UNIT-2

REGULAR EXPRESSIONS AND PROPERTIES OF REGULAR LANGUAGES

REGULAR EXPRESSIONS:

The language accepted by finite automata are easily described by simple expressions called regular expression.

OPERATIONS OF REGULAR EXPRESSIONS:

* union
* concatenation
* closure
  - star
  - kleene

⇒ The union of two languages L_1 and L_2 is $L_1 \cup L_2$, is the set of strings that are either in L_1 (or) L_2 (or) both in L_1 and L_2

Eg:
$L_1 = \{0, 11, 110\}$
$L_2 = \{\epsilon, 0, 00\}$
$L_1 \cup L_2 = \{\epsilon, 0, 00, 11, 110\}$

⇒ The concatenation of two languages L_1 & L_2 is, $L_1 \cdot L_2$.

Eg:
$L_1 = \{001, 10, 111\}$
$L_2 = \{\epsilon, 001\}$
$L_1 \cdot L_2 = \{001, \epsilon, 10\epsilon, 111\epsilon, 001001, 10001, 111001\}$

⇒ The kleene closure of a language L is denoted L^*, is defined as the set of strings that are formed by taking any no. of strings from L.

Eg:
$L_1 = \{10, 1\}$
$L_1^* = \{\epsilon, 10, 1, 101, 110, 011, 01 \ldots\ldots\}$
$L_1^+ = \{10, 1, 101, 110, 011, 01 \ldots\}$

$L_2 = \{ab, b\}$
$L_2^+ = \{abb, abb, bba, \ldots\}$

CONSTRUCTION OF REGULAR EXPRESSION:

⇒ Let Σ be an alphabet, the regular expression over Σ and the sets that they denoted are defined as,

i) Basis

the constant ϵ and ϕ are the regular expressions denoting the languages, $L(\epsilon) = \epsilon$ and $L(\phi) = \phi$.

⇒ For each 'a' in Σ, 'a' is a regular expression denotes the set $\{a\}$.

⇒ A variable represented in uppercase like 'L' is any language.

ii) Induction

If E and F are regular expressions then,

$E \cup F = e+f$.

$E \cdot F = e \cdot f$

$E^* = e^*$

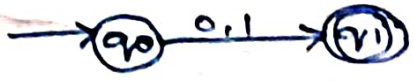
PRECEDENCE (OR) PRIORITY OF REGULAR EXPRESSION OPERATORS:

1. * operator is having the highest precedence.
2. Next to * is the . (dot) operator, after grouping all * we need to group . (dot)
3. + operator, it should be grouped from left.

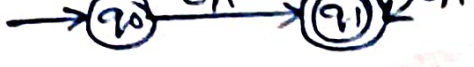
Eg: i) 0^* =

ii) 01^* =

iii) $(0+1)$ =



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



1. The set of strings over $\Sigma = \{0,1\}$ with atmost one pair of consecutives.

Sol:

Regular expression: $(0+1)^*00(0+1)^*$

(01)

$(0+1)^*00$

A set of strings 0's and 1's whose no. of 0's is divisible by 5. what is the regular expression?

sol:

Regular expression: $(00000+1)^*$

5 - zeroes → divisible by 5

3. A set of strings of A's and B's with even no. of B's. what is the regular expression?

sol:

Regular expression: $(bb+a)^*abb$

30/8/21

4. A set of strings a and b with atleast one pair of ab.

Sol:

$a(a+b)^*ab$

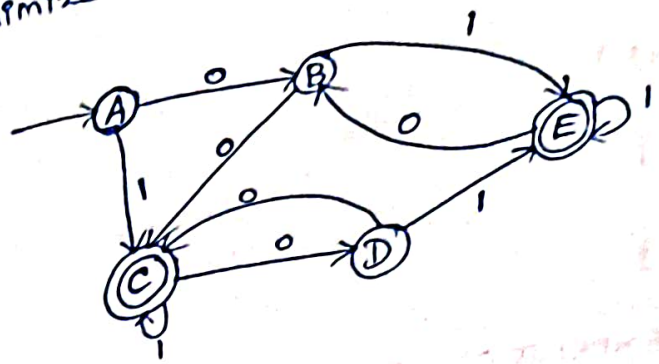
5. $(0^*1^*)^*000(0+1)^*$. write the possible language?

sol: A set of strings of 0's and 1's with the substring 000.

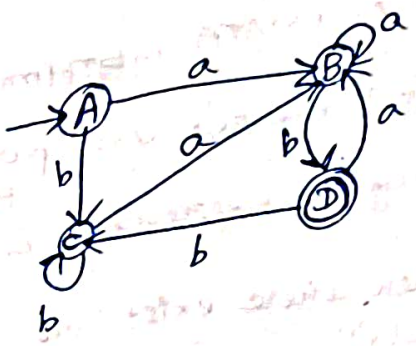
6) $a^*b + b^*a$. write the possible language.

sol: A set of strings starting and ending with a.

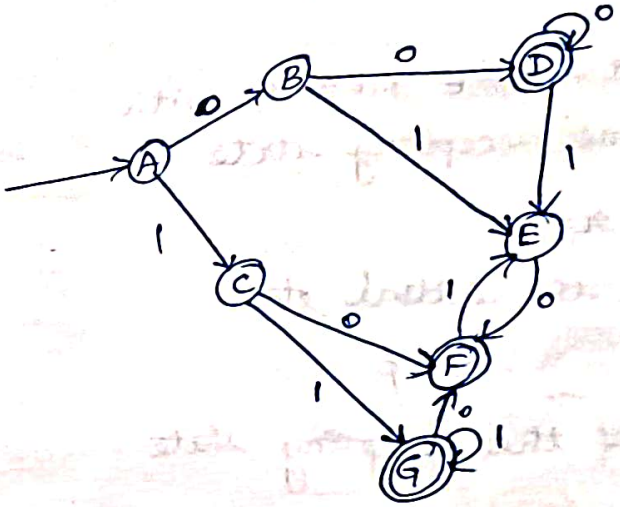
1) Minimize the DFA



2)



3)



4) NFA with ∈ to DFA

	∈	a	b	c
→ p	ϕ	{p}	{q}	{r}
q	{p}	{q}	{r}	{ϕ}
*r	{q}	{r}	{ϕ}	{p}

5)

	∈	a	b	c
→p	{q}	{p}	ϕ	ϕ
q	{r}	ϕ	{q}	ϕ
*r	ϕ	ϕ	ϕ	{r}

NFA TO DFA

	0	1
p	{q, s}	{q}
*q	{r}	{q, r}
r	{s}	{p}
*s	{ϕ}	{p}

4/9/24

CONVERTING REGULAR EXPRESSION TO AUTOMATA METHODS:

Thomson's construction method (or) kleene's first part theorem

Every language defined by a regular expression is also defined by a finite automata.

[Let R be a regular expression then there exist an NFA with ϵ transition that accepts L(r)] to prove

Proof:

To prove L = L(E) for some ϵ NFA E with

i) with exactly one accepting state.

r = e

ii) No arcs from the initial state

r = ϕ

iii) No arcs out of the accepting state.

r = a

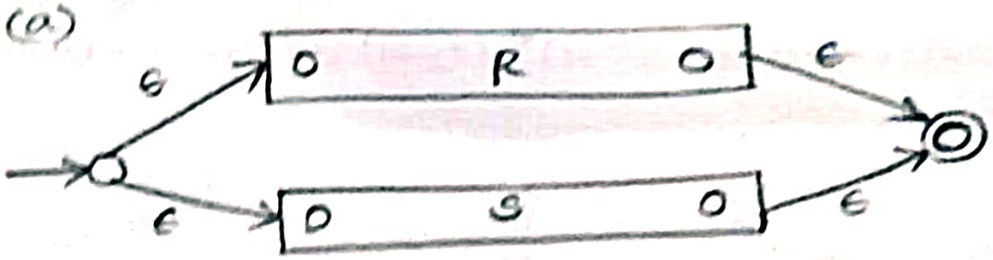
Basis:

From the above figures it is clear that expressions are must be ϵ, ϕ or 'a' for some 'a' in ϵ

Induction:

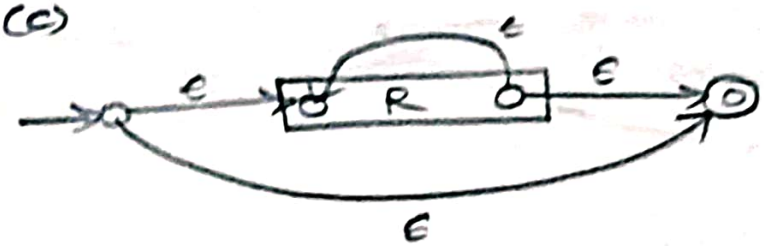
Assume that the theorem is true for the immediate sub expression of a given regular expression, (i.e) the languages of these sub expression also the language of ϵ NFA with single accepting state.

(a)



(b)

(c)



a) The expression is R+S for some small expression R & S write from the start state, we can go to the start state of either the automata of R (or) automata for S. Finally reach the accepting state of automata.

The Language of automata is $L(R) \cup L(S)$.

b) The expression is R.S, for the sub expression R and S, the state of the first automaton becomes the state (start) of the whole and accepting state two automatan becomes accepting state of whole.

The Language of automata is concatenation of 2 regular expression, $L(R) \cdot L(S)$

c) The expression is given by R^*. Here the automaton allows us to go either.

i) directly from start state to the accepting state along a path labelled ϵ where ϵ belongs to R^*.

ii) To the start state of the automata R, through that automata one or more times and then to the accepting state of R, RR, RRR, RRRR, ... which belongs to R^*.

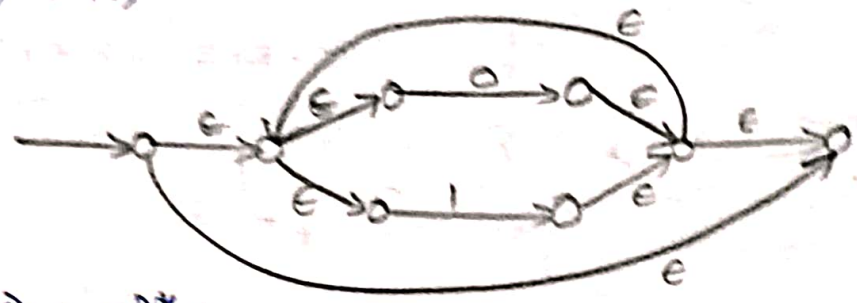
∴Thus the automata satisfy the three conditions given in inductive hypothesis one accepting state, with no arcs into the initial state or out of the accepting state.

Eg:

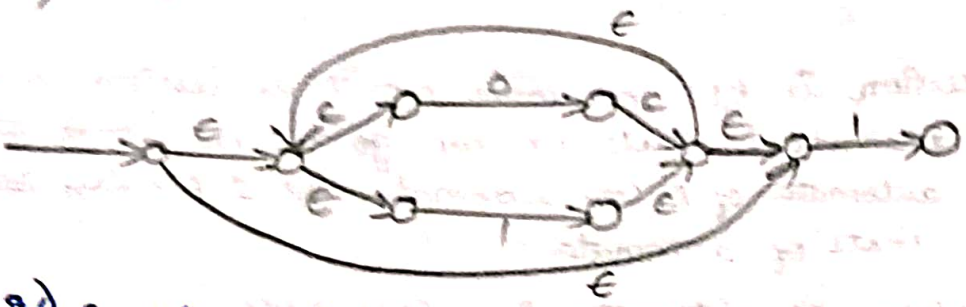
1) Convert the regular expression $(0+1)^*1(0+1)$ to an ϵ-NFA

i) $(0+1)$

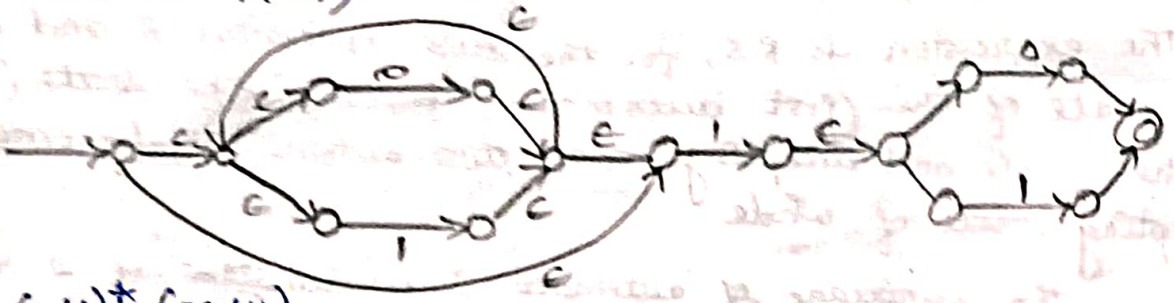
ii) $(0+1)^*$



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

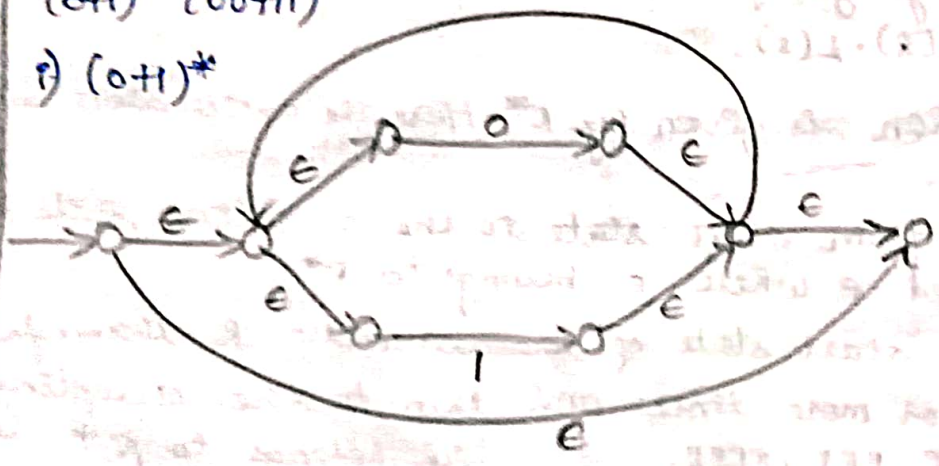


iv) $(0+1)^*1(0+1)$

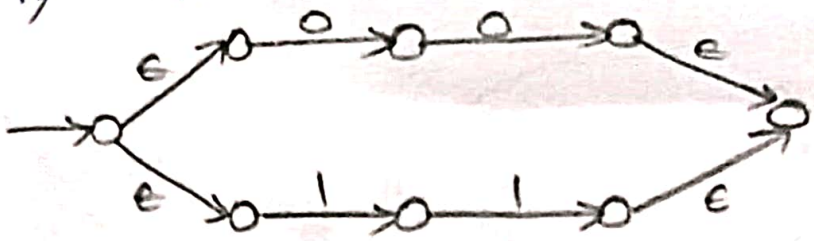


2) $(0+1)^*(00+11)$

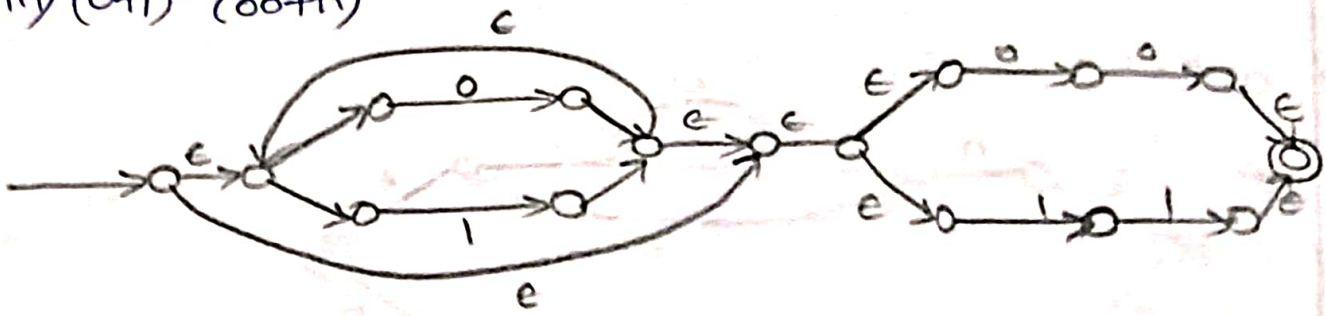
i) $(0+1)^*$



ii) 00+11.

