Unit - II :- Regular Expression

* Regular Expressions
* Regular Sets
* Identity Rules
* Equivalence of two Regular Expressions
* Manipulation of Regular Expression
* Finite Automata and Regular Expression
* Conversion of Equivalence between Finite Automata and Regular Expression
* Pumping Lemma
* Closers propertyies
* Applications of Regular expression
* Finite Automata and Regular Grammar
* Regular expression and Regular Grammar.

2. Regular Expressions

Regular Expressions :-

A Regular Expression can be defined as a language or string accepted by an FA. we know that an FA consists of 5 tuples $\{Q, \Sigma, \delta, q_0, F\}$. Among them an regular expression is a string on Σ i.e it consists of input alphabets.

In short, a regular expression is written as RE or regex or regexp

Some formal Recursive definitions of RE :-

1) Any terminals i.e the symbols that belong to Σ are RE. the null string ($\wedge$) and the null set (ϕ) are also RE
2) If P and Q are two RE's then the union of the two RE's denoted by $P+Q$ (i.e $P \cup Q$) is also an RE
3) If P and Q are two RE's, then their Concatenation denoted by P.Q is also an RE
4) If P is an RE, then the iteration (repetition or closure) denoted by P* is also an RE $P^* =$ 
5) If P is an RE, then (P) is an RE (i.e $|P|$) 
6) The expressions got by the repeated applications of the rules from (1) to (5) over Σ are also RE's.

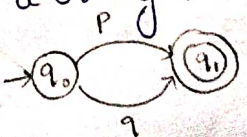
Basic Operations on Regular Expression | Regular Sets :-

There are 3 basic operations. They are:

1) Union: If R_1 and R_2 are two RE's over Σ then $L(R_1 \cup R_2)$ is denoted by $L(R_1 + R_2)$.

$L(R_1 \cup R_2)$ is a string from R_1 or a string from R_2.

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

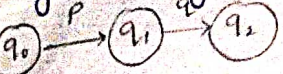
$P+q \Rightarrow$ 

Parallel transition

2) Concatenation: If R_1 and R_2 are two RE's over Σ then $L(R_1 \cap R_2)$ is denoted by $L(R_1 R_2)$.

$L(R_1 \cap R_2)$ is a string from R_1 followed by a string from R_2.

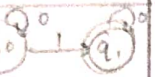
$L(R_1 R_2) = L(R_1) \cdot L(R_2)$

$P.q =$ 

3) Operation of closure of RE :-

If R_1 and R_2 are two REs over Σ then $L(R^*)$ is a string obtained by concatenating n elements for $n \geq 0$

Some RE examples:

RE's	Regular Set
1) $(0+10^*)$	$L = \{0, 1, 10, 100, 1000, 10000, \ldots\}$
2) (0^*10^*) 	$L = \{1, 01, 10, 010, 0010, 0100, \ldots\}$
3) $(0+\epsilon)(1+\epsilon)$	$L = \{\epsilon, 0, 1, 01\}$
4) $(a+b)^*$	set of strings of a's and b's of any length including the null string $L = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$
5) $(a+b)^*abb$	set of strings of a's and b's ending with the string abb $L = \{abb, aabb, babb, aaabb, ababb, \ldots\}$
6) $(11)^*$	set of strings consisting of even number of 1's including empty string. $L = \{\epsilon, 11, 1111, 111111, \ldots\}$
7) $(aa)^*(bb)^*b$	set of strings consisting of even number of a's followed by odd number of b's. $L = \{b, aab, bbb, aabbb, aabbbbb, aaaabbbbb, \ldots\}$
8) $(aa+ab+ba+bb)^*$	Strings of a's and b's of even length can be obtained by concatenating any combination of the strings aa, ab, ba and bb including null, so $L = \{\epsilon, aa, ab, ba, bb, aaab, aaba, \ldots\}$

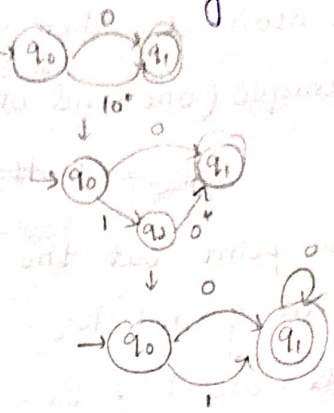
Identities related to RE's : | Identity Rules :-

Given R, P, L, Q as RE, the following identities hold

- $\phi^* = \epsilon$
- $\epsilon^* = \epsilon$
- $RR^* = R^*R$
- $R^*R^* = R^*$
- $(R^*)^* = R^*$
- $(PQ)^*P = P(QP)^*$
- $(a+b)^* = (a^*b^*)^*$

  $= (a^*+b^*)^* = (a+b^*)^* = a^*(ba^*)^*$
- $R + \phi = \phi + R = R$ (Identity for union)
- $R\epsilon = \epsilon R = R$ (Identity for Concatenation)
- $\phi L = L\phi = \phi$ (The annihilator for concatenation)
- $R+R = R$ (Idemponent law)
- $L(M+N) = LM + LN$ (Left Distribution law)
- $(M+N)L = ML + NL$ (Right Distributive law)
- $\epsilon + RR^* = \epsilon + R^*R = R^*$

$0+10^*$ =>



Applications of RE :-

* RE are useful in a wide variety of text processing tasks and more generally string processing where the data need not to be textual. Common applications include data ~~Variation~~ validation, data scrapping (especially web scrapping), data wrangling, simple parsing, the production of syntax highlighting systems and many more tasks.
* While regexps would be useful on internet search engines, processing them across the entire database could consume execessive computer resources depending on the complexity and design of Regex.

