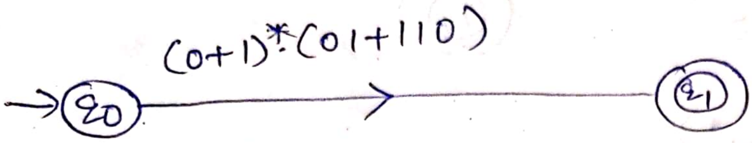


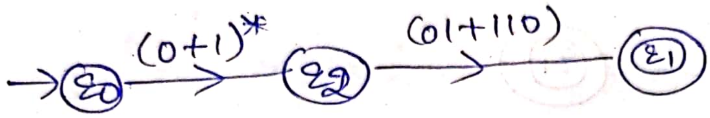
⑤ $01^* + 10^*$

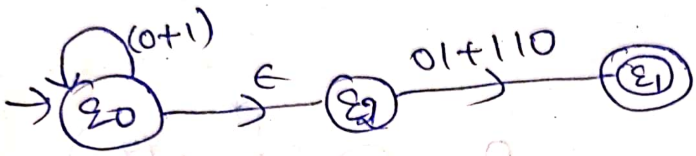


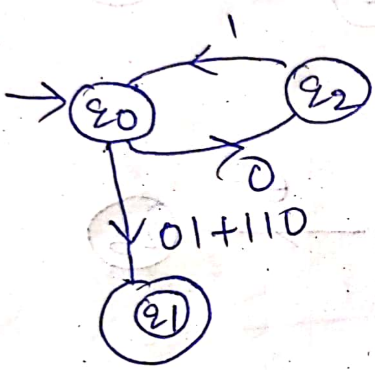
* Construct NFA without ∈ for given regular expression.

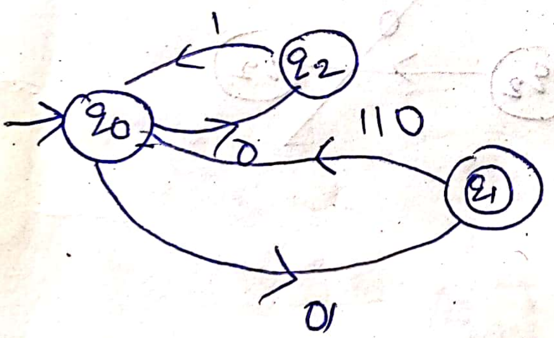
$(0+1)^*(01+110)$

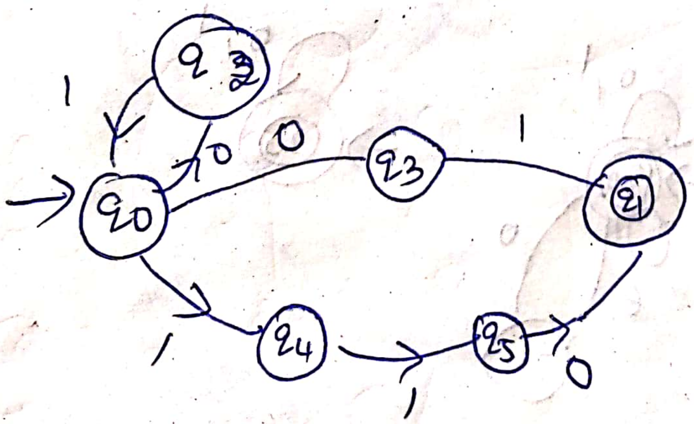


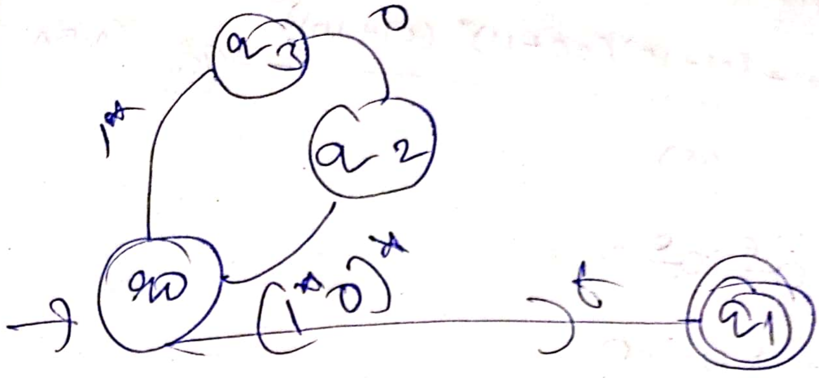


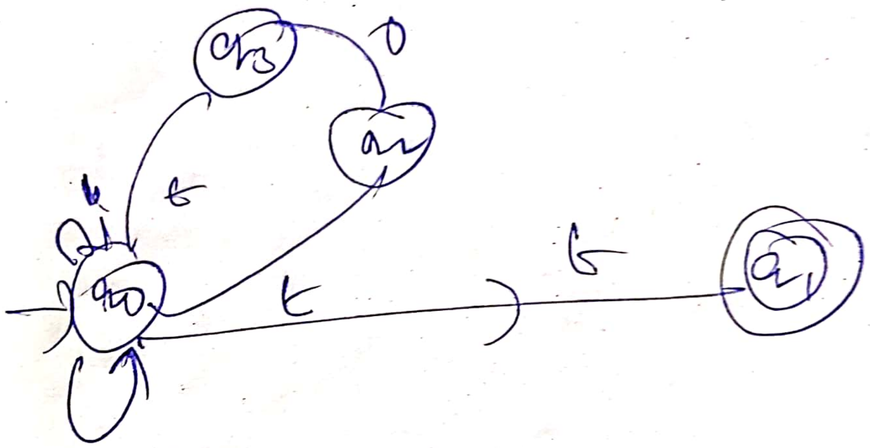


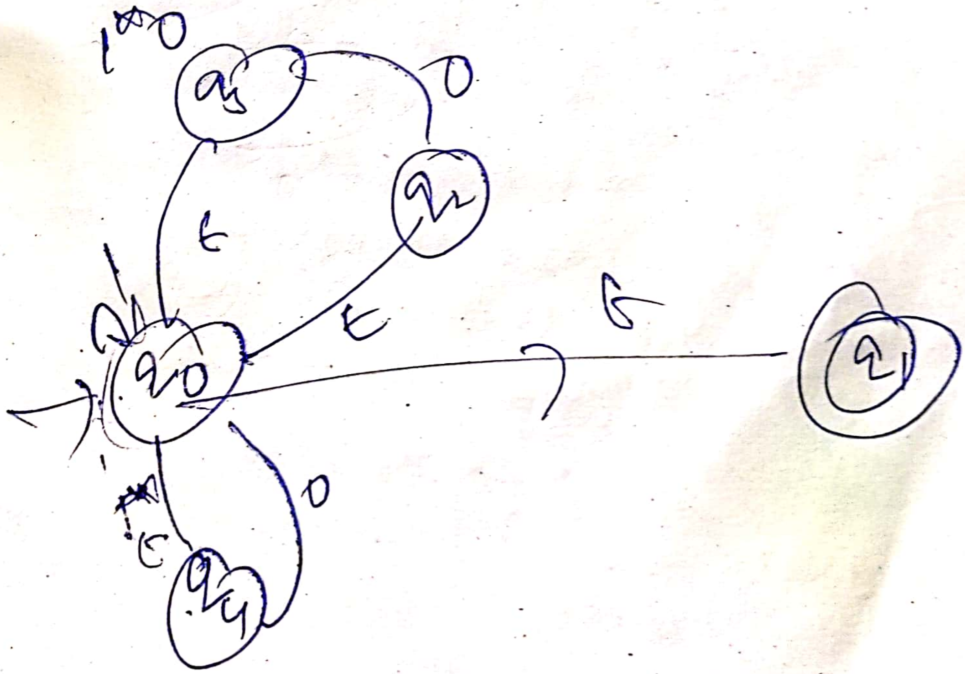








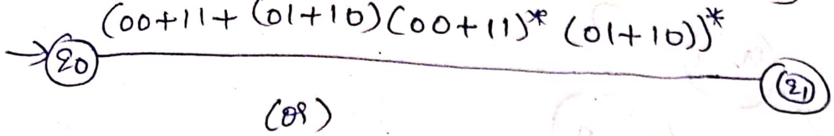




contd.

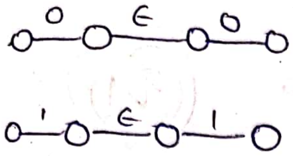
20/8/22

*Construct FA for given regular expression $[00+11+(01+10)(00+11)^*(01+10)]^*$



(NFA)

(or)



20/3/22
*Con: * closure properties of regular expressions:-

① The union of two regular languages is also regular language.

② The concatenation of two regular languages is also regular language

Q.

③ The intersection of two regular languages is also regular language

④ The complimentation of two regular languages is also regular language

⑤ kleen closure of regular language is also a regular language.

⑥ Positive closure " " " " "
" "

⑦ Inverse homomorphism of given regular language is also a regular language

*pumping lemma:- It is used to prove that set of languages (or) certain sets are not regular.

step ①:- Assume that L is a regular language
let n be no.of states in M

step ②:- choose a string w such that $|w| \geq n$. By using pumping lemma to write $w = xyz$

(i) $|y| > 0$ (ii) $|xy| \leq n$

step ③:- Find a suitable integer i such that $xy^iz \notin L$, hence the language is not regular.

*Show that the language $L = \{a^n b^n / n > 1\}$ is not regular.

Step ① Assume that L is R.L
n no.of states in M

nguages ...ns:-
is
ular languages

ular languages

regular
uage
uage is also

" "

regular
guage
ve that
are

anguage

t
na to

such
language

1}

$L = \{ a^{*}abb, aaabbb, \text{-------} \}$

step ② Let $w = aabb$

$|y| > 0$; $|xy| \leq n$

step ③ $w = aabb$

Now, we are going to divide w into 3 parts i.e., x, y, z.

$x = a$

$y = a$

$z = bb$

$xy^i z \ \forall i \geq 0$

$i=1$ $xy^i z = xy^1 z = aabb \in L$

$i=2$ $xy^i z = xy^2 z = aaabb \notin L$

$\therefore L$ is not regular.

*show that language A^{2n} where $n \geq 1$ is not regular.

step ① $L = \{AA, AAAA, \text{---------}\}$

Assume that 'L' is regular language. n no. of states in 'M'.

step ② $w = AAAA$

$|y| > 0$

$|xy| \leq n$

$x = A$

$y = A$

$z = AA$

i.e., $xy^i z \ \forall i \geq 0$

$i=1$ $xy^1 z = AAAA \in L$

$i=2$ $xy^2 z = AAAAA \notin L$

$\therefore L$ is not regular.

20/12/2-
*Cons

Co

*show that $L = A^p$ where p is a prime number is not regular

→(20)

step ①:- Assume that L is R.L
n no of states in 'M'

$L = \{AA, AAA, AAAAA, \ldots\}$

step ② ~~chor~~ $w = AAAAA$

$x = AA$

$y = A$

$z = AA$

$i=1$ $xy^iz = AAAAA \in L$

$i=2$ $xy^2z = AAAAAA \notin L$

$\therefore$ L is not regular.

* Context-free grammar:-

- Context-free grammars are applied in parser design. They are also useful for describing the block structure in programming language.

- It consists of variables, terminals, start symbol & production i.e, $G(v, t, p, s)$

Variables: Non-terminals or finite set of variables denoted by capital letters

A, B, ..., Z

terminals:- The terminals are finite set of numbers denoted by small letters or symbols

Ex a, b, ..., 0, ..., 9, ..., $ etc.

Production:- It ...
the form of $\alpha \rightarrow$ ...

start symbol:- One of ...
used as start sy...

Ex: $S \rightarrow absQA$

$S \rightarrow a$

$A \rightarrow aa$

where $G = \{V, T, P$ ...

$V = \{S, A\}$ T= ...

*derivation:-

- Deriving the te...
by using the pr...

Ex: $A \rightarrow 0A1$

$B \rightarrow \#$

- Derive a string ...

Ans The given gra...
where V = va...
T = te...
S = s...

$A \rightarrow 0A1$

$\rightarrow 00A11$

$\rightarrow 000A11$ ...

$\rightarrow 000B1$ ...

$\rightarrow 000\#1$ ...

*Derivation tr...

- A tree repre...
of the give...
free gramm...
(or) parser tre...

Ex: Derivatio...
consideri...

a prime

}

in
for
ramming

tart

s)

f

numbers

~~start~~ **Production**:- It is a set of rules are in the form of $\alpha \rightarrow \beta$ where $\alpha \in V$, $\beta \in V \cup T$

start Symbol:- One of the non-terminal is used as start symbol.

Ex:
$S \rightarrow absQA$
$S \rightarrow a$
$A \rightarrow aa$
where $G = \{V, T, P, S\}$
$V = \{S, A\}$ $T = \{a, b\}$ $S = \{S\}$

***derivation**:-
- Deriving the terminals from the non-terminals by using the productions is called derivation.

Ex: $A \rightarrow 0A1 \mid A \rightarrow B$
$B \rightarrow \#$

- Derive a string 000#111

Ans The given grammar $G = (V, t, P, S)$
where V = variables $= \{A, B\}$
T = terminals $= \{0, 1, \#\}$
S = starting symbol $= \{A\}$

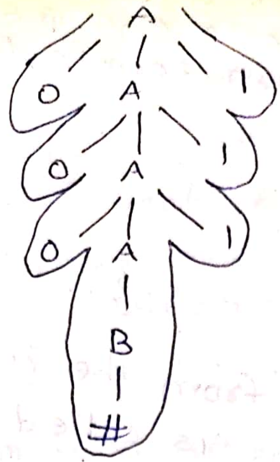
$A \rightarrow 0A1$
$\rightarrow 00A11$
$\rightarrow 000A111$
$\rightarrow 000B111$
$\rightarrow 000\#111$

***Derivation tree (OR) parser tree (OR) syntax tree**
- A tree representation for the derivation of the given productions for a context free grammar is called derivation tree (OR) parser tree (OR) syntax tree.

Ex: Derivation tree for string 000#111 by considering above Grammar

20/8/22
*Cons
[O

→(20)



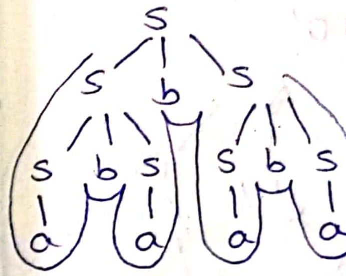
- The derivations are classified into 2 types.
① Left most derivation ② Right most derivation.
- Left most derivation :- In left most derivation of the string, the left most non-terminal is replaced first at each step in derivation.
- In Right most derivation of string the right most non-terminal is replaced first at each step in the derivation.

Q. S → absaA
S → a
A → aa
where G is grammar.

Eg: s → sbs/a
Derive a string "abababa"
The given grammar G=(V, T, P, S)
where V = {S}
T = {a, b} S=S

LMD S → sbs
S → a

s → sbs
~~s → sbsbsbs~~
s → sbsbs
s → sbsbsbs
s → abababa



* Construct lef
for following
① 00101
Grammer is
A → A | B
A → 0A | ε
B → 0B | 1B

* Given gramm
V = {A, B}
T = {0, 1,
S = A

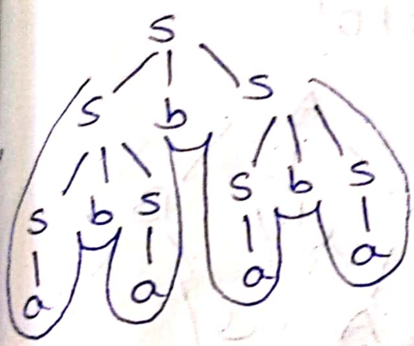
LMD

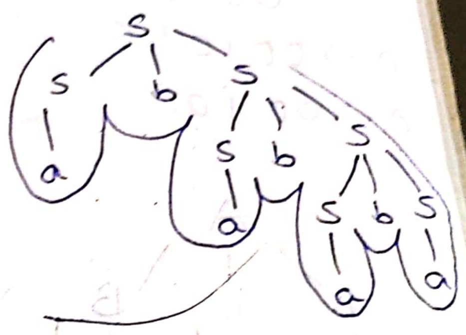
S → SbS
S → a

S → SbS
~~S → SbSbSbS~~
S → SbSbS
S → SbSbSbS
S → abababa

RMD

S → SbS
S → SbSbS
S → SbSbSbS
S → SbSbSba
S → SbSbaba
S → Sbababa
S → abababa





* Construct left most & right most derivation for following strings.