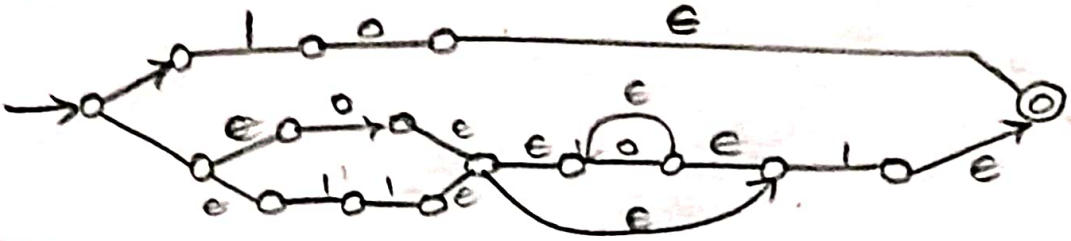


iii) (0+1)* (00+11)

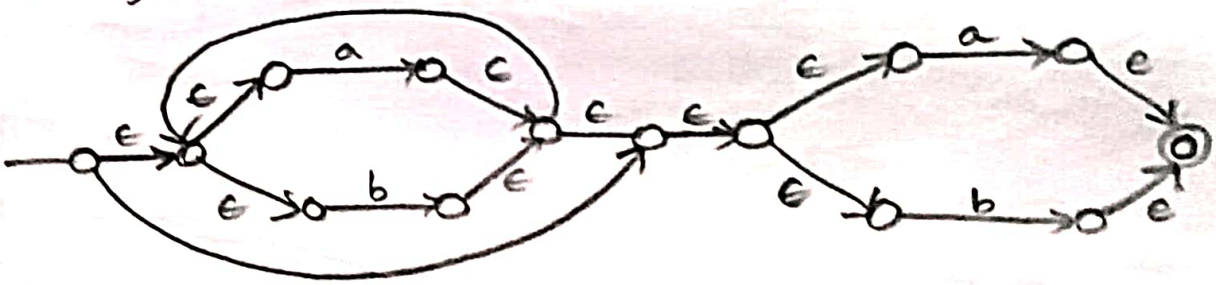


3) i) 10

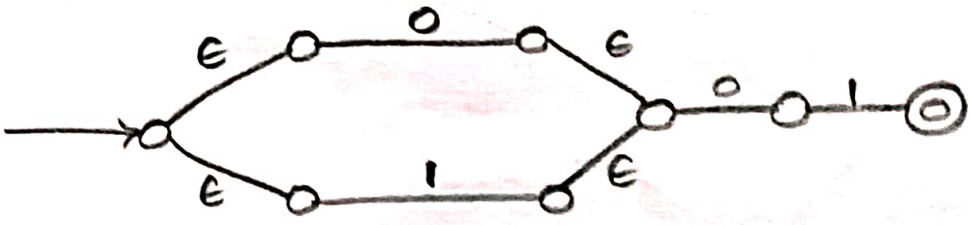
10 + (0+11)0* 1



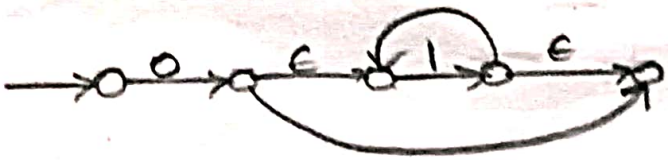
4) (a+b)*a+b



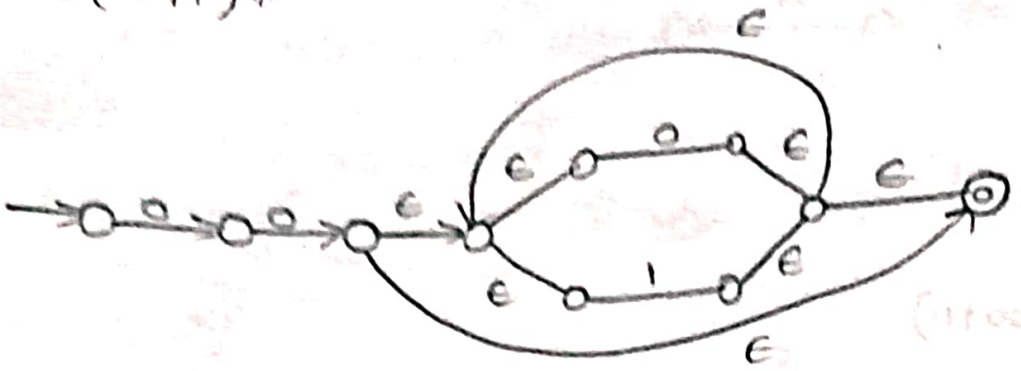
5) (0+1).01



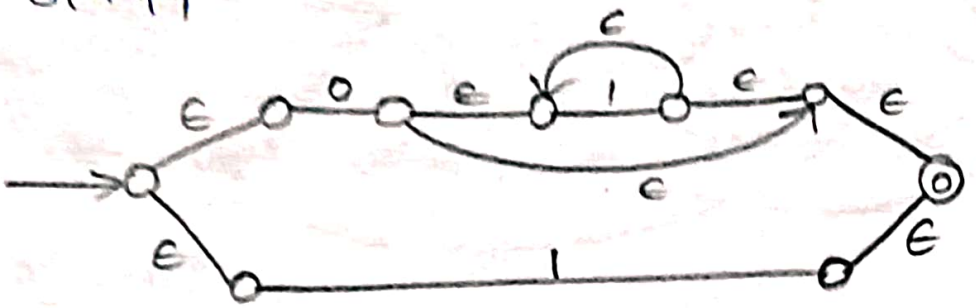
6) 01*

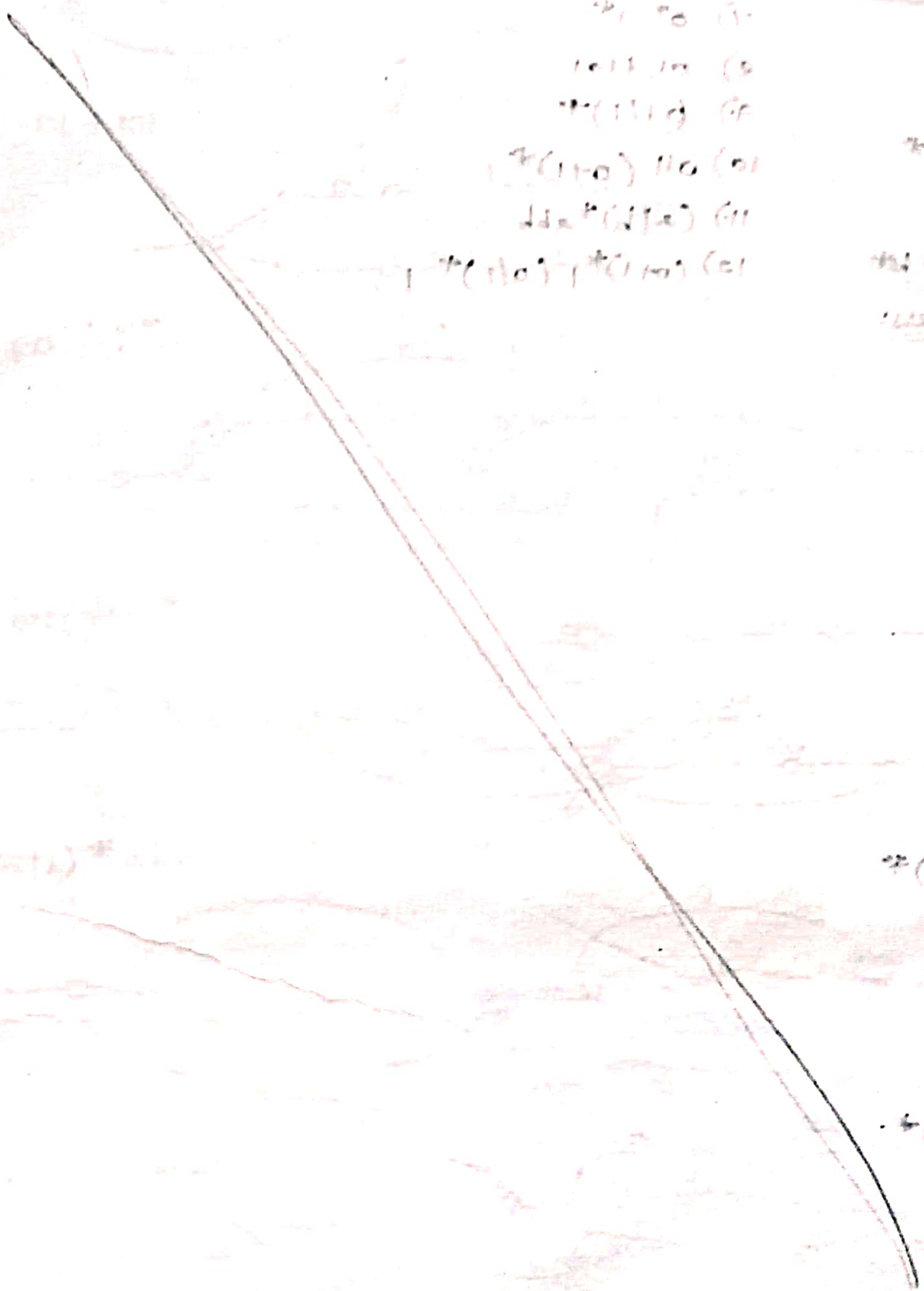


7) $00(0+1)^*$



8) 01^*+1



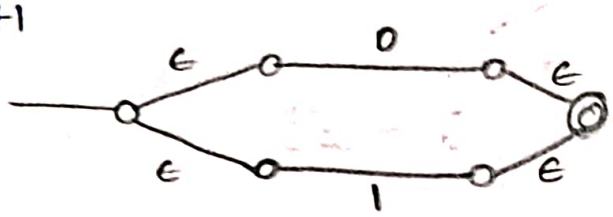


5/9/24
Thursday.

1) 0+1
2) ab
3) b*
4) (0+1)*
5) (01)*
6) a* +b*
7) 0* 1*
8) 01 +101
9) (0|1)*
10) 011 (0+1)*
11) (a|b)*abb
12) (0+1)* | (0|1)* |

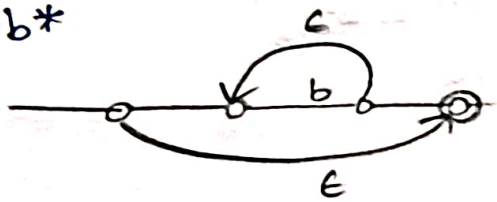
Answers:

1) 0+1

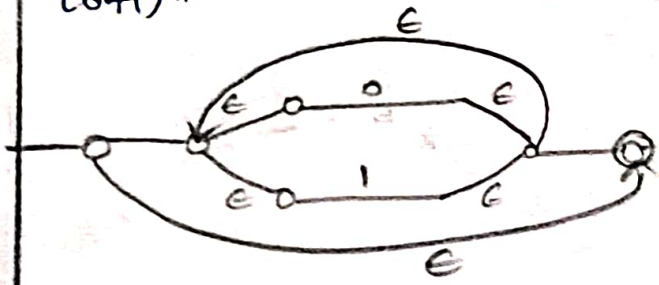


2) ab

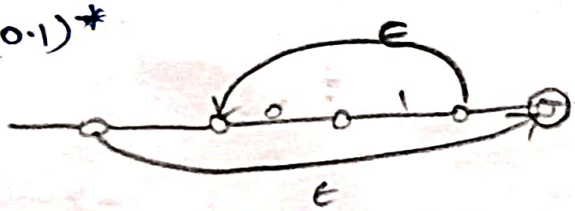
3) b*



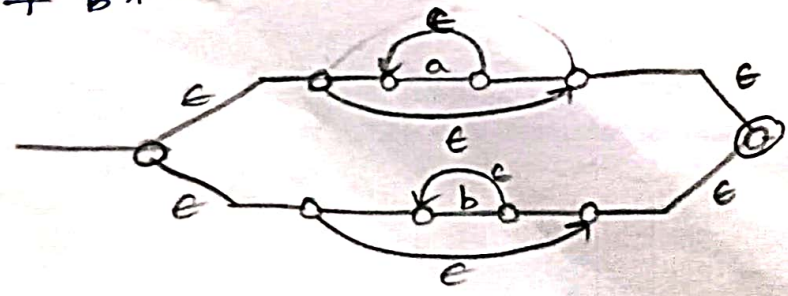
4) (0+1)*



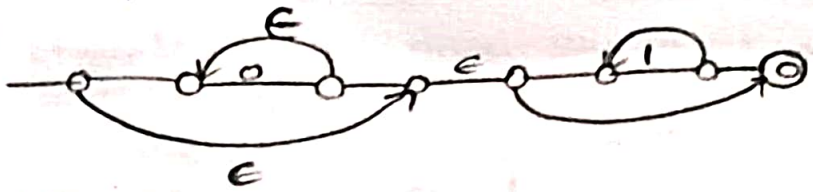
5) (0.1)*



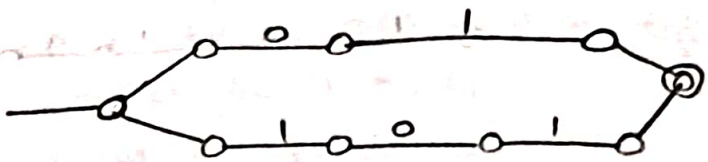
6) a* + b*



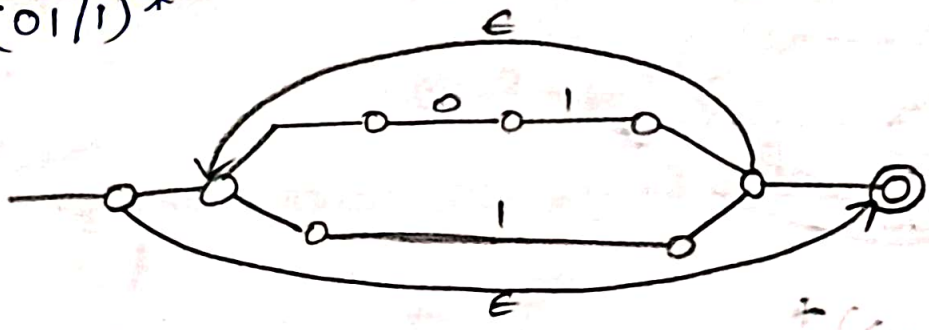
7) $0^* 1^*$



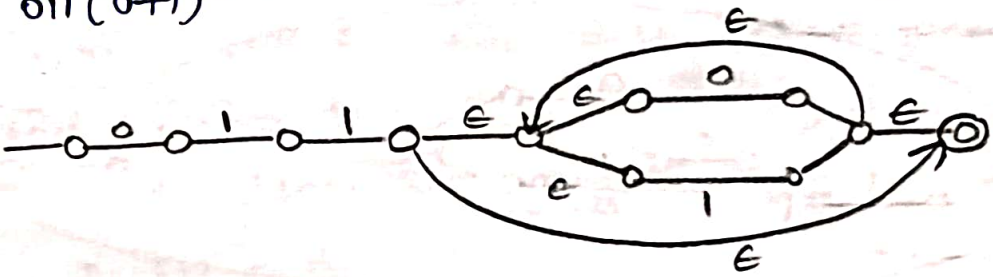
8) $01 + 101$



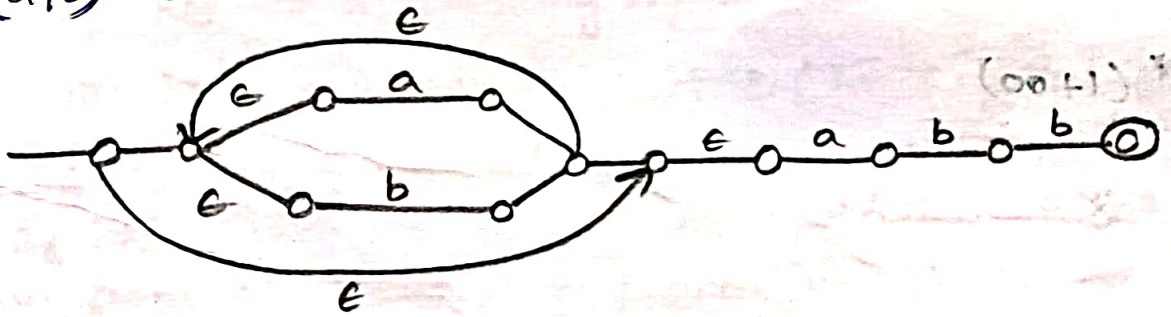
9) $(01/1)^*$



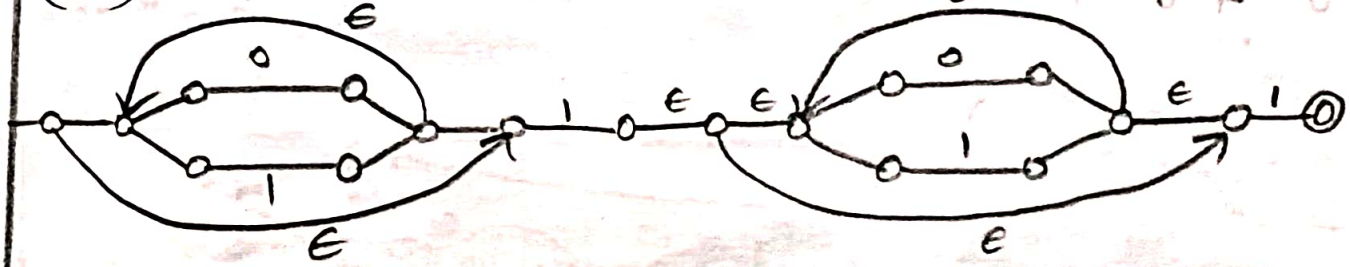
10) $011(0+1)^*$



11) $(a|b)^* abb$



12) $(0+1)^* 1 (0|1)^* 1$



13.) $(ab+c^*)^* b$

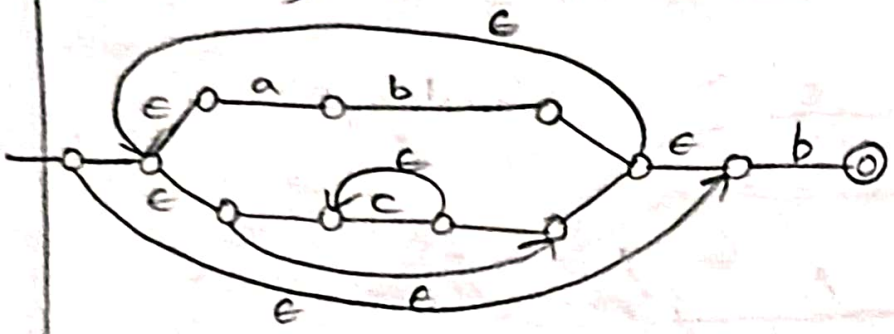
14.) $((0+1)(0+1)(0+1))^*$

15.) $(01)^*(1+00)$

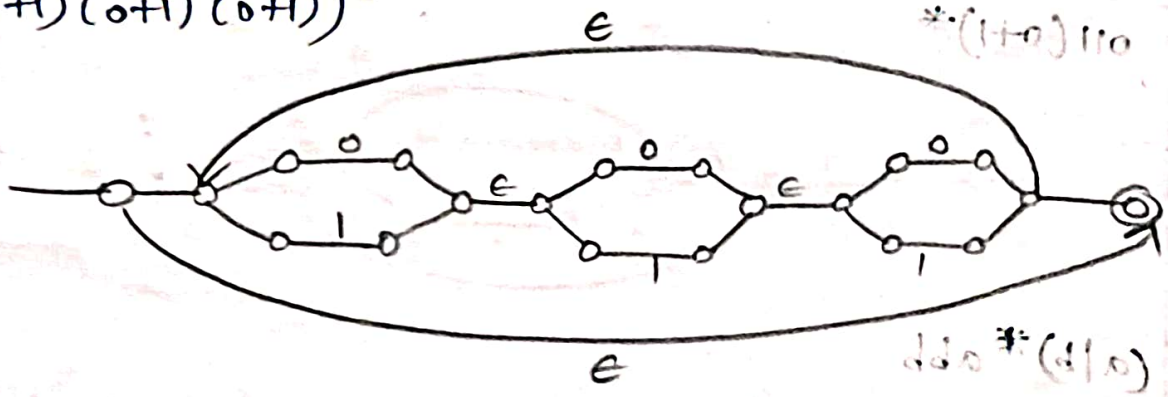
16.) $0^* + 0^* 11^*$

Answers:

