Date 3/1/18

Unit - II
Regular ~~Gram~~ Languages

* Regular languages are accepted by finite Automata.
* A Regular language can be represented by
  * Regular sets and
  * Regular expressions.

Regular Set:- A Regular set consists of set of strings, all These strings are valid In The language.

Eg:- R = { ∈, a, aa, aaa, aaaa, ------ }
* This set Represents The any number of a's
* R = { ∈, aa, aaaa, aaaaaa, ------- }
* This set Represents The evennumber of a's
* R = { ∈, ab, aabb, aaabbb, ------ }
* This set Represents no. of a's followed by same number of b's.
* R = { a, aab, aaabb, aaaabbb, -------- }
* This set Represents 'n' no. of a's followed by (n-1) no. of b's.
* R = { ∈, ab, ba, abab, baba, abba, baab, -- }
* This set Represents equal number of a's and b's.
* R = { aa, aba, aaa, abba, abaa, aaba, aaaa ------- }
* This set Represents begining with a and ending with a.
* R = { a, baa, aaa, aab, bab, aabb, aaba, aaab ------ }

* The set Represents The strings consists of substring a.
* $R=\{0,1\}$
* The set Represents The Binary set.
* $R=\{a,b\}$
* The set Represents That The set consists of a and b.
* $R=\{\epsilon, a, b, aa, bb, aaa, bbb, aaaa, bbbb, \text{---}\}$
* The set Represents That The set consists of any no. of a's or any no. of b's.
* $R=\{\epsilon, a, b, ab, abb, abbb, aab, aabb, aabbb, aaab, aaabb, aaabbb \text{----}\}$
*
  $R_1=\{\epsilon, a, aa, aaa, aaaa \text{-----}\}$

  $R_2=\{\epsilon, b, bb, bbb, bbbb \text{-------}\}$
* No. of a's and No. of b's is represented by This set.
* $R=\{0, 1, 00, 11, 10, 01, 100, 011, 110, 001, 101, 010 \text{--------}\}$
* The set Represents number and its one's complement.
* $R=\{2, 5, 8, 11, 14, 17, \text{------}\}$
* The set consists of Residue Modulo-3. Numbers.
* $R=\{0, 1, 1, 2, 3, 5, 8, 13, \text{------}\}$
* The set consists of fibonacii series.
* $R=\{1, 1, 2, 2, 3, 6, 4, 24, 5, 120, 6, 720 \text{----}\}$
* The set represents The Number & its factorial.
* $R=\{01, 11, 10, 11, 101, 100, 111, \cancel{101}, \cancel{100}, 001, 011, 010, \text{------}\}$

* This set represents begining with 0 (or) 1 and ending with 0 (or) 1.

Regular Expressions:-

* The languages accepted by finite Automata are easily described by simple expressions called Regular expressions.
* The definition of Regular Expression is Recursive.

4/1/18 * For a given character set (or) input set Σ of a language L The RE is defined by the following Rules:-

* Every character (or) alphabet belonging to Σ is a Regular Expression (RE).
* Null string 'ϵ' is a Regular Expression.
* If R_1 and R_2 are two Regular Expressions, Then $R_1 + R_2$ is also a RE.
* If R_1 and R_2 are two Regular Expressions Then $R_1 \cdot R_2$ is also a Regular Expression.
* If R is a Regular Expression Then (R) is a RE.
* If R is a Regular Expression Then R^* is also RE [also called kleens closure]
* Any combination of preceding Rules is also a Regular Expression.

Eg:- $a, b, \epsilon, a+b, a.b, (a+b), (a.b), (ab)^*, (a+b)^*, (a+b)^* a.(ab)^*$

Difference between Regular set And Regular Expressions:-

Regular Set	Regular Expression
* {a}	* a
* {a, b}	* a+b
* {ab, ba}	* ab+ba
* {abd, abe}	* ab.(d+e)
* {ε, a, aa, aaa, -----}	* a^* [Kleen's closure]
* {ε, aa, aaaa, aaaaaa, ----}	* $(aa)^*$
* {a, aa, aaa, aaaa, -----} → a.{ε, a, aa, aaa. } {a, aa, aaa, aaaa. }	* aa^* (or) a^+ (positive closure) $\Sigma^+ = \Sigma^* - \epsilon$
* {ε, ab, aabb, aaabbb, ----}	* no Regular Expression
* {ε, a, aaaa, aaaaaaaaa -----}	* no RE.
* {ε, a, aa, aaa, aaaaa, aaaaaaa ------}	* no RE.
* {a, ac, acc, accc, ----- b, bc, bcc, bccc, -----}	* $(a+b)c^*$.
* {ε, ab, abab, ababab, ---}	* $(ab)^*$.
* {ε, a, aa, ab, b, ba, bab, abab -------}	* $(a+b)^*$
* {11, 011, 0011, 0111, -----111, 1011, 1111, -----}	* $(0+1)^*11$
*	*

Regular Expressions And Regular Languages:-

* Design a RE for The language accepting all combination of a's Except 'ε'.