1. 00101
2. 1001
3. 00011

Grammer is

A → A | B
A → 0A | ε
B → 0B | 1B | ε

A: Given grammar G = (V, T, P, S) where

V = {A, B}
T = {0, 1, ε}
S = A

20/8/22

*Const ① LMD 00101

$A \to A1B$

$A \to 0A1B$

$A \to 00A1B$

(20) $A \to 00\epsilon 1B$

$A \to 001B$

$A \to 0010B$

$A \to 00101B$

$A \to 00101\epsilon$

$A \to 00101$

RMD

$A \to A1B$

$A \to A10B$

$A \to A101B$

$A \to A101\epsilon$

$A \to A101$

$A \to 0A101$

$A \to 00A101$

$A \to 00\epsilon 101$

$A \to 00101$

LMD

$A \to A1B$

$A \to \epsilon 1B$

$A \to 1B$

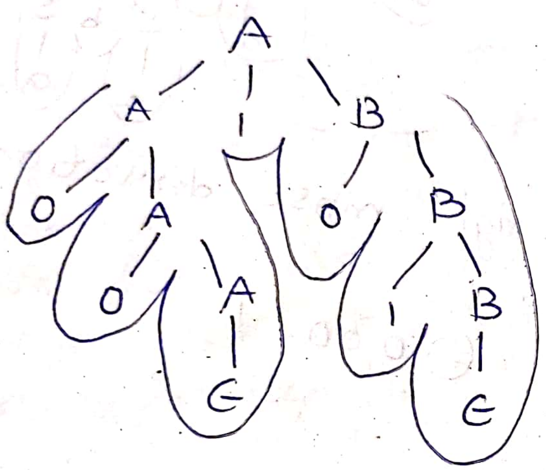
$A \to 10B$

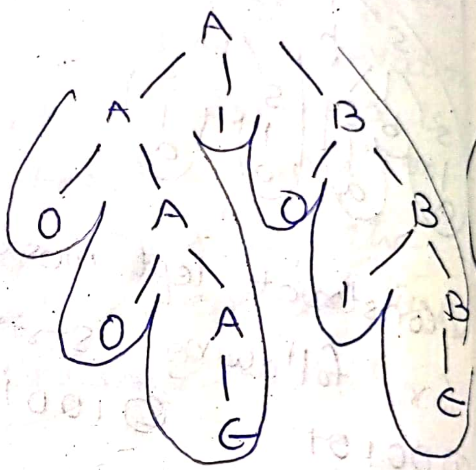
$A \to 100B$

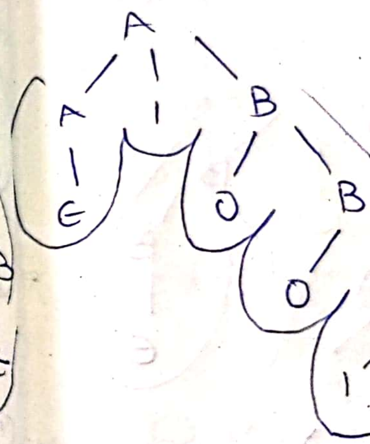
$A \to 1001B$

$A \to 1001\epsilon$

$A \to 1001$







② 1001

$A \to A1B$

$A \to 0A / \epsilon$

$B \to 0B / 1B / \epsilon$

③ 00011

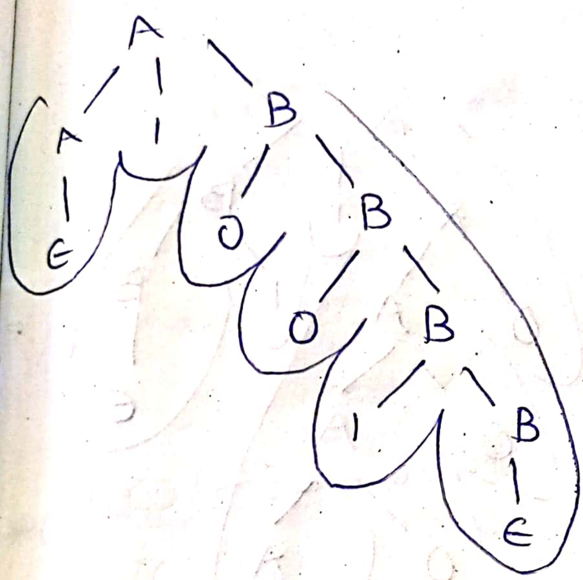
A

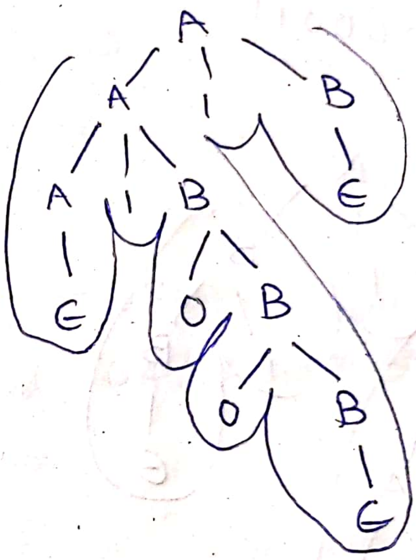
LMD

$A \to A1B$
$A \to \epsilon 1B$
$A \to 1B$
$A \to 10B$
$A \to 100B$
$A \to 1001B$
$A \to 1001\epsilon$
$A \to 1001$

RMD

$A \to A1B$
$A \to A1\epsilon$
$A \to A1$
$A \to A1B1$
$A \to \epsilon 1B1$
$A \to 1B1$
$A \to 10B1$
$A \to 100B1$
$A \to 100\epsilon 1$
$A \to 1001$





(3) 00011

4

20/8/22

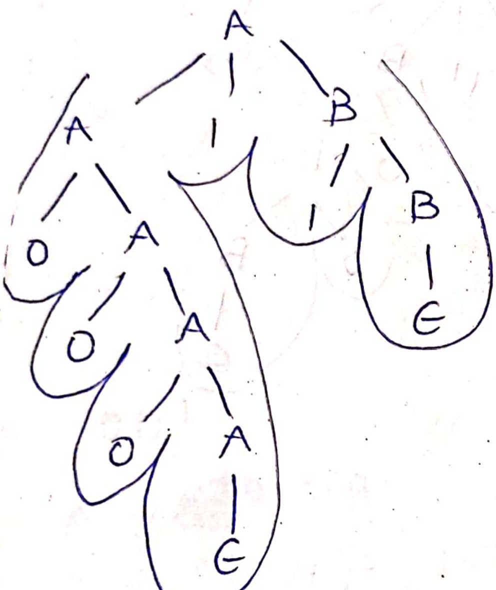
*Consti

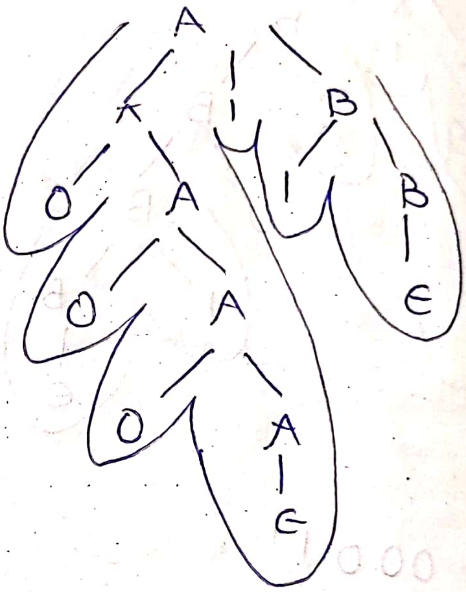
Co

00011

LMD	RMD
$A \to A1B$	$A \to A1B$
$A \to 0A1B$	$A \to A11B$
$A \to 00A1B$	$A \to A11\epsilon$
$A \to 000A1B$	$A \to A11$
$A \to 000\epsilon 1B$	$A \to 0A11$
$A \to 0001B$	$A \to 00A11$
$A \to 00011B$	$A \to 000A11$
$A \to 00011\epsilon$	$A \to 000\epsilon 11$
$A \to 00011$	$A \to 00011$

→ 20





25/8/22

Equivalence between regular grammar and finite automata:-

- For each production of the form $A \to aA_j$, then $s(A_i, a) = A_j$. If the production is $A_i \to a$ then $s(A_i, a) = D$

*Construct FA fo
regular linear

$A_0 \to aA_1$

$A_1 \to bA_1 | b$

Given data G

$V = \{A_0, A_1\}$

$T = \{a, b\}$

$S = A_0$

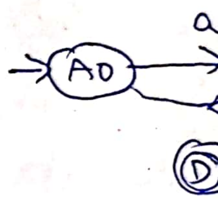
The required FA
A_0, A_1, D. D i

$A_0 \to aA_1$

$s(A_0, a) = A$

$A_1 \to bA_1$

$s(A_1, b) = A_1$



*Construct FA

$S \to 01A$

$A \to 10B$

$B \to 0A$

Given dat

$\therefore V = \{S, A$

$T = \{0$

$S = \{S\}$

$S \to 01A$

$s(S, 01) = A$

$A \to 10B$

$s(A, 10) = B$

*Construct FA for given regular grammar (or) regular linear grammar. The grammar is

$A_0 \to aA_1$

$A_1 \to bA_1 \mid bA_0 \mid a$

Given data $G=(V,T,P,S)$ where

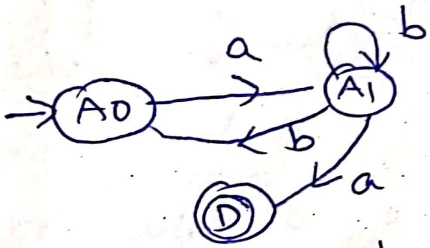
$V=\{A_0, A_1\}$

$T=\{a,b\}$

$S = A_0$

The required FA will have 3 states, those are A_0, A_1, D. D is new state or final state.

$A_0 \to aA_1$	$A_1 \to bA_0$
$\delta(A_0, a) = A_1$	$\delta(A_1, b) = A_0$
$A_1 \to bA_1$	$A_1 \to a$
$\delta(A_1, b) = A_1$	$\delta(A_1, a) = D$



*Construct FA for the given regular grammar.

$S \to 01A$

$A \to 10B$

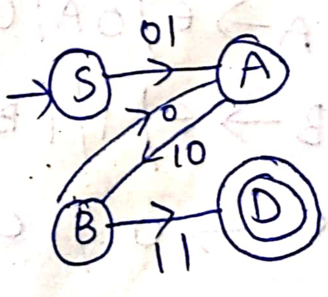
$B \to 0A \mid 11$

Given data is $G=(V,T,P,S)$

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

$T=$ ~~{0,1,0}~~ $\{01, 10, 11, 0\}$

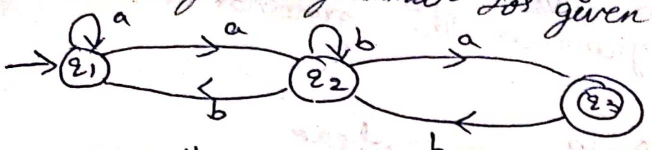
$S=\{S\}$



$S \to 01A$	$B \to 0A$
$\delta(S, 01) = A$	$\delta(B, 0) = A$
$A \to 10B$	$B \to 11$
$\delta(A, 10) = B$	$\delta(B, 11) = D$

20/12/22

* Find a regular grammar for given FA.



Rule ① If there is a transition $\delta(q_i, a) = q_j$ then $q_i \to a q_j$ is the production

Rule ② If the transition is $\delta(q_i, a) = q_j$ (final) then production is $q_i \to a$

Ans

$\delta(q_1, a) = q_1$	$\delta(q_2, b) = q_2$	$\delta(q_2, b) = q_1$
$q_1 \to a q_1$	$q_2 \to b q_2$	$q_2 \to b q_1$
$\delta(q_1, a) = q_2$	$\delta(q_2, a) = q_3$	$\delta(q_3, b) = q_2$
$q_1 \to a q_2$	$q_2 \to a / a q_3$	$q_3 \to b q_2$

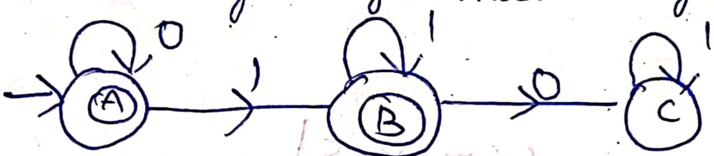
Therefore the grammar is

$q_1 \to a q_1 / a q_2$

$q_2 \to b q_2 / b q_1 / a / a q_3$

$q_3 \to b q_2 / \epsilon$

* Construct regular grammar for given FA.

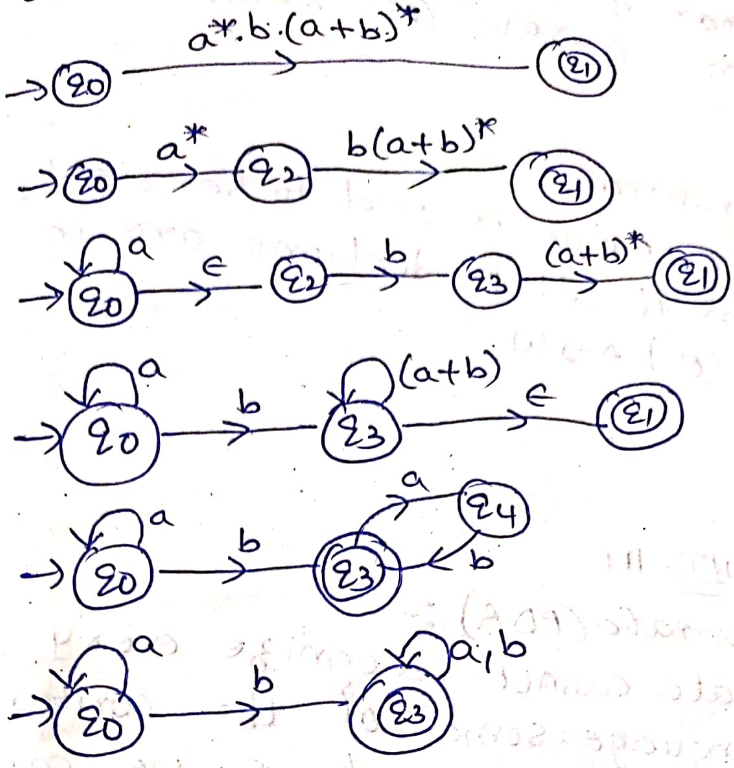


Sol

$A \to \epsilon / 0A / 0 / 1 / 1B$	$\delta(A, 0) = A \Rightarrow A \to 0 / 0A$
$B \to \epsilon / 1 / 1B / 0C$	$\delta(A, 1) = B \Rightarrow A \to 1B / 1$
$C \to 0C / 1C$	$\delta(B, 0) = C \Rightarrow B \to 0C$
	$\delta(B, 1) = B \Rightarrow B \to 1B / 1$
	$\delta(C, 0) = C \Rightarrow C \to 0C$
	$\delta(C, 1) = C \Rightarrow C \to 1C$

* Construct regular grammar for given regular expression.

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



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

$\delta(q_0, a) = q_0$

$q_0 \rightarrow a q_0$

$\delta(q_0, b) = q_3$

$q_0 \rightarrow b \mid b q_3$

$\delta(q_3, a) = q_3$

$q_3 \rightarrow a \mid a q_3$

$\delta(q_3, b) = q_3$

$q_3 \rightarrow b \mid b q_3$

$q_0 \rightarrow a q_0 \mid b \mid b q_3$

$q_3 \rightarrow \epsilon \mid a \mid b \mid a q_3 \mid b q_3$.

- Context free grammar is of 2 types.

1. Left linear grammar
2. Right linear grammar

20/8/22
*Consti -Left linear grammar:-
[Co Context free grammar is said to be left linear grammar if all the productions are in form $A \to BW$ (or) $A \to W$

→(20) Eg $A \to Ba$
$A \to a$

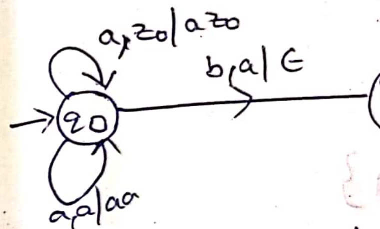
-Right linear grammar:-
Context free grammar is said to be right linear grammar if all productions are in form $A \to wB$ (or) $A \to w$

Ex $A \to Ba$
$A \to a$

26/8/22 UNIT-III

*Push-down Automata(PDA):-
- A finite automata cannot recognize every context-free language. some of the context free languages are not regular. so, we can add some extra features to finite automata to accept any context free language. so, we can say that push down automata is an increment over finite automata.
- A finite automata with stack view as push down automata.
- The operations on stack is based on last in first out.
- The PDA consists of 4 components.
①Input tape ②Read unit ③Control unit ④stack.
- we can define PDA by 7 tuple values. Those are
$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
here
Q = Finite set of states
Σ = input alphabet
q_0 = initial state.
δ = Transition function
Z_0 = Top of stack
F = Final state
Γ = set of symbols put onto stack.

*Designing PDA
*Design a PDA which
$L = \{a^n b^n, n \geq 1\}$
Ans $L = \{ab, aabb,$
aabbB-Blank
$\delta(q_0, a, z_0) = (q_0,$
$\delta(q_0, a, a) = (q_0$
$\delta(q_0, b, a) = (q_1$
$\delta(q_1, b, a) = (q$
$\delta(q_1, \epsilon, z_0) = ($



*Design a PDA whi
$L = \{wcw^R / w \in (a$
Ans $(a,b)^* = \{\epsilon$
w = ab ; w
$L = \{aca, bcb$
a b c b a B-Bla
$\delta(q_0, a, z_0) = ($
$\delta(q_0, b, a) = ($
$\delta(q_1, c, z_0) =$
$\delta(q_2, b, z_0) =$
$\delta(q_3, a, b)$

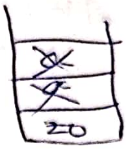
*Designing PDA

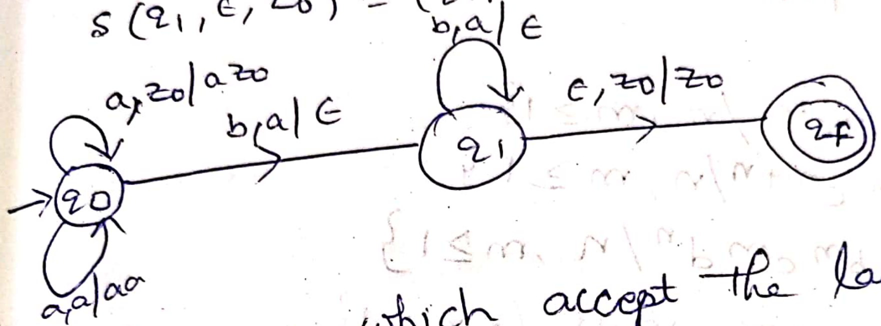
*Design a PDA which accept the language
$L=\{a^n b^n, n \geq 1\}$

Ans $L = \{ab, aabb, aaabbb, \ldots\ldots\}$

aabbB-Blank

$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, a) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$



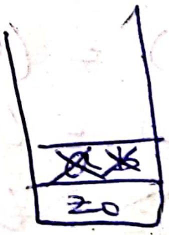


*Design a PDA which accept the language
$L = \{wcw^R \mid w \in (a,b)^*$

Ans $(a,b)^* = \{\epsilon, a, b, aa, ab, bb, ba, \ldots\ldots\}$
$w = ab \quad ; \quad w^R = ba$

$L = \{aca, bcb, aacaa, abcba, bacab, bbcbb, \ldots\ldots\}$

a b c b a B-Blank

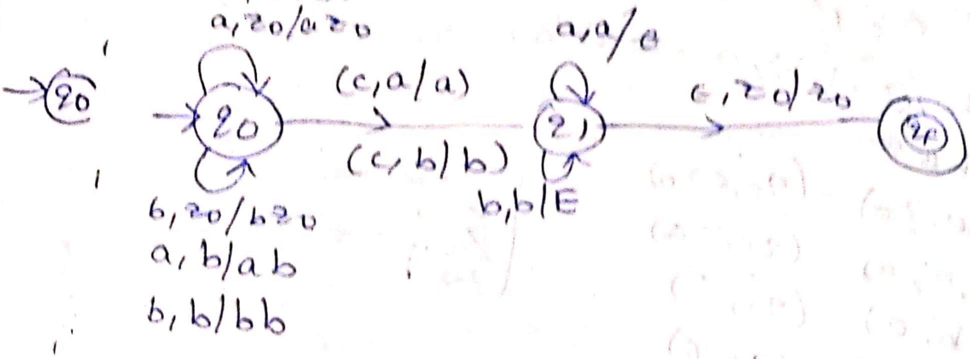


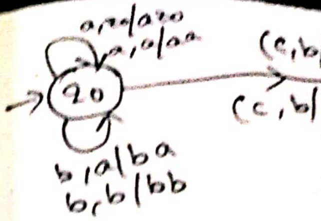
$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, c, z_0) = (q_2, z_0)$
$\delta(q_2, b, z_0) = (q_3, bz_0)$
$\delta(q_3, a, b) = (q_4, \epsilon)$