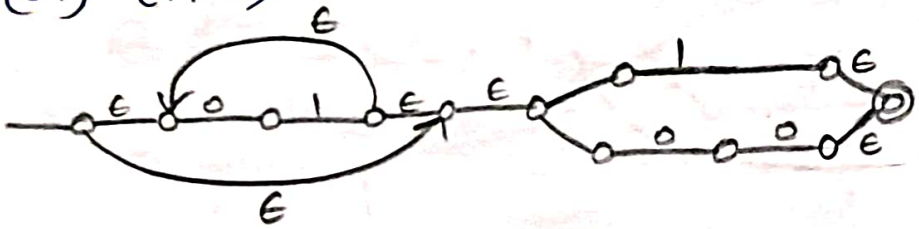
13.) $(ab+c^*)^* b$



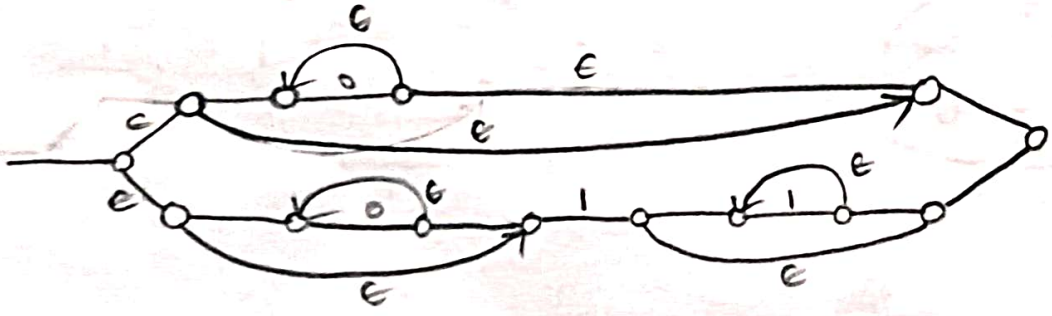
14.) $((0+1)(0+1)(0+1))^*$



15.) $(01)^*(1+00)$



16.) $0^* + 0^* 11^*$



DFA TO REGULAR EXPRESSION:

Kleene's Second Part Theorem:

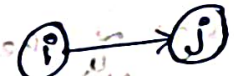
If $L = L(A)$ for some DFA 'A', then there is a regular expression R such that, $L = L(R)$.

PROOF:

Let A be a DFA which defines the language L. Let the states be {1, 2, 3, ..., n}. For constructing RE, $R_{ij}^{(K)}$ – a set of strings w that takes the DFA from state i to state j without going to any state labelled greater than K.

Basis:

If K=0, there will be no intermediate nodes between i and j.

i) 

ii)

↳ There will be two cases, an arc from node i to j and an arc from node to itself.

Then the transition may be,

i) $R_{ij}^{(0)} = \phi$

ii) $R_{ij}^{(0)} = a$ (ie, any symbol)

iii) $R_{ij}^{(0)} = a_1 + a_2 + \dots + a_l$ (i.e, l symbol from i to j)

If $i = j$, then $R_{ij}^{(0)} = \epsilon + a$.

If $i \neq j$, then the only possibility is, there must be an arc from the state i to j. (i.e) $R_{ij}^{(0)} = a_1 + a_2 + \dots + a_l$.

If there is no any direct arc from i to j, then, $R_{ij}^{(0)} = \phi$

Induction:

Let there is a path from state i to j then,

i) suppose if there is a path does not go through the state K then, $R_{ij}^{(K)} = R_{ij}^{(K-1)}$

ii) If the path goes through the state K atleast once then break the path into several pieces.

The set of tables for all paths is represented by the regular expression, $R_{ik}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$

By combining the two parts by going through k and not going through k, the regular expression can be constructed,

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$$

By induction hypothesis we can conclude that the regular expression $R_{ij}^{(n)}$ can be constructed for all values of i, j and n.

The regular expression for the language of the automata is the sum of add expression $R_{ij}^{(n)}$ such that state 'i' is the start state and state 'j' is the accepting state.

NOTE: $R_{ij}^{(0)}$

MINIMIZATION RULES:

1. $\phi + R = R$
2. $\phi R = R\phi = \phi$
3. $\epsilon R = R\epsilon = R$
4. $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$
5. $R + R = R$
6. $R^* R^* = R^*$
7. $RR^* = R^* R$
8. $(R^*)^* = R^*$
9. $\epsilon + RR^* = \epsilon + R^* R = R^*$
10. $(PQ)^* P = P(QP)^*$
11. $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
12. $(P+Q)R = PR + QR$
13. $R^* + \epsilon = R^*$
14. $(R+\epsilon)^* = R^*$
15. $R^* R + R = R^* R$
16. $(R+\epsilon)R^* = R^*(R+\epsilon) = R^*$
17. $(R+\epsilon)(R+\epsilon)^*(R+\epsilon) = R^*$
18. $0 + \epsilon = \epsilon$
19. $(\epsilon + 1)^* = 1^*$
20. $1^*(\epsilon + 1) = 1^*$
21. $(\epsilon + 1) + 1^* = 1^*$
22. $0 + 1^* 0 = 1^* 0$
23. $0 + 01^* = 01^*$
24. $\epsilon + 00^* = 0^*$
25. $(1+0)^* = (1^* 0^*)^*$
26. $\phi 0 = \phi$, $\phi + 0 = 0$
27. $\phi^* = \epsilon$
28. $11^* = 1^*$
29. $1^* 1 = 1^*$

30. $(01)^* \neq 0^* 1^*$ e.g. $01 + 1^* = 1^*$

31. $\epsilon 0 = 0$

32. $(01)^* 0 = 0(10)^*$

33. $(0^* 1)^* 0^* = (0+1)^*$

Problem 1

Convert the following to a regular expression.

i - initial state
j - ending state
k - no. of states

Find $R_{ij}^{(k)}$ for all values of i, j and k.

Sol:

Basis: $k=0$.

$$R_{ij}^{(k)} = R_{ij}^{(0)}$$

~~(ε only for self loop) and also for final state~~

(ε only for i=j)

$i=1, j=1$

$$R_{11}^{(0)} = 1 + \epsilon$$

$i=1, j=2$

$$R_{12}^{(0)} = 0$$

$i=2, j=1$

$$R_{21}^{(0)} = \phi$$

$i=2, j=2$

$$R_{22}^{(0)} = 0 + 1 + \epsilon$$

Induction:

$k=1$

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

when $i=j$,

$i=1, j=1$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{1k}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$R_{11}^{(1)} = 1+\epsilon + 1+\epsilon \ (1+\epsilon)^* (1+\epsilon)$$

$$= (1+\epsilon) + (1+\epsilon)\, 1^* (1+\epsilon)$$

$$= (1+\epsilon) + 1^* (1+\epsilon)$$

$$= (1+\epsilon) + 1^*$$

$$R_{11}^{(1)} = \boxed{1^*}$$

when, $i=1, j=2$

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

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

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

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

$$R_{12}^{(1)} = \boxed{1^*0}$$

when $i=2, j=1$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$= \phi + \phi\, (1+\epsilon)^* (1+\epsilon)$$

$$= \phi + \phi\, 1^* (1+\epsilon)$$

$$= \phi + \phi\, 1^* = \boxed{\phi}$$

11/9/24 Wednesday when $i=2, j=2$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

$$= (0+1+\epsilon) + \phi\, (1+\epsilon)^* (0)$$

$$= (0+1+\epsilon) + \phi\, 1^* (0)$$

$$= (0+1+\epsilon) + \phi \qquad (\because R\phi = \phi R = \phi)$$

$$R_{22}^{(1)} = \boxed{(\epsilon+0+1)}$$

$k=2$.

$$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)} \left(R_{22}^{(1)}\right)^* R_{2j}^{(1)}$$

when $i=1, j=1$

$$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^* R_{21}^{(1)}$$

$$= 1^* + (1^*0)(\epsilon+0+1)^* \phi$$

$$= 1^* + \phi$$

$$R_{11}^{(2)} = \boxed{1^*}$$

when $i=1, j=2$

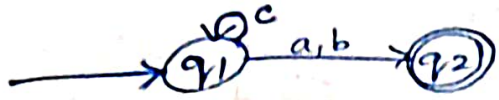
$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$$

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

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

$$R_{12}^{(2)} = \boxed{1^*0\, (0+1)^*}$$

$\therefore$ R_{ij} is $R_{12} = 1+0(0+11)^*$

a) 

Sol:

The DFA has one final state, $\therefore R_{ij}^{(k)} = R_{12}^{(2)}$.

i) Basis:

$K=0$

$i=1, j=1$

$R_{11}^{(0)} = c + \epsilon$

$i=1, j=2$ $R_{12}^{(0)} = a+b$

$i=2, j=1$

$R_{21}^{(0)} = \phi$

$i=2, j=2$

$R_{22}^{(0)} = \phi + \epsilon$

ii) Induction:

$k=1$

$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{i1}^{(0)} \left(R_{11}^{(0)}\right)^* R_{1j}^{(0)}$

$i=1, j=1$

$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$

$= (c+\epsilon) + (c+\epsilon)(c+\epsilon)^*(c+\epsilon)$ - $(c+\epsilon)[\epsilon + (c+\epsilon)(c+\epsilon)^*]$

$= (c+\epsilon) + (c+\epsilon)(c+\epsilon)^*$ - $(c+\epsilon)^*(c+\epsilon)$

$= (c+\epsilon) + c^*$ - $c^*(c+\epsilon)$

$= (c+\epsilon) + c^*$ - c^*

$R_{11}^{(1)} = \boxed{c^*}$

$i=2, j=2$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$

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

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

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

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

$R_{12}^{(1)} = \boxed{c^*(a+b)}$

$i=2, j=1$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}\left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$= \phi + \phi(\epsilon+\epsilon)^*(\epsilon+\epsilon) \quad [\because \phi r = \phi]$$

$$= \phi + \phi$$

$$R_{21}^{(1)} = \boxed{\phi}$$

$i=2, j=2$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)}$$

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

$$= \phi + \epsilon + \phi$$

$$= \phi + \epsilon$$

$$R_{22}^{(1)} = \phi \boxed{\epsilon}$$

$k=2$

$i=1, j=2$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)}$$

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

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

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

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

$$R_{12}^{(2)} = c^*(a+b)$$

$\therefore R_{ij}^{(k)}$ is $R_{12}^{(2)} = c^*(a+b)$

3)

i) Basis:

$k=0$

$i=1, j=1$

$$R_{11}^{(0)} = \phi + \epsilon$$

$$R_{12}^{(0)} = 1$$

$$R_{21}^{(0)} = 0$$

$$R_{22}^{(0)} = 0+1+\epsilon$$

ii) Induction

$k=1$

$i=1, j=1$

$$R_{ij}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* \left(R_{11}^{(0)}\right)$$

$$= \phi + \phi (\phi^*)^* (\phi)$$

$$= \phi + \phi$$

$$R_{11}^{(1)} = \phi$$

$i=1, j=2$

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* \left(R_{12}^{(0)}\right)$$

$$= 1 + \phi (\phi)^* (1)$$

$$R_{12}^{(1)} = 1$$

$i=2, j=1$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$= \phi + \phi (\phi)^* (\phi)$$

$$R_{21}^{(1)} = \phi$$

$i=2, j=2$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

$$= (0+1+\epsilon) + \phi (\phi)^* (1)$$

$$R_{22}^{(1)} = 0+1+\epsilon$$

$k=2$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$$

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

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

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

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

14/9/22 Saturday

1)

00^*11^*

Basis:

$k = 0$

$i = 1, j = 1$

$R_{11}^{(0)} = \phi + \epsilon$

$R_{12}^{(0)} = 0 + 1$

$R_{21}^{(0)} = \phi$

$R_{22}^{(0)} = \phi + \epsilon$

Induction:

$k = 1$

$i = 1, j = 1$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* (R_{11}^{(0)})$$

$$= (\phi + \epsilon) + (\phi + \epsilon)(\phi + \epsilon)^* (\phi + \epsilon)$$

$$= \epsilon + \phi$$

$$R_{11}^{(1)} = \epsilon$$

$i = 1, j = 2$

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* (R_{12}^{(0)})$$

$$= (0+1) + (\phi + \epsilon)(\phi + \epsilon)^* (0+1)$$

$$R_{12}^{(1)} = 0 + 1$$

$i = 2, j = 1$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} (R_{11}^{(0)})^* R_{11}^{(0)}$$

$$= \phi + \phi (\phi + \epsilon)^* \phi + \epsilon$$

$$= \phi + \phi$$

$$R_{21}^{(1)} = \phi$$

$i = 2, j = 2$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)} (R_{11}^{(0)})^* R_{12}^{(0)}$$

$$= (\phi + \epsilon) + \phi (\phi + \epsilon)^* (0+1)$$

$$= \phi + \epsilon$$

$$R_{22}^{(1)} = \epsilon$$

$k=2$

$i=1, j=1$

$$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{12}^{(1)}$$

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

$$R_{11}^{(2)} = \epsilon$$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$$

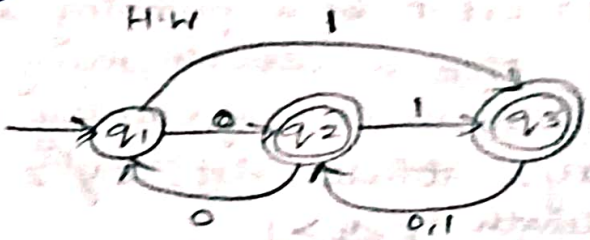
$$= (0+1) + (0+1)(\epsilon)^* (\epsilon)$$

$$R_{12}^{(2)} = 0+1$$

∴ The answer is, $R_{11}^{(2)} + R_{12}^{(2)} = \epsilon + 0 + 1 = \boxed{0+1}$ //

5)

H.W



$R_{12}^{(2)}$ $R_{13}^{(3)}$

00^*11^*

Basis:

$k=0$

$i=1, j=1$ $R_{11}^{(0)} = 1 + \epsilon$

$i=1, j=2 \Rightarrow R_{12}^{(0)} = 0$

$i=1, j=3 \Rightarrow R_{13}^{(0)} = \phi$

$i=2, j=1 \Rightarrow R_{21}^{(0)} = \phi$

$i=2, j=2 \Rightarrow R_{22}^{(0)} = \phi + \epsilon$

$i=2, j=3 \Rightarrow R_{23}^{(0)} = 1$

$i=3, j=1 \Rightarrow R_{31}^{(0)} = \phi$

$i=3, j=2 \Rightarrow R_{32}^{(0)} = 1$

$i=3, j=3 \Rightarrow R_{33}^{(0)} = 0 + \epsilon$

Induction:

$k=1$

$i=1, j=1$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* (R_{11}^{(0)})$$

$$= (1+\epsilon) + (1+\epsilon)(1+\epsilon)^* (1+\epsilon)$$

$$= (1+\epsilon) + 1^*$$

$$R_{11}^{(1)} = \boxed{1^*}$$

continued...
upside back

14/9/22 Saturday 18/9/22 Wednesday.

PROVING LANGUAGES NOT TO BE REGULAR (PUMPING LEMMA):

The powerful technique which is used to prove that certain languages are not regular is pumping lemma.

Let L be a regular language then there exist a constant n such that for every string in L, w is splitted into three substrings such as $w = xyz$ where,

i) $y \neq \epsilon$

ii) $|xy| \leq n$

iii) $xy^k z \in L, \forall k \geq 0$

Steps to show that L is not regular using pumping lemma

step 1 : Assume that L is regular.

step 2 : Let n be a pumping lemma constant

step 3 : pick a suitable string (i.e), $|w| \geq n$

step 4 : show that for every ii decomposition of string w into xyz there exist, xy^kz (i.e) that is not in L with $|xy| \leq n$ and length of $y \geq 1$.

step 5 : since xy^kz is not in L, it can be concluded as L is not regular.

Application of pumping lemma

→ It is used to prove that the certain set are the language is not regular.

1) show that $L = \{ 0^n 1^n \mid n \geq 1 \}$ is not regular.

Sol:

Step 1 : Assume, $L = \{0^n 1^n \mid n \geq 1\}$ is regular.

Step 2 : Let n be a pumping lemma constant.

Step 3 : Let $L = \{0^n 1^n \mid n \geq 1\}$

$n = 1$, $L = 0^1 1^1 = 01$

$n = 2$, $L = 0^2 1^2 = 0011$

$n = 3$, $L = 0^3 1^3 = 000111$

n = 2 is mostly applicable / works for all types of problem

∴ Suitable string = 0011

step 4: Divide the string, w = 0011 into three parts

$x = 0$
$y = 0$
$z = 11$

condition : 1 → $y \neq \epsilon$
$= 0 \neq \epsilon$

∴ The ~~string y = 0~~ first condition is true

condition : 2 → $|xy| \leq n$
$= |00| \leq 2$
$= |2| \leq 2$

∴ Second condition is true.

condition : 3 → $xy^k z \in L$, $\forall k \geq 0$

$k=0$ $= 00^0 11 \in L$
$= 011 \notin L$

∴ $011 \notin L$ (is not in Language)

$k=1$
$= 00^1 11 \in L$
$= 0011 \in L$ (satisfied)

$k=2$ $\Rightarrow 00^2 11 \in L$
$\Rightarrow 0\,00\,11 \notin L$
(not belongs to L)
not satisfied

step 5: since $xy^k z$ is not in L, the language $L = \{0^n 1^n \mid n \geq 1\}$ is not regular.