The Arden's Theorem :-

The Arden's theorem is used to construct the RE from a transitional system of FA

Statement :-

Let P and Q be two RE's over Σ, if P docsnot contain ϵ, then the equation $R = Q + RP$ has a unique (one and only one) solution i.e $R = QP^*$

Proof :-

Now point out the statements in the Arden's theorem in general form.

- P and Q are two RES
- P does not contain the A symbol.
- $R = Q + RP$ has a solution i.e

$$R = QP^*$$

This solution is the one & only solution of the equation.

If $R = QP^*$ is a solution of the equation $R = Q + RP$ then by putting the value of R in equation. we shall get the value '0'

$$R = Q + RP$$

$$R - Q - RP = 0$$

LHS $\Rightarrow R - Q - RP$

putting the values of R in LHS, we get

$$= QP^* - Q - QP^*P$$

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

$$= QP^* - QP^* \quad (\because R^* = \epsilon + R^*R)$$

$$= 0$$

So from here it is proved that $R = QP^*$ is a solution of equation $R = Q + RP$

Now, we have to prove that $R = QP^*$ is the one and only solution of the equation $R = Q + RP$.

As $R = Q + RP$, put the value of R again and again in the RHS of the equation

$$R = Q + RP$$
$$= Q + (Q + RP)P$$
$$= Q + QP + RPP$$
$$= Q + QP + (Q + RP)PP$$
$$= Q + QP + QPP + RPPP$$
$$= Q(\epsilon + P) + QPP + RPPP$$
$$= Q(\epsilon + P + PP) + RPPP$$

After several steps, we shall get

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

Now let a string w belong to R. If w is a ϵ string then in which part will be ϵ belong ?

This string will belong to either the $Q(\epsilon + P + P^2 + \cdots + P^n)$ part or RP^{n+1} part. But, according to the point number (ii) P does not contain ϵ and so the string doesn't belong to RP^{n+1} so it will obviously belong to $Q(\epsilon + P + P^2 + P^3 + \cdots + P^n)$ which is nothing but QP^*. $(\epsilon + P + P^2 + \cdots + P^n)$ is any combination of P including ϵ.

As this string belongs to only in one part, R and QP^* represent the same set. that means $R = QP^*$ is the unique solution of equation

$$R = Q + RP.$$

Procedure:

1) For getting the RE for automata we first create equations of given form for all the states

$$q_1 = q_1 w_{11} + q_2 w_{21} + q_3 w_{31} + \cdots + q_n w_{n1} + \epsilon \quad (q_1 \text{ is initial state})$$
$$q_2 = q_1 w_{12} + q_2 w_{22} + \cdots + q_n w_{n2}$$
$$\vdots$$
$$q_n = q_1 w_{1n} + q_2 w_{2n} + \cdots + q_n w_{nn}$$

w_{ij} is the RE representing the set of labels of edges from R_i to R_j

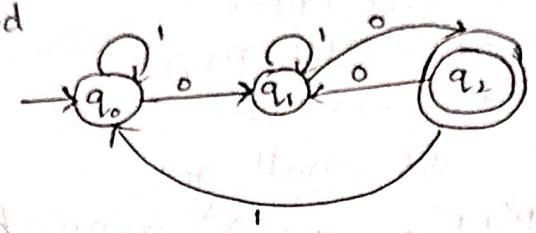
Note:- For parallel edges there will be that many expressions for that state in the expression.

2) Then we solve these equations to get the equation for q_i in terms of w_{ij} and that expression is the

required solution, where q_1 is a final state

Problems:

1) Using Arden's theorem construct RE from FA by the algebraic method



Sol:- The equations from given FA are:

$$q_0 = q_0 \cdot 1 + q_2 \cdot 1 + \epsilon \longrightarrow (1)$$

$$q_1 = q_0 \cdot 0 + q_1 \cdot 1 + q_2 \cdot 0 \longrightarrow (2)$$

$$q_2 = q_1 \cdot 0 \longrightarrow (3)$$

put the value of q_2 in (2), we get

$$q_1 = q_0 \cdot 0 + q_1 \cdot 1 + q_1 \cdot 0 \cdot 0$$

$$q_1 = q_0 \cdot 0 + q_1 (1 + 00) \longrightarrow (4)$$

This (4) is in the form of $R = Q + RP$

where $R = q_1$, $Q = q_0 \cdot 0$, $P = (1+00)$

So, the solution of equation is $R = QP^*$

$$q_1 = q_0 \cdot 0 (1+00)^* \longrightarrow (5)$$

put the value of q_2 in (1), we get

$$q_0 = q_0 \cdot 1 + q_1 \cdot 0 \cdot 1 + \epsilon$$

Substitute q_1 value in above equation

$$q_0 = q_0 \cdot 1 + q_0 \cdot 0 (1+00)^* \cdot 0 \cdot 1 + \epsilon$$

$$q_0 = q_0 (1 + 0(1+00)^* \cdot 0 \cdot 1) + \epsilon \longrightarrow (6)$$

(6) is in form of $R = Q + RP$.