$\delta(q_4, \epsilon, z_0) = (q_f, z_0)$

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

$L = \{wcw^R\}$





① Design a PDA which accept following language

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

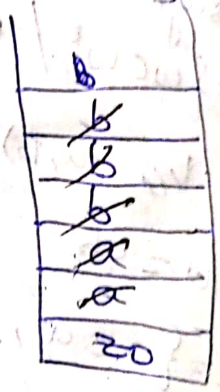
② $L = \{a^m b^n c^{m+n} / n, m \geq 1\}$

③ $L = \{a^n b^m c^m d^n / n, m \geq 1\}$

Sol ③ $L = \{a^n b^m c^m d^n / n, m \geq 1\}$

Let $n = 2$, $m = 3$

$x = $ aabbbcccdd



$\delta(q_0, a, z_0) = (q_0, az_0)$

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

$\delta(q_0, b, a) = (q_0, ba)$

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

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

~~$\delta(q_0, b, b) = (q_0, bb)$~~

$\delta(q_0, c, b) = (q_1, \epsilon)$

$\delta(q_1, c, b) = (q_1, \epsilon)$

$\delta(q_1, c, b) = (q_1, \epsilon)$

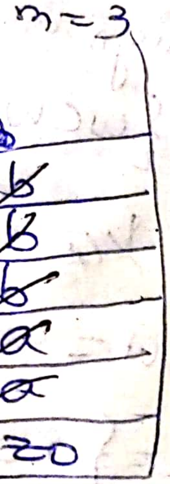
$\delta(q_1, d, a) = (q_1, \epsilon)$

$\delta(q_1, d, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$

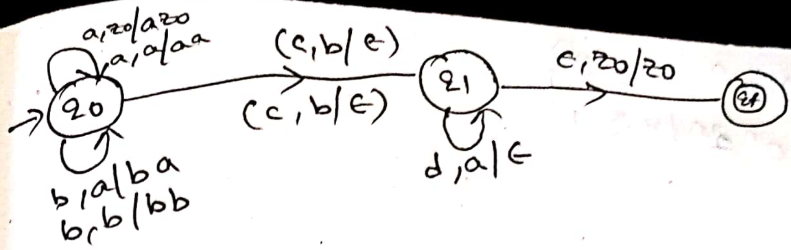
wing

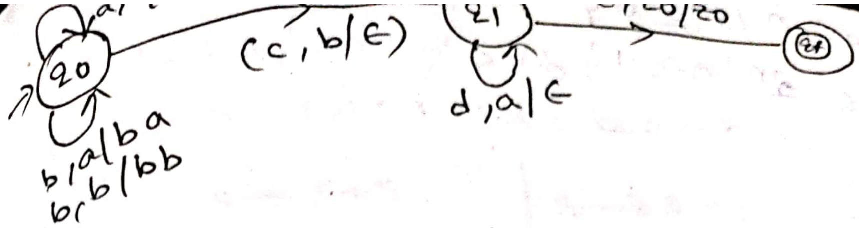
$n = 2$
$m = 3$



$= (q_1, \epsilon)$

$= (q_f, z_0)$

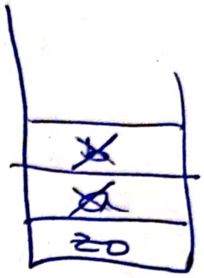




$L = \{ wcw^R : w \in \{a,b\}^* \}$

$L = \{ aca, bcb, abcba, \ldots \}$

abcba



$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_1, b, b) = (q_1, \epsilon)$

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

$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, a) = (q_0, ab)$

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

$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_1, b, b) = (q_1, \epsilon)$

$\delta(q_1, b, b) = (q_1, \epsilon)$

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

$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$ ✓

① $L = \{a^3 b^n c^n / n \geq 0\}$

② $L = \{0^n 1^{2n} / n \geq 1\}$

① $a^n b^n c^m \quad n, m \geq 1$

aabbccc

$\delta(q_0, a, z_0) = (q_0, a)$

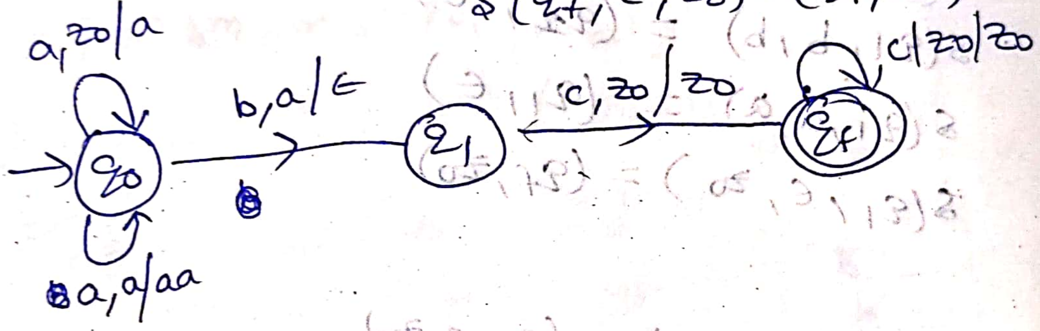
$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, c, z_0) = (q_f, z_0)$

$\delta(q_f, c, z_0) = (q_f, z_0)$



UNIT-II

6/9/22

***ambiguity (or) ambiguous grammar :-**

– The grammar is said to be ambiguous for deriving a terminal, it can produce more than one left most derivation (or) more than one right most derivation (or) more than one derivation tree (syntax tree, Parser tree) then we can say that grammar is ambiguous grammar.

Ex:- S→S+...
S→S+S|...
Derive a s...
Leftmost derivation S→...
S→...
S→...
S→...
S→...
– As there ...
grammar ...
* Test whet...
ambiguou...
S→S...
w = aa...
LMD S→...
S→...
S→...
S→...
S→...
S→...
S→...
⇒ Ambi...
*Test ...
or not...
S→a|...
A→a...

Ex: ~~S ⇒ S+S~~ ~~(or) S*S~~ (or) a (or) b

$S \to S+S \,/\, S*S \,/\, a \,/\, b$

Derive a string $w = a+a*b$

Leftmost derivation	
$S \to S+S$	$S \to S*S$
$S \to a+S$	$S \to S+S*S$
$S \to a+S*S$	$S \to a+S*S$
$S \to a+a*S$	$S \to a+a*S$
$S \to a+a*b$	$S \to a+a*b$

As there is more than one LMD, the given grammar is ambiguous grammar.

* Test whether the given grammar is ambiguous or not.

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

$V = \{S\}$; $T = \{a\}$, $S = \{S\}$

$S \to SS/a$

$w = aaaa$

LMD	
$S \to SS$	$S \to SS$
$S \to SSS$	$S \to aS$
$S \to SSSS$	$S \to aSS$
$S \to aSSS$	$S \to aaS$
$S \to aaSS$	$S \to aaSS$
$S \to aaaS$	$S \to aaaS$
$S \to aaaa$	$S \to aaaa$

⇒ Ambiguous grammar

* Test whether given grammar is ambiguous or not. The grammar is

$S \to a / aAb / abSb$

$A \to aAAb \,|\, bS$

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

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

$T = \{a, b\}$

$w = abab$.

$S \to aAb$.
$S \to absb$
$S \to abab$

$S \to absb$
$S \to abab$.

$\Rightarrow$ Ambiguous grammar.

* **Unambiguous grammar:-** The grammar is said to be unambiguous if it produces only one left most derivation, one right most derivation (or) only one derivation tree.

Ex: ① $S \to A/B$
$A \to aAb/ab$
$B \to abB/\epsilon$

$G = (V, T, P, S)$
$V = \{S, A, B\}$
$T = \{a, b\}$
$S = \{S\}$

$W = aabb$.

$S \to A$
$S \to aAb$
$S \to aabb$

$\Rightarrow$ only 1 LMD. $\Rightarrow$ Unambiguous.

* **Minimization of context free grammar:-**
(or)
(reduced)

- A reduced context free grammar G consists of the following properties.

① we can eliminate null prodictions. The productions are in form $A \to \epsilon$

② Eliminate unit productions. The productions are in the form $A \to B$

③ Eliminate the useless symbols. The variables/terminals not appear in any derivation of the terminal string from the starting symbol.

$S \rightarrow AaB / aaB$

$A \rightarrow D$

$B \rightarrow bbA / \epsilon$

$D \rightarrow E$

$E \rightarrow F$

$F \rightarrow as$

Minimize (or) reduce the context free grammar.

step ①:- The given grammar will have only one null production i.e., $B \rightarrow \epsilon$. So, we have to eliminate the null production.

As B is present in $S \rightarrow AaB / aaB$

$S \rightarrow Aa\epsilon / aa\epsilon$

$S \rightarrow Aa / aa$

∴ The grammar after eliminating null production is:-

$S \rightarrow AaB / aaB / aa / Aa$

$A \rightarrow D$

$B \rightarrow bbA$

$D \rightarrow E$

$E \rightarrow F$

$F \rightarrow as$

step ②:- In the grammar, there are 3 unit productions those are:-

$A \rightarrow D$

$D \rightarrow E$

$E \rightarrow F$

We can eliminate the unit productions

$E \rightarrow F$

$E \rightarrow as$

$D \rightarrow E$

$D \rightarrow as$

$A \rightarrow D$

$A \rightarrow as$

The grammar is

$S \rightarrow AaB / aaB / aa / Aa$

$A \rightarrow as$

$B \rightarrow bbA$

$D \rightarrow as$

$E \rightarrow as$

$F \rightarrow as$

step ③ We can eliminate useless symbols in the above grammar. D, E, F are useless symbols. So we can eliminate those symbols from grammar. Hence, the minimized grammar is

$S \to AaB / aaB / aa / Aa$

$B \to bbA$

$A \to aS$

*Reduce the context free grammar. The grammar is

$S \to AB$

$A \to a$

$B \to C$

$C \to D$

$D \to b$

Ans step ① The given grammar has no null productions. So, there is no change in grammar at this step

step ② The unit productions in given grammar are $B \to C$, $C \to D$

$C \to b$

$B \to b$

The grammar is:

$S \to AB$

$A \to a$

$B \to b$

$C \to b$

$D \to b$

step ③: The useless symbols in above grammar are C, D. So, we can eliminate them.

$S \to AB$; $A \to a$; $B \to b$

The minimize
$S \to AB$
$A \to a$
$B \to b$.

*Minimize th
① $S \to aA | bB$
$A \to aA | a$
$B \to bB$
$D \to aa | E$
$E \to ac | d$

Ans ① step ① The
step ② Th
step ③ T
So, we c
$S \to aA$
$A \to aA$
$B \to b$
The minim
$S \to aA$
$A \to a$
$B \to b|$

② step ① T
step ②
step ③
elimina

The min
$S \to aS |$
$A \to a$
$C \to ac$

the minimized context free grammar is

S → AB
A → a
B → b.

*Minimize the context free grammars

① S → aA/bB
A → aA/a
B → bB
D → aa/Ea
E → ac/d

② S → as/AC
A → a
B → aa
C → acd

① step ① There are no null productions.
step ② There are no unit productions.
step ③ The useless symbols are D, E.
So, we can eliminate them.

S → aA/bB
A → aA/a
B → bB

the minimized context free grammar is

S → aA/bB
A → aA/a
B → bB.

② step ① there are no null productions.
step ② There are no unit productions
step ③ The useless symbol is B. So, we eliminate it.

S → as/AC
A → a
C → acd

the minimized context free grammar is

S → as/AC
A → a
C → acd

*Normal forms:-

- The normal forms are classified into two types.

① Chomsky normal form (CNF) ② Gribach normal form (GNF)

- **CNF**: The context free grammar is in CNF, the productions are at the form

$A \to AB$ (or) $A \to a$ where A, B are variables and a is terminal.

- the CNF restrict the no. of symbols on the right hand side of the production to be two.

CNF procedure

For every context free grammar there is an equivalent grammar G in CNF.

Proof: step ① Eliminate the null productions, unit productions and useless symbols from context free grammar.

step ② Eliminate terminals on right hand side of production as follows:-

① All productions in P of the form of $A \to a, A \to AB$

② Consider $A \to x_1, x_2, \ldots x_n$ will be some terminals on right hand side of the production, then add a new variable C_{a_i} i.e., $C_{a_i} \to a_i$

Repeat a same for all terminals.

step ③ Restricting no. of variables on right hand side as follows.

① All productions in P are added to P' if they are in the regular form

Consider ~~$A \to a$~~ $A \to A_1, A_2, \ldots A_n$ where

$n \geq 3$ then we can introduce a new productions are:-

$A \rightarrow A_1 C_1$

$C_1 \rightarrow A_2 C_2$

$C_2 \rightarrow A_3 C_3$

Therefore the given grammar is in CNF form.

*Find the equivalent grammar in CNF for the grammar

$S \rightarrow aAbB$

$A \rightarrow aA/a$

$B \rightarrow bB/b.$

Given grammar is $G=(V,T,P,S)$

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

$T = \{a, b\}$

$S = \{S\}$

step ① There are no null productions, unit productions & useless symbols

step ② Eliminate the terminals on RHS of production.

$S \rightarrow aAbB$ ×

$A \rightarrow aA$ ×

$A \rightarrow a$ ✓

$B \rightarrow bB$ ×

$B \rightarrow b$ ✓

$A \rightarrow AB$

$A \rightarrow a$

$S \rightarrow XAYB$

$A \rightarrow XA$

$A \rightarrow a$

$B \rightarrow YB$

$B \rightarrow b$

$X \rightarrow a$

$Y \rightarrow b$

step ③ Restricting no. of variables on RHS.

$S \rightarrow ZQ$

$A \rightarrow XA$

$A \rightarrow a$

$B \rightarrow YB.$

$B \rightarrow b$

$X \rightarrow a$

$Y \rightarrow b$

$Z \rightarrow XA$

$Q \rightarrow YB$

∴ The CNF of given context free grammar
$G' = (V', T', P', S')$ where
$V' = \{S, A, B, X, Y, Z, Q\}$
$T' = \{a, b\}$
$S' = \{S\}$
$P' = \{ S \to ZQ$
$A \to XA$
$A \to a$
$B \to YB$
$B \to b$
$X \to a$
$Y \to b$
$Z \to XA$
$Q \to YB \}$

* Find out equivalent CNF for given grammar

$S \to aB / bAB$
$B \to b$
$D \to d$

Given grammar $G = (V, T, P, S)$
$V = \{S, B, A, D\}$ $S = \{S\}$
$T = \{a, b, d\}$

Ans step ① There are no null productions, unit productions. There is one useless symbol D. So, we eliminate them.

$S \to aB / bAB$
$B \to b$

step②
$S \to aB$ ×
$S \to bAB$ ×
$B \to b$ ✓

$S \to XB$
$S \to YAB$
$B \to b.$
$X \to a$
$Y \to b$

step ② $S \to XB$

$S \to ZB$

$B \to b$

$X \to a$

$Y \to b$

$Z \to YA$

∴ The CNF of given context free grammar

$G' = (V', T', P', S')$

where $V' = \{S, B, X, Y, Z\}$

$T' = \{a, b\}$

$S' = \{S\}$

$P' = \{ S \to XB$

$S \to ZB$

$B \to b$

$X \to a$

$Y \to b$

$Z \to YA \}$

* Find equivalent CNF form of grammar

$S \to abAB$

$A \to bAB \mid \epsilon$

$B \to Baa \mid A \mid \epsilon$

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

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

$T = \{a, b\}$

$S = \{S\}$

step ① There are null productions, unit productions

$A \to \epsilon$, $B \to \epsilon$.

$S \to ab\epsilon B$	$S \to abA\epsilon$	$S \to abAB \mid abB \mid abA \mid ab$
$S \to abB$	$S \to abA$	$A \to bAB \mid bA \mid bB \mid b$
		$B \to Baa \mid aa$

step ③

$S \to abAB$ ×
$S \to abB$ ×
$S \to abA$ ×
$S \to ab$ ×
$A \to bAB$ ×
$A \to bA$ ×
$A \to bB$ ×
$A \to b$ ✓
$B \to Baa$ ×
$B \to aa$ ×

$S \to XYAB$
$X \to a$ ✓
$Y \to b$ ✓
$S \to XYB$
$S \to XYA$
$S \to XY$ ✓
$A \to YAB$
$A \to YA$ ✓
$A \to YB$ ✓
$A \to b$ ✓
$B \to BXX$
$B \to XX$ ✓

step ④

$S \to ZQ$
$Z \to XY$
$Q \to AB$
$X \to a$
$Y \to b$
$S \to ZB$
$S \to ZA$
$S \to XY$
$A \to YQ$
$A \to YA$
$A \to YB$
$A \to b$
$B \to RX$
$R \to BX$
$B \to XX$

$G' = (V', T', P', S'$
$V' = \{S, A, B,$
$T' = \{a, b\}$
$S' = \{S\}$
$P' = \{ S \to ZQ$
$Z \to X$
$Q \to A$
$X \to$
$Y \to$
$S \to$
$S \to$
$S \to$
$A \to$
$A \to$
A
A
B
R

* Find equiv
$S \to ABA$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$

Sol $G = (V, T$
$V = \{S,$
$T = \{a$
$S = \{$

Step① Ther
$B \to$

step ③

S → abAB ×
S → abB ×
S → abA ×
S → ab ×
A → bAB ×
A → bA ×
A → bB ×
A → b ✓
B → Baa ×
B → aa ×

S → XYAB
X → a ✓
Y → b ✓
S → XYB
S → XYA
S → XY ✓
A → YAB
A → YA ✓
A → YB ✓
A → b ✓
B → BXX
B → XX ✓

step ④

S → ZQ
Z → XY
Q → AB
X → a
Y → b
S → ZB
S → ZA
S → XY
A → YQ
A → YA
A → YB
A → b
B → RX
R → BX
B → XX

G' = (V', T', P', S')
V' = {S, A, B, Z
T' = {a, b}
S' = {S}
P' = { S → ZQ
Z → XY
Q → AB
X → a
Y → b
S → Z
S → Z
S →
A →
A →
A →
A →
B →
R →
B →

* Find equivale
S → ABA
A → aA/ε
B → bB/ε

Step G = (V, T,
V = {S, A,
T = {a,
S = {S}

Step① There
B → ε

$G' = (V', T', P', S')$

$V' = \{S, A, B, Z, Q, X, Y, R\}$

$T' = \{a, b\}$

$S' = \{S\}$

$P' = \{ S \to ZQ$

$Z \to XY$

$Q \to AB$

$X \to a$

$Y \to b$

$S \to ZB$

$S \to ZA$

$S \to XY$

$A \to YQ$

$A \to YA$

$A \to YB$

$A \to b$

$B \to RX$

$R \to BX$

$B \to XX \}$

* Find equivalent CNF of grammar

$S \to ABA$

$A \to aA / \epsilon$

$B \to bB / \epsilon$

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

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

$T = \{a, b\}$

$S = \{S\}$

Step① There are two null productions $A \to \epsilon$, $B \to \epsilon$

$S \to ABA$
$A \to aA / \epsilon$
$B \to bB / \epsilon$.

