UNIT- III

→ **Regular Languages:-**

The Language 'L' is defined over the alphabet Σ, is said to be Regular if it is accepted by some Finite Automata.

→ **Non-Regular Language:-**

The Language 'L' which is not a Regular is called as Non-Regular Language.

Ex:-

$L = \emptyset$ (Empty Language)

L= Finite Language (Countable strings)

Since Every FL is Regular

$L = \{0,1\}$ $\quad$ $L = \{00, 11, 000, 0110, 000\}$

$L = \{0^n / n \geq 1\}$ - 0^+ (positive closure) - Regular

$L = \{W \in \{0,1\}^* / |W| \geq 3\}$ - Regular

$L = \{W \in \{0,1\}^* / |W| = r(\text{mod})\, n\}$ - Regular

Construct FA

→ $L = \{0^n 1^n / n \geq 1\}$

Regular Expressions:-

The Expression which generates the regular language is called Regular Expression

(or)

An expression obtained string $*$, $+$, $\cdot$ is called as Regular Expression

Ex:- ① $r = a+b \Rightarrow L = \{a, b\}$

② $r = ab \Rightarrow L = \{ab\}$

③ $r = ab+ba \Rightarrow L = \{ab, ba\}$

④ $r = a^* \Rightarrow L = \{\epsilon, a, aa, \ldots\}$

⑤ $r = a^* b \Rightarrow L = \{b, ab, aab, \ldots\}$

⑥ $r = (ab)^* \Rightarrow L = \{\epsilon, ab, abab, ababab, \ldots\}$

Regular operators:-

The Regular operators are $+$, $\cdot$, $*$

$*$ - The kleen closure

$\cdot$ - Concatination (ab, a.b)

$+$ - union

Order of preference

$*$ - 1

$\cdot$ - 2

$+$ - 3

$$(\underbrace{ab}_{2} \underbrace{+a}_{3})^{\overset{1}{*}}$$

Types of Regular expressions

Regular Expressions

① ② ③

① Restricted RE $(*, \cdot, +)$ | ② Semi Restricted RE $(*, \cdot, +, \cap)$ | ③ Unrestriced RE $(*, \cdot, +, \cap, \sim)$

Ex:-

S NO	R.E	R=L
①	$r = \phi$	$L = \{ \}$
②	$r = \epsilon$	$L = \{\epsilon\}$
③	$r = a$	$L = \{a\}$
④	$r = ab$	$L = \{ab\}$
⑤	$r = a+b+c$	$L = \{a, b, c\}$
⑥	$r = W_1 + W_2 + \cdots + W_n$	$L = \{W_1, W_2, \ldots W_n\}$
⑦	$r = a^+$	$L = \{a, aa, aaa \ldots\}$

Note:-

① If R is Regular Expression then $L(R)$ is the language generated by Regular Expression R

② Every finite language is Regular

③ If R_1, R_2 be the 2 Regular expressions then both $R_1 + R_2$, $R_1 R_2$ are also regular Expressions then

$L(R_1 + R_2) = L(R_1) + L(R_2)$

$L(R1 \cdot R2) = L(R1) \cdot L(R2)$

Ex: $S1 = \{01, \epsilon\}$ $S2 = \{1, 11\}$

$R1 = \epsilon + 01$ $R2 = 1 + 11$

$= 1(\epsilon + 1)$

$R1 + R2 = \epsilon + 01 + 1(\epsilon + 1)$ $\boxed{R_1 + R_2 = R_2 + R_1}$

$= \epsilon + 01 + 1 + 11$

$L(R_1 + R_2) = \{\epsilon, 1, 01, 11\} = L(R1) + L(R2)$

$R1 \cdot R2 = (\epsilon + 01)(1 + (\epsilon + 1))$

$= (\epsilon + 01)(1 + 11)$

$= \{1 + 11 + 011 + 0111\}$

$L(R1 \cdot R2) = L(R1) \cdot L(R2)$ $\boxed{R_1 \cdot R_2 \neq R_2 \cdot R_1}$

(4) If R is a regular Expression then both R^*, R^+ are also regular expressions.

$R^* = \{\epsilon, R, RR, RRR, \dots\}$

$R^+ = \{R, RR, RRR, \dots\}$

$L(R^*) = [L(R)]^*$

$L(R^+) = [L(R)]^+$

⑤ If $r = \phi$ then $r^* = \{\epsilon\}$ and $r^+ = \phi$

⑥ If $r = \epsilon$ then $r^* = \{\epsilon\}$ and $r^+ = \{\epsilon\}$

$\epsilon, \epsilon\epsilon, \epsilon\epsilon\epsilon$

⑦ If r is a Regular Expression then

In general r^*, r^+ are infinite languages except for $r = \phi$ or ϵ

⑧ If $r = \epsilon$ or ϕ then r^*, r^+ are finite Languages

⑨ If R is Regular Expression then

a) $R^+ \subset R^*$

b) $R^+ \cup R^* = R^*$

c) $R^+ \cap R^* = R^+$

d) $R^+ + \epsilon = R^*$

e) $R^* R^+ = R^+ R^* = R^+$

f) $(R^*)^* = R^*$

g) $R^{*\,*\,*\,*\cdots} = R^*$

h) $R^{+++} = R^+$

i) $(R^*)^+ = (R^+)^* = R^*$

j) $(((R^+)^*)^*)^{+\cdots} = R^*$

k) $(R^*)^* \cdot R^+ = R^+$

(10) If R_1, R_2 be two Regular expressions then

$$(R_1 + R_2)^* = (R_1^* + R_2^*)^*$$

(or)

$$= (R_1^* + R_2)^*$$

$$= (R_1 + R_2^*)^*$$

$$= (R_1^* \cdot R_2^*)^*$$

→ $(R_1 R_2)^* R_1 = R_1 (R_2 \cdot R_1)^*$

** → Regular Expression is not unique

Note:-

→ If $\Sigma = \{0, 1\}$ then

(1) $(0+1)^*$ ⇒ Generates of all strings of 0's and 1's including ϵ

(2) $(0+1)^+$ ⇒ Generates of all strings of 0's and 1's Excluding ϵ

(3) $(0+1)^n$ ⇒ Generates all strings of 0's and 1's of length n

⇒

Representor (Regular expression) ← $\boxed{\text{Regular Language } L = \{w_1, w_2, \ldots, w_n\}}$ → Generator (Regular Grammer)

↓

Acceptor (Finite Automata)