2) show that $L = \{0^i 1^i \mid i \geq 1\}$ is not regular.

sol:

step 1 : Assume, $L = \{0^i 1^i \mid i \geq 1\}$ is regular.

step 2 : Let i be a pumping lemma constant.

step 3 : Let $L = \{0^i 1^i \mid i \geq 1\}$

$n=1$, $L = 0^1 1^1 = 01$
$n=2$, $L = 0^2 1^2 = 0011$
$n=3$, $L = 0^3 1^3 = 000111$

∴ suitable string = 0011

3) $L = \{0^n 1^{3n} \mid n \geq 1\}$
4) $L = \{a^n b^n \mid n \geq 1\}$
5) $L = \{0^n 1^m \mid n > m \text{ & } n \geq 0\}$

14/9/22
Saturday

Step 4: Divide the string, $w = 0011$ into

$x = 0$
$y = 0$
$z = 11$

condition: 1 $\rightarrow y \neq \epsilon$
$0 \neq \epsilon$
$\therefore$ satisfied

condition: 2 $\rightarrow |xy| \leq n$
$= |00| \leq 2$
$= 2 \leq 2$
$\therefore$ satisfied.

condition: 3 $\rightarrow xy^k z \in L$
$k = 0$, $00^0 11 \in L$
$011 \notin L$
$\therefore$ not satisfied.

Step 5: Since $xy^k z$ is not in L, the language, $L = \{0^n 1^n \mid n \geq 1\}$ is not regular.

3) show that $L = \{0^n 1^{2n} \mid n \geq 1\}$ is not regular.

sol:

step 1: assume, $L = \{0^n 1^{2n} \mid n \geq 1\}$ is regular.

step 2: Let n be pumping lemma constant.

Step 3: Let $L = \{0^n 1^{2n} \mid n \geq 1\}$

$n = 1$, $L = 0^1 1^2 = 011$

$n = 2$, $L = 0^2 1^4 = 001111$

$n = 3$, $L = 0^3 1^6 = 000111111$

$\therefore$ suitable string = 001111

Step 4: Divide the string, $w = 001111$ into three parts.

$x = 0$
$y = 0$
$z = 1111$

condition: 1 $\rightarrow y \neq \epsilon$
$= 0 \neq \epsilon$
$\therefore$ satisfied

condition: 2 $\rightarrow |xy| \leq n$
$= |00| \leq 2$
$= 2 \leq 2$
$\therefore$ satisfied.

condition: 3 $\rightarrow xy^k z \in L$
$k = 0$, $L = 00^0 1111 = 01111 \notin L$
$\therefore$ not satisfied.

Step 5: since $xy^k z$ is not in L, the language, $L = \{0^n 1^{2n} \mid n \geq 1\}$ is not regular.

4.) show that $L = \{0^n 1^m \mid n > m \text{ and } n \geq 0\}$ is not regular

sol:

step 1: Assume $L = \{0^n 1^m \mid n > m \text{ and } n \geq 0\}$ is regular

step 2: Let n be pumping lemma constant

step 3: Let $L = \{0^n 1^m \mid n > m \text{ and } n \geq 0\}$

$n=1, m=0 \Rightarrow L = 0^1 1^0 = 0$

$n=2, m=1 \Rightarrow L = 0^2 1^1 = 001$

$n=3, m=2 \Rightarrow L = 0^3 1^2 = 00011$

$\therefore$ suitable string is 001

step 4: Divide the string, $w = 001$

$x = 0$

$y = 0$

$z = 1$

condition: 1 $\rightarrow$ $y \neq \epsilon$

$0 \neq \epsilon$ $\therefore$ satisfied

condition: 2 $\rightarrow$ ~~$xy^k z \in L$~~ $|xy| \leq n$

$= |00| \leq n$

$= |2| \leq 2$

$\therefore$ satisfied

condition: 3 $\rightarrow$ $xy^k z \in L$

$k=0, \rightarrow 0^0 1 \in L$

$\Rightarrow 01 \notin L$

$\therefore$ not satisfied.

step 5: since $xy^k z$ is not in L, the language, $L = \{0^n 1^m \mid n > m \text{ and } n \geq 0\}$ is not regular..

23/9/24 Monday

5.) show that $L = \{a^{i^2} \mid i \geq 1\}$ is not regular.

sol:

step 1: Assume $L = \{a^{i^2} \mid i \geq 1\}$ is regular.

step 2: Let i be pumping lemma constant.

step 3: Let $L = \{a^{i^2} \mid i \geq 1\}$

$i=1 \Rightarrow a^1 \Rightarrow a$

$i=2 \Rightarrow a^4 \Rightarrow aaaa$ $\therefore$ suitable string is aaaa

$i=3 \Rightarrow a^9 \Rightarrow aaaaaaaaa$

step 4: Divide the string, $w = aaaa$

$x = a$

$y = a$

$z = aa$

condition: 1 $\rightarrow$ $y \neq \epsilon$

$a \neq \epsilon$

(satisfied)

condition: 2 $\rightarrow$ $|xy| \leq n$

$|aa| \leq 2$ (satisfied)

condition: 3 $\rightarrow$ $xy^k z \in L$

$k=0 \Rightarrow a a^0 aa = aaa$

(∴ not satisfied)

∴ Not Regular.

14/9/24 Saturday

6.) show that $L = \{a^p \mid p \text{ is a prime}\}$ is not regular.

sol:

Step 1: Assume $L = \{a^p \mid p \text{ is a prime}\}$ is regular.

Step 2: Let p be pumping lemma constant.

Step 3: Let $L = \{a^p \mid p \text{ is a prime}\}$

$p = 2 \Rightarrow a^2 \Rightarrow aa$

$p = 3 \Rightarrow a^3 \Rightarrow aaa$

$p = 5 \Rightarrow a^5 \Rightarrow aaaaa$.

$\therefore$ suitable string is aaa.

Step 4: Divide the string, $w = aaa$. $\rightarrow$ $x = a$, $y = a$, $z = a$

condition: 1 $\Rightarrow$ $y \neq \epsilon$

$a \neq \epsilon$

(satisfied)

condition: 2 $\Rightarrow$ $|xy| \leq p$

$|aa| \leq 3$

$2 \leq 3$

(satisfied)

condition: 3 $\Rightarrow$ $xy^k z \in L$

$k = 0 \Rightarrow a a^0 a \Rightarrow aa \in L$

$k = 1 \Rightarrow aa^1 a \Rightarrow aaa \in L$

$k = 2 \Rightarrow aa^2 a \Rightarrow aaaa \notin L$

(:Not satisfied)

$\therefore$ Not Regular.

7.) Show that $L = \{ww^R \mid w \in (a,b)^*\}$ is not regular. (w^R: Reverse; $(a,b)^*$: use a and b compulsorily)

8.) show that $L = \{ww \mid w \in (a+b)^*\}$ is not regular.

9.) show that $L = \{w \in (a,b) \mid w = w^R\}$ is not regular.

10.) show that $L = \{1^k \mid k = n^2, n \geq 1\}$ is not regular.

11.) show that $L = \{0^n 1^m 2^{n+m}, m,n \geq 1\}$ is not regular.

answers:

7.) $L = \underbrace{abb}_{w}\underbrace{a}_{w^R}$.

Step 1: Assume $L = \{ww^R \mid w \in (a,b)^*\}$ is regular $\therefore$

Step 2: ~~Let Divide the string, w~~

$n = 1$, abba

$n = 2$, abbabbaba

$n = 3$, abababbababa

Step 3: Divide the string, w = abba

$x = a$
$y = b$
$z = ba$

condition: 1 → $y \neq \epsilon$
$b \neq \epsilon$
(satisfied)

condition: 2 → $|xy| \leq 1$
$|ab| \leq 1$
$2 \not\leq 1$
(not satisfied) ∴ Not Regular.

8) $L = \{ww \mid w \in (a+b)^*\}$
$w = \underbrace{ab}_{w}\underbrace{ab}_{w}$.
$n = 1 \Rightarrow abab$
$n = 2 \Rightarrow abababab$
$n = 3 \Rightarrow abababababab$
Divide the string $w = abab$. $(n=1)$

$x = a$
$y = b$
$z = ab$

condition: 1 → $y \neq \epsilon$
$b \neq \epsilon$
(satisfied)

condition: 2 → $|xy| \leq n$
$|ab| \leq 1$
$2 \not\leq 1$
(∵ not satisfied)

∴ Not regular.

9) $L = \{w \in (a,b) / w = w^R\}$
$L = aaa$. (if reversed also aaa)
$n=1 \Rightarrow aaa$
$n=2 \Rightarrow aaaaaa$
$n=3 \Rightarrow aaaaaaaaa$.
Divide the string $w = aaa$. $(n=1)$

$x = a$
$y = a$
$z = a$

condition: 1 → $y \neq \epsilon$
$a \neq \epsilon$
(satisfied)

condition: 2 → $|xy| \leq n$
$|aa| \leq 1$
$2 \not\leq 1$ (∵ not satisfied) ∴ Not Regular.

14/9/24
Saturday

10) $L = \{ 1^k \mid k = n^2, n \geq 1 \}$

$n = 1 \Rightarrow 1^1 \Rightarrow 1$

$n = 2 \Rightarrow 1^4 \Rightarrow 1111$

$n = 3 \Rightarrow 1^9 \Rightarrow 111111111$

Divide the string, $w = 1111$ $(n=2)$

$x = 1$

$y = 1$

$z = 11$

condition : 1 $y \neq \epsilon$

$1 \neq \epsilon$

$(\because$ satisfied$)$

condition : 2 $|xy| \leq n$

$|11| \leq n$

$2 \leq 2$

$(\because$ satisfied$)$

condition : 3 $xy^k z \in L$

$k = 0 \Rightarrow 1\,1^0\,1 = 11 \notin L$

$(\because$ not satisfied$)$

$\therefore$ It is not regular.

11) $L = \{ 0^n 1^m 2^{n+m} \mid m, n \geq 1 \}$

$n = 1 \Rightarrow 0^1 1^1 2^2 = 0122$

$n = 2 \Rightarrow 0^2 1^2 2^4 = 00112222$

$n = 3 \Rightarrow 0^3 1^3 2^6 \Rightarrow 000111222222$

Divide the string, $w = 0122$ $(n=1)$

$x = 0$

$y = 1$

$z = 22$

condition : 1 $y \neq \epsilon$

$1 \neq \epsilon$

$(\because$ satisfied$)$

condition : 2 $|xy| \leq n$

$|01| \leq n$

$2 \nleq 1$

$(\because$ not satisfied$)$

$\therefore$ It is not regular.

12) $L = \{0^n \mid n \text{ is a perfect cube}\}$

$n=1 \Rightarrow 0^1 = 0$

$n=8 \Rightarrow 0^8 = 00000000$

$n=27 \Rightarrow 0^{27} = 000000000000000000000000000$

Divide the string, $w = 00000000$ $(n=8)$ $x=0$, $y=0$, $z=000000$

condition: 1 $y \neq \epsilon$

$0 \neq \epsilon$

(∵ satisfied)

condition: 2 $|xy| \leq n$

$|00| \leq 8$

$2 \leq 8$

(∵ satisfied)

condition: 3 $xy^kz \in L$

$k=0, \Rightarrow 00^0000000 = 0000000 \notin L$

(∵ not satisfied)

∴ It is not regular.

CLOSURE PROPERTIES OF REGULAR LANGUAGES:

If the classes of language is closed under a particular operation means that the resultant of an operation is same class of language, then the property is called as closure property.

The principle closure properties of regular languages are as follows:

1. The union of two regular language is regular.
2. The intersection of two regular language is regular.
3. The complement of two regular language is regular.
4. The difference of two regular language is regular.
5. The reversal of a regular language is regular.
6. The closure of a regular language is regular.
7. The concatenation of a regular language is regular.
8. The Homomorphism of a regular language is regular.
9. The Inverse homomorphism of a regular language is regular.

Theorem: 1

Union of two regular language is regular.

PROOF:

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

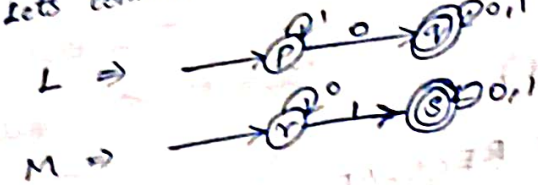
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ accepts the language L_1 and L_2 respectively.

Problem Solving

construct a finite automata M that accepts $L_1 \cup L_2$. (i.e) $M = (Q, \Sigma, \delta, q_0, F)$ where,

1. $Q = Q_1 \times Q_2$
2. $\Sigma = \Sigma$
3. $q_0 = \{q_1, q_3\}$
4. δ is defined by $\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a))$
5. If $F = \{(p,q) \mid p \in F_1 \text{ (or) } q \in F_2\}$

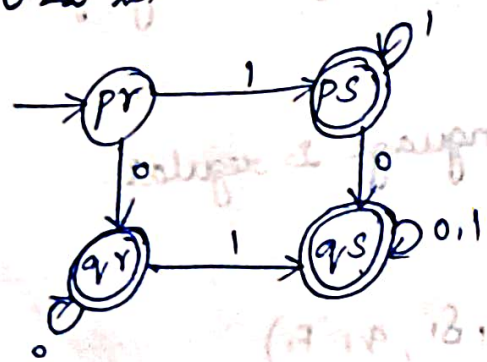
Lets consider the automata: L



construct a machine $M = \{Q, \Sigma, \delta, q_0, F\}$ where,

1. $Q = Q_1 \times Q_2$
   $= \{pr, ps, qr, qs\}$
2. $\Sigma = \{0,1\}$
3. $q_0 = \{pr\}$
4. $\delta = (pr,0) = \{qr\}$, $(pr,1) = \{ps\}$
   $(ps,0) = \{qs\}$, $(ps,1) = \{ps\}$
   $(qr,0) = \{qr\}$, $(qr,1) = \{qs\}$
   $(qs,0) = \{qs\}$, $(qs,1) = \{qs\}$
5. $F = \{(pq) \mid p \in F_1 \text{ (or) } q \in F_2\}$
   $F = \{ps, qr, qs\}$

$\therefore L_1 \cup L_2$ is,



$\therefore$ The union of two regular language is regular

Theorem : 2

Intersection of two regular language is regular

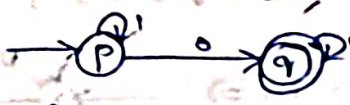
PROOF: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

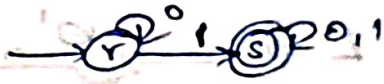
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ accepts the language L_1 and L_2 respectively.

construct a finite automata M that accepts the language $L_1 \cap L_2$ (ie), $M = (Q, \Sigma, \delta, q, F)$ where,

1. $Q = Q_1 \times Q_2$
2. $\Sigma = \Sigma$
3. $q_0 = \{q_1, q_2\}$
4. δ is defined by $(\delta(p,q), a) = (\delta_1(p,a), \delta_2(q,a))$
5. If $F = \{(p,q) \mid p \in F_1$ (and) $q \in F_2\}$

Lets consider the automata,

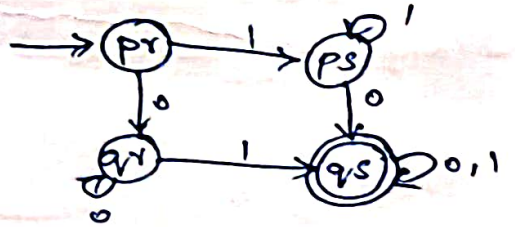
$L \Rightarrow$ 

$M \Rightarrow$ 

Construct a machine, $M = \{Q, \Sigma, \delta, q, F\}$ where,

1. $Q = Q_1 \times Q_2$
   $= \{pr, ps, qr, qs\}$
2. $\Sigma = \{0, 1\}$
3. $q_0 = \{pr\}$
4. $\delta \Rightarrow (pr,0) = \{qr\}$ $(pr,1) = \{ps\}$
   $(ps,0) = \{qs\}$ $(ps,1) = \{ps\}$
   $(qr,0) = \{qr\}$ $(qr,1) = \{qs\}$
   $(qs,0) = \{qs\}$ $(qs,1) = \{qs\}$
5. $F = \{(pq) \mid p \in F_1$ (and) $q \in F_2\}$
   $F = \{qs\}$

$\therefore L_1 \cap L_2$,



$\therefore$ The intersection of two regular language is regular.

14/9/24 Saturday 25/9/24 Wednesday

Theorem: 3

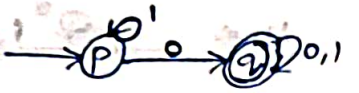
Difference of two regular language is language.

PROOF:

Let $M_1 = \{Q_1, \Sigma, \delta, q_1, F_1\}$

$M_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$ accepts the language L_1 and L_2 then finite automata accepts the language $L_1 - L_2$ where, $M = \{Q, \Sigma, \delta, q, F\}$

1. $Q = Q_1 \times Q_2$
2. $\Sigma = \Sigma$
3. $q_0 = \{q_1, q_2\}$
4. $\delta = \delta(q_1, q_2) = (\delta_1(q_1, a), \delta_2(q_2, a))$
5. $F = \{(p, q) \mid p \in F_1 \text{ \& } q \notin F_2\}$

$L \Rightarrow$ 

$M \Rightarrow$ 

Construct a machine $M = \{Q, \Sigma, \delta, q, F\}$ where,

1. $Q = \{pr, ps, qr, qs\}$
2. $\Sigma = \{0, 1\}$
3. $q_0 = \{pr\}$
4. $\delta \Rightarrow$

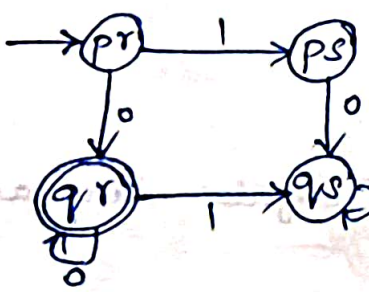
$(pr, 0) = \{qr\}$	$(pr, 1) = \{ps\}$
$(ps, 0) = \{qs\}$	$(ps, 1) = \{ps\}$
$(qr, 0) = \{qr\}$	$(qr, 0) = \{qs\}$
$(qs, 0) = \{qs\}$	$(qs, 0) = \{qs\}$

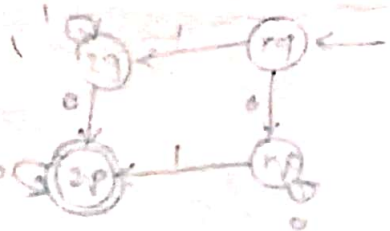
5. $F = \{(p, q) \mid p \in F_1 \text{ \& } q \notin F_2\}$

$F = \{qr\}$ → since we are subtracting final state of M (s)

↓ 1st automata final state

$\therefore L_1 - L_2$ is,





Theorem: 4

The complement of a regular language $\bar{L}$ is regular.