$S \to \epsilon BA$	$S \to A\epsilon A$	$S \to AB\epsilon$	$S \to \epsilon\epsilon A$
$S \to BA$	$S \to AA$	$S \to AB$	$S \to A$

$S \to A\epsilon\epsilon$	$S \to \epsilon B\epsilon$
$S \to A$	$S \to B$.

step② $S \to \epsilon$

$S \to ABA$ ✗ $\quad A \to aA$ ✗

$S \to BA$ ✓ $\quad A \to a$

$S \to AA$ ✓ $\quad B \to bB$ ✗

$S \to AB$ ✓ $\quad B \to b$

$S \to A$ ✗

$S \to B$ ✗ need to be removed

$P' = \{$

$S \to \epsilon$

$S \to XA$

$X \to AB$

$S \to BA$

$S \to AA$

$S \to AB$

$A \to YA$

$Y \to a$

$A \to a$

$B \to ZB$

$Z \to b$

$B \to b. \}$

$G' = (V', T', P', S')$

$V' = \{S, X, A, Y, B, Z\}$

$T' = \{a, b\}$

$S' = \{S\}$

Convert conte

① $S \to asa$

step① Th

unit

step② $S \to$

$S \to XSX$

$X \to a$ ✓

~~XAXB~~.

$S \to YSY$

$Y \to b$ ✓

$S \to a$ ✓

$S \to b$ ✓

step③:

$S \to ZX$

$Z \to XS$

$X \to a$

$S \to PY$

$P \to YS$

$Y \to b$

$S \to a$

$S \to b$

Is the CNF

$\rightarrow \epsilon \in A$

$\rightarrow A$

Convert context free grammar into CNF form.

① $S \rightarrow aSa / bSb / a / b$

② $S \rightarrow aB \mid bA$

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

$B \rightarrow b \mid aS \mid aBB$

Ans ① step ① There are no null productions, unit productions & useless symbols.

step ②

$S \rightarrow aSa$ ×

$S \rightarrow bSb$ ×

$S \rightarrow a$ ✓

$S \rightarrow b$ ✓

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

$V = \{S\}$

$T = \{a, b\}$

$S = \{S\}$

$S \rightarrow XSX$

$X \rightarrow a$ ✓

$S \rightarrow YSY$

$Y \rightarrow b$ ✓

$S \rightarrow a$ ✓

$S \rightarrow b$ ✓

$, B, Z\}$

step ③:

$S \rightarrow ZX$

$Z \rightarrow XS$

$X \rightarrow a$

$S \rightarrow PY$

$P \rightarrow YS$

$Y \rightarrow b$

$S \rightarrow a$

$S \rightarrow b$

$G' = (V', T', P', S')$

$V' = \{S, X, Y, Z, P\}$

$T' = \{a, b\}$

$S' = \{S\}$

$P' = \{S \rightarrow ZX$

$Z \rightarrow XS$

$X \rightarrow a$

$S \rightarrow PY$

$P \rightarrow YS$

$S \rightarrow a$

$S \rightarrow b\}$

Is the CNF form of context free grammar.

→ step ① there are no null productions, unit productions and useless symbols

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

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

$T = \{a, b\}$

$S = \{S\}$

step ①

$S \to aB$
$S \to bA$
$A \to a$
$A \to aS$
$A \to bAA$
$B \to b$
$B \to bS$
$B \to aBB$

$S \to XB$
$X \to a$
$S \to YA$
$Y \to b$
$A \to a$
$A \to XS$
$A \to YAA$
$B \to b$
$B \to YS$
$B \to XBB$

step ③

$S \to XB$
$X \to a$
$S \to YA$
$Y \to b$
$A \to a$
$A \to XS$
$A \to ZA$
$Z \to YA$
$B \to b$
$B \to XS$
$B \to PB$
$P \to XB$

$G' = (V', T', P', S')$

$V' = \{S, X, A, B, Y, Z, P\}$

$T' = \{a, b\}$

$S' = \{S\}$

step① There are no null productions, unit productions and useless symbols.

$G = (V, T, P, S)$
$V = \{S, A, B\}$
$T = \{a, b\}$
$S = \{S\}$

step① $S \to aB$
$S \to bA$
$A \to a$ ✓
$A \to aS$
$A \to bAA$
$B \to b$ ✓
$B \to aS$
$B \to aBB$

$S \to XB$ ✓
$X \to a$ ✓
$S \to YA$ ✓
$Y \to b$ ✓
$A \to a$ ✓
$A \to XS$ ✓
$A \to YAA$ ×
$B \to b$ ✓
$B \to XS$ ✓
$B \to XBB$ ×

step③ $S \to XB$
$X \to a$
$S \to YA$
$Y \to b$
$A \to a$
$A \to XS$
$A \to ZA$
$Z \to YA$
$B \to b$
$B \to XS$
$B \to PB$
$P \to XB$

$G' = (V', T', P', S')$
$V' = \{S, X, A, B, Y, Z, P\}$
$T' = \{a, b\}$
$S' = \{S\}$

Here, it is in CNF fo[rm]

$P' = \{S \to XB$
$X \to a$
$S \to YA$
$Y \to b$
$A \to a$
$A \to XS$
$A \to ZA$
$Z \to YA$
$B \to b$
$B \to XS$
$B \to PB$
$P \to XB\}$

9/9/22
GNF (Gribach no[rmal form])
- The context free[e] ... productions are ...
$A \to a$

Lemma① - $A \to B\alpha$
$B \to \beta_1 |$ ...
if all production[s] ...
$A \to \beta_1 \alpha | \beta_2 \alpha |$ ...

Ex:- $A \to Aba | c$ ...
$A \to Bba$
$B \to A | CD |$ ...

Lemma② - If
$A \to A\alpha_1 |$ ...
$A \to \beta_1 |$ ...
then
A ...
A ...

Here, it is in CNF form.

$P' = \{ S \to XB$
$X \to a$
$S \to YA$
$Y \to b$
$A \to a$
$A \to XS$
$A \to ZA$
$Z \to YA$
$B \to b$
$B \to XS$
$B \to PB$
$P \to XB \}$

9/9/22

*GNF (Gribach normal form):-

- The context free grammar is in GNF if all productions are at the form $A \to a\alpha$ (or) $A \to a$

Lemma① - $A \to B\alpha$
$B \to \beta_1 / \beta_2 / \ldots / \beta_n$

if all productions are,
$A \to \beta_1 \alpha / \beta_2 \alpha / \beta_3 \alpha / \ldots / \beta_n \alpha$

Ex:- $A \to Aba / cDba / zba / Fba \ldots$

$A \to Bba$
$B \to A / cD / z / F \ldots$

Lemma② - If the productions are in the form
$A \to A\alpha_1 / A\alpha_2 \ldots A\alpha_n$
$A \to \beta_1 / \beta_2 \ldots \beta_n$

then
$A \to \beta_i$
$A \to \beta_i z$

$z \to \alpha_i$
$z \to \alpha_i z$

$S \to Sa/Sab/SBA/SAaB/aA/aD/d/aAB$

$S \to aA/aD/d/aAB$

$S \to aAZ/aDZ/dZ/aABZ$

$Z \to a/ab/BA/AaB$

$Z \to aZ/abZ/BAZ/AaBZ$

*Find GNF grammar equivalent to the context free grammar. The grammar is

$S \to AA/0$

$A \to SS/1$

Given grammar is $G=(V,T,P,S)$

$V=\{S,A\}$

$T=\{0,1\}$

$S=\{S\}$

step① The grammar is already in CNF form.

step② Rename the variables (or) non-terminals

$S=A_1$

$A=A_2$

$A_1 \to A_2A_2/0$

$A_2 \to A_1A_1/1$

step③ Select production in which subscript values on LHS > RHS subscript value. Choose production

$A_2 \to A_1A_1/1$

$A_2 \to A_2A_2A_1/0A_1/1$

$A_2 \to 0A_1/1$

$A_2 \to 0A_1Z/1Z$

$Z \to A_2A_1$

$Z \to A_2A_1Z$

$Z \to 0A_1A_1/1A_1$

$Z \to 0A_1A_1Z/1A_1Z$

Therefore GNF for the

$A_1 \to 0A_1A_2/1A_2/0$

$A_1 \to 0A_1A_2/1A_2/0$

~~$A_1 \to A_2A_2$~~

$A_2 \to 0A_1/1/0A$

$Z \to 0A_1A_1/1A_1$

*Convert the follow... the grammar is

$S \to AB$

$A \to BS/b$

$B \to SA/a$

Given grammar

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

$T=\{a,b\}$

$S=\{S\}$

step① The give...

step② Rename

$S=A_1$

$A=A_2$

$B=A_3$

$A_1 \to A_2A_3$

$A_2 \to A_3A_1$

$A_3 \to A_1A$

step③ choos... values on

$A_3 \to A_1A$

$A_3 \to A_2A$

$A_3 \to A_3$

therefore GNF for the given context-free grammar

$A_1 \rightarrow 0A_1A_2 / 1A_2 / 0A_1ZA_2 / 1ZA_2 / 0$

$A_1 \rightarrow 0A_1A_2 / 1A_2 / 0A_1ZA_2 / 1ZA_2 / 0$

~~$A_1 \rightarrow A_2A_2$~~

$A_2 \rightarrow 0A_1 / 1 / 0A_1Z / 1Z$

$Z \rightarrow 0A_1A_1 / 1A_1 / 0A_1A_1Z / 1A_1Z$

*Convert the following grammar into GNF.

the grammar is

$S \rightarrow AB$

$A \rightarrow BS/b$

$B \rightarrow SA/a$

As Given grammar is $G=(V,T,P,S)$

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

$T=\{a,b\}$

$S=\{S\}$

step① the given grammar is in CNF.

step② Rename variables (or) non-terminals.

$S = A_1$

$A = A_2$

$B = A_3$

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 / b$

$A_3 \rightarrow A_1A_2 / a$

step③ choose production in which subscript values on LHS>RHS.

$A_3 \rightarrow A_1A_2 / a$

$A_3 \rightarrow A_2A_3A_2 / a$

$A_3 \rightarrow A_3 \underbrace{A_1A_3A_2}_{\alpha_1} / \underbrace{bA_3A_2}_{\beta_1} / \underbrace{a}_{\beta_2}$

$\underset{A}{A_3} \rightarrow \underset{A}{A_3}$

$A_3 \rightarrow bA_3A_2 / a$

$A_3 \rightarrow bA_3A_2Z / aZ$

$Z \rightarrow A_1A_3A_2$ ×

$Z \rightarrow A_1A_3A_2Z$ ×

$Z \rightarrow A_1A_3A_2$

$Z \rightarrow A_2A_3A_3A_2$

$Z \rightarrow A_3A_1A_3A_3A_2 \cancel{A_1Z}$

$Z \rightarrow bA_3A_2A_1A_3A_3A_2$

$Z \rightarrow A_1A_3A_2Z$

$Z \rightarrow bA_3A_2A_1A_3A_3A_2Z$

$A_2 \rightarrow A_3A_1 / b$

$A_2 \rightarrow bA_3A_2A_1 / b$

$\cancel{Z \rightarrow bA_3A_2A_1Z / bA_3Z}$

$A_1 \rightarrow A_2A_3$

$A_1 \rightarrow bA_3A_2A_1A_3 / bA_3$

$\cancel{Z \rightarrow bA_3A_2A_1A_3Z / bA_3Z}$

$G' = (V', T', P', S')$

$\cancel{V' = \{A_1, A_2, A_3\}}$

$T = \{a, b\}$

*Construct GNF for equivalent CNF grammar

$E \rightarrow E+T / a / (E)$

$T \rightarrow a / (E)$

Sol

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

$V = \{E, T\}$

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

$S = \{E\}$

step① given

i.e.) $E \rightarrow$

$E \rightarrow ($

T

a) $E \rightarrow EAT$

$A \rightarrow +$

$E \rightarrow BE$

$B \rightarrow ($

$G \rightarrow)$

$T \rightarrow B$

The CN

$E \rightarrow D$

$D \rightarrow E$

$A \rightarrow$

$E \rightarrow$

$F \rightarrow$

$B \rightarrow$

$G \rightarrow$

$T \rightarrow$

$E \rightarrow$

$T \rightarrow$

Now it

step②

E

step ① Given grammar is not in CNF form.

i.e) $E \to E+T$

$E \to (E)$

$T \to (E)$ are not in CNF.

a) $E \to EAT$

$A \to +$

$E \to BEG$

$B \to ($

$G \to)$

$T \to BEG$

b) $E \to DT$

$D \to EA$

$A \to +$

$E \to FG$

$F \to BE$

$B \to ($

$G \to)$

$T \to FG$

The CNF is

$E \to DT$

$D \to EA$

$A \to +$

$E \to FG$

$F \to BE$

$B \to ($

$G \to)$

$T \to FG$

$E \to a$

$T \to a$

Now it is in CNF form.

step ② Rename variables/Non-terminals

$E = A1$

$D = A2$

$T = A3$

$A = A4$

$F = A5$

$G = A6$

$B = A7$

Scanned with CamScanner

step ③ $A_1 \to A_2A_3 / A_5A_6 / a$

$A_2 \to A_1A_4$

$A_4 \to +$

$A_5 \to A_7A_1$

$A_7 \to ($

$A_6 \to)$

$A_3 \to A_5A_6 / a$

step ④ choose production in which subscript on LHS > RHS.

$A_2 \to A_1A_4$

$\underset{A}{A_2} \to \underset{A}{A_2}\underset{\alpha}{A_3A_4} / \underbrace{A_5A_6A_4}_{\beta_1} / \underbrace{aA_4}_{\beta_2}$

~~$A_2 \to A_3A_4 / A_5A_6A_4 / aA_4$~~

~~$A_2 \to A_3A_4Z$~~

$\boxed{A_2 \to aA_4 \quad A_2 \to aA_4Z}$

$A_2 \to A_5A_6A_4 / aA_4$ | $Z \to A_3A_4$

$A_2 \to A_5A_6A_4Z / aA_4Z$ | $Z \to A_3A_4Z$

$A_2 \to A_7A_1A_6A_4 / aA_4$

$A_2 \to (A_1A_6A_4 / aA_4 / (A_1A_6A_4Z / aA_4Z$

- $A_4 \to +$
- $A_7 \to ($
- $A_6 \to)$

$A_1 \to A_1A_4A_3 / A_7A_1A_6 / a$

- $A_1 \to aA_4A_3 / (A_1A_6 / a /$

$A_3 \to A_7A$

$A_3 \to (A_1$

$A_5 \to (A_1$

$G' = (V', T',$

$V' = \{A_1, A_2$

$T' = \{a, +$

$S' = \{A_1\}$

14/9/22

* closure pr
1) Context fre union (or)
2) Context f Kleen clo
3) Context substitut
4) Context intersec

pumping

* show th language

Note:-

① $|VY|$ where

3) $L = \{$

step ① A n r

step ② s divid

$w =$

$A_3 \to A_7 A_1 A_6 / a$

$A_3 \to (A_1 A_6 / a$

$A_5 \to (A_1$

$G' = (V', T', P', S')$

$V' = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$

$T' = \{a, +, (,)\}$

$S' = \{A_1\}$

14/9/22

* **closure properties of context free language(or) grammar:-**

1. Context free languages are closed under union (or) concatenation.
2. Context free languages are closed under Kleen closure.
3. Context free languages are closed under substitution
4. Context free languages are not closed under intersection (or) complimentation.

pumping lemma:-

* show that the language is not context free language. The language $L = \{a^i b^i c^i / i \geq 1\}$

Note:- $w = uv^i x y^i z$

① $|vy| > 0$ ② $|vxy| \leq n$

where $uv^i x y^i z \in L \ \forall i \geq 0$

Sol $L = \{abc, aabbcc, aaabbbccc, \ldots\}$ a context free language,

Step ① Assume that L is n no: of states in machine M.

step ② select a string w then the string is dividing into 5 parts i.e., $uv^i x y^i z$

$w = aabbcc$

ch subscript

$|a A_4 z$

where $U = a$

$V = ab$

$X = b$

$Y = c$

$Z = c$

$\omega = u v^i x y^i z$

① $|vy| > 0$ ② $|vxy| \leq n$

$u v^i x y^i z \in L \quad \forall i \geq 0$

step ③: we can take a suitable integer value for i. where $i = 1$

$uvxyz = aabbcc \in L$

$i=2$ $uv^2 x y^2 z = aababbccc \notin L$

∴ Given language is not context free language

* show that the given language is not context free language. $L = a^p$ where p is prime no.

Sol $L = \{aa, aaa, aaaaa, aaaaaaa, ---\}$

step ① Assume that L is context free language. n = no. of states in machine M.

step ② select a string w then string is divided into 5 parts i.e, $u v^i x y^i z$

$w = aaaaaaa$

$u = a$, $v = aa$, $x = aa$, $y = a$, $z = a$

$w = u v^i x y^i z$

① $|vy| > 0$

② $|vxy| \leq n$

$^i x y^i z \in L \quad \forall i \geq 0$

step for ③ We can take a suitable integer value i.

$i=1$ $uvxyz = aaaaaaa \in L$

$i=2$ $uv^2\cancel{x}y^2z = aaaaaaaaaa \notin L$

∴ Given language is not context free language.

UNIT-III

* Language accepted by PDA:-

- There are two ways to describe the acceptance of PDA.

① Language accepted by empty stack.

② PDA enter into the final state.

* Instantaneous description of PDA :- (ID)

- The ID of PDA is defined as tuples (q, w, s) where q is the current state, w is remaining input string, s is current stack content.

- It uses the symbol "$\vdash$" to represent.