④ Construct Regular Expression for following language

① $L = \{a, b\} \Rightarrow r = a + b$

② $L = \{\epsilon, a, b\} \Rightarrow r = \epsilon + a + b$

③ $L = \{00, 01, 10, 11\} \Rightarrow r = (0+1)^2$

④ $L = \{ W \in \{a+b\}^* / |W| \text{ atleast } 3 \}$

$\hookrightarrow r = (a+b)^3 (a+b)^*$

⑤ $L = \{ W \in \{a+b\}^* / |W| \text{ atmost } 3 \}$

$\hookrightarrow r = \epsilon + (a+b) + (a+b)^2 + (a+b)^3$

⑥ $L = \{ W \in \{0,1\}^* / \text{W starts with 1 and ends with 0} \}$

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

⑦ $L = \{ W \in \{0,1\}^* / W \text{ ends with } 11 \}$

$r = (0+1)^* 11$

⑧ $L = \{ W \in \{0,1\}^* / W \text{ contains } 11 \text{ as substring} \}$

$r = (0+1)^* 11 (0+1)^*$

⑨ $L = \{ 00, 001, 0011, 00111, \ldots \}$

$r = 001^*$

⑩ $L = \{ W \in \{a,b\}^* / W \text{ has exactly } 2a\text{'s} \}$

$r = b^* a b^* a b^*$

⑪ $L = \{ W \in \{a,b\}^* / W \text{ has atleast } 2a\text{'s} \}$

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

(12) $L = \{W \in \{0,1\}^* / W$ has exactly one $0\}$

$r = 1^* 0 1^*$

(13) $L = \{W \in \{0,1\}^* / W$ has atmost one $0\}$

$r = 1^* + 1^* 0 1^*$

(14) $L = \{W \in \{a,b\}^* / W$ have even length string$\}$

$r = ((a+b)^2)^*$

(15) $L = \{W \in \{a,b\}^* / |W|$ at most $2\}$

$r = \epsilon + (a+b) + (a+b)^2$

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

Atmost $|W| = 1 \Rightarrow a + b + \epsilon$

(16) $L = \{W \in \{a,b\}^* / W$ has set of all old length string$\}$

$r = (a+b)((a+b)^2)^*$

(17) $L = \{W \in \{a,b\}^* /$ length of the string is divisible by $3\}$

$r = ((a+b)^3)^*$

(18) $L = \{W \in \{a,b\}^* /$ No. of a's are atmost $2\}$

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

(19) $L = \{W \in \{a,b\}^* /$ No of a's in W are even$\}$

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

(20) $L = \{ w \in \{0,1\}^* \,/\, w$ starts and ends with different symbol $\}$

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

(21) $L = \{ w \in \{0,1\}^* \,/\, w$ starts and ends with same symbol $\}$

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

(22) $L = \{ w \in \{a,b\}^* \,/\, w$ contain atleast 2 a's $\}$

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

(23) $L = \{ w \in \{a,b\}^* \,/\, 4^{th}$ symbol from L.H.S is always b $\}$

$r = (a\cancel{+b)}^* \cancel{(a+}b \; (a+b)(a+b)(a+b)b(a+b)^*$

$r = (a+b)^3 b (a+b)^*$

(24) $L = \{ w \in \{a,b\}^* \,/\, 3^{rd}$ symbol from R.H.S is always a $\}$

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

$= (a+b)^* a (a+b)^2$

(25) $L = \{ w \in \{a,b\}^* \,/\,$ No 2a's should come together $\}$

$r = a(ba^{\bullet})^* b^* + b^*(ab)^*(a+b^*)$

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

(26) $L = \{ w \in \{0,1\}^* \,/\, w$ doesn't have substring 11 $\}$

$r = \cancel{(}1(01)^*0^* + 0^*(10)^*(1+0^*)$

(27) $L = \{w \in (0,1)^* /$ Atmost one pair of consecutive 1's $\}$

Identities

1. $\phi + R = R + \phi = R$

2. $\epsilon + R = R + \epsilon$

3. $\epsilon \cdot R = R \cdot \epsilon = R$

4. $\phi \cdot R = R \cdot \phi = \phi$

5. $R + R = R$

6. $\epsilon^* = \epsilon$

7. $RR^* = R^* R$

8. $\epsilon + RR^* = \epsilon + R^* R = R^*$

9. $(P+Q)R = PR + QR$

10. $(PQ)^* P = P(QP)^*$

11. $(R^*)^* = R^*$

12. $(P+Q)^* = (P^* + Q^*)^* = (P^* Q^*)^*$

$= P^* + Q^* = (P + Q^*)^* = \boxed{(P^* + Q)^*}$

$= P^* (Q P^*)^* = Q^* (P Q^*)^*$

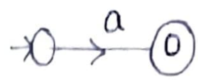
→ Equivalence between FA and RE

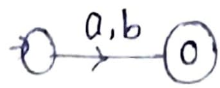
FA → RE

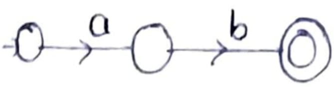
RE to FA

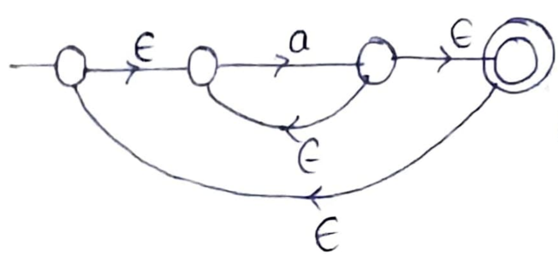
(1) $r = \emptyset \Rightarrow$ 

(2) $r = \epsilon \Rightarrow$ 

(3) $r = a \Rightarrow$ 

(4) $r = a+b \Rightarrow$ 

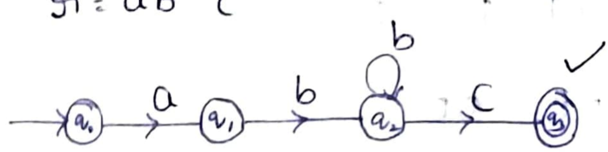
(5) $r = ab \Rightarrow$ 

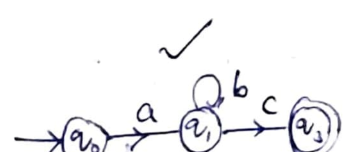
(6) $r = a^* \Rightarrow$ 

(preferable because $r \Rightarrow$ ε-NFA ↓ NFA ↓ DFA ↓ Minimal DFA

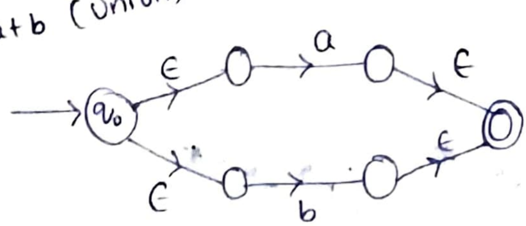
→ Construct FA for the Regular Expression