→ RE = a^+

* Design a RE for The language contain all The strings having any no's of a's and b's.

→ RE = $(a+b)^*$.

* Design a RE for language 'L' over {0,1} such That every string in 'L' ends with 11.

→ RE = $(0+1)^*11$

* Set of strings over {a,b,c} begining with c and ending with cc.

→ RE = $c(a+b+c)^*cc$.

* write a RE for a Language 'L' over {a,b,c} such That every string in 'L' contains a substring cc.

→ RE = $(a+b+c)^*cc(a+b+c)^*$.

* write a RE to denote a Language which accepts all The strings which begin (or) ends with either 00 (or) 11.

→ RE = $[(00+11)(0+1)^*] + [(0+1)^*(00+11)]$

→ $\left.\begin{array}{l} 11(0+1)^* \\ 00(0+1)^* \end{array}\right\} (00+11)(0+1)^*$

$\left.\begin{array}{l} (0+1)^*00 \\ (0+1)^*11 \end{array}\right\} (0+1)^*(00+11)$.

Identity Rules :-

* $\phi + R = R + \phi = R$. (Rule - 1)
* $\epsilon + R^* = R^* + \epsilon = R^*$ (Rule - 2)
* $\epsilon R = R\epsilon = R$. (Rule - 3)
* $R + R = R$. (Rule - 4)
* $\epsilon^* = \epsilon$ (Rule - 5)
* $R.R^* = R^*R = R^+$ (Rule - 6)
* $\epsilon + RR^* = \epsilon + R^*R = R^*$ (Rule - 7)
* $(P+Q)R = PR + QR$. (Rule - 8)
* $(PQ)^*P = P(QP)^*$ (Rule - 9)
* $(R^*)^* = R^*$ (Rule - 10)
* $(P+Q)^* = (P^* + Q^*)^* = (P^*Q^*)^*$ (Rule - 11)
* $\phi R = R\phi = \phi$. (Rule - 12).

V.v.v Imp ***** (14M)

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

Sol:- Given That,

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

$= (1+00^*1).(\epsilon + \underbrace{(0+10^*1)^*}_{R^*}\underbrace{(0+10^*1)}_{R})$ ($\epsilon + R^*R$)

$= (1+00^*1)(0+10^*1)^*$ $(\because \epsilon + R^*R = R^*)$

$= 1(\underbrace{\epsilon + 00^*}_{\epsilon + RR^*})(0+10^*1)^*$

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

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

$=$ RHS

$\therefore$ LHS = RHS.

Hence proved

* Prove $\epsilon + 1^*(011)^*(1^*(011^*))^* = (1+011)^*$ ①

Sol: Given That,

$LHS = \epsilon + 1^*(011)^*(1^*(011^*))^*$

$\epsilon + R \quad R^* = R^*$

$= (1^*(011^*))^*$

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

$= (1+011)^*$

$= RHS$

$LHS = RHS$.

Hence proved.

V.V.V Imp ***** (14M) **Arden's Theorem :-**

Let P and Q be The two regular Expressions over The input set Σ. The Regular Expression R is given as $\boxed{R = Q + RP}$ which has a unique solution as $\boxed{R = QP^*}$

Proof:- Let P and Q be two Regular Expression over Input string Σ.

If P does not contain ϵ Then There exist R such That $R = Q + RP \longrightarrow$ ①

~~consider~~

1) Replace $R = QP^*$ in RHS of Eq ①

$R = Q + RP$.

$R = Q + (QP^*)P$

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

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

$R = QP^*$.

2) replace $R = Q + RP$ in RHS of Eq (1)

$$R = Q + (Q + RP)P$$

$$R = Q + QP + RPP$$

Again Replace $R = Q + RP$.

$$R = Q + QP + (Q + RP)PP$$

$$R = Q + QP + QPP + RPPP$$

$$R = Q + QP + QPP + (Q + RP)PPP$$

$$R = Q + QP + QPP + QPPP + RPPPP \text{------}$$

$$R = Q(\epsilon + P + PP + PPP + \text{-----}) + RP^{n+1}$$

$$R = Q(\epsilon + P + P^2 + P^3 + \text{------} + P^n) + RP^{n+1}$$

Let 'w' be the string of length 'n' in RP^{n+1} has no string of less than $n+1$ length. So, 'w' is not in set RP^{n+1}. So, neglect RP^{n+1} term.

$$R = Q(\epsilon + P + P^2 + P^3 + \text{-------} + P^n)$$

$$R = QP^*$$

> Hence the Equation $R = Q + RP$ has a unique solution $R = QP^*$.

