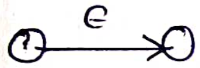
Unit-2

* **Regular Expression**:- The language accepted by finite automata is easily described by simple expression called regular expression.

* **Regular Language**:- Language accepted by regular expression.

The regular expression over Σ can be defined

1. ϕ is a regular expression which denotes empty set $\{\}$.
2. ϵ is a RE denotes the set $\{\epsilon\}$. It is called as null string.

   eg:- 

3. for each "A" in Σ just A is a regular expression if the language is $\{a\}$.
4. If "r" and "s" are RE's denoting languages L1 and L2 respectively then the regular expression is $r+s$; $L_1 \cup L_2$

   eg:- $a+b$. $\{a\} \cup \{b\}$

5. Concatenation: rs ; $L_1 L_2$

   eg:- ab $\{a\}\{b\}$

$a^* = \{\epsilon, a, aa, aaa, ----\}$

$a^+ = \{a, aa, aaa, ----\}$

Q:- Design a regular . E for the language contains all the strings with a and b combinations.

Ans:- Universal language for a and b is

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

Q:- Construct RE for lang accepting all string should end with 00 / $\Sigma = \{0,1\}$

$RE = (0+1)^* 00$

Q:- Construct RE starts with 1 ends with 0

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

Q:- Atleast 1 a followed by b

$RE = a^+ \cdot b$ $\qquad a^+b$

Q:- Substring 110

$RE = (0+1)^* 110 (0+1)^*$.

Q:- Having atleast 2 0's.

~~$RE = (0+1)^* (00)^+ (0+1)^*$~~ (wrong)

$RE = (0+1)^* 0 (0+1)^* 0 (0+1)^* = \Sigma^* 0 \Sigma^* 0 \Sigma^*$

Q:- Exactly 2 0's.

$RE = 1^* 0 1^* 0 1^*$

Q:- atmost 2 0's.