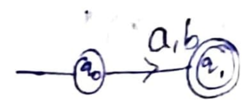
(1) $r = ab^*c$



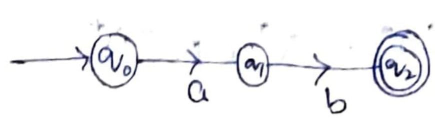


(7) $r = a+b$ (union)





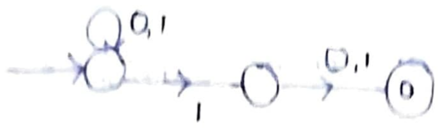
(8) $r = ab$ (concatination)



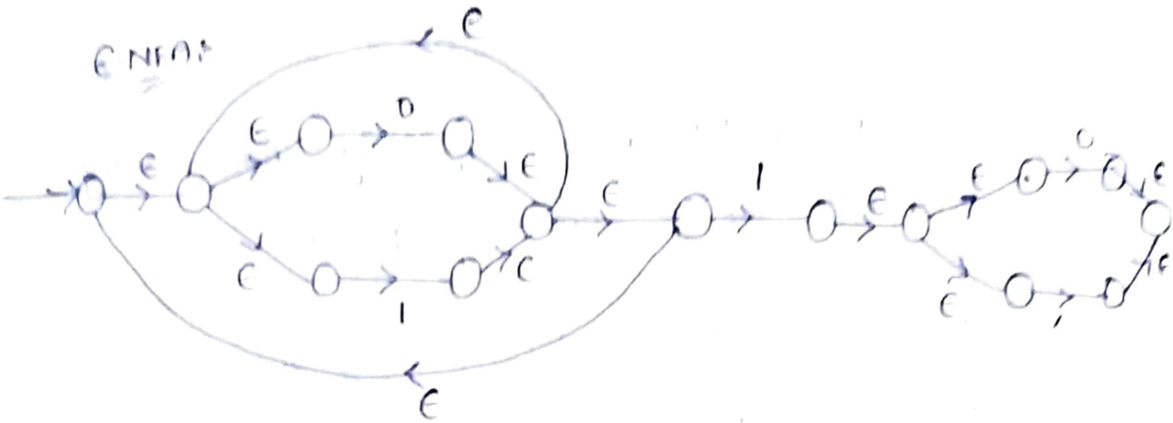
ab*c

(2) r (0+1)*1(0+1)

NFA:



ε NFA:



FA → RE

1. Arden's Lemma Method
2. State Elimination method

1

→ Arden's Lemma

This mechanism used for DFA and NFA only.

→ If P, Q be the 2 Regular Expressions such that

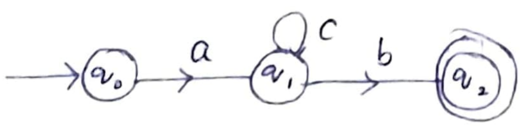
$R = Q + RP$ then

a) The equation have unique solution if P doesn't contains Epsilon and then solution is QP^*

i.e, $R = Q + RP \Rightarrow R = QP^*$ if P is free from ϵ

b) The $R = Q + RP$ has infinitely has many solutions if P contains Epsilon

Ex: Find RE (r) for following F.A by implementing Arden's method



$r = ac^*b$

$q_0 = \epsilon \rightarrow (1)$

$q_1 = q_0 a + q_1 c \longrightarrow (2)$

$q_2 = q_1 b \longrightarrow (3)$

$q_1 = \epsilon . a + q_1 c$

$q_1 = a + q_1 c$ (It is in the form of $R = Q + RP$)

$P = C$

C is free from Epsilon

$\therefore R = QP^*$

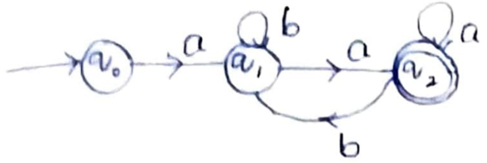
$\boxed{q_1 = ac^*}$

substitute q_1 in eq (3)

$\boxed{\therefore q_2 = ac^*b}$

$\therefore R = ac^*b$

(2)



$q_0 = \epsilon \rightarrow (1)$

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

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

substitute q_0 in eq(1)

$q_1 = q_0 a + q_1 b + q_2 b$

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

$q_1 = a + q_2 b + q_1 b$

$R = Q + RP \Rightarrow R = QP^*$

$q_1 = (a + q_2 b) b^*$

susbstitute q_1 in eq(3)

$q_2 = ((a + q_2 b) b^*) a + q_2 a$

$R = Q + RP \Rightarrow R = QP^*$

$\cancel{q_2 = a +}$

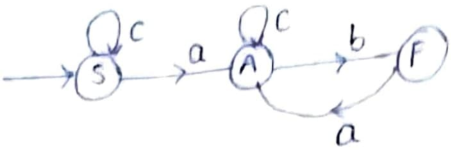
$q_2 = a b^* a + q_2 b b^* a + q_2 a$

$q_2 = a b^* a + q_2 (b b^* a + a)$

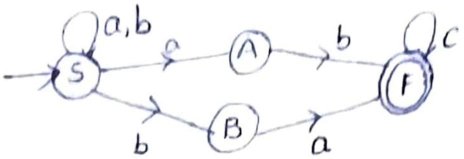
$R = Q + RP^* \Rightarrow R = QP^*$

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

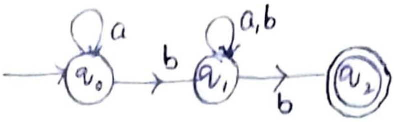
③



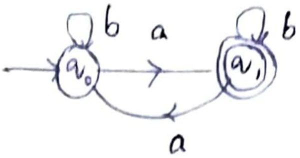
④



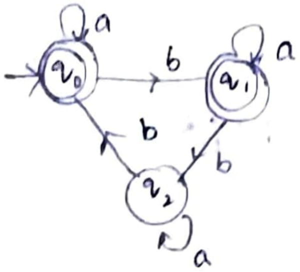
⑤



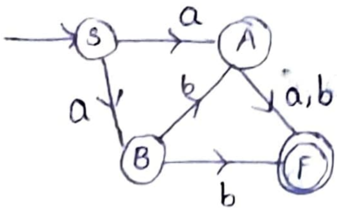
⑥



⑦



⑧



③ A)

$S = \epsilon + Sc \longrightarrow ①$

$A = Sa + Ac + Fa \longrightarrow ②$

$F = Ab \longrightarrow ③$

Eq ①

$S = \epsilon + Sc \Rightarrow S = \epsilon . c^* \Rightarrow S = c^* \longrightarrow ⓐ$