11. Equivalence between Finite Automata And Regular Expression:-

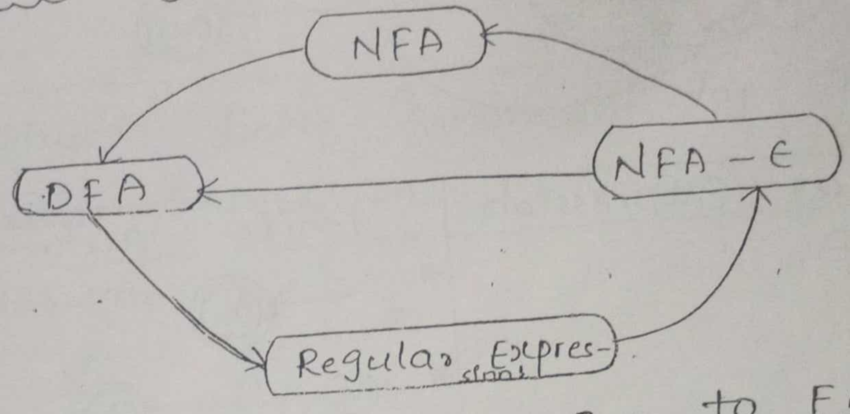
* Construction of Finite Automata from the given Regular Expression:-

Let us starts with Kleen's closure Theorem which can be defined in two parts as follows:-

* The first part of The Theorem says That if There is a Regular Language L, Then There must exist a corresponding FA, M which accepts all strings belonging to it.

* The second part of The Theorem states That if There is a finite Automata, it should be possible to create a Regular Expression associated with The Regular Language from The FA.

Relation between FA and RE:-

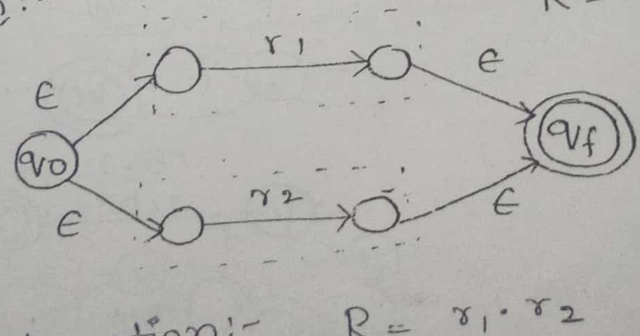


Regular Expression to FA:-

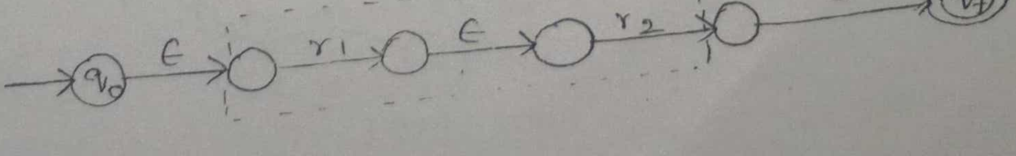
* ~~const~~ Regular Expression on 1

There are 3 types of operations.

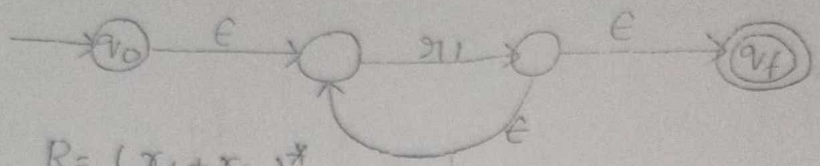
* **union:-** $R = r_1 + r_2$.