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

* Identity Rules :-

$I_1 : \phi + r = r$

$I_2 : \phi r = r \phi$

$I_3 : \epsilon r = r \epsilon = r$

$I_4 : \epsilon^* = \epsilon$ and $\phi^* = \epsilon$

$I_5 : r + r = r$

$I_6 : r^* r^* = r^*$

$I_7 : r r^* = r^* r = r^+$

$I_8 : (r^*)^* = r^*$

$I_9 : \epsilon + r r^* = r^*$

$I_{10} : (pq)^* p = p(qp)^*$

$I_{11} : (p+q)^* = (p^* q^*)^* = (p^* + q^*)^*$

I_{12}: $(p+q)r = pr + qr \not\equiv r(p+q) = rp + rq$

Q:- Prove that $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

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

$L.H.S = (1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= P + PQ^*Q$

$= P(\epsilon + Q^*Q)$

$= P(\epsilon + Q^+)$

$= PQ^*$

$= (1+00^*1)(0+10^*1)^*$

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

$= (\epsilon + 0^+)1(0+10^*1)^*$

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

$= R.H.S.$

Q:- $0^*(10^*)^*$

a) $(1^*0^*)1^*$ b) $0+(0+10)^*$

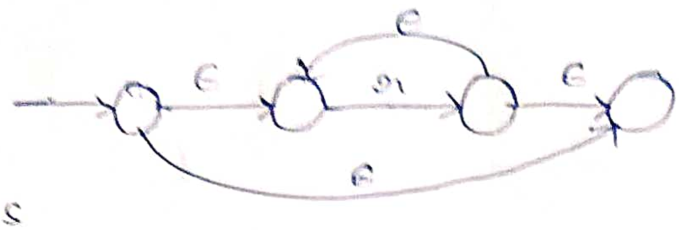
c) $(0+1)^*10(0+1)^*$ d) $(0+10)^*$

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

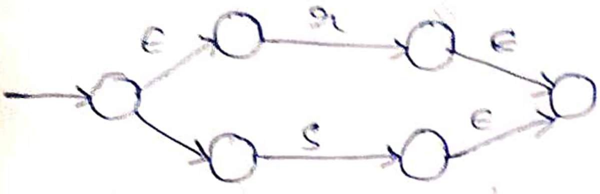
$0^*1 + 0^*1^*$

* Conversion from expression to finite automata

→ r^*

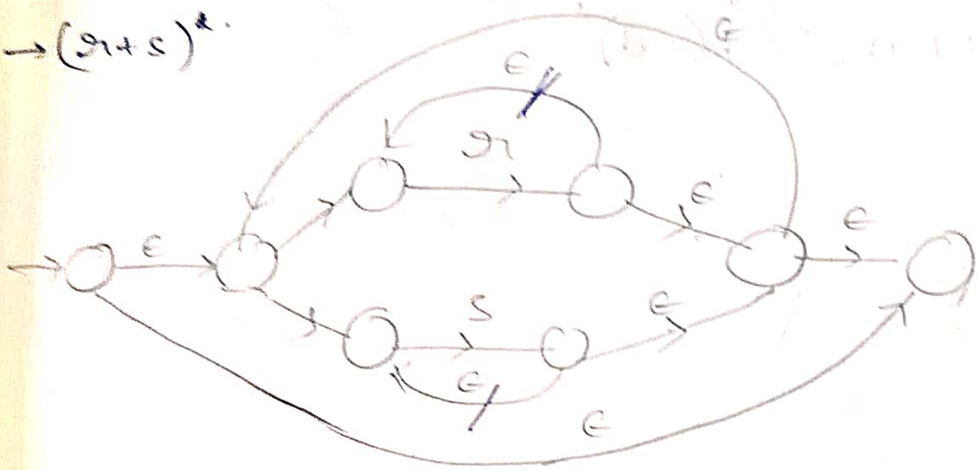


→ $r+s$



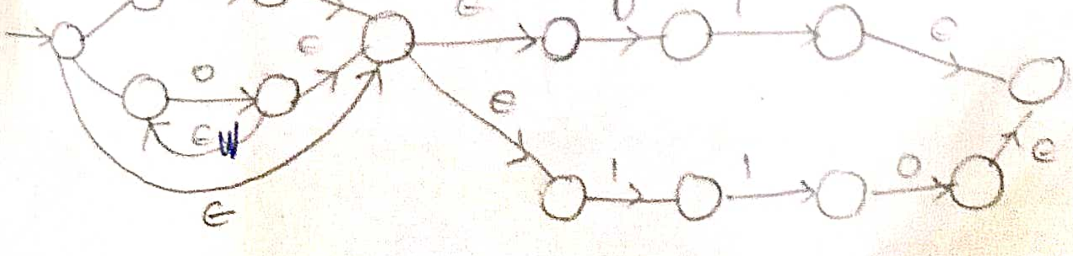
→ rs

→ $(r+s)^*$

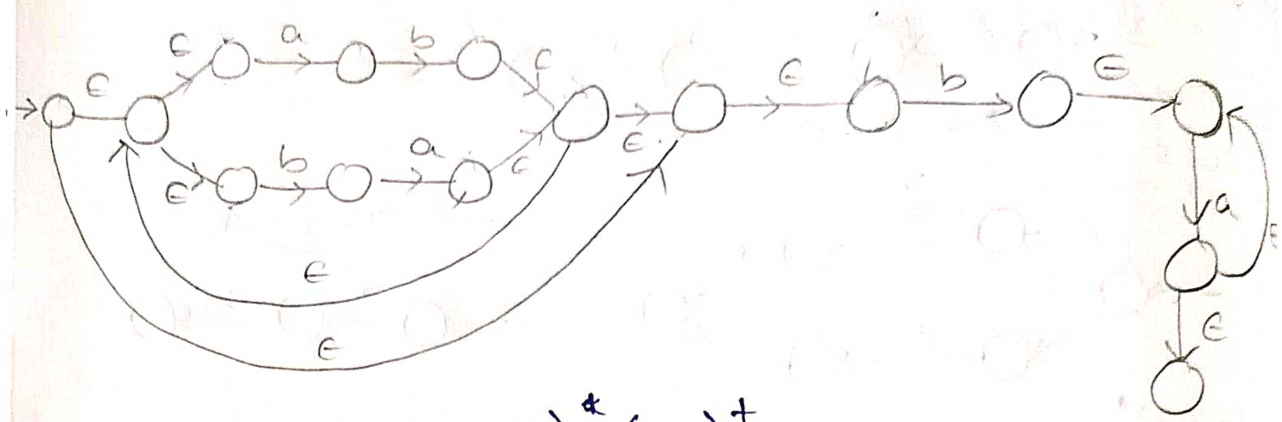


→ Construct finite automata for $b^*(aa)^*b^*$.