$R = Q + RP \Rightarrow R = QP^*$

substitute ⓐ in eq ②

$A = Sa + Ac + Fa$

$A = c^*a + Ac + Fa$

$A = \underline{c^*a + Fa} + Ac$

$R = Q + RP \Rightarrow R = QP^*$

$A = (c^*a + Fa)c^* \rightarrow (b)$

Substitute A in eq (3)

$F = Ab$

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

$F = c^*ab + Fac^*b$

$R = Q + RP \Rightarrow R = QP^*$

$\therefore F = c^*ac^*b(ac^*b)^*$

$$\boxed{\therefore RE = c^*ac^*b(ac^*b)^*}$$

(4)

$S = \epsilon + Sa + Sb \rightarrow (1)$

$A = Sa \rightarrow (2)$

$B = Sb \rightarrow (3)$

$F = Ab + Ba + FC \rightarrow (4)$

Eq (1)

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

$R = Q + RP \Rightarrow R = QP^*$

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

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

Substitute S in eq (2) and eq (3)

$A = Sa$ $\qquad$ $B = Sb$

$A = (a+b)^* a \rightarrow (b)$ $\qquad$ $B = (a+b)^* b \rightarrow (c)$

Substitute B, A in eq (4)

$F = Ab + Ba + Fc$

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

$R = \quad Q \quad + RP \Rightarrow R = QP^*$

$\therefore RE = (((a+b)^* a)b + ((a+b)^* b)a)c^*$

$\boxed{RE = (a+b)^* abc^* + (a+b)^* bac^*}$

(5A) $q_0 = \epsilon + q_0 a \longrightarrow (1)$

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

$q_2 = q_1 b \rightarrow (3)$

Eq (1)

$q_0 = \epsilon + q_0 a \Rightarrow q_0 = \epsilon . a^* \Rightarrow q_0 = a^* \rightarrow (a)$

$R = Q + RP \Rightarrow R = QP^*$

substitute eq (a) in eq (2)

$q_1 = q_0 b + q_1 a + q_1 b$

$\underset{R}{q_1} = \underset{Q}{a^* b} + \underset{RP}{q_1 (a+b)} \Rightarrow R = QP^*$

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

Substitute q_1 in eq (3)

$q_2 = q_1 b$

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

$\boxed{\therefore\ R.E = a^* b (a+b)^* b}$

⑥ A) $q_0 = \epsilon + q_0 b + q_1 a \rightarrow ①$

$q_1 = q_0 a + q_1 b \rightarrow ②$

eq①

$q_0 = \underline{\epsilon + q_1 a} + \underline{q_0 b}$

$R = Q + RP \Rightarrow R = QP^*$

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

Substitute ⓐ in eq②

$q_1 = q_0 a + q_1 b$

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

$= b^* a + q_1 a b^* a + q_1 b$

$q_1 = b^* a + q_1 (a b^* a + b)$

$R = Q + RP \Rightarrow R = QP^*$

$q_1 = b^* a (a b^* a + b)^*$

$\boxed{\therefore\ R.E = b^* a (a b^* a + b)^*}$

⑦ A) $q_0 = \epsilon + q_0 a + q_2 b \rightarrow ①$

$q_1 = q_0 b + q_1 a \rightarrow ②$

$q_2 = q_1 b + q_2 a \rightarrow ③$

Eq ①

$q_0 = \epsilon + q_0 a + q_2 b$

$q_0 = (\epsilon + q_2 b) + q_0 a.$

$R = Q + RP \Rightarrow R = QP^*$

$q_0 = (\epsilon + q_2 b) a^* \rightarrow$ ⓐ

Substitute eq ⓐ in eq ②

$q_1 = q_0 b + q_1 a$

$q_1 = (\epsilon + q_2 b) a^* b + q_1 a$

$R = Q + RP \Rightarrow R = QP^*$

$q_1 = (\epsilon + q_2 b) a^* b a^* \rightarrow$ ⓑ

Substitute eq ⓑ in eq ③

$q_2 = q_1 b + q_2 a$

$= (\epsilon + q_2 b) a^* b a^* b + q_2 a$

$= a^* b a^* b + q_2 b a^* b a^* b + q_2 a$

$\underbrace{q_2}_{R} = \underbrace{a^* b a^* b}_{Q} + \underbrace{q_2}_{R} \underbrace{(b a^* b a^* b + a)}_{P} \Rightarrow R = QP^*$

$\boxed{\therefore RE = a^* b a^* b (b a^* b a^* b + a)^*}$

⑧ $S = \epsilon \rightarrow$ ①

$A = Sa + Bb \rightarrow$ ②

$F = Aa + Ab + Bb \rightarrow$ ④

$B = Sa \rightarrow$ ③

eq (2)

$A = Sa + Bb$

$A = \epsilon . a + Bb \quad \because S = \epsilon$

$A = a + Bb \rightarrow$ (i)

eq (3)

$B = Sa$

$B = \epsilon . a \quad \because S = \epsilon$

$B = a \rightarrow$ (ii)

substitute eq (ii) in eq (i)

$A = a + Bb$

$A = a + ab \rightarrow$ (iii)

Substitute A,B in eq (iii)

$F = Aa + Ab + Bb$

$= A(a+b) + Bb$

$\boxed{RE = (a+ab)(a+b) + ab}$

(2) $\rightarrow$ State Elimination Method.

This mechanism is not only for FA, Also for Transition graph

FA : $M = \{Q, \Sigma, q_0, F, \delta\}$

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

NFA : $\delta : Q \times \Sigma \rightarrow 2^Q$

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

For ~~So,~~ Any Automata

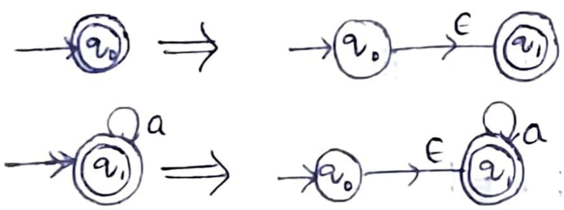
$\delta :- Q \times \Sigma^* \longrightarrow 2^Q$

→ Algorithm (steps has to be followed in order to get RE)

Step1: Simplify the Transition Graph / Finite Automata such that it has only one intial state and one final state.

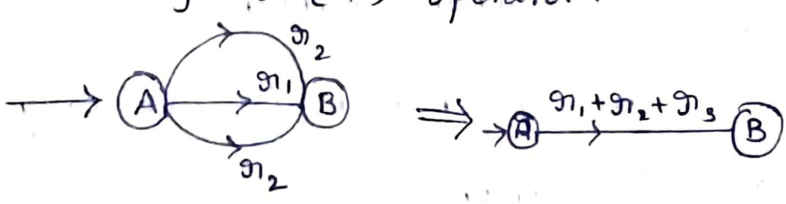
Step2: Simplify the system such a way that it has different initial and Final set

Ex:-

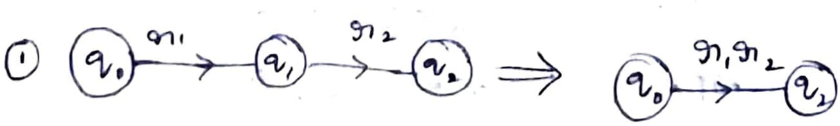


Step3: If there exists more than one edge/Transition between a pair of states then they are called parallel Edges and then combine parallel edges using or (+) operator.