* **concatenation:-** $R = r_1 \cdot r_2$

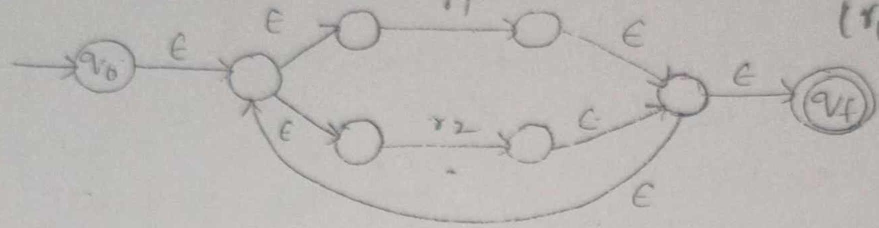


Kleen's closure:- $R = r^*$



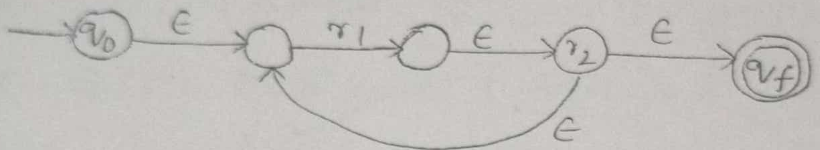
* $R = (r_1 + r_2)^*$

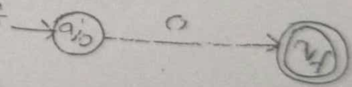
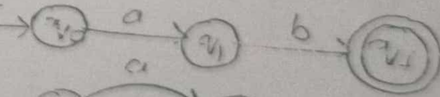
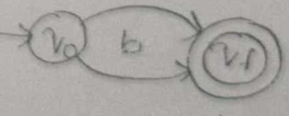
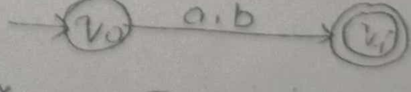
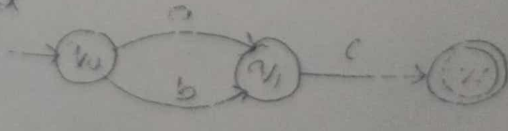
$(r_1 + r_2)^*$



* $R = (r_1 r_2)^*$

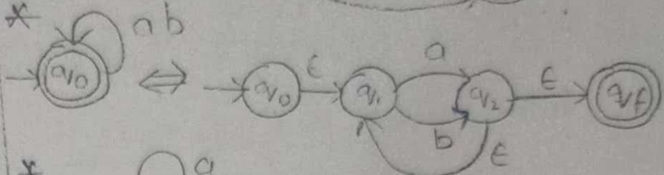
$(r_1 \cdot r_2)^*$



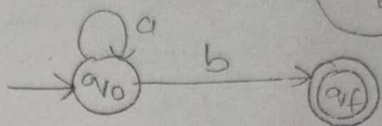
Regular Expression	Finite Automata
* $R = \epsilon$	* 
* $R = a$	* 
* $R = \phi$	* 
* $R = ab$	* 
* $R = a+b$ ✓	 (or) 
* $R = a^*$	* 
* $R = (a+b) \cdot c$	* 

* $R = a \cdot (b+c)$

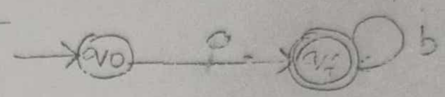
* $R = (a+b)^*$



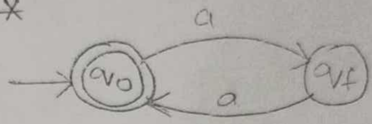
* $R = a^* b$



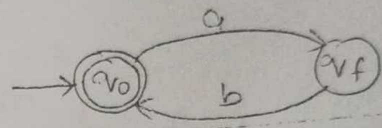
* $R = ab^*$



* $R = (aa)^*$



* $R = (ab)^*$



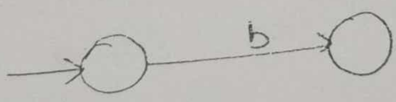
11/1/18 Construct finite Automata for The followin Regular Expression. $R = b + ba^*$.

Sol:- Assume $r_1 = b$

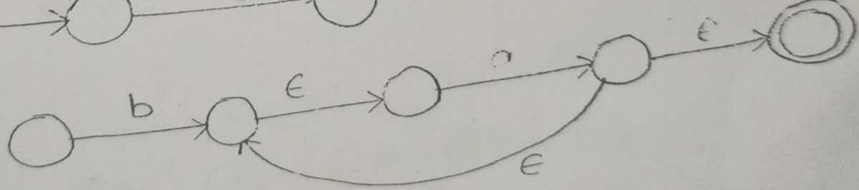
$r_2 = ba^*$

$R = r_1 + r_2$.