* Conversion of context free grammar to PDA:-

* Construct PDA to accept the strings generated by context free grammar. Context free grammar is

$S \rightarrow aABC$

$A \rightarrow aB/C$

$B \rightarrow bA/b$

$C \rightarrow a$

Sol The initial state move of PDA is to place S on the top of the stack and move to q_1 state.

i.e., $\delta(q_0, \epsilon, z_0) = (q_1, S, z_0)$

NOTE $\delta(q, a, S) = (q, \alpha)$ (where $S \rightarrow a\alpha$)

$S \rightarrow aABC$

$\delta(q_1, a, S) = (q_1, ABC)$

Scanned with CamScanner

$A \to aB$

$\delta(q_1, a, A) = (q_1, B)$

$A \to \epsilon C$

$\delta(q_1, \epsilon, A) = (q_1, C)$

$B \to bA$

$\delta(q_1, b, B) = (q_1, A)$

$B \to b\epsilon$

$\delta(q_1, b, B) = (q_1, \epsilon)$

$C \to a\epsilon$

$\delta(q_1, a, C) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$

string: aabba

LMD

$S \to aABC$

$S \to aaBBC$

$S \to aabBC$

$S \to aabbC$

$S \to aabba$

(q, w, s)

$\vdash (q_1, aabba, Sz_0)$

$\vdash (q_1, abba, ABCz_0)$

$\vdash (q_1, bba, BBCz_0)$

$\vdash (q_1, ba, BCz_0)$

$\vdash (q_1, a, Cz_0)$

$\vdash (q_1, \epsilon, \epsilon z_0)$

$\vdash (q_1, \epsilon, z_0)$

$\vdash (q_f, z_0)$

∴ string is accepted.

* Convert PDA to context free grammar.
* Context free grammar which accept PDA[N(A)]

$A = \{(q_0, q_1), (a, b), (z_0, z), \delta, q_0, z_0, \phi\}$

$\delta(q_0, b, z_0) = \{q_0, zz_0\}$

$\delta(q_0, \Lambda, z_0) = \{q_0, \Lambda\}$

$\delta(q_0, b, z) = \{q_0, zz_0\}$

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

$\delta(q_1, b, z) = \{q_1, \Lambda\}$

$\delta(q_1, a, z_0) = \{q_0, z_0\}$

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

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

$V = \{S, (q_0, z_0, q_0) (q_0, z_0, q_1), (q_0, z, q_0), (q_0, z, q_1), (q_1, z_0, q_1), (q_1, z_0, q_0), (q_1, z, q_1), (q_1, z, q_0)\}$

$S = S$

$(q_0, q_1), (z, z_0)$

P :

$P_1 \rightarrow (q_0, z_0, q_0)$

$P_2 \rightarrow (q_0, z_0, q_1)$

$\delta(q_0, b, z_0) = \{q_0, zz_0\}$

$P_3 - (q_0, z_0, q_0) \rightarrow b (q_0, z, q_0)(q_0, z_0, q_0)$

$P_4 - (q_0, z_0, q_0) \rightarrow b (q_0, z, q_1)(q_1, z_0, q_0)$

$P_5 - (q_0, z_0, q_1) \rightarrow b (q_0, z, q_0)(q_0, z_0, q_1)$

$P_6 - (q_0, z_0, q_1) \rightarrow b (q_0, z, q_1)(q_1, z_0, q_1)$

$\delta(q_0, \Lambda, z_0) = \{q_0, \Lambda\}$

$\delta(q_0, \Lambda, z_0) = \{q_0, \Lambda\}$

$P_{13} : (q_0, z_0, q_0) \rightarrow \Lambda$ ~~(q_0, Λ, q_0)~~

$P_{14} : (q_0, z_0, q_1) \rightarrow \Lambda$

$\delta(q_1, b, z) = \{q_1, \Lambda\}$

$P_{15} : (q_1, z, q_0) \rightarrow b$

$P_{16} : (q_1, z, q_1) \rightarrow b$

$\delta(q_0, b, z) = \{q_0, z z_0\}$

$P_7 : (q_0, z, q_0) \rightarrow b (q_0, z, q_0) (q_0, z_0, q_0)$

$P_8 : (q_0, z, q_0) \rightarrow b (q_0, z, q_1) (q_1, z_0, q_0)$

$P_9 : (q_0, z, q_1) \rightarrow b (q_0, z, q_0) (q_0, z_0, q_1)$

$P_{10} : (q_0, z, q_1) \rightarrow b (q_0, z, q_1) (q_1, z_0, q_1)$

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

$P_{11} : (q_0, z, q_0) \rightarrow a (q_1, z, q_0)$

$P_{12} : (q_0, z, q_1) \rightarrow a (q_1, z, q_1)$

$\delta(q_1, a, z_0) = \{q_0, z_0\}$

$P_{17} : (q_1, z_0, q_0) = a (q_0, z_0, q_0)$

$P_{18} : (q_1, z_0, q_1) = a (q_0, z_0, q_1)$

$* A = \{\{q_0,$

$\delta,$

$\delta(q_0, a,$

$\delta(q_0, b,$

$\delta(q_0, a,$

$\delta(q_0, b,$

$\delta(q_0, a,$

$\delta(q_0, b,$

$\delta(q_0, c,$

$\delta(q_0,$

$\delta(q_0,$

$\delta(q_0,$

$\delta(q_1,$

$\delta(q_1,$

Q) Given g

$T = \{a$

$V = \{$

(

(

(

(

Scanned with CamScanner

$A = \{\{q_0, q_1, q_2\}, \{a, b, c\}, \{a, b, z_0\}, \delta, q_0, z_0, q_2\}$

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, b, z_0) = (q_0, b z_0)$

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

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, a, b) = (q_0, ab)$

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

$\delta(q_0, c, z_0) = (q_1, z_0)$

$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_0, a, a) = (q_1, \epsilon)$

$\delta(q_1, b, b) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$

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

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

$V = \{S, (q_0, a, q_0), (q_0, a, q_1), (q_0, a, q_2),$
$(q_1, a, q_0), (q_1, a, q_1), (q_1, a, q_2),$
$(q_2, a, q_0), (q_2, a, q_1), (q_2, a, q_2),$
$(q_0, b, q_0), (q_0, b, q_1), (q_0, b, q_2),$
$(q_1, b, q_0), (q_1, b, q_1), (q_1, b, q_2),$
$(q_2, b, q_0), (q_2, b, q_1), (q_2, b, q_2),$
$(q_0, z_0, q_0), (q_0, z_0, q_1), (q_0, z_0, q_2),$
$(q_1, z_0, q_0), (q_1, z_0, q_1), (q_1, z_0, q_2),$
$(q_2, z_0, q_0), (q_2, z_0, q_1), (q_2, z_0, q_2)\}$

$P_1 \to (q_0, a, q_0)$

$P_2 \to (q_0, a, q_1)$

$P_3 \to (q_0, a, q_2)$

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$P_4 : (q_0, z_0, q_0) \to a\,(q_0, a, q_0), (q_0, z_0, q_0)$

$P_5 : (q_0, z_0, q_0) \to a\,(q_0, a, q_1), (q_1, z_0, q_0)$

$P_6 : (q_0, z_0, q_0) \to a\,(q_0, a, q_2), (q_2, z_0, q_0)$

$P_7 : (q_0, z_0, q_1) \to a\,(q_0, a, q_0)\,(q_0, z_0, q_1)$

$P_8 : (q_0, z_0, q_1) \to a\,(q_0, a, q_1)\,(q_1, z_0, q_1)$

$P_9 : (q_0, z_0, q_1) \to a\,(q_0, a, q_2)\,(q_2, z_0, q_1)$

$P_{10} : (q_0, z_0, q_2) \to a\,(q_0, a, q_0)\,(q_0, z_0, q_2)$

$P_{11} : (q_0, z_0, q_2) \to a\,(q_0, a, q_1)\,(q_1, z_0, q_2)$

$P_{12} : (q_0, z_0, q_2) \to a\,(q_0, a, q_2)\,(q_2, z_0, q_2)$

$\delta(q_0, b, z_0) = (q_0, b z_0)$

$P_{13} : (q_0, z_0, q_0) \to b\,(q_0, b, q_0)\,(q_0, z_0, q_0)$

$P_{14} : (q_0, z_0, q_0) \to b\,(q_0, b, q_1)\,(q_1, z_0, q_0)$

$P_{15} : (q_0, z_0, q_0) \to b\,(q_0, b, q_2)\,(q_1, z_0, q_0)$

$P_{16} : (q_0, z_0, q_1) \to b\,(q_0, b, q_0)\,(q_0, z_0, q_1)$

$P_{17} : (q_0, z_0, q_1) \to b\,(q_0, b, q_1)\,(q_1, z_0, q_1)$

$P_{18} : (q_0, z_0, q_1) \to b\,(q_0, b, q_2)\,(q_2, z_0, q_1)$

$P_{19} : (q_0, z_0, q_2) \to b\,(q_0, b, q_0)\,(q_0, z_0, q_2)$

$P_{20} : (q_0, z_0, q_2) \to b\,(q_0, b, q_1)\,(q_1, z_0, q_2)$

$P_{21} : (q_0, z_0, q_2) \to b\,(q_0, b, q_2)\,(q_2, z_0, q_2)$

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

$P_{22} : (q_0, a, q$

$P_{23} : (q_0, a, q$

$P_{24} : (q_0, a,$

$P_{25} : (q_0, a$

$P_{26} : (q_0, a,$

$P_{27} : (q_0, a,$

$P_{28} : (q_0, a$

$P_{29} : (q_0, a$

$P_{30} : (q_0,$

$\delta(q_0, b,$

$P_{31} : (q_0,$

$P_{32} : (q_0,$

$P_{33} : (q_0,$

$P_{34} : (q_0,$

$P_{35} : (q_0$

$P_{36} : (q_0$

$P_{37} : (q_0$

$P_{38} : (q$

$P_{39} : (q$

$\delta(q_0, a, b$

$P_{40} : (q_0$

$P_{41} : (q_0$

$P_{42} : (q_0$

$P_{43} : (q$

$P_{44} : (q$

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

$P_{22}: (q_0, a, q_0) \Rightarrow a\,(q_0, a, q_0)(q_0, a, q_0)$

$P_{23}: (q_0, a, q_0) \Rightarrow a\,(q_0, a, q_1)(q_1, a, q_0)$

$P_{24}: (q_0, a, q_0) \rightarrow a\,(q_0, a, q_2)(q_2, a, q_0)$

$P_{25}: (q_0, a, q_1) \rightarrow a\,(q_0, a, q_0)(q_0, a, q_1)$

$P_{26}: (q_0, a, q_1) \rightarrow a\,(q_0, a, q_1)(q_1, a, q_1)$

$P_{27}: (q_0, a, q_1) \rightarrow a\,(q_0, a, q_2)(q_2, a, q_1)$

$P_{28}: (q_0, a, q_2) \rightarrow a\,(q_0, a, q_0)(q_0, a, q_2)$

$P_{29}: (q_0, a, q_2) \rightarrow a\,(q_0, a, q_1)(q_1, a, q_2)$

$P_{30}: (q_0, a, q_2) \rightarrow a\,(q_0, a, q_2)(q_2, a, q_2)$

$\delta(q_0, b, a) = \{q_0, ba\}$

$P_{31}: (q_0, a, q_0) \Rightarrow b(q_0, b, q_0)(q_0, a, q_0)$

$P_{32}: (q_0, a, q_0) \rightarrow b(q_0, b, q_1)(q_1, a, q_0)$

$P_{33}: (q_0, a, q_0) \rightarrow b\,(q_0, b, q_2)(q_2, a, q_0)$

$P_{34}: (q_0, a, q_1) \rightarrow b(q_0, b, q_0)(q_0, b, q_1)$

$P_{35}: (q_0, a, q_1) \rightarrow b(q_0, b, q_1)(q_1, b, q_1)$

$P_{36}: (q_0, a, q_1) \rightarrow b\,(q_0, b, q_2)(q_2, b, q_1)$

$P_{37}: (q_0, a, q_2) \rightarrow b(q_0, b, q_0)(q_0, b, q_2)$

$P_{38}: (q_0, a, q_2) \rightarrow b\,(q_0, b, q_1)(q_1, b, q_2)$

$P_{39}: (q_0, a, q_2) \rightarrow b\,(q_0, b, q_2)(q_2, b, q_2)$

$\delta(q_0, a, b) = (q_0, ab)$

$P_{40}: (q_0, b, q_0) \rightarrow a\,(q_0, a, q_0)(q_0, b, q_0)$

$P_{41}: (q_0, b, q_0) \rightarrow a\,(q_0, a, q_1)(q_1, b, q_0)$

$P_{42}: (q_0, b, q_0) \rightarrow a\,(q_0, a, q_2)(q_2, b, q_0)$

$P_{43}: (q_0, b, q_1) \rightarrow a\,(q_0, a, q_0)(q_0, b, q_1$

$P_{44}: (q_0, b, q_1) \rightarrow a\,(q_0, a, q_1)(q_1, b, q$

P_{45} : $(q_0, b, q_1) \rightarrow a\,(q_0, a, q_2)\,(q_2, b, q_1)$

P_{46} : $(q_0, b, q_2) \rightarrow a\,(q_0, a, q_0)\,(q_0, b, q_2)$

P_{47} : $(q_0, b, q_2) \rightarrow a\,(q_0, a, q_1)\,(q_1, b, q_2)$

P_{48} : $(q_0, b, q_2) \rightarrow a\,(q_0, a, q_1)\,(q_1, b, q_2)$

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

P_{49} : $(q_0, b, q_0) \rightarrow b\,(q_0, b, q_0)\,(q_0, b, q_0)$

P_{50} : $(q_0, b, q_0) \rightarrow b\,(q_0, b, q_1)\,(q_1, b, q_0)$

P_{51} : $(q_0, b, q_0) \rightarrow b\,(q_0, b, q_2)\,(q_2, b, q_0)$

P_{52} : $(q_0, b, q_1) \rightarrow b\,(q_0, b, q_0)\,(q_0, b, q_1)$

P_{53} : $(q_0, b, q_1) \rightarrow b\,(q_0, b, q_1)\,(q_1, b, q_1)$

P_{54} : $(q_0, b, q_1) \rightarrow b\,(q_0, b, q_2)\,(q_2, b, q_1)$

P_{55} : $(q_0, b, q_2) \rightarrow b\,(q_0, b, q_0)\,(q_0, b, q_2)$

P_{56} : $(q_0, b, q_2) \rightarrow b\,(q_0, b, q_1)\,(q_1, b, q_2)$

P_{57} : $(q_0, b, q_2) \rightarrow b\,(q_0, b, q_2)\,(q_2, b, q_2)$

$\delta(q_0, c, z_0) = (q_1, z_0)$

P_{58} : $(q_0, z_0, q_0) \rightarrow c\,(q_1, z_0, q_0)$

P_{59} : $(q_0, z_0, q_1) \rightarrow c\,(q_1, z_0, q_1)$

P_{60} : $(q_0, z_0, q_2) \rightarrow c\,(q_1, z_0, q_2)$

$\delta(q_0, c, a) = (q_1, a)$

P_{61} : $(q_0, a, q_0) = c\,(q_1, a, q_0)$

P_{62} : $(q_0, a, q_1) = c\,(q_1, a, q_1)$

P_{63} : $(q_0, a, q_2) \Rightarrow c\,(q_1, a, q_2)$

$\delta(q_0, c, b) \Rightarrow (q_1, b)$

P_{64} : $(q_0, b, q_0) \Rightarrow c\,(q_1, b, q_0)$

P_{65} : $(q_0, b, q_1) \Rightarrow c\,(q_1, b, q_1)$

P_{66} : $(q_0, b, q_2) \Rightarrow c\,(q_1, b, q_2)$

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

P_{67} : $(q_0, a, q_0) \rightarrow$

P_{68} : $(q_0, a, q_1$

P_{69} : $(q_0, a, q$

$\delta(q_1, b, b) \rightarrow$

P_{70} : $(q_1, b, q$

P_{71} : $(q_1, b, q$

P_{72} : $(q_1, b, q$

$\delta(q_1, \epsilon, z_0$

P_{73} : $(q_1, z_0$

P_{74} : $(q_1, z_0$

P_{75} : $(q_1, z_0,$

1/9/22

Turing mach

- Turing mach

values. Those

$M = (Q,$

where Q is

Σ i

Γ

q_0

$B | b$

F

δ

i.e

$\delta(q_0, a, a) = (q_1, \epsilon)$

$P_{67} : (q_0, a, q_0) \Rightarrow a$

$P_{68} : (q_0, a, q_1) \Rightarrow a$

$P_{69} : (q_0, a, q_2) \Rightarrow a$

$\delta(q_1, b, b) \Rightarrow (q_1, \epsilon)$

$P_{70} : (q_1, b, q_0) \Rightarrow b$

$P_{71} : (q_1, b, q_1) \Rightarrow b$

$P_{72} : (q_1, b, q_2) \Rightarrow b$

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$

$P_{73} : (q_1, z_0, q_0) \Rightarrow \wedge (\epsilon)$

$P_{74} : (q_1, z_0, q_1) \rightarrow \wedge (\epsilon)$

$P_{75} : (q_1, z_0, q_2) \rightarrow \wedge (\epsilon)$

21/9/22

* Turing machine :-

- Turing machine M consists of 7 tuple values. Those are:

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

where Q is finite no: of states
- Σ is input alphabet
- Γ is tape symbols
- q_0 is initial state
- B / b is blank symbol.
- F is final state
- δ is transition function

i.e., $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times [L, R]$

Scanned with CamScanner

given strings are accepted by turing machine by using its transition table given below.

δ	0	1	x	y	b
→q_1	xRq_2	—	—	—	bRq_5
q_2	$0Rq_2$	YLq_3	—	YRq_2	—
q_3	$0Lq_4$	—	xRq_5	YLq_3	—
q_4	$0Lq_2$	—	xRq_1	—	—
q_5	—	—	—	YRq_5	bRq_6
*q_6	—	—	—	—	—

i) 011 (ii) 0011 (iii) 001

Sol i) $011 \rightarrow x11 \rightarrow xY1 \rightarrow xY1 \rightarrow$ with heads at q_1, q_2, q_3, q_1

Hence string is not accepted.

(ii) $0011 \rightarrow x011 \rightarrow x011 \rightarrow x0Y1 \rightarrow x0Y1$ with heads at q_1, q_2, q_2, q_3, q_4

$\rightarrow x0Y1 \rightarrow xxY1 \rightarrow xxY1 \rightarrow xxYY$ with heads at q_1, q_2, q_2, q_3

$\rightarrow xxYY \rightarrow xxYY \rightarrow xxYY \rightarrow xxYYb$ with heads at q_3, q_5, q_5, q_5

$\rightarrow xxYYbb$ with head at q_6 accepted.