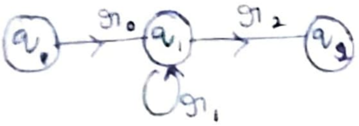
Ex:



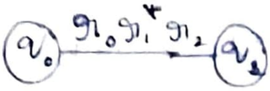
Step4: Eliminate the state q_1 from the following graph



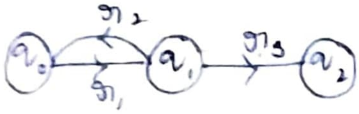
(2)



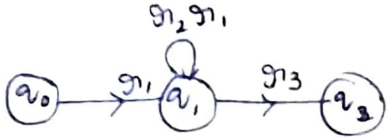
$\Rightarrow$



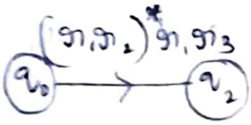
(3)

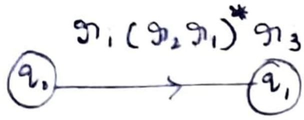


P-1 P-2



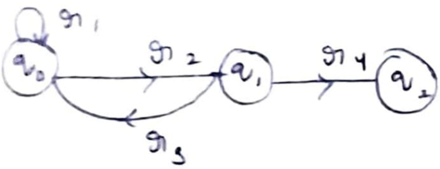
$\Downarrow$ $\Downarrow$



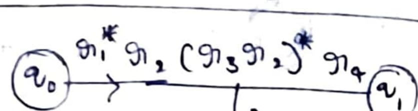


✓ (Better to shift the states to Non-initial and Non final states)

(4)



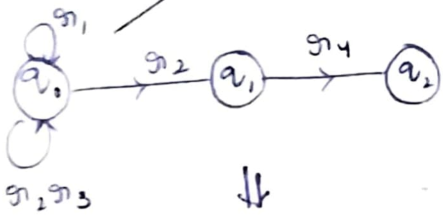
$\Rightarrow$

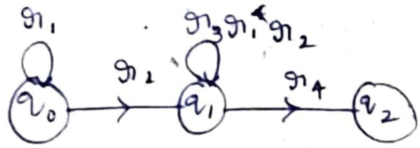


can't give like this because of r_1 loop

$\Downarrow$

P-1 P-2



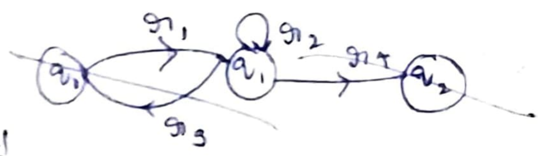


$\Downarrow$ $\Downarrow$

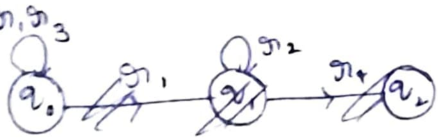
$(r_1 + r_2 r_3)^* r_2 r_4$

$r_1 r_2 (r_3 r_1^* r_2)^* r_4$

(5)

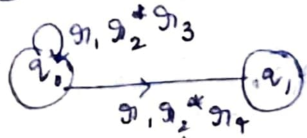


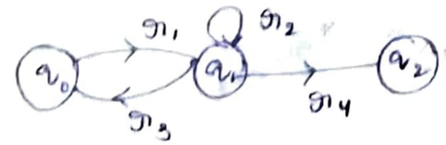
P-1



$(r_1 \cancel{r_3})^* r_1 r_2^* r_4$

P-2



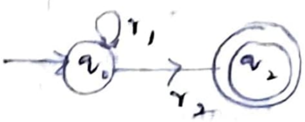
(5) 

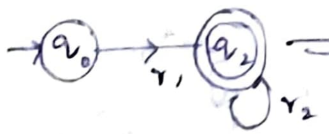
P-1 P-2

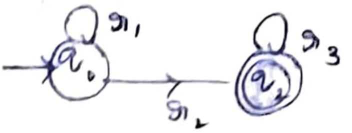
Steps: Continue the elimination of states until it takes any of the following form

(1)  $\Rightarrow r^*$

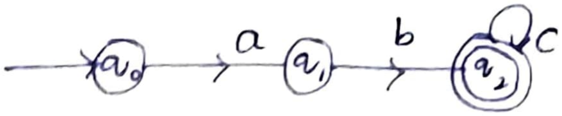
(2)  $\Rightarrow (r_1 + r_2)^*$

(3)  $\Rightarrow r_1^* r_2$

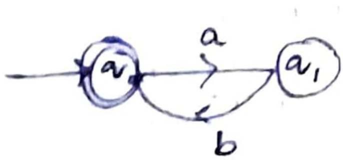
(4)  $\Rightarrow r_1 r_2^*$

(5)  $\Rightarrow r_1 r_2^* r_3$

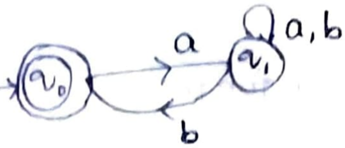
→ Construct the Regular Expression For Finite Automata

(1) 

(2) 

(3) 

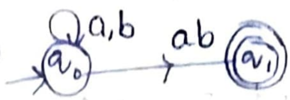
(4)



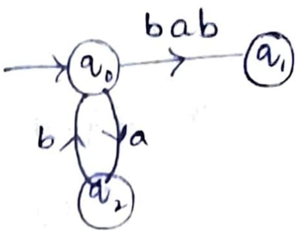
(5)

(Trasition graph)

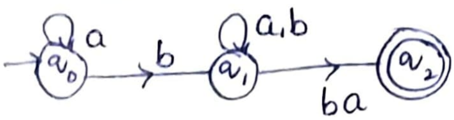
(6)



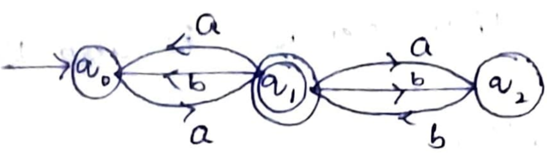
(7)