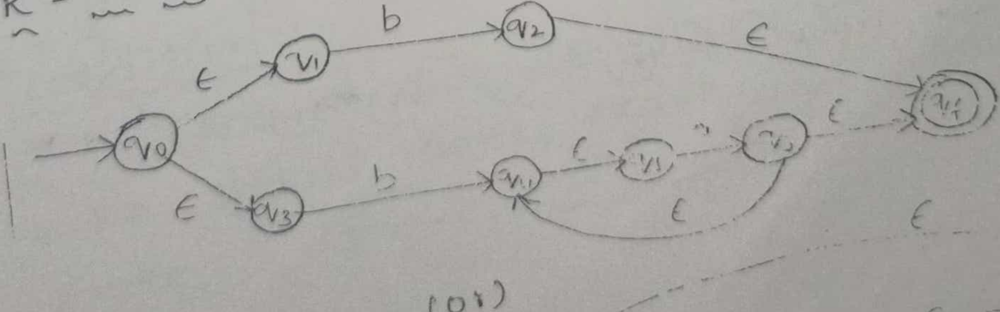
r_1 :-



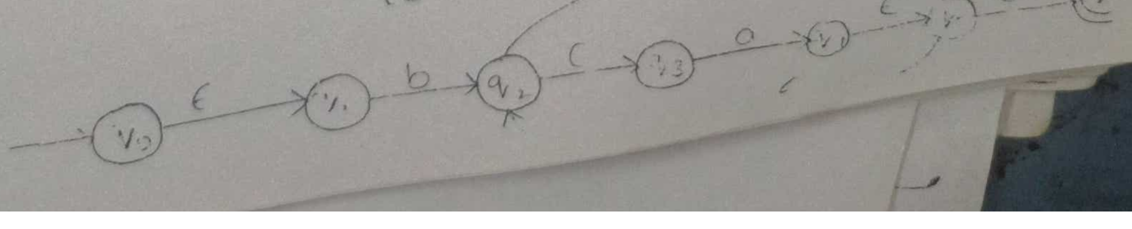
r_2 :-



$R = r_1 + r_2$.



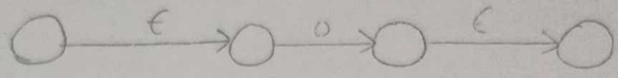
(or)



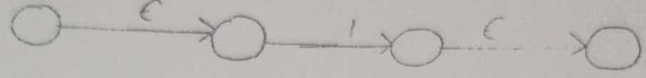
* construct finite Automata for the Regular Expression $(0+1)^*$