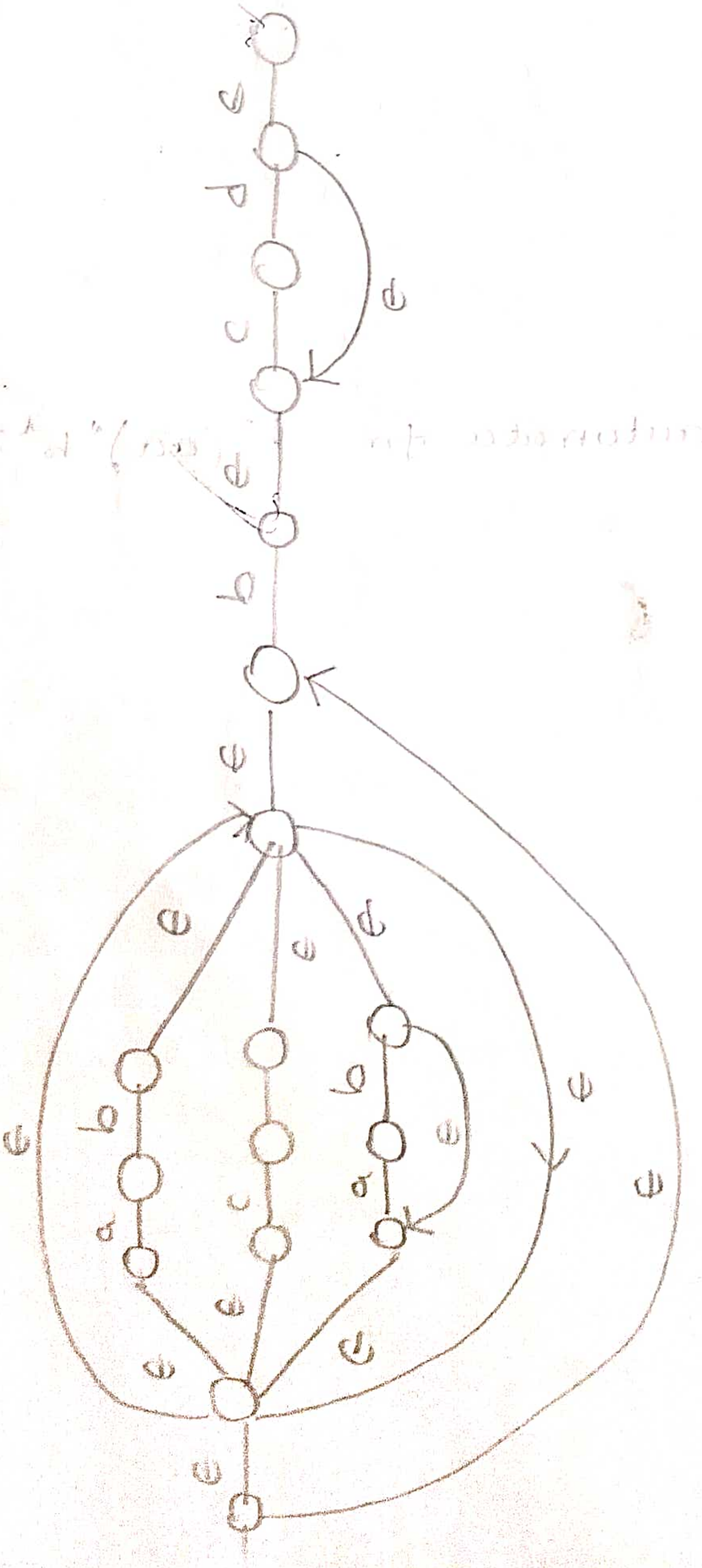
→ $(0+1)^*(01+110)$



Ex1:- $(ab+ba)^* ba^+$

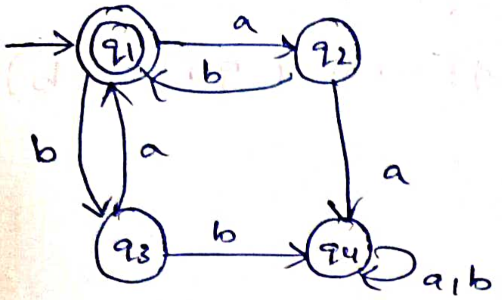


Q:- $(ab+cd+(ab)^*)^* b(cd)^+$



* finite automata to Regular expression by using Ardens theorm :-

Ex:-



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

→ only 1 initial state

→ no null transitions

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

$q_2 = q_1 a$

$q_3 = q_1 b$

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b.$

for final state

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

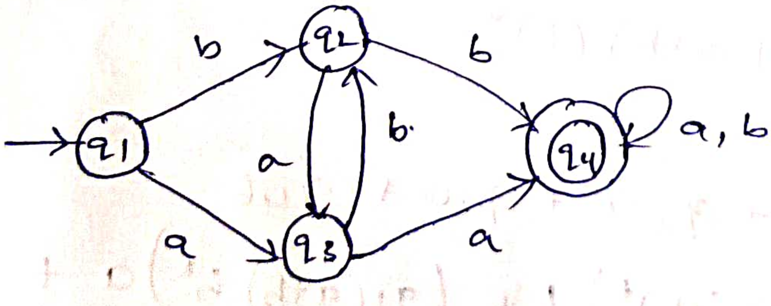
$= \epsilon + q_1 ab + q_1 ba$

$= \epsilon + q_1 (ab + ba)$

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

$= (ab + ba)^*$

Ex:-



$q_1 = \epsilon.$

$q_2 = q_1 b + q_3 b = (q_1 + q_3) b.$

$q_3 = q_1 a + q_2 a$

$q_4 = q_2 b + q_3 a + q_4 a + q_4 b.$

For final state