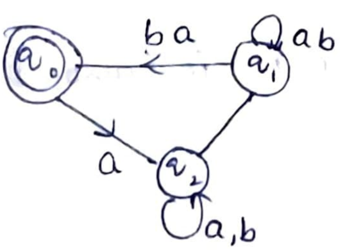
(8)



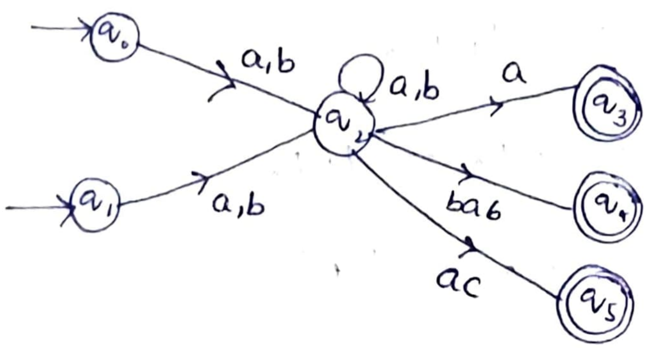
(9)



(10)



(11)



→ closure properties of Regular Language

Regular Language satisfy the closure properities with respect to the following operations

1. If L is Regular, then compliment of L is also regular

   Complement; $L \to L^c$

   If $L = \{\} \to R$ then

   $L^c = \{\} \to R$

2. Reversal : $L \to L^R$

   If L is Regular, then Reversal is also Regular

3. prefix ; $L \to$ prefix(L)

   If L is RE, then prefix(L) is also RE

4. kleen closure : $L \to L^*$

5. Union : $L_1, L_2 \Rightarrow L_1 \cup L_2$

6. Concatenation : $L_1, L_2 \Rightarrow L_1 \cdot L_2$

7. Intersection : $L_1, L_2 \Rightarrow L_1 \cap L_2$

8. Difference operator : $L_1, L_2 \Rightarrow L_1 - L_2$ is RE

   $L_2 - L_1$ is also RE

9. Symmetric Difference : $L_1, L_2 \Rightarrow L_1 \Delta L_2$

10. Quotient operator ; $L_1, L_2 \Rightarrow \frac{L_1}{L_2}$

11. Substitution

(12) Homomorphism

(13) Inverse Homomorphism

(14) Maximum, max(L)

(15) Minimum, min(L)

(16) Cycle operator

cycle = $\{xy / yx \in L\}$

(17) Init operator

(18) Complement OR

(19) Not OR

① <u>Complement of Finite Automata</u>

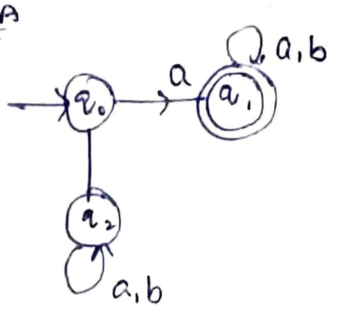
The FA which is obtained by interchanging final and Non-final states. is called as complement of FA

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

$L = \{ W \in (a+b)^* / W$ starts with $a)$

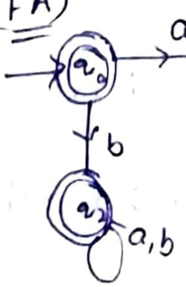
$RE = a(a+b)^* \Rightarrow RE^c = (a+b)^* - a(a+b)^*$

DFA



$\Rightarrow$

$(FA)^c$



$\boxed{L(FA^c) = \Sigma^* - L(FA)}$

① $L(FA) = \{ w \mid w$ ends with $ab \}$

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

② Reverse Operator

If 'L' is a Regular Language then Reverse of L is also Regular

process to get FA^R

① Interchange Initial and Final state

② Change the direction of edges

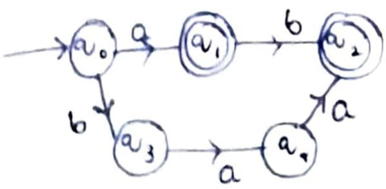
③ If more than one initial state occurs then combine those initial states into single Initial state by adding ϵ-transition

Ex: $L = \{a, ab, baa\}$

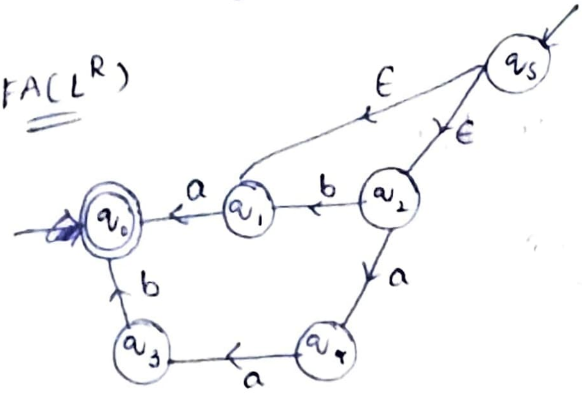
$L^R = \{a, ba, aab\}$

DFA for L

FA



⇒ FA(L^R)



③ prefix operator

If L is RL then prefix of 'L' is also RL

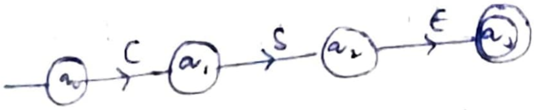
process

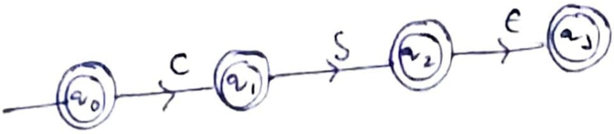
① In case of NFA, make all the states as final states.

② In case of DFA, make all the states as final states except dead states

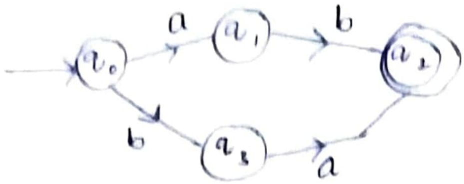
Ex: W = CSE

P(W) = ε, C, CS, CSE

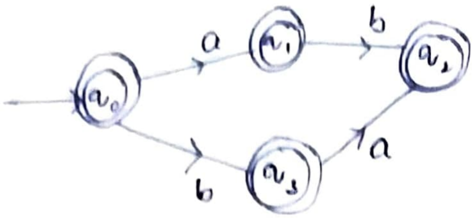




Ex:- $L = \{ab, ba\}$



prefix of $L = \{\epsilon, a, ab, b, ba\}$



(4) Kleen closure:-

If L is Regular then L^* is also Regular

(5) Union: Let L_1, L_2 be 2 Regular Languages then $L_1 \cup L_2$ is also regular.