ol:-

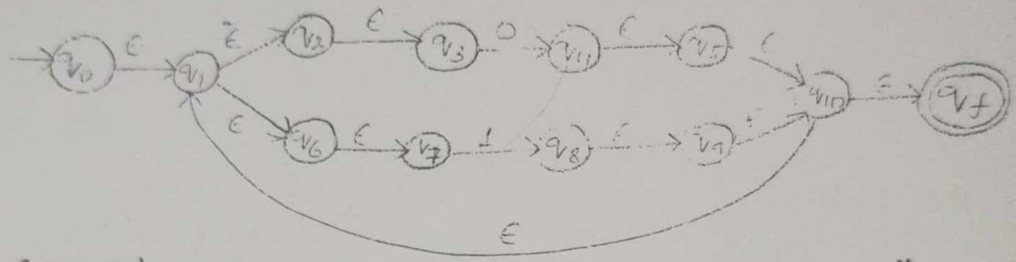
$r_1 = 0$:-



$r_2 = 1$:-

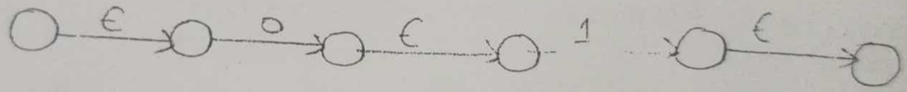


$R = (0+1)^*$:-

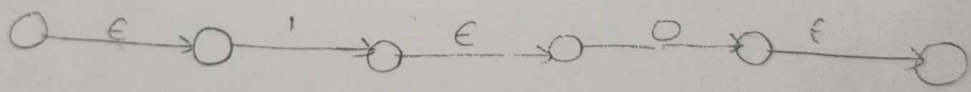


* construct F.A for The R.E $(01+10)^*$

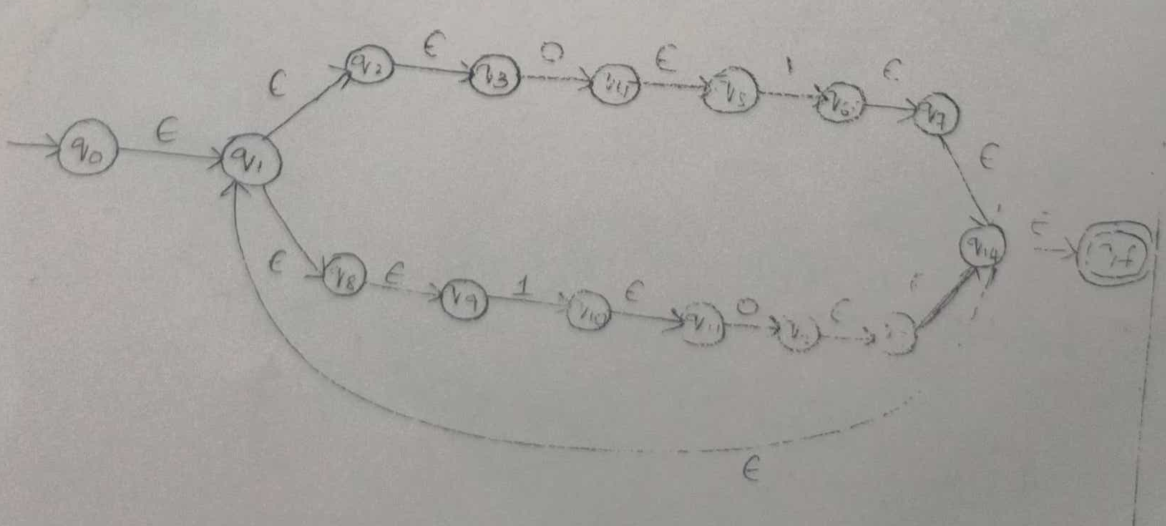
$r_1 = 01$:-



$r_2 = 10$:-



$R = (01+10)^*$



Construction of Regular Expression from Finite Automata:- ①

* A String w in The F.A M is accepted if and only iff There is a path from The initial state to The final state for 'w'.
The total path from initial state to Final state in a FA- contain The following type of component paths.
* serial path
* Parallel path
* loop (or) cycle.
* **Serial Path**:- It can be obtained by concatinating The R.E's of The individual component paths connected in series.
* **Parallel Path**:- It can be represented with The '+' symbol. between The Individual R.E's of The component parallel path's.
* **loop (or) cycle**:- The Regular Expression for a loop is a* where a is The input symbol forming The loop.

conditions:-
* FA should not contain 'ε' moves.
* FA should have exactly one initial state.

Procedure:-

1) Q_i is the R.E representing the set of strings accepted by the system even though Q_i is the final state.

2) α_{ij} denotes the RE representing the set of labels of edges from Q_i to Q_j. when there is no such edge, $\alpha_{ij} = \phi$. consequently we can get the following set of equations in Q_1, Q_2 and ------ Q_n.

$$q_1 = q_1\alpha_{11} + q_2\alpha_{21} + q_3\alpha_{31} + \cdots + q_n\alpha_{n1} \rightarrow ①$$

$$q_2 = q_1\alpha_{12} + q_2\alpha_{22} + q_3\alpha_{32} + \cdots + q_n\alpha_{n2} \rightarrow ②$$

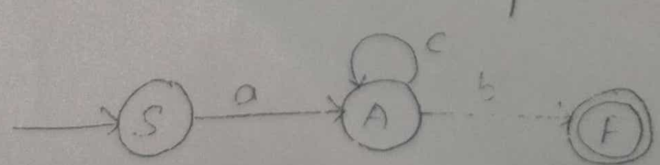
$$\vdots$$

$$q_n = q_1\alpha_{1n} + q_2\alpha_{2n} + q_3\alpha_{3n} + \cdots + q_n\alpha_{nn} \rightarrow ③$$

* we solve this equations, q_i in terms of α_{ij}'s and it will be required. R.E. one thing should be noted that we add ϵ in the equation starts with the starting (or) initial state and solve the equation to findout q_i (final state) in terms of α_{ij}'s, it is the RE for given DFA.

Note:- If more than one final state in DFA leads to union of 2 R.E's.

* Find the RE corresponding to F.A.



sol:-

Sol:-

$S = \epsilon \longrightarrow ①$

$A = Sa + Ac \longrightarrow ②$

$F = Ab \longrightarrow ③$

Substitute ① in ②

$A = Sa + Ac \quad (\because S \rightarrow \epsilon)$

$A = \epsilon \cdot a + Ac$

$A = a + Ac$ is in The form of Arden's Theorem $\boxed{R = Q + RP}$ and its solution is $\boxed{R = QP^*}$

$A = ac^* \longrightarrow ④.$

substitute ④ in ③, we get

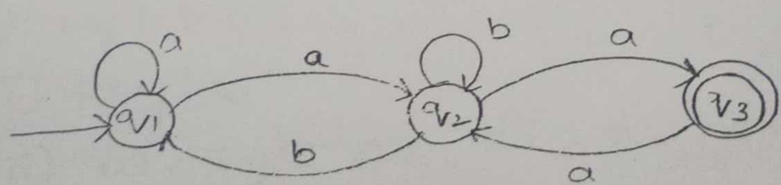
$F = Ab$

$\boxed{F = ac^*b.}$

∴ The Regular Expression for given FA

$\boxed{R = ac^*b.}$

* consider The transition diagram given below and prove that The strings recognised That $(a+a(b+aa)^*b)^* a(b+aa)^*a.$