$$q_4 = q_2 b + q_3 a + q_4 a + q_4 b$$

$$= (q_1 b + q_3 b) b + (q_1 a + q_2 a) a + q_4 (a+b)$$

$$= (q_1 + q_3) bb + (q_1 + q_2) aa + q_4 (a+b)$$

$\Rightarrow$ $q_2 = q_1 b + q_3 b$

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

$$= q_1 b + q_1 ab + q_2 ab$$

$$= q_1 b (\epsilon + a) + q_2 ab$$

$$= \underbrace{q_1 ab}_{Q} + \underbrace{q_2}_{r}\underbrace{ab}_{P}$$

$$q_2 = q_1 ab (ab)^*$$

$\Rightarrow$ $q_3 = q_1 a + q_2 a$

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

$$= q_1 \underbrace{(a+b)}_{Q} + \underbrace{q_3}_{r}\underbrace{b}_{P}$$

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

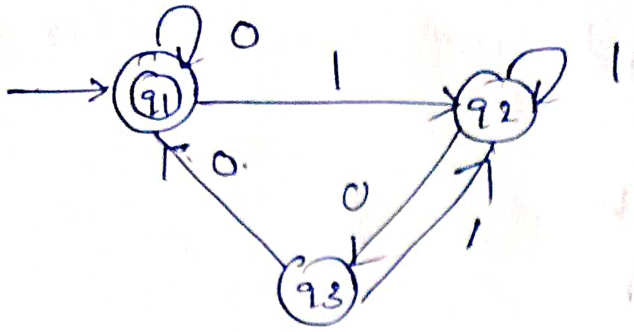
$\Rightarrow$ $q_4 = q_2 b + q_3 a + q_4 a + q_4 b$

$$= ((q_1 ab)(ab)^*) b + (q_1 (a+b) b^*) a + q_4 a + q_4 b$$

$$= q_1 (ab(ab)^* b + (a+b) b^* a) + q_4 (a+b)$$

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

Ex:



$q_1 = q_1 0 + q_3 0$

$q_2 = q_1 1 + q_2 1 + q_3 1 = (q_1 + q_3) 1 + q_2 1 = (q_1 + q_3) 1 1^*$

$= (q_1 + q_3) 1^+$

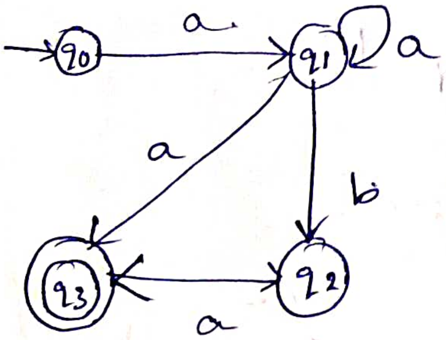
$q_3 = q_2 0$

$q_1 = q_1 0 + q_3 0$

$= q_1 0 + q_2 0 0$

$= q_1 0 +$

ex:-



$q_0 = \epsilon$

$q_1 = q_0 a + q_1 a = q_0 a a^* = q_0 a^+ = \epsilon a^+ = a^+$

$q_2 = q_1 b = a^+ b$

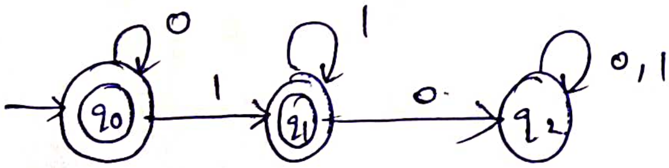
$q_3 = q_2 a + q_1 a = a^+ b a + a^+ a = a^+ a (b+\epsilon)$

$= a^+ a b.$

$q_3 = q_2 a + q_1 a$

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

ex:-



$q_0 = q_0 0 + \epsilon = \epsilon 0^* = 0^*$

$q_1 = q_0 1 + q_1 1 = \cancel{(q_0 1 1^* = q_0 0 1^+)} = 0^* 1 + q_1 1$

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

$q_2 = q_1 0 + q_2 0 + q_2 1$

$\cancel{= q_1 0 + q_2(0+1)}$

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

$\cancel{= q_0 0 0 1^+ (0+1)^*}$

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

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

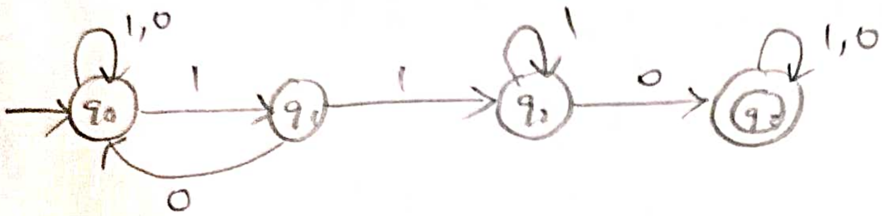
final state $= 0^* + 0^* 1^+ = 0^* (\epsilon + 1^+)$

$= 0^* 1^*$

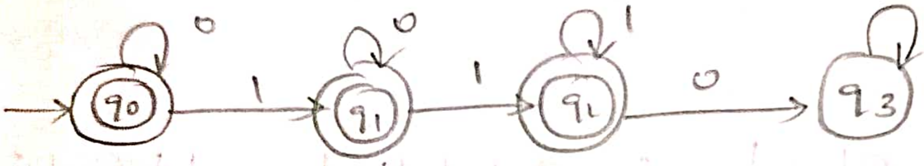
Ex:- The string doesn't have 110 as substring.

$L = \{000, 001, 101,$

$L' = \{110, 1110, 0110, 1101, 1100$



– contains 110.