(6) Concatination: $L_1 \cdot L_2$ is Regular

(7) Intersection:- $L_1 \cap L_2$ Regular

(8) Difference operator :- $L_1 - L_2$, $L_2 - L_1$ are Regular

$L_1 = \{\epsilon, a, b, aa, bb\}$ $L_2 = \{a, ab, ba, bb\}$

$L_1 - L_2 = \{\epsilon, b, aa\}$

$L_2 - L_1 = \{ab, ba\}$

(9) Symmetric Difference : $L_1 \Delta L_2$

$$L_1 - L_2 = L_2 - L_1$$

(10) Quotient operator :-

Let L_1, L_2 be 2 Regular languages then $\frac{L_1}{L_2}$ is also Regular Language where

$$\frac{L_1}{L_2} = \{ x \mid xy \in L_1 , y \in L_2 \}$$

$\frac{xy}{y} = x$ (Right Quotient)

$\frac{xy}{x} = y$ (Left Quotient)

(11) Note:- (1) $\frac{x}{x} = \epsilon$

(2) $\frac{x}{\epsilon} = x$

(3) $\frac{x}{y} = \phi$ (if $x \neq y$)

Ex:- (1) $\frac{101}{1} = 01$ or 10 → Right Quotient

Left Quotient

(2) $\frac{101}{101} = \epsilon$ (3) $\frac{1010}{01} = \phi$

(4) $\frac{1001}{01} = 10$ (5) $\frac{1000}{000} = 1$

Ex:- $L_1 = \{101, 011, 000\}$, $L_2 = \{10, 01, 11\}$

$$\frac{L_1}{L_2} = \{0, 1\}$$

$\frac{101}{10} = 1$, $\frac{011}{10} = 1$, $\frac{000}{10} = \phi$

$\frac{101}{01} = \phi$, $\frac{011}{01} = 1$, $\frac{000}{11} = 0$

$\frac{101}{11} = \phi$, $\frac{011}{11} = 0$, $\frac{000}{11} = \phi$

Note: If L is any language defined over Σ

① $\frac{\Sigma^*}{L} = \Sigma^*$

② $\frac{L}{\Sigma^*} = \text{prefix}(L)$

③ $\frac{RL}{RL}$, $\frac{RL}{NRL}$ → RL

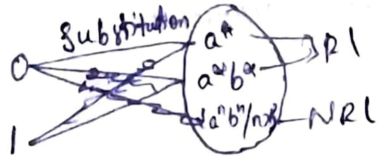
④ $\frac{NRL}{RL}$, $\frac{NRL}{NRL}$ → NRL

⑪ Substitution

$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

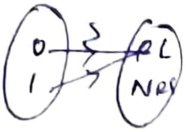
$\Delta^* = \{\epsilon, a, b, aa, bb, \ldots\}$

Σ 2^{Δ^*} $2^{\Delta^*} = \{$set of finite language$\}$



RL NL

$S(0) = a^*$, ~~$S(0)$~~
$S(1) = a^\alpha b^\alpha$



Def:- Let Σ and Δ be the two alphabets, then substitution is a mapping from $\Sigma \rightarrow \Delta$ where the symbols of Σ are mapped to the Regular Language of another alphabet Δ.

i.e, Substitution (S) $\longrightarrow$ $\Sigma \longrightarrow 2^{\Delta^*} \longrightarrow S(x) = L$

where $x \in \Sigma$

L is a RL over Δ

<u>Note:</u> If 'L' is a Regular and S is a substitution over "L" then S(L) is also Regular Language

$S(\epsilon) = \epsilon$

$S(\emptyset) = \emptyset$

$S(x^*) = (S(x))^*$

$S(x_1 + x_2 + \cdots + x_n) = S(x_1) + S(x_2) + \cdots + S(x_n)$

$S(x_1 \cdot x_2 \cdots x_n) = S(x_1) \cdot S(x_2) \cdots S(x_n)$

Ex: $\Sigma = \{0,1\}$ $\qquad \Delta = \{a,b\}$

$S(0) = a^*$ $\qquad S(1) = bab^*$

compute

① $S(0^*) = (S(0))^* = (a^*)^* = a^*$

② $S(1^*) = (S(1))^* = (bab^*)^*$

③ $S(01) = S(0) \cdot S(1)$

$= a^* \cdot bab^*$

(12) Homomorphism :-

Σ and Δ be the 2 alphabets then Homomorphism is a kind of substitution from Σ to Δ $\exists$ the symbols of Σ is mapped to only a single string of another alphabet &.

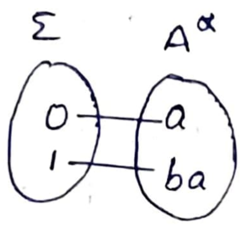
$$h : \Sigma \rightarrow \Delta^* \Rightarrow h(x) = y \quad \text{where } x \in \Sigma$$
$$y \in \Delta^*$$

Note:- If L is a RL and H is Homomorphism defined over L then

$h(L)$ is the homomorphic image of L and it is a RL

Ex: $\Sigma = \{0, 1\}$ $\quad \Delta = \{a, b\}$

$h(0) = a \quad h(1) = ba$



$L : \{00, 101\}$

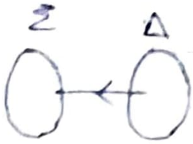
$h(L) = \{aa, baaabb\}$

$L = 0^* (10)^*$

$h(L) = a^* (baa)^*$

(13) Inverse Homomorphism

If L is Regular Language defined over alphabet Σ, & h is a homomorphism over L then $h^{-1}(L)$ is also Regular



where $h^{-1}(L) = \forall\ x \in h(L)$

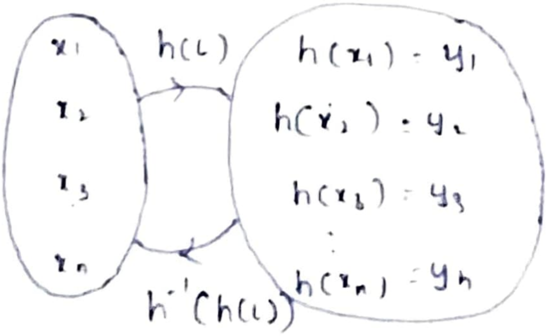
$$z \in h(L) \Leftrightarrow \exists\ x \in L \Rightarrow h(x) = z$$

$z \notin h(L) \Leftrightarrow$ there is no any x is in L such that $h(x) = z$

$$h^{-1} = \{x \in L \,/\, h(x) \in h(L)\}$$

$$= \{x \in L \,/\, \exists\, y \in h(L) \Rightarrow h(x) = y\}$$

Ex:



Ex: $\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

$h(0) = a^*$ $h(1) = ab^*$

$L = \{01\}$ $h(L) = a^* ab^*$

$h^{-1}(L) = h^{-1}(a^* ab^*)$

$= h^{-1}(a^*)\, h^{-1}(ab^*) = h^{-1}(h(0))\, h^{-1}(h(1))$

$= 01$

→ If L is Regular then min(L) is also Regular

→ Cycle operator:

cycle: $\{xy / yx \in L\}$

$\{abc / cba \in L\}$

→ Init operator

Init : $\{x / xy \in L_1, y \in L_2\}$

→ Consider a language L

L < Regular (Maybe) / Irregular (Maybe)

$L = \{ \ldots \}$ ⇒ By seeing language we can't say either RL or NRL.