(iii) 001

001 →
↑
q_1

→ xxY b

String is

*check the
not by
table is

δ	1
→q_0	xR
q_1	1R
q_2	
q_3	1
q_4	
q_5	
* q_6	

Sol i) 12
↑
q_0

(iii) 001

001 → x01 → x01 → x0y →
↑ q_1 ; ↑ q_2 ; ↑ q_2 ; ↑ q_3

x0y → x0y → xxy
↑ q_4 ; ↑ q_1 ; ↑ q_2

→ xxyb →
↑ q_2

String is not accepted.

* Check the following strings are accepted or not by using turing machine. Its transition table is given below. (i) 123 (ii) 112233 (iii) 111222333

S	1	2	3	X	Y	Z	B
→ q_0	xRq_1	–	–	–	YRq_4	–	–
q_1	$1Rq_1$	YRq_2	–	–	YRq_1	–	–
q_2	–	ZRq_2	ZLq_3	–	–	ZRq_2	–
q_3	$1Lq_3$	ZLq_3	–	xRq_0	YLq_3	ZLq_3	–
q_4	–	–	–	–	YRq_4	ZRq_5	–
q_5	–	–	–	–	–	ZRq_5	BRq_6
* q_6	–	–	–	–	–	–	–

(i) 123 → x23 → xY3 → xYZ → xYZ →
↑ q_0 ; ↑ q_1 ; ↑ q_2 ; ↑ q_3 ; ↑ q_3

xYZ → xYZ → xYZb → xYZBb
↑ q_0 ; ↑ q_4 ; ↑ q_5 ; ↑ q_6

accepted

(ii) 112233

$112233 \rightarrow x12233 \rightarrow x12233 \rightarrow x1Y233$ (q_0, q_1, q_1, q_2)

$\rightarrow x1Y233 \rightarrow x1Y2Z3 \rightarrow x1Y2Z3$ (q_2, q_3, q_3)

$\rightarrow x1Y2Z3 \rightarrow x1Y2Z3 \rightarrow x1Y2Z3$ (q_3, q_3, q_0)

$\rightarrow xxY2Z3 \rightarrow xxY2Z3 \rightarrow xxYYZ3$ (q_1, q_1, q_2)

$\rightarrow xxYYZ3 \rightarrow xxYYZZ \rightarrow xxYYZZ$ (q_2, q_3, q_3)

$\rightarrow xxYYZZ \rightarrow xxYYZZ \rightarrow xxYYZZ$ (q_3, q_3, q_0)

$\rightarrow xxYYZZ \rightarrow xxYYZZ \rightarrow xxYYZZ$ (q_4, q_4, q_5)

$\rightarrow xxYYZZb \rightarrow xxYYZZBb$ (q_5, q_6)

Accepted.

(iii) 111222333

$111222333 \rightarrow x11222333 \rightarrow x1\ldots$ (q_0, q_1)

$x11222333 \rightarrow x11Y22333 \rightarrow$ (q_1, q_2)

$\rightarrow x11Y22333 \rightarrow$ (q_3)

$\rightarrow x11Y22Z33$ (q_3)

$\rightarrow xx1Y22Z33$ (q_1)

$\rightarrow xx1YY2Z33$ (q_2)

$\rightarrow xx1YY2ZZ\ldots$ (q_3)

$\rightarrow xx1YY2ZZ3$ (q_3)

$\rightarrow xxxYY2Z\ldots$ (q_1)

$\rightarrow xxxYYYZ\ldots$ (q_2)

$\rightarrow xxxYYYZ\ldots$

$\ldots xxYYYZZZ \rightarrow xxxYYYZZZ \rightarrow xxxYYYZZZ \rightarrow$ (q_3, q_3, q_3)

$\ldots YYYZZZ \rightarrow xxxYYYZZZ \rightarrow xxxYYYZZZ \rightarrow xxxYYYZZZ$ (q_0, q_4, q_4, q_4)

$\rightarrow xxxYYYZZZ$ (q_5)

$\rightarrow xxxYYYZZZ$ (q_5)

$\rightarrow xxxYYYZZZ$ (q_5)

$\rightarrow xxxYYYZZZb \rightarrow xxxYYYZZZBb \rightarrow$ accepted (q_5, q_6)

(iii) 111222333

111222333 → x11222333 → x11222333 →
$\uparrow q_0$ $\uparrow q_1$ $\uparrow q_1$

x11222333 → x11y22333 → x11y22333
$\uparrow q_1$ $\uparrow q_2$ $\uparrow q_2$

→ x11y22333 → x11y22z33 → x11y22z33
$\uparrow q_2$ $\uparrow q_3$ $\uparrow q_3$

→ x11y22z33 → x11y22z33 → x11y22z33
$\uparrow q_3$ $\uparrow q_3$ $\uparrow q_3$

→ x11y22z33 → x11y22z33 → xx1y22z33
$\uparrow q_3$ $\uparrow q_0$ $\uparrow q_1$

→ xx1y22z33 → xx1y22z33 → xx1yy2z33
$\uparrow q_1$ $\uparrow q_1$ $\uparrow q_2$

→ xx1yy2z33 → xx1yy2z33 → xx1yy2zz3
$\uparrow q_2$ $\uparrow q_2$ $\uparrow q_3$

→ xx1yy2zz3 → xx1yy2zz3 → xx1yy2zz3
$\uparrow q_3$ $\uparrow q_3$ $\uparrow q_3$

→ xx1yy2zz3 → xx1yy2zz3 → xx1yy2zz3
$\uparrow q_3$ $\uparrow q_3$ $\uparrow q_0$

→ xxxyy2zz3 → xxxyy2zz3 → xxxyy2zz3
$\uparrow q_1$ $\uparrow q_1$ $\uparrow q_1$

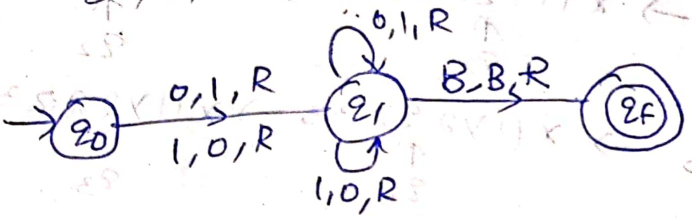
→ xxxyyyzz3 → xxxyyyzz3 → xxxyyyzz3
$\uparrow q_2$ $\uparrow q_2$ $\uparrow q_2$

→ xxxyyyzzz → xxxyyyzzz → xxxyyyzzz
$\uparrow q_3$ $\uparrow q_3$ $\uparrow q_3$

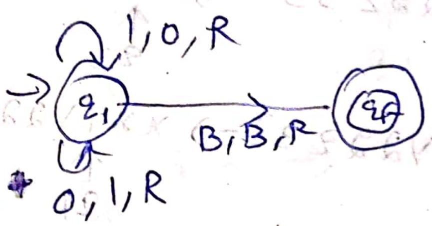
*Design of turing machine:-

* Design a turing machine for 1's compliment where $\Sigma=\{0,1\}$

Sol: $L=\{0,1,00,01,10,11,000,001,\ldots\}$



(or)



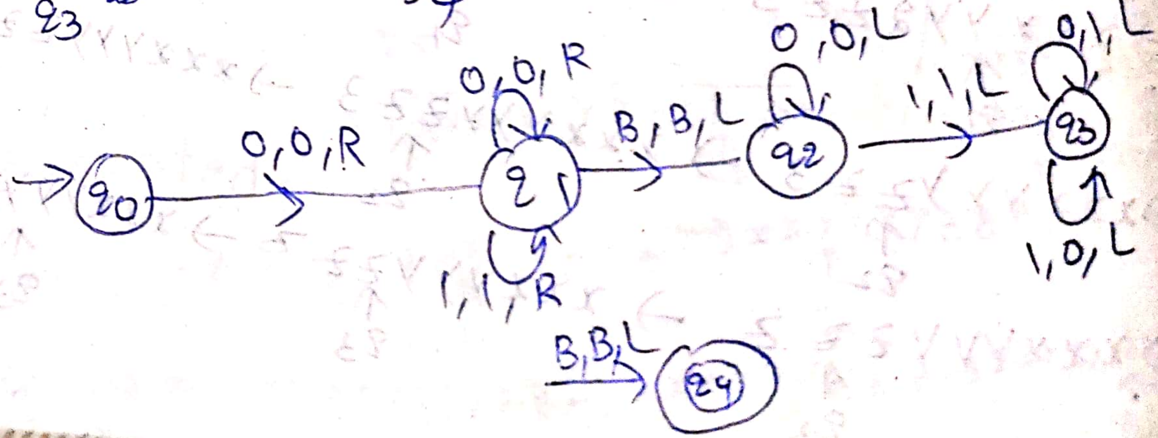
* Design a turing machine for 2's compliment where $\Sigma=\{0,1\}$

Sol: 1100

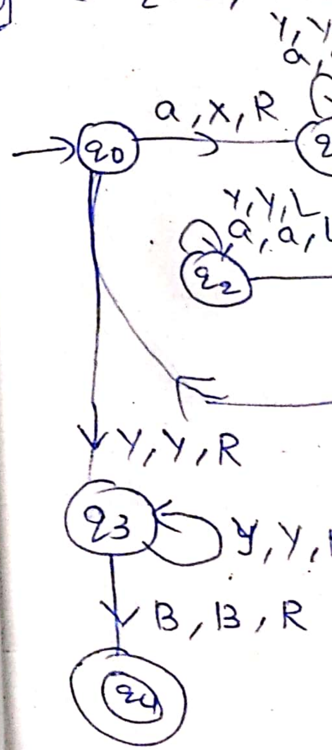
1100 → 1100 → 1100 → 1100 → 1100b
(q0) (q0) (q0) (q0) (q0)

→ 1100b → 1100 → 1100 → 1100 →
(q1) (q1) (q2) (q2)

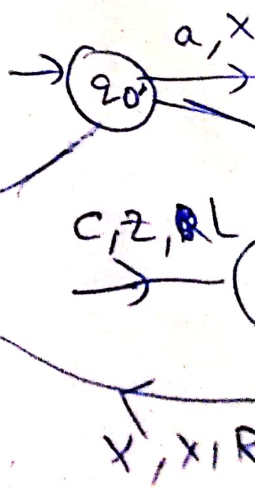
b0100b → bb0100
(q3) (q4)



Design a turing
language L=
Sol L={ab,aab



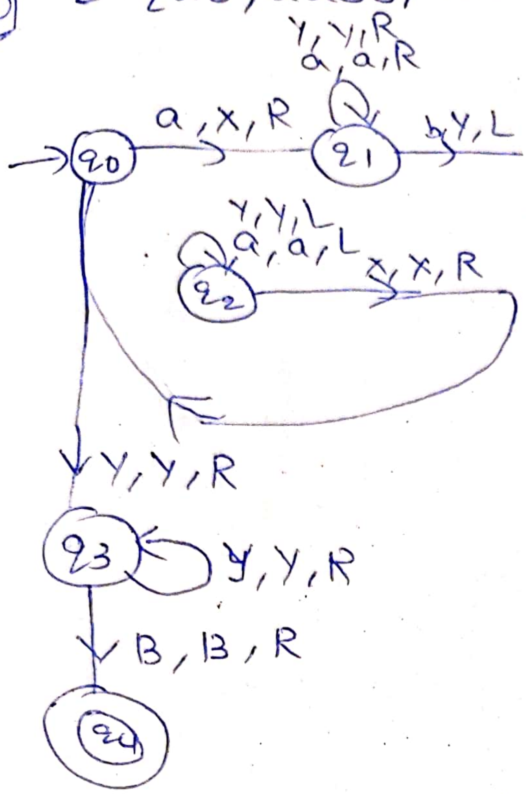
* L={aⁿbⁿcⁿ
* Design a tu
n≥1

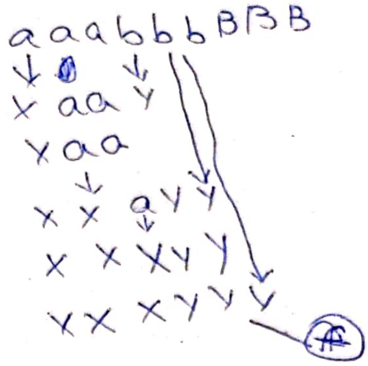


Scanned with CamScanner

* Design a turing machine which accept language $L=\{a^n b^n$ where $n>=1\}$

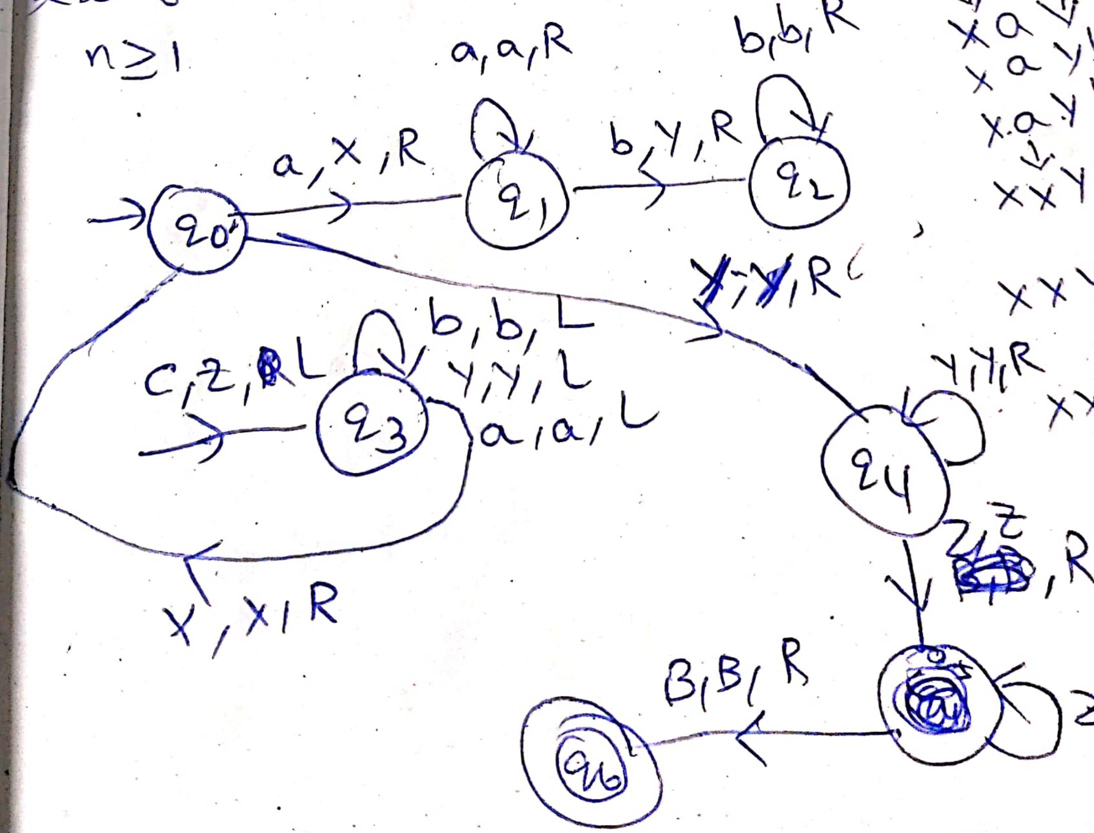
Sol $L=\{ab, aabb, aaabbb, \text{-----}\}$

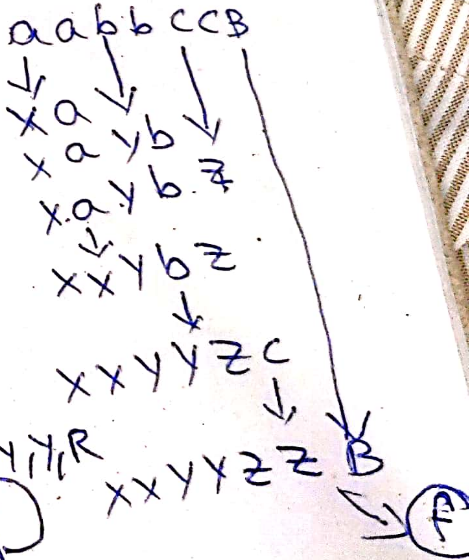




* $L=\{a^n b^n c^n \mid n>=1\}$

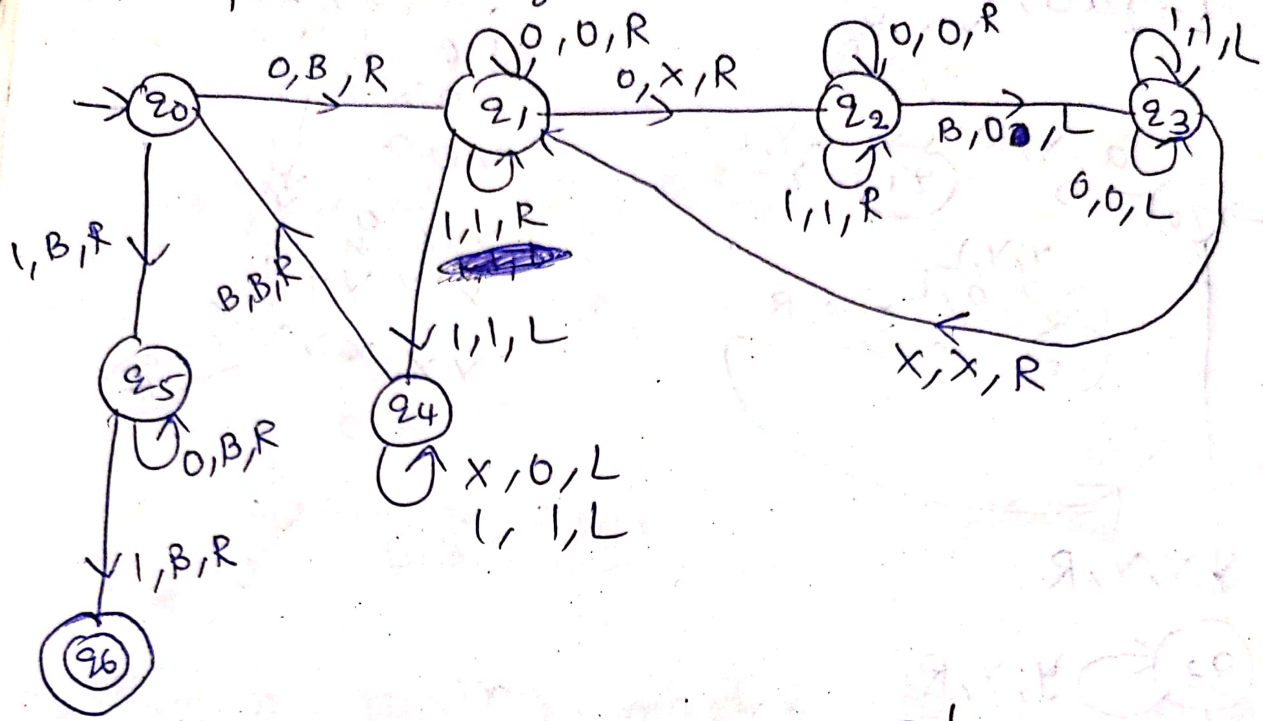
* Design a turing machine for $a^n b^n c^n$ where $n \geq 1$





Design a turing machine for add 2 numbers. Design a TM for ~~[illegible]~~

— Multiplication of 2 numbers



$0^n 1 0^n 1$; $n=2 \Rightarrow 001001$

$n=1 \Rightarrow 0101$

--- BBB 001001BBB ----

B 0 1 X 0 1 0 0

X 1 0 0

B 0 1 0 0 1

B B 1 X 0 1 0 0 0

B B B B B B 0 0 0 0

There are 2 languages — recursive language
(Turing machine) — Recursive Enumerable language

Recursive language: 2 states
- Halt & accept
- Halt & reject

Recursive Enumerable language: 3 states
- Halt & accept
- Halt & reject
- Never halt

- If the language L is recursive language, there are 2 states ① Halt & accept ② Halt & reject.
- The recursive languages are closed under all properties except homomorphism and substitution.