$q_0 = q_0 0 + \epsilon = \epsilon 0^* = 0^*.$

$q_1 = q_0 1 + q_1 0 = 0^* 1 + q_1 0 = 0^* 1 0^*.$

$q_2 = q_1 1 + q_2 1 = 0^* 1 0^* 1 + q_2 1 = 0^* 1 0^* 1 1^*$

$q_3 = q_2 0 + q_3 0 + q_3 1 \qquad = 0^* 1 0^* 1^+.$

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

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

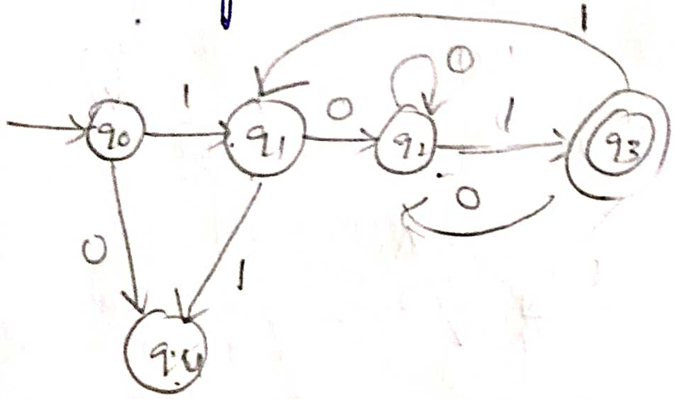
$= 0^* + 0^* 1 0^* 1^*$

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

$= 0^* + 0^* 1 0^* 1^*.$ ~~wrong~~

convert 1st diagr

Q:- string starts with 10 ends with 01



$q_0 = \epsilon$

$q_1 = q_0 1 \cancel{= \epsilon 1 = 1} = q_0 1 + q_3 1 = 1 + q_3 1 = 11^*$

$q_2 = q_1 0 + q_2 0 + q_3 0 \cancel{+ q_5 0} = 1^+ 0$

$q_4 = q_0 0 + q_1 1 = \epsilon 0 + 11 = 0 + 11$

$[\cancel{q_5 = q_3 1 + q_5 1}]$

Q/s

1, construct the expression for string ends with '1'.

2, never starts with 1 but ends with 1

3, atmost 3 1's in particular string.

① 

$q_0 = q_0 0 + q_1 0 + \epsilon$

$= \underbrace{q_1 0}_{Q} + \underbrace{q_0}_{r} \underbrace{0}_{P} + \epsilon$

$= QP^*$

$= q_1 0\, 0^*$

$= q_1 0^+$

$q_1 = q_0 1 + q_1 1$

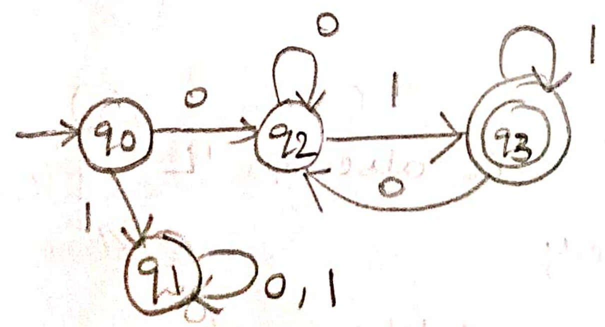
$q_1 = q_1 0 0^* 1 + q_1 1$

$= q_1 (00^* 1 + 1)$

$= q_1 (0^+ 1 + 1)$

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

$= q_1 0^* 1$

② 

$q_0 = \epsilon$

$q_1 = q_0 1 + q_1 0 + q_1 1$

$q_2 = q_0 0 + q_2 0 + q_3 0$

$q_3 = q_2 1 + q_3 1$

$q_2 = \epsilon 0 + q_2 0 + q_3 0$

$= (0 + q_3 0) + q_2 0$

$= (0 + q_3 0) 0^*$.

$q_3 = q_2 1 + q_3 1$

$= q_2 1 1^* = q_2 1^+$

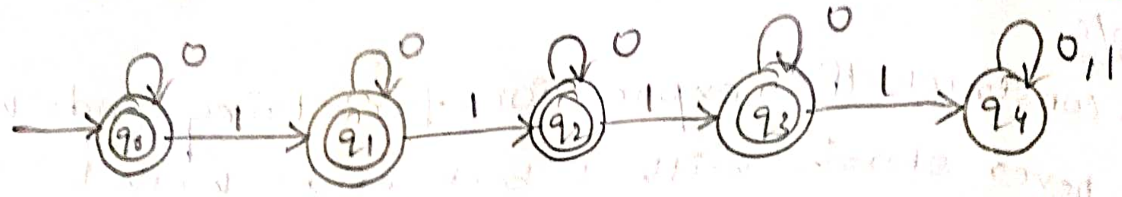
$= [(0 + q_3 0) 0^*] 1^+$

$q_3 = 0^+ 1^+ + q_3 0^+ 1^+$

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

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

(3)



$q_0 = q_0 0 + \epsilon = 0^*$

$q_1 = q_0 1 + q_1 0 = 0^* 1 + q_1 0 = 0^* 1 0^*$

$q_2 = q_1 1 + q_2 0 = 0^* 1 0^* 1 + q_2 0 = 0^* 1 0^* 1 0^*$

$q_3 = q_2 1 + q_3 0 = 0^* 1 0^* 1 0^* 1 + q_3 0 = 0^* 1 0^* 1 0^* 1 0^*$

$q_4 = q_3 1 + q_4 0 + q_4 1$

final states $= q_0 + q_1 + q_2 + q_3$

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

* Pumping Lemma

Let 'L' be a regular language there exits a constant 'n' for every string 'w' in 'L' such that $|w| \geq n$, now we can break 'w' into $w = xyz$ such that

i, $y \neq \epsilon$

ii, $|xy| \leq n$

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

Ex:- $L = \{a^n b^n / n \geq 0\}$

$L = \{ab; aabb, aaabbb, ---\}$

$w = aabb$, (4) → const

$\underline{a}\,\underline{ab}\,\underline{b}$ (x, y, z)