→ Pumping Lemma Theorem is used to prove the given L is NRL.

PL → statisfies < Maybe Regular / May be Non-Regular

PL → doesn't satisfies → NRL

Pumping Lemma Theorem

Note points

→ PL is used to prove that some of the languages are not regular

→ PL is used only for a Non-Regular Languages i.e, For PL, The input is NRL, O/P is also Non-Regular.

$$\xrightarrow{\text{NRL}} \boxed{\text{Pumping Lemma}} \xrightarrow{\text{NRL}}$$

→ Every RL satisfies the PL properties

→ The language which doesn't satisfy the PL property is always NRL.

→ The language which satisfies the PL property then it may be Regular or Non-Regular.

Statement

Let L be the RL then there exists a constant n (which depends on L) such that for every string W in L such that $|W| \geq n$.

We can break w into 3 strings $w = xyz$

such that

① $y \neq \epsilon$

② $|xy| \leq n$

③ $\forall\ k \geq 0$, the string xy^iz is also in L

Process of pumping Lemma

① Assume that L is a Regular Language

- L satisfies pumping property

② Choose $w \in L$, $|w| \geq n$ where n is PL constant

③ We can split w into 3 parts.

x y z

① $y \neq \epsilon$, $|y| \neq 0$

② $|xy| \leq |w|$

③ $\forall\ i \geq 0$, $xy^iz \in L$, since L is RL

④ If there exists one value for i, such that $xy^iz \notin L \Rightarrow$ L is Non-Regular Language

Ex: prove that the Language $L = \{a^n b^n / n \geq 1\}$ is NRL by using PL.

step1: Assume L is Regular Language { L satisfies PL property }

step2: Let $W \in L$, $|W| \geq n$

$W = a^k b^k$, $k \geq 1$ $\quad |W| = 2k$

step3:-

$$W = \underbrace{a^{k-1}}_{(x)} \; \underbrace{a}_{(y)} \; \underbrace{b^k}_{z}$$

can take (a^{k-2}, a^2, b^k)

① $y \neq \epsilon$

② $|xy| \leq |W|$

Impstep:- $\forall i \geq 0$, $xy^iz \in L$

$i=0 \Rightarrow xy^iz \Rightarrow xy^0z \Rightarrow xz \Rightarrow a^{k-1}.b^k \in L$

But $a^{k-1}.b^k \notin L$

$i=1 \Rightarrow xy^1z \Rightarrow xyz \Rightarrow a^{k-1}.a.b^k \Rightarrow a^k b^k \in L$

$i=2 \Rightarrow xy^2z \Rightarrow a^{k-1}.a^2 b^k \Rightarrow a^{k+1}.b^k \in L$

But $a^{k+1}.b^k \notin L$

$i=3 \Rightarrow xy^3z \Rightarrow a^{k-1} a^3 b^k \Rightarrow a^{k+2} b^k \in L$

But $a^{k+2}.b^k \notin L$

* Except $(i=1)$ $\forall i>1$ ~~and~~, $i=0$, $xy^iz \notin L$)

∴ The language failed to satisfy PL property

∴ Our assumption is ~~not~~ ~~Regular~~, wrong

∴ The language is Non-Regular Language

② prove that the language $L = \{ a^{n^2} / n \geq 0 \}$ is Non-Regular by PL

$L = \{ \epsilon, a, a^4, a^9, \ldots \}$

step1:- Assume L is RL (L satisfies PL)

step2:- Let $W \in L$, $|W| \geq n$

$W = a^{k^2}$

step 3:

$W = a^{k^2}$

$a^{k^2-1} \quad a \quad \epsilon$

① $y \neq \epsilon$

② $|xy| \leq |W|$

$\forall i \geq 0 \quad xy^i z \in L$

$i = 2 \Rightarrow a^{\cancel{k}\,k^2-1} a^2 \epsilon \Rightarrow a^{k^2+1} \in L$

∴ But $a^{k^2+1} \notin L$

∴ It doesn't satisfy PL property

Our assumption is wrong

∴ Language is NRL

Assignment

(3) prove that $L = \{ a^{n!} / n \geq 0 \}$ is NRL

(4) prove that $L = \{ W / W$ is a palindrome number over $\{0,1\} \}$ is NRL

$L = \{0, 1, 101, 010, 0110, 1001, \ldots \}$

(5) prove that $L = \{ a^p / p$ is prime number$\}$ is NRL

Note:- <u>weak form of pumping lemma</u>

If L is any language defined over alphabet containing only one symbol

$\Sigma = \{a\}$ Then L is Regular if length of strings of Language L are in some Arithmetic progression (AP) otherwise L is non-Regular.

Ex:- (1) $L = \{ a^n / n \geq 0 \}$

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

$0, 1, 2, 3, 4 \ldots$ in AP

$\therefore$ L is RL

(2) $L = \{ a^{2n} / n \geq 0 \}$

$0, 2, 4, 6,$ in AP $\therefore$ L is RL

(3) $L = \{ a^{3n+2} / n \geq 0 \}$

$2, 5, 8, 11, \ldots$ in AP

$\therefore$ L is RL

④ $L = \{ a^{n^2} / n \geq 0 \}$

$0, 1, 4, 9, 16, \ldots$ Not in AP

$\therefore$ L is NRL

⑤ $L = \{ a^{n^3} / n \geq 0 \}$

$0, 1, 8, 27, \ldots$ Not in AP

$\therefore$ L is NRL

⑥ $L = \{ a^{n^2+n+1} / n \geq 0 \}$

$1, 3, 7, \ldots$ Not in AP

$\therefore$ L is NRL

⑦ $L = \{ a^{n!} / n \geq 0 \}$

$1, 1, 2, 6$ Not in AP

$\therefore$ L is NRL

⑧ $L = \{ a^p /$ p is prime number $\}$

$2, 3, 5, 7, 11, \ldots$ Not in AP

$\therefore$ L is NRL

Note:-

* Union of NRL is need not be NR

$L_1 = \{ a^p /$ p is prime number $\}$

$L_2 = \{ a^p /$ p is not a prime number $\}$

$L_1 \cup L_2 = \{ a^n / n \geq 1 \} \Rightarrow RL$