Recursive enumerable

- If the language L is recursive enumerable then there are 3 states. ① Halt & accept ② Halt & reject ③ Never halt.
- Recursive enumerable languages are closed under all properties except complimentation.

Turing Machine

*Moves of Turing Machine
- There are 3 possible moves.
① Move to the left. ② Move to the right
③ No move.

The string classes in Turing Machine:-
- Every turing machine over Σ divides the set of strings w into 3 classes

① Accept ② Reject ③ Loop

- Role of Turing Machine:-

① Accepting the devices for languages

② Computation of functions

③ Enumerator of strings of languages that output the strings of languages in some systematic order i.e., list.

3 ⇒ 0001000 1

--B B B B B 0001 000 1 B B B B B B B B B

B 001 X 001 0

X 010 0

X 1 000

B 0 0 1 0 0 0 1 0 0 0

B B 0 1 X 001 000 0

X 0 1 0 0 0 0 0

X 1 0 0 0 0 0 0

B B 0 1 0 0 0 1 0 0 0 0 0 0

B B B 1 X 0 0 1 0 0 0 0 0 0 0

X 0 1 0 0 0 0 0 0 0 0

X 1 0 0 0 0 0 0 0 0 0

B B B 1 0 0 0 1 0 0 0 0 0 0 0 0 0

$\boxed{\text{B B B B B B B 1 } \boxed{0\,0\,0\,0\,0\,0\,0\,0\,0}}$

Instantane
- The compl
at any po
that may
① The name
machine i
② The symb
③ The cell
- The desc
Instantar
- Types of
① Turing m
② Multiple
③ Non det
④ Multi di
⑤ Offline
⑥ Multi h
* Construct
transition

ε

1

ε

The give
Q = {1,2
Σ = {
δ : Q X

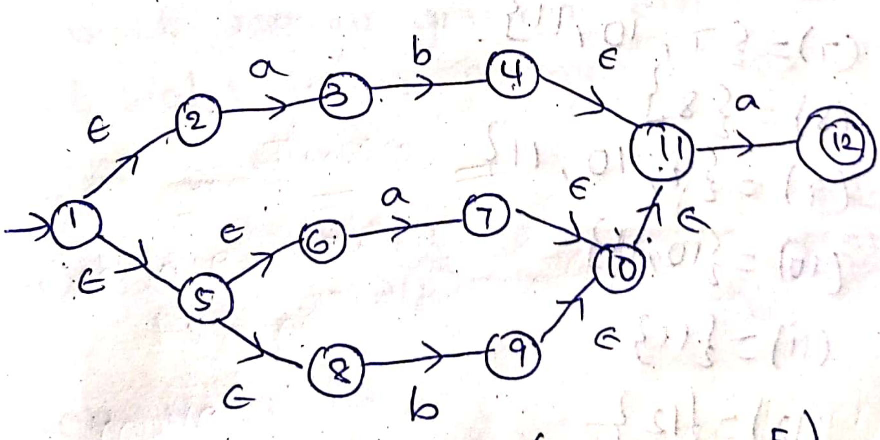
Instantaneous description of turing machine:-

- The complete state of turing machine at any point during the computation that may be described as
① The name of the state in which the machine is
② The symbol in input tape.
③ The cell being currently scanned.

- The description of these 3 is called Instantaneous description or configuration.

- Types of turing machine:-
① Turing machine with two way infinite input tape
② Multiple Turing machines or tapes.
③ Non deterministic turing machine
④ Multi dimensional "
⑤ Offline "
⑥ Multi head "

* Construct DFA for the following NFA. The NFA transition diagram is given below:-



Sol: The given NFA $M = (Q, \Sigma, \delta, q_0, F)$

$Q = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

$\Sigma = \{a, b\}$ $\quad q_0 = 1$

$F = 12$

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

S	a	b	ε
→1	–	–	2,5
2	3	–	–
3	–	4	–
4	–	–	11
5	–	–	6,8
6	7	–	–
7	–	–	10
8	–	9	–
9	–	–	10
10	–	–	11
11	12	–	–
* 12	–	–	–

ε-closure(1) = {1, 2, 5, 6, 8}

ε-closure(2) = {2}

" (3) = {3}

" (4) = {4, 11}

" (5) = {5, 6, 8}

" (6) = {6}

" (7) = {7, 10, 11}

" (8) = {8}

" (9) = {9, 10, 11}

" (10) = {10, 11}

" (11) = {11}

" (12) = {12}

current state

A
A
B
B
C
C
D
D
E
E

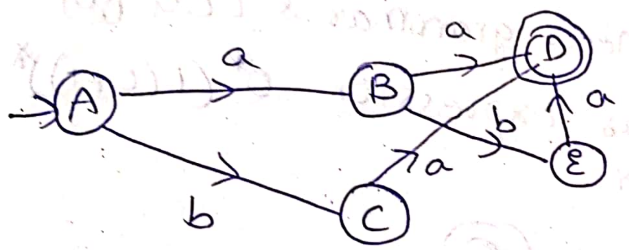
→(A)

*Construc
and r
finite

Given

A→
B→
C→
This is

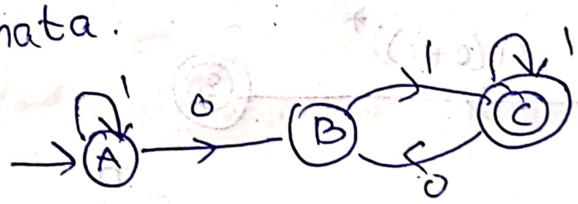
current state	current i/p	δ(state, i/p)	ε-closure(δ(state, i/p)) =new state
A	a	{3,7}	{3,7,10,11} =B
A	b	{9}	{9,10,11} = C
B	a	{12}	{12} = D
B	b	{4}	{4,11} = E
C	a	{12}	{12} = D
C	b	φ	φ
D	a	φ	φ
D	b	φ	φ
E	a	{12}	{12} = D
E	b	φ	φ



UNIT-II

*Construct regular grammar(left linear grammar) and right linear grammar for the given finite automata.

~~xxxx~~



Sol Given data is

A → 1A/0B

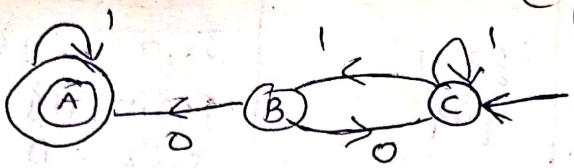
B → 1C/1

C → 0B/1C/1

This is right linear grammar.

Left linear grammar

(Make initial state as final state & final state as initial state)



C → 1B / 1C

B → 0C / 0A / 0

A → 1A / 1

LLG

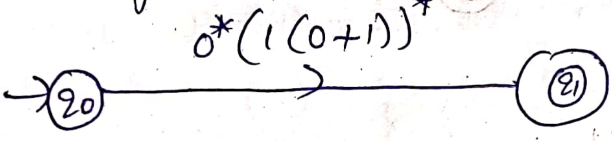
$G = (V, T, P, S)$

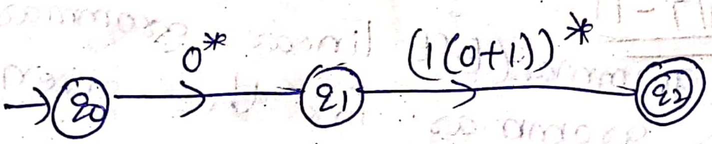
C → B1 / C1

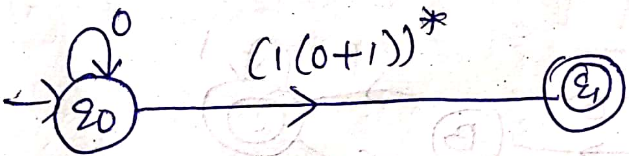
B → C0 / A0 / 0

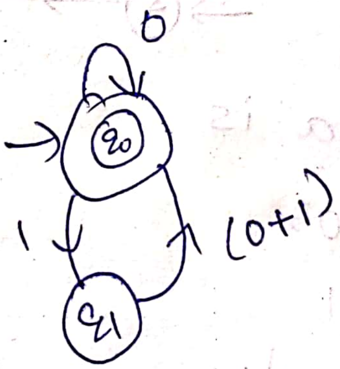
A → A1 / 1

*Construct Right linear grammar & LLG for the given regular expression $0^*(1(0+1))^*$







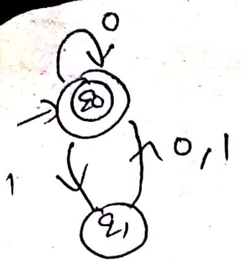


make initial state as final state & final state as initial state)

(V,T,P,S)

LLG for

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

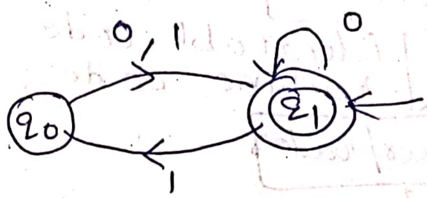


RLG

$q_0 \rightarrow 0q_0 \mid 0 \mid 1q_1$

$q_1 \rightarrow 0q_0 \mid 1q_0 \mid 0 \mid 1$

LLG



$q_1 \rightarrow 0q_1 \mid 0 \mid 1q_0$

$q_0 \rightarrow 0q_1 \mid 0 \mid 1q_1 \mid 1$

30/9/22

Compiler Design

High-level language

↓

Pre-processor

file inclusion
macro expansion

↓ Pure HLL

Compiler

(Intermediate language)

↓ Assembly language / Low level " " / Machine " "

Assembler

↓ Relocatable code / Machine code

Linker/loader

↓

absolute code

Imp

— structure of phases of compiler :-

High level language

↓

Lexical Analysis

↓ lexem/Tokens → stream of tokens

Syntax Analysis — Tool: syntax Analyzer

↓ Parse tree/Syntax tree

semantic Analysis — Tool: semantic Analyzer

↓ Parse tree semantically verified

Intermediate code generation

↓ Three address code

code optimizer — (Reducing the code)

↓

Target code generation

↓

Target code (Assembly level language)

Symbol table

Error Handler

Example:-

$x = a + b * c;$ (HLL)

↓

$\boxed{\text{Lexical Analyser}}$

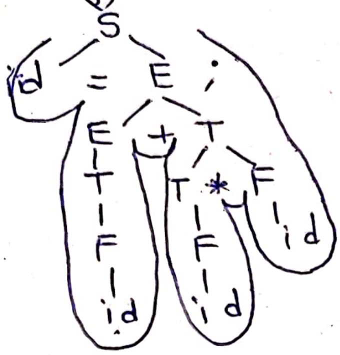
↓

$id = id + id * id;$

↓

$\boxed{\text{Syntax Analyser}}$

↓



$S \rightarrow id = E;$
$E \rightarrow E + T \mid T$
$F \rightarrow T * F \mid F$
$F \rightarrow id$

↓

$\boxed{\text{Semantic Analysis}}$

↓

$\boxed{\text{Intermediate code generation}}$

↓

$t_1 = b * c$
$t_2 = a + t_1$
$x = t_2$

↓

$\boxed{\text{code optimiser}}$

↓

$t_1 = b * c$
$x = a + t_1$

↓

$\boxed{\text{target code generation}}$

↓

mul R_1, R_2
add R_0, R_2
mov R_2, x

$a = R_0$
$b = R_1$
$c = R_2$

Syntax Analyzer :-

Types of parsers

- Top down parsers (Based on when to use the productions)
  - Full backtracking
    - Brute force method
  - without backtracking (predictive parser)
    - Recursive descent
    - Non-recursive decent (LL(1))
- Bottom up parsers (Based on when to reduce the productions) (SR parsers) SR-shift Reduce
  - operator precedence
  - LR parsers
    - LR(0)
    - SLR(1)
    - LALR(1)
    - CLR(1)

Syntax Analyser -

Types of parsers

- Top Down parsers (Based on when to use the productions)
  - Full Back Tracking
    - Brute Force method
  - without Back Tracking (Predictive Parser)
    - Recursive Descent
    - Non-Recursive Descent (LL(1))
- Bottom up Parsers (Based on when to reduce the productions) (SR parsers) SR - Shift Reduce
  - Operator Precedence
  - LR Parsers
    - LR(0)
    - SLR(1)
    - LALR(1)
    - CLR(1)

3/10

- For elimination of the unambiguous grammer by using
  1. Eliminating left recursion.
  2. ~~Eliminating~~ left factoring.

Left Recursion: $A \to A\alpha_1 | A\alpha_2 - | A\alpha_n$

Right Recursion: $A \to \alpha_1 A_1 | \alpha_2 A_2 | -- | \alpha_n A_n$

① ⇒ let, $G = \{V, T, P, S\}$ be a context free grammar where $A \to A\alpha_1 | A\alpha_2 --- A\alpha_n | \beta_1 | \beta_2 -- | \beta_n$, then eliminate the left recursion by using formula

$$\boxed{\begin{array}{l} A \to \beta A' \\ A' \to \alpha A' | \epsilon \end{array}}$$

Eg: eliminate the left recursion for the given grammar.

$S \to \underset{A}{\underline{S}}\ \underset{\alpha}{\underline{0S1S}} / \underset{\beta}{\underline{01}}$

Sol:

$A \to 01S'$

$S' \to 0S1S\,S' | \epsilon$

Eg: Eliminate the left recursion for the given grammar

$E \to E + T / T$

$T \to T * F / F$

$E \to id / (E)$

Sol:

$E \to \underset{A}{\underline{E}} \underset{\alpha}{\underline{+T}} / \underset{\beta_1}{\underline{T}}$

⇒ $E \to TE'$

⇒ $E' \to +TE' | \epsilon$

$T \to \underset{A}{\underline{T}} \underset{\alpha}{\underline{*F}} / \underset{\beta}{\underline{F}}$

⇒ $T \to FT'$

⇒ $T' \to *FT' | \epsilon$

⇒ $F \to id | (E)$

Now,
Grammar is

$E \rightarrow TE'$

$E' \rightarrow +T | \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' | \epsilon$

$F \rightarrow id | (E)$

Q: Eliminate the left recursion from the given grammar

① $S \rightarrow (L), a$

$\underbrace{L}_{A} \rightarrow \underbrace{L}_{A}\underbrace{, S}_{\alpha} | \underbrace{S}_{\beta}$

$L \rightarrow SL'$

$L' \rightarrow , SL' | \epsilon$

∴ The final grammar is

$S \rightarrow (L), a$

$L \rightarrow SL'$

$L' \rightarrow , SL' | \epsilon$

2Q. Eliminate the left recursion from the given grammar

$Expr \rightarrow \underbrace{Expr}_{A} \underbrace{+Expr}_{\alpha_1} | \underbrace{Expr}_{A} \underbrace{*Expr}_{\alpha_2} | \underbrace{id}_{\beta}$

∴ The final grammar is:

$Expr \rightarrow id\, Expr'$

$Expr' \rightarrow +Expr\, Expr' | \epsilon | *Expr\, Expr'$

~~$Expr' \rightarrow *Expr\, Expr' | \epsilon$~~

Left Factoring:

• If the Grammar $A \rightarrow \alpha\beta_1 | \alpha\beta_2 | \alpha\beta_3 \dots \alpha\beta_n$ then we can eliminate the left factoring using

$$\boxed{\begin{array}{l} A \rightarrow \alpha A' \\ A' \rightarrow \beta_1 | \beta_2 \dots \beta_n \end{array}}$$

Eg: $S \rightarrow \underbrace{a}_{\alpha}\underbrace{AB}_{\beta_1} | \underbrace{a}_{\alpha}\underbrace{CD}_{\beta_2} | \underbrace{a}_{\alpha}\underbrace{EF}_{\beta_2} | bd$

$S \rightarrow aS' | bd$

$S' \rightarrow AB | CD | EF$

Q. Eliminate left factor for the given grammar.

The grammar is $S \rightarrow \underbrace{iEtS}_{\alpha} | \underbrace{iEtS}_{\alpha} \underbrace{eS}_{\beta_1} | a$

$E \rightarrow d$

The final grammar is:

$S \rightarrow iEtSS' / a$

$S' \rightarrow eS / \epsilon$

$E \rightarrow d$.