PROOF:

Let L be a recognized DFA with $M = (Q, \Sigma, \delta, q, F)$ then $\bar{L} = L(B)$ where $B = (Q, \Sigma, \delta, q_0, Q-F)$. B is exactly like M, but the accepting state of M have become non-accepting state B.

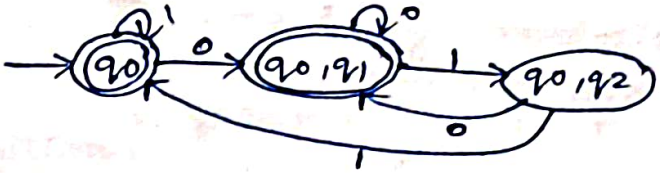
Ex:

$F = Q - F$

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

$F = \{q_0\}\ \{q_0, q_1\}$

The final state is not in q_0, q_2.



Theorem: 5

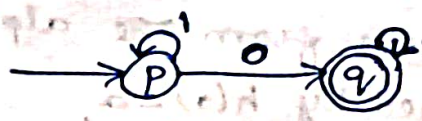
The closure of a regular language L^* is regular.

PROOF:

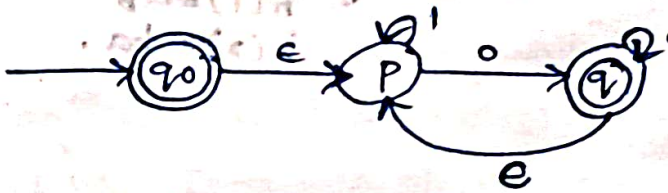
Let $N_1 = (Q_1, \Sigma, \delta, q_1, F_1)$ then construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize L^*

1. $Q = \{q_0\} \cup Q_1$ (i.e) the states of N_1 + a new start state
2. q_0 = the new start state.
3. $F = q_0 \cup F_1$ (i.e) the accepting states are the old accepting state + the new start state.
4. δ is defined as $\delta(q, a) = \delta_1(q, a)$ $q \in Q_1$ & $q \notin F_1$

Ex:



Ans:



14/9/24
Saturday

Theorem:6

The reversal of a string $a_1, a_2, \ldots a_n$ is the string written backwards $a_n, a_{n-1}, \ldots a_1$.

PROOF:

Let $L=\{001, 10, 111\}$ then $L^R = \{100, 01, 111\}$

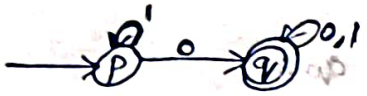
Given a language L (i.e) for some finite automata with NFA and ϵ, construct an automata L^R.

Step 1: Reverse all the arcs in the transition diagram of A

Step 2: Move the start of A, be the only accepting state of the new automata.

Step 3: create a new start state with transition on ϵ to all the accepting states of A

Ex:



Ans:

Theorem:7

Homomorphism - substituting a particular string for each symbol.

A string homomorphism is a function on strings that works by substituting a particular string for each symbol.

Ex:

$h(0) = ab$

$h(1) = \epsilon$

$w = 0011$

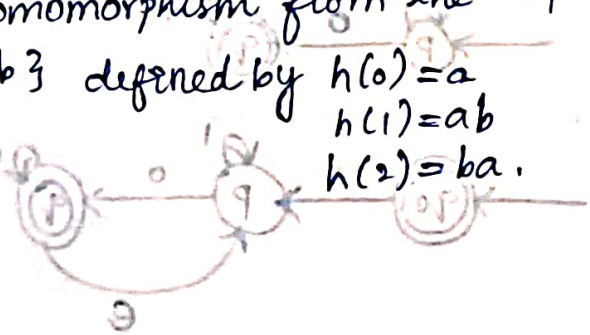
$h(0011) = h(0)\, h(0)\, h(1)\, h(1)$

$= abab\epsilon\epsilon$

$h(0011) = abab$

3

1. Suppose h is the homomorphism from the alphabets $\{0,1,2\}$ to the alphabet $\{a,b\}$ defined by $h(0)=a$, $h(1)=ab$, $h(2)=ba$.



Find i) $h(0120)$

ii) $h(0(1)^*2)$

Sol:

i) $h(0120) = h(0)\ h(1)\ h(2)\ h(0)$
$= a\ a\ b\ b\ a\ a.$

ii) $h(0(01)^*2) = (h(0))(h(1))^*\ h(2)$
$= a(ab)^*ba$

6/2/24 Thursday

UNIT-3

CONTEXT FREE GRAMMARS AND PUSH DOWN AUTOMATA (PDA)

APPLICATIONS OF CFG:

* The context free languages (CFL) are very similar to regular sets notably,
  i) In defining programming languages.
  ii) In formalising the notation of parsing.
  iii) simplifying translation of programming languages.
  iv) string processing applications.
  v) YACC parser generator

Definition:

context free grammar is defined by four tuples, $G = (V, T, P, S)$ where,

→ V = finite set of variables (or) non-terminals.

→ T = finite set of symbols called terminals.

→ P = set of productions (or) rules which is of the form $A \rightarrow \alpha$, $\alpha \in (V \cup T)^*$ where,

A ⇒ variable

α ⇒ string of zero (or) more terminals and strings.

→ S = starting symbol.

Eg: $E \rightarrow E+E$

E is variable (capital letter)

E → starting symbol

+ → terminal

CONTEXT FREE LANGUAGE: (CFL)

A language L is said to be a context free language if there exists a CFG with $L = L(G)$ where,

$L(G)$ is a set of strings derived from the grammar G.

Types of Grammar:

There are 4 types.

1. Type 0 - Phrase structure grammar → doesn't have any restrictions
2. Type 1 - context sensitive grammar. → LHS & RHS
3. Type 2 - Context free grammar. → Production are of length greater than or equal to 0
4. Type 3 - Regular grammar → RHS should always start with terminal (symbols and small letters) should be greater than α.

TYPE	GRAMMAR	LANGUAGE	MODEL
0	Phrase structure grammar.	Recursively enumerable.	Turing machine
1	context sensitive grammar	context sensitive Language (CSL)	Linear bounded automata
2	context free grammar	context free Language (CFL)	Push docon automata (PDA)
3	Regular grammar	Regular Language	Finite automata

Example:

Find the language for $G = (\{S\}, \{a, b\}, P, S)$ where,

$P = (S \to aSa,\ S \to bSb,\ S \to \epsilon)$.

Sol:

$S \to aSa$

$S \to bSb$

$S \to \epsilon$

i) $S \to aSa$

$S \to aa \quad [S \to \epsilon]$

ii) $S \to aSa$

$S \to aaSaa \quad [S \to aSa]$

$S \to aaaa \quad [S \to \epsilon]$

iii) $S \to bSb$

$S \to bbb \quad [S \to \epsilon]$

$L = \{ww^R / w \in (a,b)^*\}$

iv) $S \to aSa$

$S \to abSba \quad [S \to bSb]$

$S \to abba \quad [S \to \epsilon]$

For given grammar this is the language,

$L = \{\omega^n / n \geq 1\}\ \omega \in (a, b)$

2) Let G be a CFG with $N = S$, $T = (a, b)$ and $P = \{S \to aSbb, S \to abb\}$ Find $L(G)$.

sol:

i) $S \to aSbb$
$S \to aabbbb$ [$S \to abb$]

ii) $S \to aSbb$
$S \to aaSbbbb$ [$S \to aSbb$]
$S \to aaabbbbbb$ [$S \to abb$]

$\therefore L = \{ a^n b^{2n} \mid n \geq 1 \}$

3) $S \to bT \mid \lambda$ ($\lambda = \epsilon$). Find $L(G)$
$T \to ba \mid bS$

sol:

i) $S \to bT$
$S \to bba$ [$T \to ba \mid bS$]

ii) $S \to bT$
$S \to bbS$ [$T \to ba \mid bS$]
$S \to bbbT$ [$S \to bT \mid \lambda$]
$S \to bbbba$ [$T \to ba$]

iii) $S \to bT$
$S \to bbS$ [$T \to ba \mid bS$]
$S \to bb$ [$S \to \lambda$]

iv) $S \to bT$
$S \to bbS$ [$T \to bT \mid \lambda$]
$S \to bb$ [$S \to ba \mid bS$]

$L(G) = \{ bba, bbbba, b, bb, bbbb \}$
$L = \{ b^{2n} a^m \mid n \geq 0, m = 0 \text{ (or) } 1 \}$

4) $S \to SaS \mid b$

sol:

i) $S \to SaS$
$S \to bab$

ii) $S \to SaS$
$S \to baS$
$S \to baSaS$
$S \to bababab$

iii) $S \to SaS$
$S \to SaSab$
$S \to SaSabab$
$S \to bababab$

iv) $S \to b$

$L(G) = \{ b, bab, bababab, bababab \}$

$\therefore$ Language, $L = \{ (ba)^n b \mid n \geq 0 \}$ $(ba)^0 b = b$, $(ba)^1 b = bab$

$S \to 1A/1$

$A \to 0S/0$

Sol:

i) $S \to 1A$
$S \to 10$

ii) $S \to 1$

iii) $S \to 1A$
$S \to 10S$
$S \to 101$

iv) ~~$S \to 1A$~~
~~$S \to 10S$~~
~~$S \to 101A$~~

v) $S \to 1A$
$S \to 10S$
$S \to 101A$
$S \to 1010$

vi) $S \to 1A$
$S \to 10S$
$S \to 101A$
$S \to 1010S$
$S \to 10101$

$L = \{1, 10, 101, 1010, 10101, \ldots\}$

3/9/24
Monday.

6) construct a CFG for the language $L = \{a^n \mid n \text{ is odd}\}$.

Sol:

$n = 1, 3, 5, \ldots$

$n=1 \Rightarrow a^1 = a$

$n=3 \Rightarrow a^3 = aaa$

$n=5 \Rightarrow a^5 = aaaaa$

i) Expression with length = 1

$S \to a$

ii) Expression with length $2n+1$

$n=1$, $S \to aaS$

$n=2$, $S \to aaaaS$

The possible productions are, $S \to a$, $S \to aaS$

$\therefore$ CFG $= (V, T, P, S)$

where, $V = \{S\}$

$T = \{a\}$

$P = \{S \to a, S \to aaS\}$

$S = \{S\}$

7.) construct a CFG for $L = \{ wcw^R \mid w \text{ is } (a+b)^* \}$

Sol:

If $w = ab$; c is intermediate symbol.

$L = abCba$

If $w = a$, $L = aCa$

If $w = b$, $L = bCb$

$E \to C$

$E \to aEa$

$E \to bEb$

$E \to abEba$

The possible productions are ($E \to C$, $E \to aEa$, $E \to bEb$, $E \to abEba$)

$\therefore CFG = (V, T, P, S)$

where, $V = \{C, E\}$

$T = \{a, b\}$

$P = (E \to C, E \to aEa, E \to bEb, E \to abEba)$

$S = \{E\}$

8) construct a CFG representing a set of palindrome over (0+1)

Sol:

$L = \{ ww^R \mid w \text{ is } (0+1)^* \}$

$w = 01$	$w = 0101$	$w = 0110$
$w^R = 10$	$w^R = 1010$	$w^R = 0110$
$ww^R = 0110$	$ww^R = 01011010$	$ww^R = 01100110$

i) Palindrome with length = 1

$S \to 0/1/\epsilon$

ii) Palindrome with length > 1

$S \to 0S0$

$S \to 1S1$

$\therefore CFG = (V, T, P, S)$

where, $V = \{S\}$

$T = \{0, 1\}$

$P = \{S \to 0/1/\epsilon, S \to 0S0, S \to 1S1\}$,

$S = \{S\}$.

DERIVATIONS:

Derivations are the set of strings that are derived from the start symbol, after applying the production rules a finite no. of times.

$$S \overset{*}{\underset{G}{\Rightarrow}} w \mid w \in T^*$$

(w: Terminal)

while inferring whether the given input string belongs to a given CFG we have two approaches,

i) Recursive inference (using the rule from body to head)
ii) Derivations (using the rule from head to body)

This derivations include two types,

i) Leftmost derivation (LMD)
ii) Rightmost derivation (RMD)

LMD:

If at each step in a derivation, a production is applied to the leftmost variable then it is called leftmost derivation. It is denoted as, $\overset{*}{\underset{lm}{\Rightarrow}}$

RMD:

If at each step in a derivation, a production is applied to the rightmost variable then it is called rightmost derivation. It is denoted as, $\overset{*}{\underset{rm}{\Rightarrow}}$

Example:

Let $G = (V, T, P, S)$ where $V = \{E\}$, $T = \{+, *, id\}$, $S = \{E\}$ and P is given by $\{E \to E+E, E \to E*E, E \to id\}$. construct LMD and RMD for $id + id * id$.

Sol: LMD:

$E \to E+E$
$\overset{lm}{\Rightarrow} id + E \quad (\because E \to id)$
$\overset{lm}{\Rightarrow} id + E*E \quad (\because E \to E*E)$
$\overset{lm}{\Rightarrow} id + id * E \quad (\because E \to id)$
$\overset{lm}{\Rightarrow} id + id * id \quad (\because E \to id)$

RMD: $E \to E+E$

$\xrightarrow{rm}$ $E+E*E$ $(\because E \to E+E)$

$\xrightarrow{rm}$ $E+E*id$ $(\because E \to id)$

$\xrightarrow{rm}$ $E+id*id$ $(\because E \to id)$

$\xrightarrow{rm}$ $id+id*id$ $(\because E \to id)$

solve for $id+id*id$

$E \to E+T/T$

$T \to T*F/F$

$F \to E/id$

so:

LMD: $E \to E+T$

$\xrightarrow{lm}$ $T+T$ $(\because E \to T)$

$\xrightarrow{lm}$ $F+T$ $(\because T \to F)$

$\xrightarrow{lm}$ $id+T$ $(\because F \to id)$

$\xrightarrow{lm}$ $id+T*F$ $(\because T \to T*F)$

$\xrightarrow{lm}$ $id+F*F$ $(\because T \to F)$

$\xrightarrow{lm}$ $id+id*F$ $(\because F \to id)$

$\xrightarrow{lm}$ $id+id*id$ $(\because F \to id)$

RMD: $E \to E+T$

$\xrightarrow{rm}$ $E+T*F$ $(\because T \to T*F)$

$\xrightarrow{rm}$ $E+T*id$ $(\because F \to id)$

$\xrightarrow{rm}$ $E+F*id$ $(\because T \to F)$

$\xrightarrow{rm}$ $E+id*id$ $(\because F \to id)$

$\xrightarrow{rm}$ $T+id*id$ $(\because E \to T)$

$\xrightarrow{rm}$ $F+id*id$ $(\because T \to F)$

$\xrightarrow{rm}$ $id+id*id$ $(\because F \to id)$

SENTENCIAL FORM:-

The strings are produced from the start symbol is called sentenceal form. If $G=(V,T,P,S)$ is a CFG, then any string α in $(V \cup T)^*$ such that $S \xRightarrow{*} \alpha$ is a sentencial form if $S \xRightarrow[lm]{*} \alpha$ then α is a left sentencial form. If $S \xRightarrow[rm]{*} \alpha$, then α is a right sentencial form.

PARSE TREE:

→ The representation of a derivation in the form of a tree is called a parse tree or derivation tree

→ A parse tree is used to show how the symbols of terminal strings are grouped into substrings each of which belongs to the language in the grammar.

→ The parse for G is the tree with the following conditions,

i) Each interial node is labelled by a variable in V

ii) Each leaf is labelled either by a variable, a terminal or ϵ.

iii) If an interior node is labelled A and its children are labelled $X_1, X_2, X_3, \ldots X_k$ respectively from the left, then $A \to X_1 X_2 \ldots X_k$ is a production in P.

iv) If $A \to \epsilon$, then A is considered to be the label.

THE YIELD OF A PARSE TREE:

→ The string obtained by concatenating the leaves of a parse tree from the left is called the yield of a parse tree.

→ The yield is always derived from the root.

→ Root is the start symbol

→ All leaves are labelled either with a terminal or with ϵ, thus the yield is a terminal string.