Sol:-

$q_1 = q_1a + q_2b + \epsilon \longrightarrow ①$

$q_2 = q_1a + q_2b + q_3a \longrightarrow ②$

$q_3 = q_2a \longrightarrow ③$

Substitute ③ in ②

$q_2 = q_1a + q_2b + q_2aa$

$q_2 = q_1a + q_2(b + aa)$

This is in the form of $R = Q + RP$.

$$q_2 = q_1 a(b+aa)^* \rightarrow (4)$$

Substitute (4) in (1)

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

$$\underbrace{q_1}_{R} = \underbrace{q_1}_{R}\underbrace{(a + a(b+aa)^* b)}_{P} + \underbrace{\epsilon}_{Q}$$

This is again in the form of $R = Q + RP$

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

$$q_1 = (a + (a(b+aa)^* b)^* \rightarrow (5)$$

Substitute (5) in (4)

$$q_2 = (a + (a(b+aa)^* b)^* a(b+aa)^* \rightarrow (6)$$

Substitute (6) in (3) we get

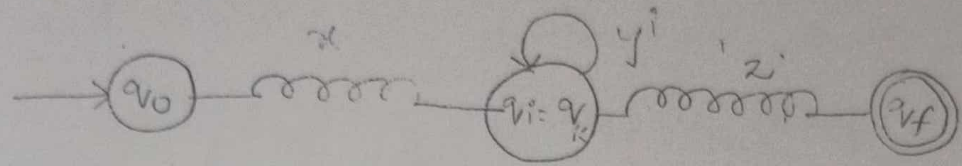
$$q_3 = q_2 a.$$

$$q_3 = (a + (a(b+aa)^* b)^* a(b+aa)^* a$$

$\therefore$ Hence it is proved.

imp 14M *****

Pumping Lemma For Regular Languages



* Pumping Lemma gives a necessary condition for an input string to belong to a Regular set and it is a powerful tool for providing certain languages are not regular. It is also useful to answer certain Questions such as whether the language accepted by a given FA is finite (or) infinite.

→⑥ - **Statement:-** Let L be a Regular Language There exists a constant n such That for every string w in L, such That $|w| \geq n$, we can break w into three substrings. $w = xyz$ follows The following condition.

i) $|y| > 0$.

ii) $|xy| \leq n$

iii) $w = xy^iz \in L, \forall i \geq 0$.

Theorem:-

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a F.A with n states. Let L be The Regular set accepted by M. Let $w \in L$ and $|w| \geq n$ if $m > n$, Then There exist x, y, z such That $w = xyz$, $y \neq \epsilon$ and $xy^iz \in L$ for each $i \geq 0$.

Proof:-

Let $w = a_1, a_2, \ldots\ldots a_m$, $m > n$.

$\delta(q_0, a_1, a_2, \ldots\ldots a_m) = q_i$ for $i = 1, 2, \ldots m$

In The path $w = a_1, a_2, \ldots\ldots a_m$.

As There are only n distinct states, Atleast 2 states must coincide among These, we take The first pair. Let us take Them as q_j and q_K $(q_j = q_K)$, j and K satisfies The condition $0 \leq j < K \leq n$.

The string w can be decomposed into 3 substrings.

$$a_1, a_2 \ldots\ldots a_j$$

$$a_{j+1}, a_{j+2} \ldots\ldots a_K$$

$$a_{K+1}, a_{K+2} \ldots\ldots a_m$$

Let xyz denotes 3 strings

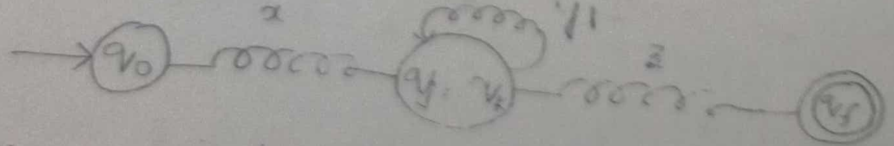
$$x = a_1, a_2 \ldots\ldots a_j$$

$$y = a_{j+1}, a_{j+2} \ldots\ldots a_K$$

$$z = a_{K+1}, a_{K+2} \ldots\ldots a_m$$

as $K \leq m$ and $|xy| \leq n$ and $w = xyz$

The path ~~Nlorkue~~ with The path value w in The transition diagram of 'm' is



* on Applying string x it reaches to q_j (or) q_K. The Automata is in The same state q_j

After Application of string y^i for each $i \geq 0$.