Q. Eliminate left factor for the given grammar.

The grammar is

$A \rightarrow aAB / aA / a$

$B \rightarrow bB / b$

The final grammar is

$A \rightarrow aA'$

$A' \rightarrow AB / A / \epsilon$

$B \rightarrow bB'$

$B \rightarrow B / \epsilon$

Q. Eliminate ~~left~~ factor for the given grammar.

$S \rightarrow assbS / asaSb / abb / b$

Sol: The ~~given~~ final grammar is

$S \rightarrow aS' / b$

$S' \rightarrow SSbS / SaSb / bb$

$S' \rightarrow SS'' / bb$

$S'' \rightarrow SbS / aSb$

$\Rightarrow$

$S \rightarrow aS' / b$

$S' \rightarrow SSbS / SaSb / bb / SS''$

$S'' \rightarrow SbS / aSb / S''$

FIRST() and FOLLOW():

- If $A \rightarrow a\alpha$ where $\alpha \in (V \cup T)^*$ then $FIRST(A) = \{a\}$
- If $A \rightarrow \epsilon$, then $FIRST(A) = \{\epsilon\}$
- If $A \rightarrow BC$, then $FIRST(A) = FIRST(B)$ if B doesnot contains any ϵ.
- If B contain ϵ, then $FIRST(A) = FIRST(B) \cup FIRST(C)$

Eg: Find out 1st symbol for the given grammar

$E \rightarrow E+T / T$

$T \rightarrow T*F / F$

$F \rightarrow id / (E)$

Sol: ~~FIRST(E) = FIRST(E) ∪ FIRST(T)~~

$E \rightarrow \underset{A}{E} \underset{\alpha}{+T} / \underset{\beta}{T}$

$E \rightarrow TE'$

$E' \rightarrow +TE' / \epsilon$

$T \rightarrow \underset{A}{T} \underset{\alpha}{*F} / \underset{\beta}{F}$

$T \rightarrow FT'$

$T' \rightarrow *FT' / \epsilon$

After eliminating left recursion, the grammar is,

$E \to TE'$
$E' \to +TE'/\epsilon$
$T \to FT'$
$T' \to *FT'/\epsilon$
$F \to id/(E)$

$(A \to Bd/\epsilon$
$= \{a, \epsilon$

$FIRST(E) = FIRST(T) = FIRST(F) = \{id, (\}$
$FIRST(E') = \{+, \epsilon\}$
$FIRST(T) = FIRST(F) = \{id, (\}$
$FIRST(T') = \{*, \epsilon\}$
$FIRST(F) = \{id, (\}$

Q. Find out first f^n's for the given grammar.

$S \to aBDh$
$B \to cC$
$C \to bC/\epsilon$
$D \to EF$
$E \to g/\epsilon$
$F \to f/\epsilon$

$FIRST(S) = \{a\}$
$FIRST(B) = FIRST(C) = \{b, \epsilon\}$
$FIRST(C) = \{b, \epsilon\}$
$FIRST(D) = FIRST(E) = \{g, \epsilon\}$
$FIRST(E) = \{g, \epsilon\}$
$FIRST(F) = \{f, \epsilon\}$

Q. Find out first f^n's for given grammar

$S \to A$
$A \to aB/Ad$
$B \to b$
$C \to g$

left recursion

$\underline{A} \to \underbrace{aB}_{\beta}/ \underbrace{\underline{A}d}_{\alpha}$

$A \to aBA'$
$A' \to dA'/\epsilon$

The grammar G is

* Syntax Analyses

Types of Parsers

- Top Down Parsers (Based on when to use the productions)
  - Full Back Tracking
    - Brute Force method
  - Without Backtracking (predictive parser)
    - Recursive Descent
    - Non-Recursive Decent (LL(1))
- Bottom up parsers (Based on when to reduce the productions (SR Parser) SR-Shift Reduce
  - Operator precedence
  - LR Parser
    - LR(0)
    - SLR(1)
    - LALR(1)
    - CLR(1)

⇒ For the elimination of the unambiguous grammer by using

1. Eliminating left recursion
2. Left factoring

Grammer → Deterministic
Grammer → Non ambiguous
Grammer → Support to generate parse tree.
Right recursion

$\boxed{A} \to \alpha_1 \boxed{A} \mid \alpha_2 A \mid \dots \alpha_n A$
↳ Right recursion.

⇒ Elimination of Left Recursion

Let $G = (V, T, P, S)$ be a Context-free grammer with

$$A \to A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$$

Then eliminate the ~~right~~ left recursion by using the formula

$$\boxed{\begin{array}{l} A \to \beta A' \\ A' \to \alpha A' / \epsilon \end{array}}$$

e.g:- eliminate the left recursion for the given grammer

$S \to SOS1S \mid 01$

rs (Based on
reduce the
tions (SR Parsers
ift Reduce

LR Parsers
- LR(0)
- SLR(1)
- LALR(1)
- CLR(1)

r by using

→ Non Determinis

ammer

→ Non ambiguous

Right recursion

– $\alpha_n A$

ecursion.

sing

mmer

Ans:- $S \rightarrow \underbrace{S}_{A}\underbrace{0}_{A}\underbrace{S1S}_{\alpha} \mid \underbrace{01}_{\beta}$

$A \rightarrow \beta A'$
$A' \rightarrow \alpha A' \mid \epsilon$ $\Rightarrow$ $S \rightarrow 01S'$
$S' \rightarrow 0S1SS' \mid \epsilon$

Q. Eliminate the left recursion for the given grammar

$E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow id \mid (E)$

$E \rightarrow E+T \mid T$

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow \underbrace{T}_{A}\underbrace{*F}_{\alpha} \mid \underbrace{F}_{\beta}$

$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \epsilon$

$E \rightarrow id \mid (E)$

$\Rightarrow$
$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \epsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \epsilon$
$F \rightarrow id \mid (E)$

Eliminate the left recursion for the given grammar

(ii) $S \rightarrow (L), a$
$L \rightarrow L,S \mid S$

(iii) $Expr \rightarrow Expr + Expr \mid Expr * Expr \mid id$

(ii) $S \rightarrow (L), a$

$\underbrace{L}_{A} \rightarrow \underbrace{L}_{A}\underbrace{,S}_{\alpha} \mid \underbrace{S}_{\beta}$

$L \rightarrow SL'$
$L' \rightarrow SL' \mid \epsilon$

$\Rightarrow$ $S \rightarrow (L), a$
$L \rightarrow SL'$
$L' \rightarrow SL' \mid \epsilon$

(iii) $\underbrace{Expr}_{A} \rightarrow \underbrace{Expr}_{A}\underbrace{+Expr}_{\alpha} \mid \underbrace{Expr * Expr}_{1} \mid id$

The grammar G is

* Left Factoring:

If the grammer $A \rightarrow \alpha\beta_1 | \alpha\beta_2 \dots | \alpha\beta_n$ then we can eliminate the left factoring.

i.e $A \rightarrow \alpha A'$

$A' \rightarrow \beta_1 | \beta_2 \dots \beta_n$

i, Eliminate the left factor for the given grammer. The Grammer is $S \rightarrow iEtS | iEtSeS | a$

$E \rightarrow d$

Ans:- $S \rightarrow \underbrace{iEtS}_{\alpha} | \underbrace{iEtS}_{\alpha}\underbrace{eS}_{\beta_1} | a$

$E \rightarrow d$

$\Rightarrow S \rightarrow iEtSS' | a$

$S' \rightarrow eS | \epsilon$

$E \rightarrow d.$

ii, Eliminate the LF for the given grammer. The grammer is $A \rightarrow aAB | aA | a$

$B \rightarrow bB | b$

Ans:- $A \rightarrow \underbrace{a}_{\alpha}\underbrace{AB}_{\beta_1} | \underbrace{a}_{\alpha}\underbrace{A}_{\beta_2} | \underbrace{a}_{\alpha}$ $\quad B \rightarrow \underbrace{b}_{\alpha}\underbrace{B}_{\beta_1} | \underbrace{b}_{\alpha}$

$A \rightarrow aA'$ $\quad B \rightarrow bB'$

$A' \rightarrow AB | A | \epsilon$ $\quad B' \rightarrow B | \epsilon$

Eliminate the LF for the given Grammer.

iii, $S \rightarrow \underbrace{aS}_{\alpha}\underbrace{Sbs}_{\beta_1} | \underbrace{aS}_{\alpha}\underbrace{aSb}_{\beta_2} | abb | b$

$S \rightarrow aSS' | abb | b$

$S' \rightarrow \underbrace{Sb}_{\alpha}\underbrace{S}_{\beta} | \underbrace{aSb}_{\alpha}$

$S'' \rightarrow SbS''$

$S'' \rightarrow S | \epsilon$

then

mer. The

rammer

* FIRST() & FOLLOW() fun's

FIRST() :- If $A \rightarrow a\alpha$, where $\alpha \in (V \cup T)^*$ then

FIRST(A) = {a}

(2) $A \rightarrow \epsilon$

FIRST(A) = {ϵ}

(3) $A \rightarrow BC$ then

FIRST (A) = FIRST(B) If B doesn't contain any ϵ

If B contain ϵ then FIRST(A) = FIRST(B) ∪ FIRST(C)

$A \rightarrow aBc/b$

$B \rightarrow d/d$

$C \rightarrow dd/D$

$D \rightarrow C$

* find out 1st symbols for the given grammer.

$E \rightarrow \underset{A}{E} + \underset{\alpha}{T} \,/\, \underset{\beta}{T}$

$T \rightarrow T \underset{\alpha}{*F} \,/\, \underset{\beta}{F}$

$F \rightarrow id \,/\, (E)$

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' / \epsilon$

$E \rightarrow TE'$ $\quad E' \rightarrow +TE' / \epsilon$

$T \rightarrow FT'$ $\quad T' \rightarrow \alpha FT' / \epsilon$

$F \rightarrow id / (E)$

Final Grammer

$E \rightarrow TE'$

$E' \rightarrow +TE' / \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' / \epsilon$

$F \rightarrow id / (E)$

FIRST(E) = {id, (}

FIRST(E') = {+, ϵ}

FIRST (T) = {id, (}

FIRST (T') = {α, ϵ}

FIRST(F) = {id, (}

(Q) Findout first fun^n for the given grammer

(1) $S \to aBDh$
$B \to cC'$
$C' \to bC' | \epsilon$
$D \to EF$
$E \to g | \epsilon$
$F \to f | \epsilon$

2) $S \to A$
$A \to aB | Ad$
$B \to b$
$C \to g$

(1) ⇒ FIRST(S) = {a}
FIRST(B) = FIRST(C) = {b}
FIRST(C') = {b}
FIRST(D) = FIRST(E) = {g, ε}
FIRST(E) = {g}
FIRST(F) = {f, ε}

(2) ⇒ $A \to \underbrace{aB}_{\beta} | \underbrace{Ad}_{\alpha}$

$A \to aBA'$
$A' \to dA' | \epsilon$

$A \to aBA'$
$A' \to dA' | \epsilon$
$S \to A$
$B \to b$
$C \to g$

FIRST(A) = {a}
FIRST(A') = {d}
FIRST(S) = FIRST(A) = {a}
FIRST(B) = {b}
FIRST(C) = {g}

(3) ⇒ $S \to \underbrace{cL}_{\alpha} | \underbrace{a}_{\beta}$
$L \to SL'$
$L' \to SL' | \epsilon$

$A \to \alpha A'$
$A' \to \beta_1 | \beta_2 \cdots \beta_n$

FIRST(S) = {c}
FIRST(L) = FIRST(S) = {c}
FIRST(L') = FIRST(S) = {c}

(4) $S \to AaAb | BbBa$
$A \to \epsilon$
$B \to \epsilon$

(5) $S \to AcB | cbB | Ba$
$A \to da | BC$
$B \to g | \epsilon$
$C \to h | \epsilon$

(4) ⇒ S → AaAb | B

FOLLOW() :-
(1) If S is a
(2) If A → αBβ
FOLLOW(B
FIRST(
(3) If A → αB
FOLL
(4) A → αBβ

(1) E → T
E' →
T →
T' →
F →

Scanned with CamScanner

(4) $S \to AaAb \mid BbBa$

FOLLOW( ) :-

(1) If S is a start symbol then follow(S) = {\$}

(2) If $A \to \alpha B \beta$; $\alpha, \beta (V \cup T)^*$

FOLLOW(B) = FIRST(β). If

FIRST(β) ∉ ε

(3) If $A \to \alpha B$ then

FOLLOW(B) = Follow(A)

(4) $A \to \alpha B \beta$ where $\beta \to \epsilon$ then

follow(B) = follow(A)

(1)
$$E \to TE'$$
$$E' \to +TE' \mid \epsilon$$
$$T \to FT'$$
$$T' \to *FT' \mid \epsilon$$
$$F \to id \mid (E)$$

follow(E) = {), \$ }

Follow(E') = {), \$ }

follow(T) = {+,), \$ }

Follow(T') = { +,), \$ }

Follow(F) = { *, +,), \$ }

ACB | cbB | Ba

da | BC

g | ε

h | ε

Step 1 :-

1) If a production at a form $A \to \alpha$ where $\alpha \in (V \cup T)^*$ then FIRST(α) contains 'a' then add $A \to \alpha$ to M[A, a]

Step 2 :- 2) If the production is at the form $A \to \epsilon$ or FIRST(A) = {ε} then Follow(A) = {b} then add $A \to \epsilon$ To M[A, b]

3) :- The remaining all (productions) the entries of the Parsing table are filled with errors

eg:- $E \to TE'$

$E' \to +TE' / \epsilon$

$T \to FT'$

$T' \to *FT' / \epsilon$

$F \to id / (E)$

Parsing table

M	+	*	(	)	id	$
E			$E \to TE'$		$E \to TE'$	
E'	$E' \to +TE'$			$E' \to \epsilon$		$E' \to \epsilon$
T			$T \to FT'$		$T \to FT'$	
T'	$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$		$T' \to \epsilon$
F			$F \to (E)$		$F \to id$	

Rule ① $A \to \alpha$

$\alpha \in (V \cup T)^*$

FIRST(α) = a

② $A \to \alpha$ to M[A, a]

$A \to \epsilon$ (or) FIRST(A) = {ε}

Follow(A) = {b}

$A \to \epsilon$ To M[A, b]

Check the string is accepted or not id+id

$E \to TE'$

$\to FT'E'$

$\to id / (E) T'E'$

$\to id \epsilon +TE'$

$\to id +TE'$

$\to id + FT'E'$

$\to id + id T'E'$

$\to id + id \epsilon E'$

$\to id + id \epsilon\epsilon$

$\to id + id$

String is accepted.

First (L

Stack
$E
$E'T
$E'T'F
$E'T'id
$E'T'
$E'ε
$E'T+
$E'T
$E'T'F
$E'T'id
$E'T'
$E'
$

Check the

Parser the

S → (L) /

L → L, S

Ans:- Give

(1) Elimin

S → (

L →

L' →

Now,

ere $\alpha \in (V \cup T)^*$
$A \rightarrow \alpha$ to

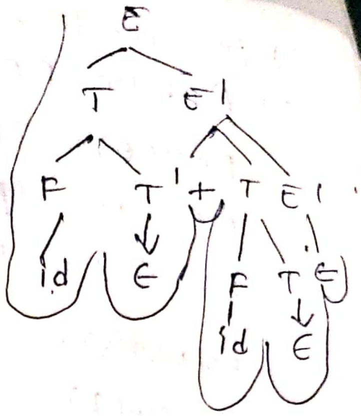
$A \rightarrow \epsilon$ &
then add $A \rightarrow \epsilon$

entries of the

	id	$
	$E \rightarrow TE'$	
		$E' \rightarrow \epsilon$
	$\rightarrow FT'$	
		$T' \rightarrow \epsilon$
	id	

First (L) ⊇ First (S) = {(, a}

Stack	i/p buffer	Action
$E	id+id$	$E \rightarrow TE'$
$E'T	id+id$	$T \rightarrow FT'$
$E'T'F	id+id$	$F \rightarrow id$
$E'T'id	id+id$	pop
$E'T'	+id$	$T' \rightarrow \epsilon$
$E'ε	+id$	$E' \rightarrow +TE'$
$E'T+	+id$	pop
$E'T	id$	$T \rightarrow FT'$
$E'T'F	id$	$F \rightarrow id$
$E'T'id	id$	pop
$E'T'	$	$T' \rightarrow \epsilon$
$E'	$	$E' \rightarrow \epsilon$
$	$	Accepted



Check the given string is accepted or not by using LL1 Parser the string is ((a)) & the Grammer G is

$S \rightarrow (L) / a$

$L \rightarrow L, S / S$

Ans:- Given Grammer $G = (V, T, P, S)$

$V = \{S, L\}$ $S = \{S\}$

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

(1) Eliminate the left recursion

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' / \epsilon$

$S \rightarrow (L) | a$

$L \rightarrow SL'$

$L' \rightarrow ;SL' | \epsilon$

Now, the Grammer is $G' =$

$S = \{S\}$

$V = \{S, L, L'\}$

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

2) Does not have left factoring
3) Find the FIRST and FOLLOW functions.

FIRST(S) = { (, a }
FIRST(L) = FIRST(S) = { (, a }
FIRST(L') = { ,, ϵ }

FOLLOW(S) = { $,), , }
FOLLOW(L) = {) }
FOLLOW(L') = {) }

Ex) - $E \to TE'$
$E' \to +TE' / \epsilon$
$T \to FT'$
$T' \to *FT' / \epsilon$
$F \to id / (E)$.

	FIRST ( )	Follow( )
E	{ id , (}	{ $,) }
E'	{ +, ϵ }	{ $,) }
T	{ id, (}	{ +, $,) }
T'	{ *, ϵ }	{ +, $,) }
F	{ id, (}	{ *, +, $,) }

Scanned with CamScanner

First (L) ⊇ First (S) = {(, a}

1) $A \rightarrow \alpha B \beta$,

Follow (B) = FIRST (β)

2) $A \rightarrow \alpha B$

$\beta \rightarrow \epsilon$

Follow (B) = Follow (A)

$A \rightarrow \alpha B \beta$

Follow (B) = FIRST (β)

= {ϵ}

Follow(E) :-

$F \rightarrow (E)$

$A \quad \alpha \; B \; \beta$

Follow (E) = FIRST (")")

= {) }

Follow E' :

$E \rightarrow T E'$ / $E' \rightarrow +TE'$

$A \quad \alpha \; B$ $A \quad \alpha \; B$

Follow (E') = Follow (E)   Follow (E') = Follow (E')

= {\$,)}

Follow T :-

$E \to TE'$, $E' \to +TE'$

$\cancel{A \to \alpha B}$

A α B β ; β β

Follow (T) = FIRST (E')

= {+, ε}

E → T E'

→ T +

A α B β

= {+

E → T E'

→ T ε

E → T.

A → α B

A → BCDE

B → a | ε

C → c | ε

D → d | ε

E → e

FIRST (A) = {a, c, d, e}

First (L

Follow

Foll

Foll

First (L) $\Rightarrow$ First (S) = {(a}

Follow T':-

$T \to F T'$ ✓

$T' \to * F T'$ ×

β

$T \to F T'$

$A \quad \alpha \; B \; \beta$

Follow (E)

$T \to \underline{F T'}$

$T' \to * F T'$

$T \to F T'$

$A \to \alpha \; B \; \beta$

Follow (F) = FIRST (T')

$T \to F T'$

= (*, ϵ)

$\to F *$.

= {*, +,), \$}

$T \to F \epsilon$

$T \to F$.