3/10/24
Thursday

1) Let G be a grammar, $S \to 0B \mid 1A$

$A \to 0 \mid 0S \mid 1AA$

$B \to 1 \mid 1S \mid 0BB$

For the string 00110101 find LMD and parse tree.

sol:

$S \to 0B$

$\xrightarrow{lm} 00BB \quad [B \to 0BB]$

$\xrightarrow{lm} 001B \quad [B \to 1]$

$\xrightarrow{lm} 0011S \quad [B \to 1S]$

$\xrightarrow{lm} 00110B \quad [S \to 0B]$

$\xrightarrow{lm}$ 00110108 [S → 0B]

$\xrightarrow{lm}$ 00110101 [B → 1]

RMD:

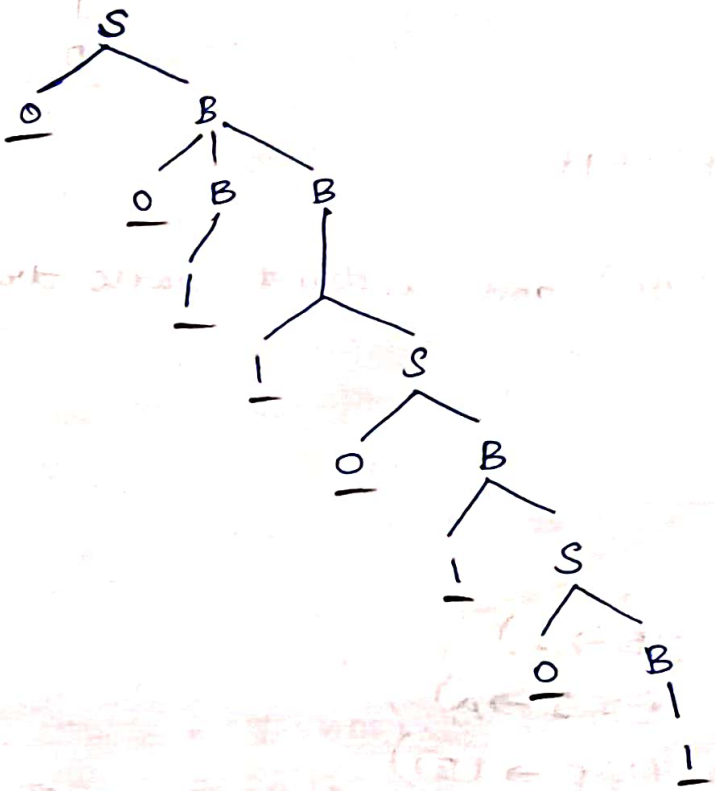
S → 0 B

$\xrightarrow{rm}$ 00BB [B → 0BB]

$\xrightarrow{rm}$ 00B1 [B → 1]

$\xrightarrow{rm}$

Parse tree:



Yeild of a parse tree: 00110101

RMD: S → 0B

$\xrightarrow{rm}$ 00BB [B → 0BB]

$\xrightarrow{rm}$ 00B1 [B → 1]

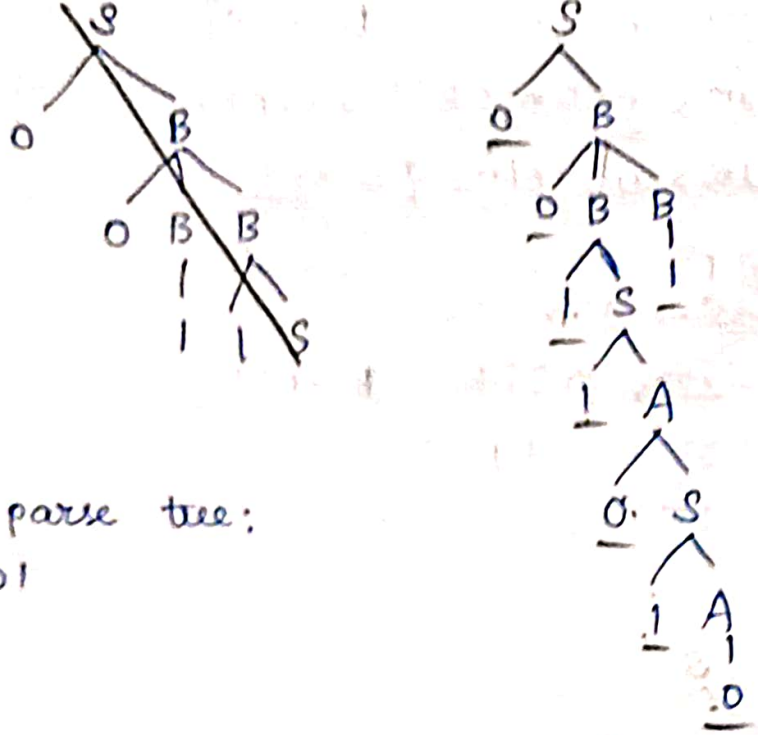
$\xrightarrow{rm}$ 001S1 [B → 1S]

$\xrightarrow{rm}$ 0011A1 [S → 1A]

$\xrightarrow{rm}$ 00110S1 [A → 0S]

$\xrightarrow{rm}$ 001101A1 [S → 1A]

$\xrightarrow{rm}$ 00110101 [A → 0]



Yield of parse tree:
00110101

a. $E \rightarrow E+E \mid I \mid (E) \mid E+E$
$I \rightarrow a \mid b \mid I00$

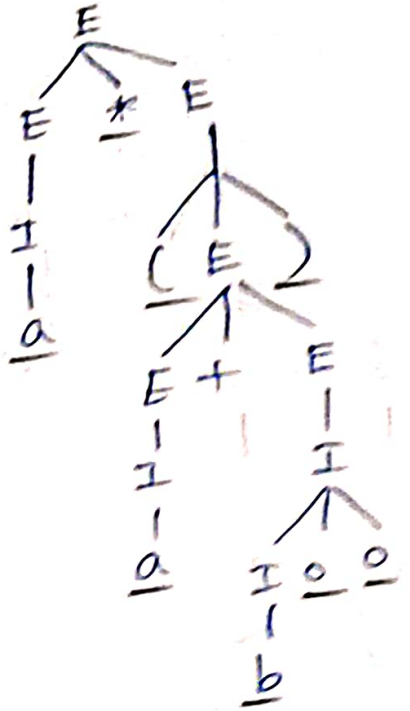
Derive $a*(a+b00)$ and construct parse tree. write yield.

Sol:

LMD:

$E \rightarrow E*E$
$\xrightarrow{lm} I*E \quad (\because E \rightarrow I)$
$\xrightarrow{lm} a*E \quad (\because I \rightarrow a)$
$\xrightarrow{lm} a*(E) \quad (\because E \rightarrow (E))$
$\xrightarrow{lm} a*(E+E) \quad (\because E \rightarrow E+E)$
$\xrightarrow{lm} a*(I+E) \quad (\because E \rightarrow I)$
$\xrightarrow{lm} a*(a+E) \quad (\because I \rightarrow a)$
$\xrightarrow{lm} a*(a+I) \quad (\because E \rightarrow I)$
$\xrightarrow{lm} a*(a+I00) \quad (\because I \rightarrow I00)$
$\xrightarrow{lm} a*(a+b00) \quad (\because I \rightarrow b)$

Parse tree:



yield of tree:

a * (a+b00)

RMD:

$E \to E * E$

$\xrightarrow{rm} E * (E)$ ($\because E \to (E)$)

$\xrightarrow{rm} E * (E+E)$ ($\because E \to E+E$)

$\xrightarrow{rm} E * (E+I)$ ($\because E \to I$)

$\xrightarrow{rm} E * (E+I00)$ ($\because I \to I00$)

$\xrightarrow{rm} E * (E+b00)$ ($\because I \to b$)

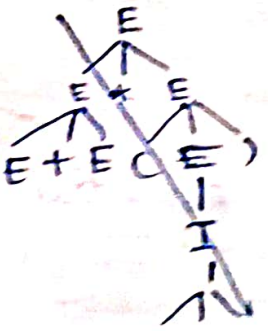
$\xrightarrow{rm} E * (I+b00)$ ($\because E \to I$)

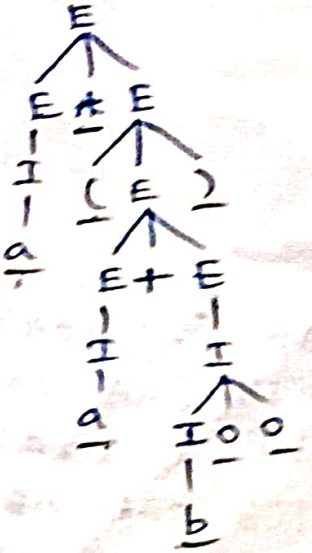
$\xrightarrow{rm} E * (a+b00)$ ($\because I \to a$)

$\xrightarrow{rm} I * (a+b00)$ ($\because E \to I$)

$\xrightarrow{rm} a * (a+b00)$ ($\because I \to a$)

Parse tree:





yield of tree:

a * (a+b00)

3) For the grammar

$S \to A1B/B$

$A \to 0A/\epsilon$

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

Find parse tree for

i) 1001

ii) 00101

iii) 00011.

Ans:

i) 1001

LMD:

$S \to A1B$

$\xrightarrow{lm} 1B \quad (\because A \to \epsilon)$

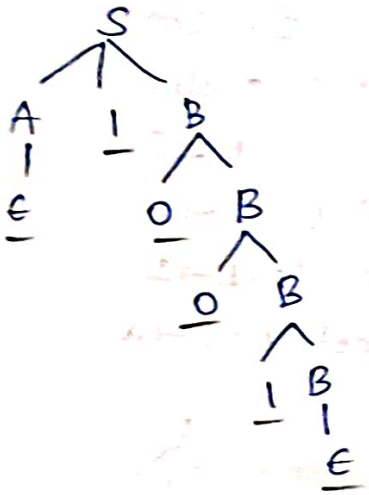
$\xrightarrow{lm} 10B \quad (\because B \to 0B)$

$\xrightarrow{lm} 100B \quad (\because B \to 0B)$

$\xrightarrow{lm} 1001B \quad (\because B \to 1B)$

$\xrightarrow{lm} 1001 \quad (\because B \to \epsilon)$

Parse tree:



yield : 1001

ii) 00101

LMD:

$S \to A1B$

$\xrightarrow{lm} 0A1B \quad (\because A \to 0A)$

$\xrightarrow{lm} 00A1B \quad (\because A \to 0A)$

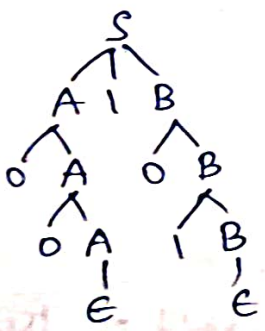
$\xrightarrow{lm} 001B \quad (\because A \to \epsilon)$

$\xrightarrow{lm} 0010B \quad (\because B \to 0B)$

$\xrightarrow{lm} 00101B \quad (\because B \to 1B)$

$\xrightarrow{lm} 00101 \quad (\because B \to \epsilon)$

Parse tree:



yield: 00101

iii) 00011

LMD:

$S \to A \mid B$

$\xrightarrow{lm}$ $0A \mid B$ ($\because A \to 0A$)

$\xrightarrow{lm}$ $00A \mid B$ ($\because A \to 0A$)

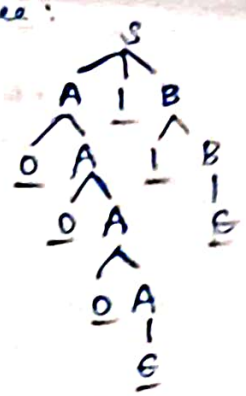
$\xrightarrow{lm}$ $000A \mid B$ ($\because A \to 0A$)

$\xrightarrow{lm}$ $000 \mid B$ ($\because A \to \epsilon$)

$\xrightarrow{lm}$ $000 \mid 1B$ ($\because B \to 1B$)

$\xrightarrow{lm}$ 00011 ($\because B \to \epsilon$)

Parse tree:



yield : 00011

H.W obtain the string aaabbabbba from the grammar,

$S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$

AMBIGOUS GRAMMAR:

⇒ A CFG, $G = (V, T, P, S)$ is ambigous if there is atleast one string w for which we can find two different parse trees, each with root labelled S any yield w.

⇒ If each string has atmost one parse tree in the grammar then the grammar is unambigous.

1. $E \to E+E \mid E*E \mid id$.

Input string : id + id * id

Sol: LMD:

i) $E \to E+E$

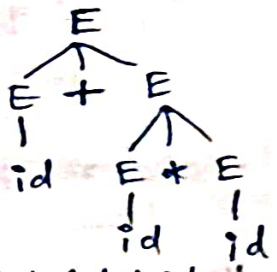
$\xrightarrow{lm}$ id + E ($\because E \to id$)

$\xrightarrow{lm}$ id + E * E ($\because E \to E*E$)

$\xrightarrow{lm}$ id + id * E ($\because E \to id$)

$\xrightarrow{lm}$ id + id * id ($\because E \to id$)

Parse tree:



yield : id + id * id

ii) $E \to E*E$

$\xrightarrow{lm}$ id * E ($\because E \to id$)

$\xrightarrow{lm}$ id * E + E ($\because E \to E+E$)

$\xrightarrow{lm}$ id * id + E ($\because E \to id$)

$\xrightarrow{lm}$ id * id + id. ($\because E \to id$)

HW: $S \to S*S \mid S+S \mid a$

show that the given grammar G is ambiguous.
a*a+

Sol:

i) Let us consider, $w = a+a*a$.

Should do LMD and RMD. Not two separate strings (S+S*S and S+S)

LMD

p) $S \to S*S$

$\xrightarrow{lm}$ $S+S*S$ ($\because S \to S+S$)

$\xrightarrow{lm}$ $a+S*S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a+a*S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a+a*a$ ($\because S \to a$)

pp) $S \to S+S$

$\xrightarrow{lm}$ $S+S*S$ ($\because S \to S*S$)

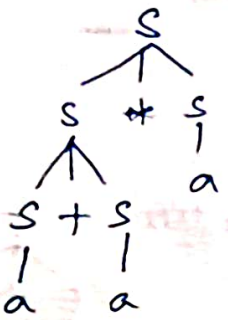
$\xrightarrow{lm}$ $a+S*S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a+a*S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a+a*a$ ($\because S \to a$)

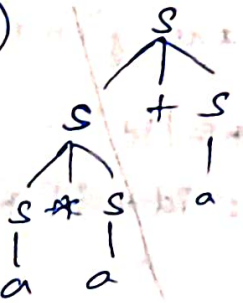
Parse tree:

p)



yield = $a+a*a$

pp)



yield = $a*a+a$

ii) ∴ The yields are diffelent so this not ambigous. So let us consider another example, $w = a*a+a*a$.

p) $S \to S*S$

$\xrightarrow{lm}$ $a*S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a*S+S$ ($\because S \to S+S$)

$\xrightarrow{lm}$ $a*a+S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a*a+S*S$ ~~($\because S \to a*S$)~~ ($S \to S*S$)

$\xrightarrow{lm}$ $a*a+a*S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a*a+a*a$ ($\because S \to a$)

pp) $S \to S+S$

$\xrightarrow{lm}$ $S*S+S$ ($\because S \to S*S$)

$\xrightarrow{lm}$ $a*S+S$ ($\because S \to a$)

$\xrightarrow{lm}$ $a*a+S$ ($\because S \to a$)

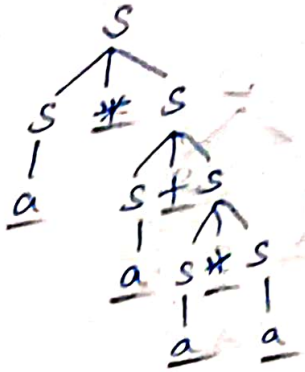
$\xrightarrow{lm}$ $a*a+S*S$ ($\because S \to S*S$)

$\xrightarrow{lm}$ $a*a+a*S$ ($\because S \to a$)

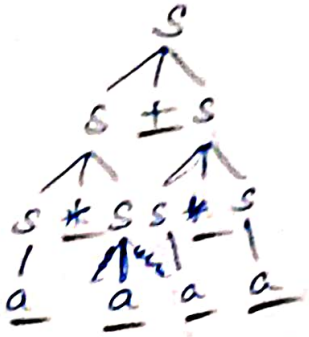
$\xrightarrow{lm}$ $a*a+a*a$ ($\because S \to a$)

Parse tree :

i)



ii)



yield = a * a + a * a

yield = a * a + a + a

Both the trees yield some string. so it is ambigous.

H.W aaabbabbba

$S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$

Sol:

$S \to aB$

$\xrightarrow{lm}$ aaBB ($\because B \to aBB$)

$\xrightarrow{lm}$ aaaBBB ($\because B \to aBB$)

$\xrightarrow{lm}$ aaabBB ($\because B \to b$)

$\xrightarrow{lm}$ aaabbB ($\because B \to b$)

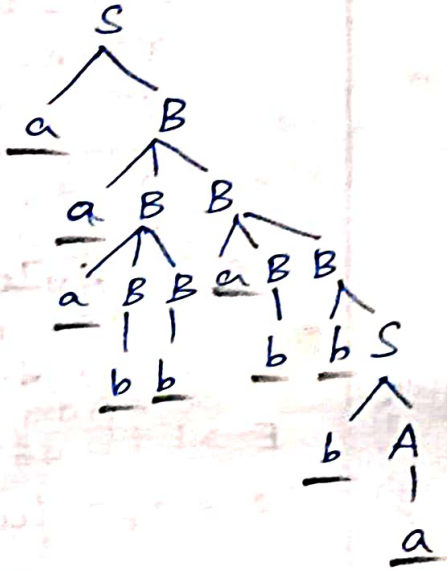
$\xrightarrow{lm}$ aaabbaBB ($\because B \to aBB$)

$\xrightarrow{lm}$ aaabbabB ($\because B \to b$)

$\xrightarrow{lm}$ aaabbabbS ($\because B \to bS$)

$\xrightarrow{lm}$ aaabbabbbA ($\because S \to bA$)

$\xrightarrow{lm}$ aaabbabbba ($\because A \to a$)



14/ady

grammar defined by the production rule

$S \to SS \mid a \mid b$ is ambiguous for the input string aab

4c)

LMD: $S \to SS$

$\underline{lm}\rightarrow aS \quad (S \to a)$

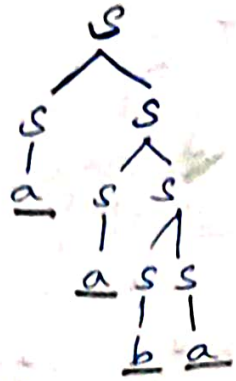
$\underline{lm}\rightarrow aSS \quad (S \to SS)$

$\underline{lm}\rightarrow aaS \quad (S \to a)$

$\underline{lm}\rightarrow aaSS \quad (S \to SS)$

$\underline{lm}\rightarrow aabS \quad (S \to b)$

$\underline{lm}\rightarrow aaba \quad (S \to a)$



yield : aaba

RMD: $S \to SS$

$\underline{rm}\rightarrow Sa \quad (S \to a)$

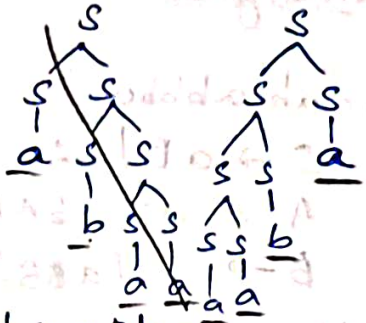
$\underline{rm}\rightarrow SSa \quad (S \to SS)$

$\underline{rm}\rightarrow Sba \quad (S \to b)$

$\underline{rm}\rightarrow SSba \quad (S \to SS)$

$\underline{rm}\rightarrow Saba \quad (S \to a)$

$\underline{rm}\rightarrow aaba \quad (S \to a)$



yield : aaba

$\therefore$ It is ambiguous (Two different trees can be derived)

H.W.