* On applying z it reaches to q_f The final state.

Note:- Decomposition is valid only for string of The length greater Than (or) Equals to number of states.

18/1/18 Application of Pumping Lemma:-

* Assume L is Regular.
* Let 'n' be The no. of states corresponding to FA.
* choose a string w, such That $|w| \geq n$.

use pumping Lemma to write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$.

* Find a suitable integer i, such That $xy^iz \notin L$, which contridicts our assumption

Hence L is not Regular.

Note:- * The crucial (impl (or) major (or) significant) point of The procedure is to find The value of i such That xy^iz does not ($\notin$) L.

* In some cases we prove $xy^iz \notin L$ by considering $|xy^iz|$ (Length). in some cases we may have to use The structure of The string in 'L'.

Pigeonhole principle:-

* Prove $L = \{a^{i^2} / i > 0\}$ is not Regular

Sol:- using Pumping Lemma, we can prove The given Language is Regular (or) not.

Given data:-

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

$L = \{a, aaaa, aaaaaaaaa, aaaaaaaaa$
$aaaaaaa \ldots\ldots\ldots \}$

Apply Pumping lemma:-

* choose 'n'.

$w = aaaa$, $n = 3$.

$|w| \geq n$, $|w| = 4 \geq 3$ (True).

* $w = xyz$, $|xy| < n$, $|zy| > 0$.

$w = \underset{x}{a}\,\underset{y}{a}\,\underset{z}{aa}$ $\quad 2 < 3$, $|y| > 0$

* $i = 3$;

$w = xy^iz$

$\Rightarrow xyyyz$

$\Rightarrow \underset{x}{a}\,\underset{y}{a}\,\underset{y}{a}\,\underset{y}{a}\,\underset{z}{aa} \notin L$.

Hence L is not Regular.

* ~~Le~~ prove $L = \{a^p \mid p \text{ is prime}\}$ is not Regular?

Sol:- using Pumping Lemma, we can prove The given Language is Regular (or) not.

Given data:- $L = \{a^p \mid p \text{ is prime}\}$

$L = \{a, aa, aaa, aaaaa, aaaaaaa, aaaaaaaaaaa,$
$\ldots\ldots\ldots \}$.

Applying pumping Lemma:-

* choose n

$w = aaaaa \quad n = 4$

$|w| \geq n \quad , \quad |w| = 5 \geq 4.$

* $w = xyz \quad , \quad |xy| < n, \quad |y| > 0.$

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

* $i = 3 \ ; \ w = xyyyz$

$w = \underset{x}{a}\ \underset{y}{aa}\ \underset{y}{aa}\ \underset{y}{aa}\ \underset{z}{aa} \notin L.$

Hence, it is not Regular.

* prove $L = \{a^n b^n \mid n > 0\}$ is not Regular?

sol:- using Pumping Lemma, we can prove The given Language is Regular (or) not.

Given That :- $L = \{a^n b^n \mid n > 0\}$.

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

Applying pumping Lemma. (structure oriented problem).

* choose 'n'.

$w = aabb \ , \ n = 3$

$|w| \geq n \Rightarrow |w| = 4 > 3$ (True).

* $w = aabb. \quad |xy| \leq n \ , \ |y| > 0.$

case1:- $a^n b^n \Rightarrow \underset{x}{a^{n-l}}\ \underset{y}{a^l}\ \underset{z}{b^n}$

case2:- $a^n b^n = \underset{x}{a^n}\ \underset{y}{b^{n-l}}\ \underset{z}{b^l}$

case 3:- $a^n b^n = \underset{x}{a^{n-k}}\ \underset{y}{a^k\, b^{n-l}}\ , \underset{z}{b^l}$

* $W = \underset{x}{a}\ \underset{y}{ab}\ \underset{z}{b}$

choose $i = 2.$

$w = xyyz = aababb \notin L.$

UNIT - III

1/1/18

Regular Grammar

Regular Grammar:-

Regular Grammar $G = (V, \Sigma, P, S)$

where V = a finite-set of variables (or) Non-Terminal

Σ = a finite set of input symbols.

S = start symbol.

P = Set of productions

P is in the form of either terminal followed by non-terminal (or) P is in the form of either non-terminal followed by terminal.

* $P \rightarrow T\,NT$ (or) [Right linear]

$P \rightarrow NT\,T$ (or) [left linear]

Note:-

* ϵ is allowed.
* $S \rightarrow \epsilon$ is not allowed, if allowed S may not appear on RHS of any other production.

Equivalence between Regular Grammar and Finite Automata:-

Regular Grammar to FA:-

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and R be a Regular Grammar accepted by DFA.