$y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aababb \notin L$ [not satisfied]

$\underline{aa}\,\underline{bb}$ (y, z)

$x = \epsilon$ $y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aaaabb \notin L$

Ex:- $L = \{ a^n b a^n / n \geq 1 \}$

$L = \{ b, aba, aabaa, \text{-----} \}$

$w = \underbrace{aa}_{x}\underbrace{b}_{y}\underbrace{aa}_{z}$, 6.

not regular expression.

$\underbrace{aa}_{x}\underbrace{b}_{y}\underbrace{aa}_{z}$ $\quad y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aabbaa \notin L$

$\underbrace{a}_{x}\underbrace{ab}_{y}\underbrace{aa}_{z}$ $\quad y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aababaa \notin L$

Ex:- $L = \{ a^n b^m / m > n \}$

$L = \{ b, abb, aabbb, \text{-----} \}$

$w = \underbrace{aa}_{y}\underbrace{bbb}_{z}$, 6. $\quad y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aaaabbb \notin L$

$\underbrace{aa}_{x}\underbrace{b}_{y}\underbrace{bb}_{z}$. $\quad y \neq \epsilon$

$|xy| \leq n$

$xy^1z = aabbbbb \in L$

[but the power of a is becoming const]

∴ not a regular language.

:- $L = \{ a^n b^m / n, m \geq 0 \}$

$L = \{ a, b, ab, aab, abb, aaab \}$

$w = aaab$, 5

$\underbrace{aa}_{x}\underbrace{ab}_{y}$ $\quad z = \epsilon, y \neq \epsilon$ ✓

$|xy| \leq n$ ✓

$xy^1z = aaabab \notin L$

$\underbrace{a}_{x}\underbrace{a}_{y}\underbrace{ab}_{z}$. $\quad y \neq \epsilon$ ✓

[partially regular]

$|xy| \leq n$. ✓

$xy^1z = aaaaab \in L$

is a regular language

Ex:- 1, $L = \{a^n / n > 0\}$

2, $L = \{a^{n^2} / n > 0\}$

① $L = \{a, aa, aaa, ----\}$

$w = \underset{x}{a}\,\underset{y}{aa}\,\underset{z}{a}$ $\qquad$ $w = \underset{y}{aaa}\,\underset{z}{a}$

$xy^1z = aaaaaa$ $\qquad$ $xy^1z = aaaaaaaa$

Regular language.

② $L = \{a, aaaa, aaaaaaaaa, ----\}$

$w = \underset{x}{aa}\,\underset{y}{aa}$ $\qquad$ $w = \underset{x}{a}\,\underset{y}{aa}\,\underset{z}{a}$

$xy^1z = aaaaaa$ $\qquad$ $xy^1z = aaaaaa$

× $\qquad$ ×

Not regular language.

Ex:- $L = \{a^n b^m / n > m$ where $n, m \in [1, 10]\}$ is a regular language.

* closure properties of Regular sets :- [explain with examples]

1, The union of two regular languages is regular

2, Intersection of two regular languages is regular

3, Complement of a regular language is regular

$[\Sigma^* - L = L']$

4, Difference of two regular languages is regular.

$[L - M \neq M - L]$ $\qquad$ $[L - M = L - (L \cap M)]$

5, Reversal or Transpose of a regular language is also regular.

[If abc regular cba also regular]

6, The closure of regular language is regular.

[closure means *]

[if r is regular, r^* will also be regular]

7, Concatenation of regular languages is regular.

Ex:- L & M regular ⇒ LM is regular.

[LM ≠ ML]

8, Homomorphism of a regular language is regular.

9, Inverse Homomorphism of regular language is also regular.

⇒ if we get repeated patterns we can give the repeated patterns a new name — This is called as Homomorphism.

⇒ This reduces the length.

Ex:- $\underbrace{aaa}_{A}\underbrace{bbb}_{B}\underbrace{ccc}_{C}c = \underbrace{ABCc}_{\text{Homomorphism}}$

Inverse homomorphism.