Ambiguous or not:

2) $E \to E+E \mid E*E \mid a$ ($\because$ using own input strings)

3) $S \to SbS \mid a$

REMOVING AMBIGUITY IN GRAMMAR:

The following are the causes for ambiguity in grammar:

1. Precedence of operators is not followed.
2. A sequence of identical operators can group either from left or the right.

The solutions to these problems are,

i) factor – is an expression that cannot be broken by any adjacent operators either + (or) *

ii) a term – is a product of one or more factors which cannot be broken by the + operator.

iii) an expression - can be written by either an adjacent * or an adjacent +. An expression is a sum of one or more terms.

INHERENT AMBIGUITY:

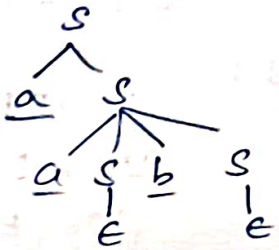
A CFL is said to inherently ambiguous if all of its grammar are ambiguous (if even one grammar in L is unambiguous then L is unambiguous grammar)

Eg: $S \to A \mid B$
$A \to a$
$B \to a \mid b$
Input string: a

$S \to a$ | $S \to B$ (∵ By using both the grammars S→A and S→B we can derive the input string so, it is inherently ambiguous)

i) $S \to aS \mid aSbS \mid \epsilon$
input string: aab

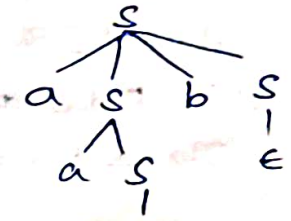
sol: i) $S \to aS$
$\xrightarrow{lm}$ $aaSbS$ (∵ $S \to aSbS$)
$\xrightarrow{lm}$ $aabS$ (∵ $S \to \epsilon$)
$\xrightarrow{lm}$ aab (∵ $S \to \epsilon$)



⇒ aab.

ii) $S \to aSbS$
$\xrightarrow{lm}$ $aaSbS$ (∵ $S \to aS$)
$\xrightarrow{lm}$ $aabS$ (∵ $S \to \epsilon$)
$\xrightarrow{lm}$ $aabS$ (∵ $S \to \epsilon$)

⇒



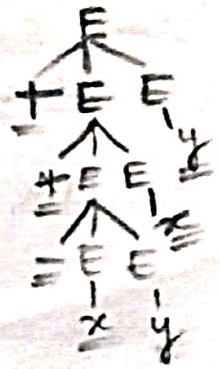
⇒ aab

∴ It is inherently ambiguous.

ii:
i) Find the derivation using +*-xyxy using
$E \to +EE \mid *EE \mid -EE \mid x \mid y$. using LMD and RMD.

sol: $E \to +EE$

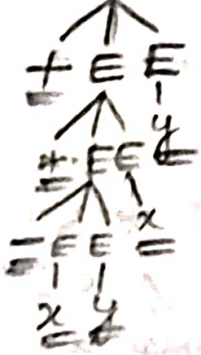
LMD:
$\xrightarrow{lm}$ $+*EEE$ (∵ $E \to *EE$)
$\xrightarrow{lm}$ $+*-EEEE$ (∵ $E \to -EE$)
$\xrightarrow{lm}$ $+*-xEEE$ (∵ $E \to x$)
$\xrightarrow{lm}$ $+*-xyEE$ (∵ $E \to y$)
$\xrightarrow{lm}$ $+*-xyxE$ (∵ $E \to x$)
$\xrightarrow{lm}$ $+*-xyxy$ (∵ $E \to y$)



RMD: E→+EE

rm→ +Ey (∵ E→y)
rm→ +*EEy (∵ E→*EE)
rm→ +*Exy (∵ E→x)
rm→ +*-EExy (∵ E→-EE)
rm→ +*-Eyxy (∵ E→y)
rm→ +*-xyxy (∵ E→x)

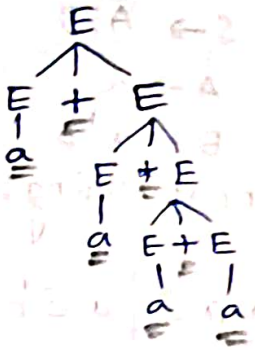
∴ It is ambiguous



a) Sol: E→ E+E | E*E | a

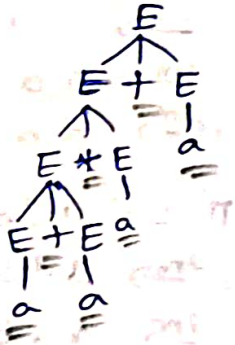
Input string: a+a*a+a

LMD: E→ E+E
lm→ a+E (E→a)
lm→ a+E*E (∵E→E*E)
lm→ a+a*E (∵E→a)
lm→ a+a*E+E (∵E→E+E)
lm→ a+a*a+E (∵E→a)
lm→ a+a*a+a (∵E→a)



RMD: E→E+E
rm→ E+a (∵E→a)
rm→ E+E+a (∵E→E*E)
rm→ E*a+a (∵E→E)
rm→ E+E*a+a (∵E→E+E)
rm→ E+a*a+a (∵E→a)
rm→ a+a*a+a (∵E→a)

∴ It is ambiguous.



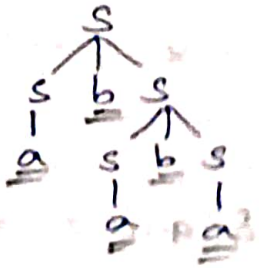
3) Sol:
S→sbs/a

Input string: abbabb

LMD: S→sbs
lm→ abs (∵s→a)
lm→ absbs (∵S→sbs)
lm→ abbabb

3) Sol: $S \to SbS/a$

Input string: ababa



LMD:

$S \to SbS$

$\xrightarrow{lm} abS$ ($\because S \to a$)

$\xrightarrow{lm} abSbS$ ($\because S \to SbS$)

$\xrightarrow{lm} ababS$ ($\because S \to a$)

$\xrightarrow{lm} ababa$ ($\because S \to a$)

RMD:

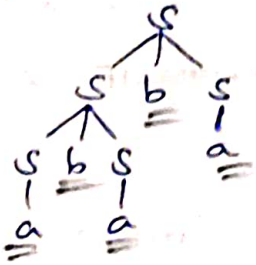
$S \to SbS$

$\xrightarrow{rm} Sba$ ($\because S \to a$)

$\xrightarrow{rm} SbSba$ ($\because S \to SbS$)

$\xrightarrow{rm} Sbaba$ ($\because S \to a$)

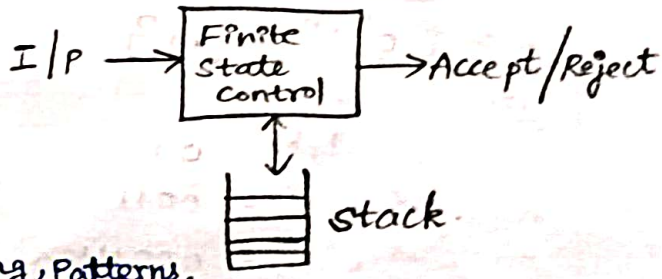
$\xrightarrow{rm} ababa$ ($\because S \to a$)



$\therefore$ It is ambiguous.

PUSHDOWN AUTOMATA:

It is essentially a finite automata with control of both input tape and stack on which we can store a string of stack symbols. With the help of a stack, the push down automata can remember an infinite amount of information.



Applications:

1. Syntax Analysis (compiler)
2. ~~Format definition~~. string matching, Patterns.
3. software engineering

FORMAL DEFINITION:

Push down automata consist of 7 tuples,

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where, $Q \to$ Finite non-empty set of states.

$\Sigma \to$ Input symbols.

$\Gamma \to$ Finite non-empty set of stack symbols.

$q_0 \to$ Initial state.

$Z_0 \to$ Initial start symbol of the stack.

$F \to$ Accepting / Final state.

δ → Transition function.

$Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to Q \times \Gamma^*$

NOTE:

$\delta(q, a, z) = (p, \alpha)$

where, q → Input state
a → Input symbol
z → stack
p → state
α → content of stack.

INSTANTANEOUS DESCRIPTION:

There are two different ways of language acceptance by PDA.

i) Acceptance by final state.

$L(M) = \{w \mid (q_0, w, z_0) \vdash^* (p, \epsilon, \gamma)\}$ for some p in M and γ in Σ^*, Γ^*

(p → final state)

ii) Acceptance by empty stack.

$N(M) = \{w \mid (q_0, w, z_0) \vdash^* (p, \epsilon, \epsilon)\}$ for some p in Q

1) Construct a PDA that accepts the language,

$L = \{0^n 1^n \mid n \geq 1\}$ (Empty stack)

Sol:

$n=1 \Rightarrow 0^1 1^1 = 01$
$n=2 \Rightarrow 0^2 1^2 = 0011$ (Select $n=2$)
$n=3 \Rightarrow 0^3 1^3 = 000111$

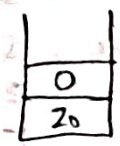
PDA,

$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where, $Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$\Gamma = \{0, z_0\}$

$\delta(q_0, 0, z_0) = \{(q_1, 0z_0)\}$

(q_0 → Initial state; 0 → i/p; z_0 → Top value of stack; q_1 → Next state; $0z_0$ → previous value; 0 → Pushed value)



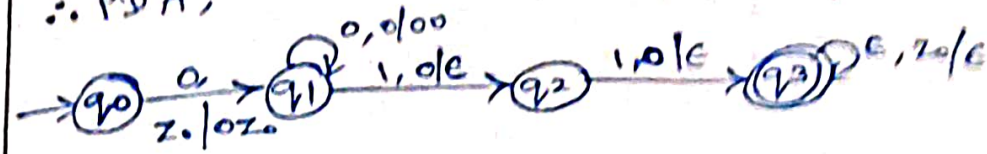
$\delta(q_1, 0, 0) = \{(q_1, 00)\}$ → 

$\delta(q_1, 1, 0) = \{(q_2, \epsilon 0)\}$ → 

$\delta(q_2, 1, 0) = \{(q_2, \epsilon)\}$ → 

$\delta(q_2, \epsilon, z_0) = \{(q_3, \epsilon)\}$ →

∴ PDA,



2) construct a PDA that accepts the language, $L = \{a^n b^n \mid n \geq 1\}$ by empty stack.

sol:

$n=1 \Rightarrow a^1 b^1 = ab$

$n=2 \Rightarrow a^2 b^2 = aabb$ (select n=2)

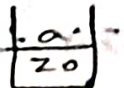
$n=3 \Rightarrow a^3 b^3 = aaabbb$.

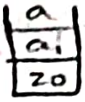
PDA, $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

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

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

$\Gamma = \{a, z_0\}$

$\delta(q_0, a, z_0) = \{(q_1, az_0)\}$ → 

$\delta(q_1, a, a) = \{(q_2, aa)\}$ → 

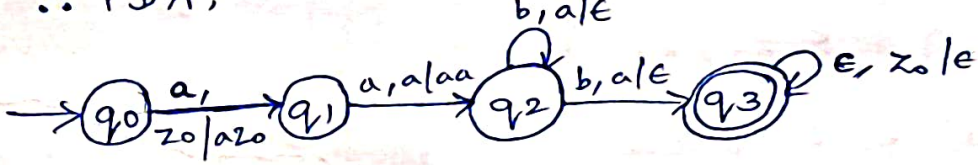
$\delta(q_2, b, a) = \{(q_2, \epsilon)\}$ → 

$\delta(q_2, b, a) = \{(q_3, \epsilon)\}$ → 

$\delta(q_3, \epsilon, z_0) = \{(q_3, \epsilon)\}$ → 

$\Sigma = \{a, b\}$
$\Gamma = \{a, z_0\}$
Initial state = q_0
Final state = $\{q_3\}$

∴ PDA,



3) construct a PDA that accepts the language,
$L = \{a^n b^{2n} \mid n \geq 1\}$.

sol:

$n=1 \Rightarrow a^1 b^2 = abb$

$n=2 \Rightarrow a^2 b^4 = aabbbb$

$n=3 \Rightarrow a^3 b^6 = aaabbbbbb$.

(Select n=2)

let us assume $1b = 2b$.

$w = aabb$

aabbbbb.
b b

PDA, $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$\delta(q_0, a, Z_0) = \{(q_1, aZ_0)\}$ → 

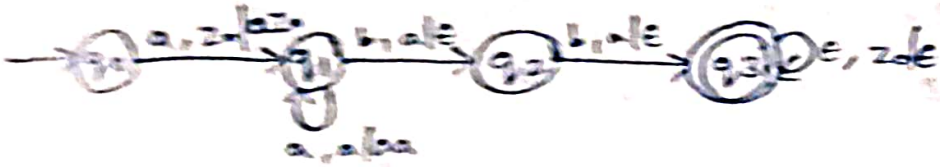
$\delta(q_1, a, a) = \{(q_1, aa)\}$ → 

$\delta(q_1, b, a) = \{(q_2, \epsilon)\}$ → 

$\delta(q_2, b, a) = \{(q_3, \epsilon)\}$ → 

$\delta(q_2, \epsilon, Z_0) = \{(q_3, \epsilon)\}$ → 

PDA:



$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

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

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

$\Gamma = \{a, Z_0\}$

starting state = q_0

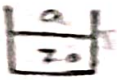
final state = $\{q_3\}$

2) construct a PDA that accepts the language, $L = \{a^m b^n a^n \mid m, n \geq 1\}$ by empty stack.

Sol:

$n = m = 1 \Rightarrow a^1 b^1 a^1 = aba$

$n = m = 2 \Rightarrow a^2 b^2 a^2 = aabbaa$

$\delta(q_0, a, Z_0) = \{(q_1, aZ_0)\}$ → 

$\delta(q_1, a, a) = \{(q_1, aa)\}$ → 

$\delta(q_1, b, a) = \{(q_1, a)\}$

$\delta(q_1, b, a) = \{(q_1, a)\}$

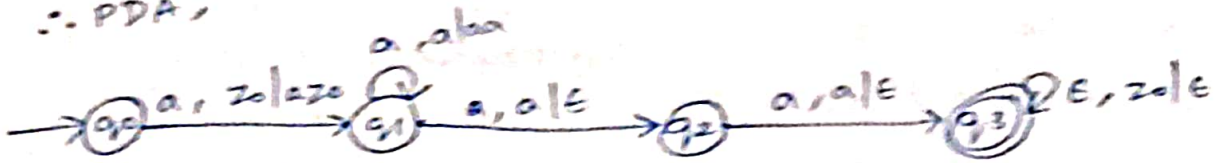
> No operation just skip the step till reads a.

$\delta(q_1, a, a) = \{(q_2, \epsilon)\}$ → 

$\delta(q_2, a, a) = \{(q_3, \epsilon)\}$ →

$\delta(q_3, \epsilon, Z_0) = \{(q_3, \epsilon)\}$ → ⊔

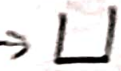
∴ PDA,



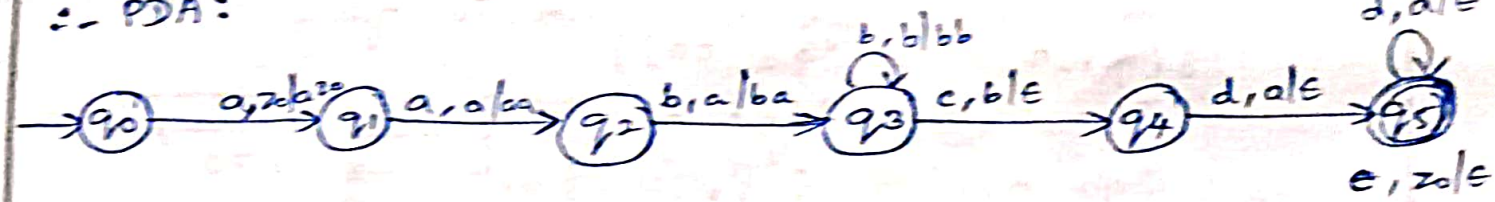
5) Construct a PDA that accepts the language, $L = \{a^n b^m c^m d^n / m, n \geq 1\}$ by empty stack.

Sol:

n=m=1 ⇒ a¹b¹c¹d¹ = abcd
n=m=2 ⇒ aabbccdd (Select)

$\delta(q_0, a, z_0) = \{(q_1, az_0)\}$ → 
$\delta(q_1, a, a) = \{(q_2, aa)\}$ → 
$\delta(q_2, b, a) = \{(q_3, ba)\}$ → 
$\delta(q_3, b, b) = \{(q_3, bb)\}$ → 
$\delta(q_3, c, b) = \{(q_4, \epsilon)\}$ → 
$\delta(q_4, c, b) = \{(q_4, \epsilon)\}$ → 
$\delta(q_4, d, a) = \{(q_5, \epsilon)\}$ → 
$\delta(q_5, d, a) = \{(q_5, \epsilon)\}$ → 
$\delta(q_5, \epsilon, z_0) = \{(q_5, \epsilon)\}$ → 

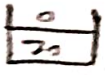
∴ PDA:



6) Construct a PDA that accepts the language, $L = \{wcw^R / w \in (a,b)^*\}$

Sol: The possible languages are,

i) aaCaa
ii) abCba
iii) baCab
iv) bbCbb

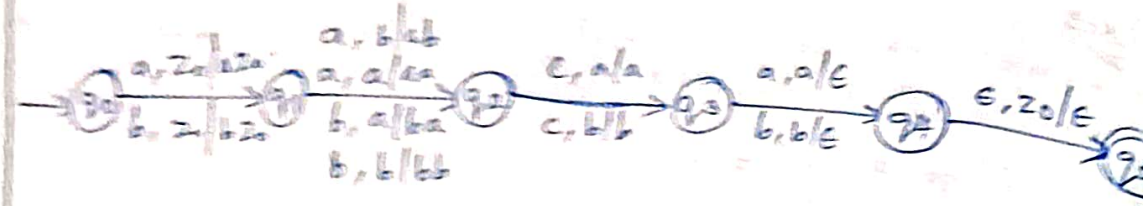
1. $\delta(q_0, a, z_0) = \{(q_1, az_0)\}$ ← a, b — 
2. $\delta(q_0, b, z_0) = \{(q_1, bz_0)\}$ ← a, b — 

1.1 $\delta(q_1, a, a) = \{(q_2, aa)\}$ — 
1.2 $\delta(q_1, b, a) = \{(q_2, ba)\}$ — 
2.1 $\delta(q_1, a, b) = \{(q_2, ab)\}$ — 
2.2 $\delta(q_1, b, b) = \{(q_2, bb)\}$ — 

$\delta(q_2, c, a) = \{(q_3, a)\}$
$\delta(q_0, c, b) = \{(q_3, b)\}$ } → Separator 'c'

$\delta(q_3, a, a) = \{(q_4, \epsilon)\}$
$\delta(q_3, b, b) = \{(q_4, \epsilon)\}$ } → Reversal

$\delta(q_4, \epsilon, z_0) = \{(q_5, \epsilon)\}$ — ⊔



→ Construct a PDA that accepts the language, $L = \{ww^R / w \in$

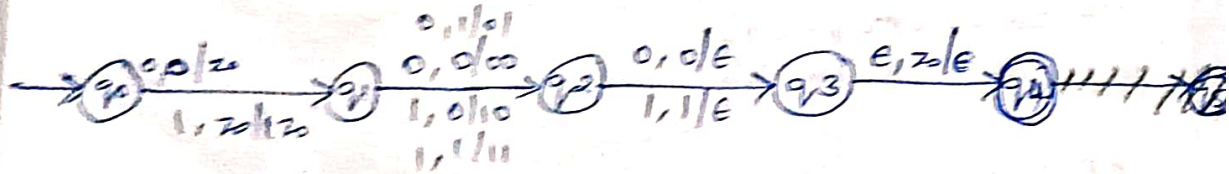
Sol: Possible languages are,

i) 0000
ii) 0110
iii) 1001
iv) 1111

- $\delta(q_0, 0, z_0) = \{(q_{11}, 0z_0)\}$ — 0/z0
- $\delta(q_0, 1, z_0) = \{(q_{11}, 1z_0)\}$ — 1/z0

1.1 $\delta(q_{11}, 0, 0) = \{(q_{12}, 00)\}$
1.2 $\delta(q_{11}, 1, 0) = \{(q_{12}, 10)\}$
2.1 $\delta(q_{11}, 0, 1) = \{(q_{12}, 01)\}$
2.2 $\delta(q_{11}, 1, 1) = \{(q_{12}, 11)\}$

$\delta(q_{12}, a, a) = \{(q_{13}, \epsilon)\}$
$\delta(q_{13}, 1, 1) = \{(q_{13}, \epsilon)\}$ } → Reversal
$\delta(q_{13}, \epsilon, z_0) = \{(q_{14}, \epsilon)\}$ — ⊔



8) Let 100C001 be the input string for the PDA check whether it is accepted or rejected

sol:

$\delta(q_0, 100C001, Z_0) \vdash (q_0, 00C001, 1Z_0)$
$\vdash (q_0, 0C001, 01Z_0)$
$\vdash (q_0, C001, 001Z_0)$
$\vdash (q_0, 001, 001Z_0)$
$\vdash (q_0, 01, 01Z_0)$ (whenever same element → pop)
$\vdash (q_0, 1, 1Z_0)$
$\vdash (q_0, \epsilon, Z_0)$

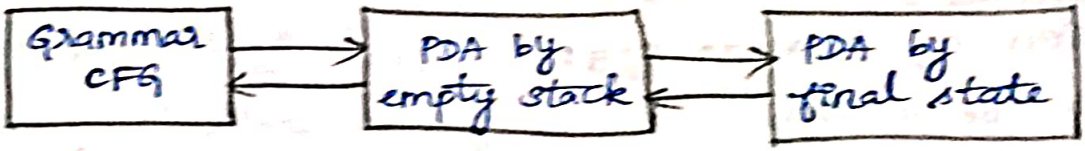
9|b) Instantaneous description:

The instantaneous description must record state and stack content if $M = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$ be a PDA then $(q, aw, z\alpha) \vdash (p, w, \beta\alpha)$ if $\delta(q, a, z) = (p, \beta)$

Equivalence of PDA and CFG

There are 3 classes of language:

i) context free language
ii) Language accepted by final stack of PDA
iii) Language accepted by empty stack of PDA



From CFG to PDA:

Let $G = \{V, T, P, S\}$ be CFG, then construct a PDA, P that accepts $L(G)$ by empty stack as follows,

$P = (\{q\}, T, VUT, \delta, q, \{\})$ where δ is defined by for each variable.

i) $\delta(q, \epsilon, A) = \{(q, p) / A \to \beta$ is a production in G
ii) $\delta(q, a, a) = \{q, \epsilon\}$

1.) Eg: Convert the grammar $E \to E+E$, $E \to id$ the PDA is given by $P = (\{q\}, \{+, id\}, \{E, id, +\}, \delta, q, E)$ where δ is defined by $\delta(q, \epsilon, E) = \{\{q, E+E\}\{q, id\}\}$

sol:

$\delta(q, +, +) = \{(q, \epsilon)\}$
$\delta(q, id, id) = \{(q, \epsilon)\}$

and check whether input id+id+id is accepted or not

1) $\delta(q, id+id+id, E) \vdash (q, id+id+id, E+E)$
$\vdash (q, id+id+id, id+E)$
$\vdash (q, id+id, E+E)$
$\vdash (q, id+id, id+E)$
$\vdash (q, id, id)$
$\vdash (q, \epsilon, \epsilon)$
$\therefore$ Accepted.

2) $S \to 0BB$, $B \to 0S/1S/0$ and check for 000.

Sol:

$P = (\{q\}, \{0,1\}, \{S, B, 0, 1\}, \delta, q, \{S\})$ where δ is

$\delta(q, \epsilon, S) = (\{q, 0BB\})$

$\delta(q, \epsilon, B) = (\{q, 0S\}\ \{q, 1S\}\ \{q, 0\})$

$\delta(q, 0, 0) = \{(q, \epsilon)\}$

$\delta(q, 1, 1) = \{(q, \epsilon)\}$

$\delta(q, 000, S) \vdash (q, 000, 0BB)$
$\vdash (q, 00, BB)$
$\vdash (q, 00, 0B)$
$\vdash (q, 0, B)$
$\vdash (q, 0, 0)$
$\vdash (q, \epsilon, \epsilon)$
$\therefore$ Accepted.

Conversion of PDA to CFG:

Let M be a PDA,

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ and $G = (V, T, P, S)$ then,

1. S is start symbol
2. V is a set of objects from $[P, a, q]$ where P and q in Q and A in Γ
3. P is a set of production defined by
   i) $S \to [q_0, z_0, q]$ from each small q in Q
   ii) Let $\delta(q, a, Y) = (Y_1, Y_2, \ldots Y_k)$ then

$[q, X, Y_k] \to a\,[Y\,Y\ ]\,[\ \ ]\ldots\ldots[Y_{k-1}\,Y_k, Y_k]$

For all states $Y_1, Y_2, \ldots Y_k$

1. construct a CFG for the PDA. Let M = ({q0, q1}, {0, 1}, {x, z0}, {δ, q0, z0, φ}) where (q0, 0, z0) = {(q0, x, z0)} push

Initial state, Initial stack symbol, Final state

δ(q1, 1, x) = {(q1, ∈)} → pop
δ(q0, z0, x) = {(q0, xx)} → push
δ(q1, ∈, x) = {(q1, ∈)} → pop
δ(q0, 1, x) = {(q1, ∈)} → pop
δ(q1, ∈, z0) = {(q1, ∈)} → pop

Sol:

V = {S, [q0, x, q0] [q0, x, q1] [q1, x, q0] [q1, x, q1] [q0, z0, q0] [q0, z0, q1] [q1, z0, q0] [q1, z0, q1]}

P1 : S → [q0, z0, q0]
P2 : S → [q0, z0, q1]

Push operation:

1) δ(q0, 0, z0) = {(q0, xz0)}

Possibility states, final state

q0 as final state P3 : [q0, z0, q0] → 0 [q0, x, q0] [q0, z0, q0]
P4 : [q0, z0, q0] → 0 [q0, x, q1] [q1, z0, q0]

q1 as final state P5 : [q0, z0, q1] → 0 [q0, x, q0] [q0, z0, q1]
P6 : [q0, z0, q1] → 0 [q0, x, q1] [q1, z0, q1]

2) δ(q0, 0, x) = {(q0, xx)}

q0 as final state P7 : [q0, x, q0] → 0 [q0, x, q0] [q0, x, q0]
P8 : [q0, x, q0] → 0 [q0, x, q1] [q1, x, q0]

q1 as final state P9 : [q0, x, q1] → 0 [q0, x, q0] [q0, x, q1]
P10 : [q0, x, q1] → 0 [q0, x, q1] [q1, x, q1]

10/26 Monday.

Pop operation:

1.) δ(q1, 1, x) = {(q1, ∈)}

P11 : [q1, x, q1] → 1

2) δ(q1, ∈, x) = {(q1, ∈)}

P12 : [q1, x, q1] → ∈

3.) δ(q0, 1, x) = {(q1, ∈)}

P13 : [q0, x, q1] → 1

4.) δ(q1, ∈, z0) = {(q1, ∈)}

P14 : [q1, z0, q1] → ∈

Replace the variables by an alphabet $[qTq] \rightarrow A$, q_0, q_1 is in Q, (x, z_0) starting symbol in stack.

$[q_0, z_0, q_0] \rightarrow A$
$[q_0, z_0, q_1] \rightarrow B$
$[q_0, z_0, q_0] \rightarrow C$
$[q_1, z_0, q_1] \rightarrow D$

$[q_0, X, q_0] \rightarrow E$
$[q_0, X, q_1] \rightarrow F$
$[q_1, X, q_0] \rightarrow G$
$[q_1, X, q_1] \rightarrow H$

Productions,

P1 : $S \rightarrow A$
P2 : $S \rightarrow B$
P3 : $A \rightarrow 0EA$
P4 : $A \rightarrow 0FC$
P5 : $B \rightarrow 0EB$
P6 : $B \rightarrow 0FD$
P7 : $E \rightarrow 0EE$
P8 : $E \rightarrow 0FG$
P9 : $F \rightarrow 0EF$
P10 : $F \rightarrow 0FH$
P11 : $H \rightarrow 1$
P12 : $H \rightarrow \epsilon$
P13 : $F \rightarrow 1$
P14 : $D \rightarrow \epsilon$

The productions should not contain unreachable and non-generating symbols.

⇒ There is no production starting with C and G, therefore remove the production containing C and G.

Remove the recursive production $E \rightarrow 0EE$,

so remove the productions starting with E.

similarly, for A there is no starting production so remove the productions starting with A.

so, the productions are,

$S \rightarrow B$
$B \rightarrow 0FD$
$F \rightarrow 0FH \mid 1$
$H \rightarrow 1 \mid \epsilon$
$D \rightarrow \epsilon$

2) construct a CFG for a PDA, $A = (\{q_0, q_1\}, \{a, b\}, \{z_0, z\}, \delta, q_0, z_0, \phi)$ where, $\delta(q_0, b, z_0) = \{(q_0, zz_0)\} \rightarrow$ push

$\delta(q_0, \epsilon, z_0) = \{(q_0, \epsilon)\} \rightarrow$ pop

$\delta(q_0, b, z) = \{(q_0, zz)\} \rightarrow$ push

$\delta(q_0, a, z) = \{(q_1, z)\} \rightarrow$ No operation

$\delta(q_0, b, z) = \{(q_1, 0)\}$ → pop

$\delta(q_1, a, z) = \{(q_0, z_0)\}$ → No operation

Sol:

$V = \{ S, [q_0, z_0, q_0] [q_0, z_0, q_1] [q_1, z_0, z_0], [q_1, z_0, q_1],$
$[q_0, z, q_0] [q_0, z, q_1] [q_1, z, q_0] [q_1, z, q_1] \}$

P1 : $S \rightarrow [q_0, z_0, q_0]$

P2 : $S \rightarrow [q_0, z_0, q_1]$

Push operation:

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

q_0 as final state P3: $[q_0, z_0, z] \rightarrow b\,[q_0, z, q_0][q_0, z_0, q_0]$

P4 : $[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_1][q_1, z_0, q_0]$

q_1 as final state P5 : $[q_0, z, q_1] \rightarrow b\,[q_0, z, q_0][q_0, z_0, q_1]$

P6 : $[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_1][q_1, z_0, q_1]$

2.) $\delta(q_0, b, z) = \{(q_0, zz)\}$

q_0 as final state P7 : $[q_0, z, q_0] \rightarrow b\,[q_0, z, q_0][q_0, z, q_0]$

P8 : $[q_0, z, q_0] \rightarrow b\,[q_0, z, q_1][q_1, z, q_0]$

q_1 as final state P9 : $[q_0, z, q_1] \rightarrow b\,[q_0, z, q_0][q_0, z, q_1]$

P10 : $[q_0, z, q_1] \rightarrow b\,[q_0, z, q_1][q_1, z, q_1]$

Pop operation

1. $\delta(q_0, \epsilon, z_0) = \{(q_0, \epsilon)\}$

P11 : $[q_0, z_0, q_0] \rightarrow \epsilon$

2. $(q_1, b, z) = \{(q_1, \epsilon)\}$

P12 : $[q_1, z, q_1] \rightarrow b$

No operation:

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

q_0 as final state P13 : $[q_0, z, q_0] \rightarrow a\,[q_1, z, q_0]$

q_1 as final state P14 : $[q_0, z, q_1] \rightarrow a\,[q_1, z, q_1]$

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

q_0 as final state : P15 : $[q_1, z, q_0] \to a[q_0, z_0, q_0]$

q_1 as final state : P16 : $[q_1, z, q_1] \to a[q_0, z_0, q_1]$

$[p, x, p] \to A$ $\quad$ $[p, z_0, p] \to E$

$[p, x, q] \to B$ $\quad$ $[p, z_0, q] \to F$

$[q, x, p] \to C$ $\quad$ $[q, z_0, p] \to G$

$[q, x, q] \to D$ $\quad$ $[q, z_0, q] \to H$

$S \to G|H$
$H \to 1DH$
$G \to 1DG$
$D \to 1D$

$C \to 1Dc|1CA$
$H \to \epsilon$
$A \to 1$

$\therefore$ F, E, B have no starting production so remove it.

$C \to 1CA$
$A \to 1$
$H \to \epsilon$

Remove the recursive production D, $D \to 1DD$

No production starting with D so remove the production with D.

$C \to 1CA$
$A \to 1$
$H \to \epsilon$

Problem

3) Let $M = (\{q\}, \{1, e\}, \{Z\}, \delta, q_1, z, \phi\}$ where δ is,

$\delta(q, i, Z) = \{(q, zZ)\}$

$\delta(q, e, Z) = \{(q, \epsilon)\}$.

4) convert the PDA $A = \{(p, q), \{0, 1\}, \{x, z_0\}, \delta, q, z, \phi\}$

$\delta(q, 1, z_0) = \{(q, xz_0)\}$ $\quad$ $\delta(q, \epsilon, z_0) = \{(q, \epsilon)\}$

$\delta(q, 1, x) = \{(q, xx)\}$ $\quad$ $\delta(p, 1, x) = \{(p, \epsilon)\}$

$\delta(q, 0, x) = \{(p, x)\}$ $\quad$ $\delta(q, 0, z_0) = \{(q, z_0)\}$

DETERMINISTIC PUSHDOWN AUTOMATA: (DPDA)

A PDA, $P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ is deterministic if and only if it satisfies the following conditions:

i) $\delta(q, a, x)$ has only one member for any q in Q and a in Σ and x in Γ

ii) If $\delta(q, a, x)$ is non-empty, for some a in Σ then $\delta(q, \epsilon, x)$ must be empty.

23/10/24 Wednesday.

UNIT-4

CONTEXT FREE LANGUAGES AND TURING MACHINES.

NORMAL FORMS OF CFG:

* Every CFL is generated by a CFG in which all productions are of the form,

$A \rightarrow BC$
$A \rightarrow a$

where, A, B, C are variable and a is terminal. This form of CFG is called chomsky Normal form (CNF)

* In order to find CNF we need to perform the following operations. $NT \rightarrow NT\ NT$, $NT \rightarrow T$

i) Eliminate useless symbols → unreachable symbol.
ii) Eliminate unit production ($A \rightarrow B$) RHS have single variable.
iii) Eliminate ϵ-production ($A \rightarrow \epsilon$) RHS have ϵ

1. Eliminate useless symbols [unreachable symbol]

The variable or terminal that do not appear in any derivation

Eg:

1. Consider the grammar $S \rightarrow AB/a$, $A \rightarrow b$ find CNF.

sol:

$S \rightarrow AB$
$S \rightarrow a$
$A \rightarrow b$

(∵ There is no unit production and ϵ-production)

B does not produce any string, so remove the production with B.

$S \rightarrow a$
$A \rightarrow b$

Now, A is not reachable, therefore remove the production with A.

∴ $\boxed{S \rightarrow a}$ //

2. Consider the grammar $S \rightarrow AB/CA$, $B \rightarrow BC/AB$, $A \rightarrow a$, $C \rightarrow ab/b$

sol:

$S \rightarrow AB/CA$
$B \rightarrow BC/AB$ (B don't have terminal)
$A \rightarrow a$
$C \rightarrow AB/b$

Check whether the string is generated.

↳ B is not generating any string therefore remove the production containing B.