where $R = q_0$, $Q = \epsilon$, $P = 1 + 0(1+00)^* \cdot 0 \cdot 1$

So solution of above equation is $R = QP^*$

$$q_0 = \epsilon (1 + 0(1+00)^* \cdot 0 \cdot 1)^*$$

$$= (1 + 0(1+00)^* \cdot 0 \cdot 1)^* \longrightarrow (7)$$

Substitute (7) in (5)

$$q_1 = (1 + 0(1+00)^* \cdot 0 \cdot 1)^* \cdot 0 (1+00)^* \longrightarrow (8)$$

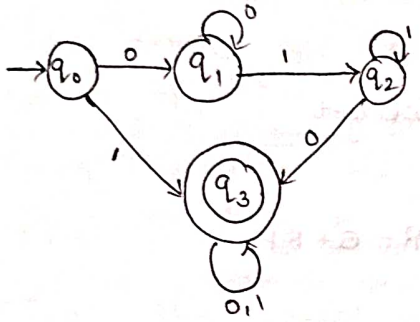
Substitute (8) in eq (3)

$$q_2 = (1+0(1+00)^*01)^*0\,(1+00)^*\cdot 0$$

As q_2 is final state of FA, all the strings will be halt on this state. Therefore the RE is:

$$(1+0\cdot(1+00)^*01)^*0\,(1+0.0)^*\cdot 0.$$

Thus accepting the FA.



$q_0 = \epsilon \longrightarrow (1)$

$q_1 = q_1\cdot 0 + q_0\cdot 0 \longrightarrow (2)$

$q_2 = q_1\cdot 1 + q_2\cdot 1 \longrightarrow (3)$

$q_3 = q_0\cdot 1 + q_2\cdot 0 + q_3\cdot 0 + q_3\cdot 1 \longrightarrow (4)$

Substitute (1) in (2) we get

$q_1 = \epsilon\cdot 0 + q_1\cdot 0$

$q_1 = 0 + q_1\cdot 0 \Rightarrow R = Q + RP.$

Solution is $q_1 = 0\cdot 0^* \Rightarrow R = QP^*$ $\longrightarrow (5)$

Substitute (5) in (3)

$q_2 = 0\cdot 0^*\cdot 1 + q_2\cdot 1 \Rightarrow R = Q + RP.$

Solution is $q_2 = (0\cdot 0^*\cdot 1)\,1^* \longrightarrow (6)$

Substitute (6), (1) in (4)

$q_3 = \epsilon\cdot 1 + (0\cdot 0^*\cdot 1)\cdot 1^*\cdot 0 + q_3\cdot 0 + q_3\cdot 1$

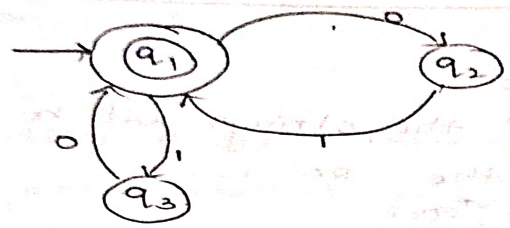
$q_3 = \epsilon\cdot 1 + (0\cdot 0^*\cdot 1)\cdot 1^*\cdot 0 + q_3(0+1)$

$q_3 = (1 + 0\cdot 0^*\cdot 1\cdot 1^*\cdot 0) + q_3(0+1) \Rightarrow R = Q + RP.$

Solution is $q_3 = (1+0\cdot 0^*\cdot 1\cdot 1^*\cdot 0)\,(0+1)^*$

This is the required RE

3)



sol:-

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

$q_2 = q_1 \cdot 0 \longrightarrow (2)$

$q_3 = q_1 \cdot 1 \longrightarrow (3)$

Substitute eq (2), (3) in (1) we get

$q_1 = \epsilon + q_1 \cdot 0 \cdot 1 + q_1 \cdot 1 \cdot 0$

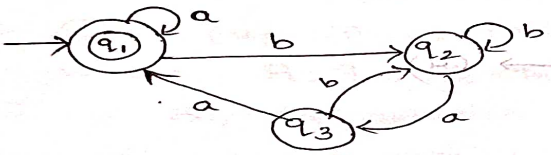
$= \epsilon + q_1 (01+10) \Rightarrow R = Q + RP.$

Solution is $q_1 = \epsilon (01+10)^*$

$q_1 = (01+10)^*$

This is the required RE

4)



sol:-

$q_1 = \epsilon + q_1 \cdot a + q_3 \cdot a \longrightarrow (1)$

$q_2 = q_1 \cdot b + q_2 \cdot b + q_3 \cdot b \longrightarrow (2)$

$q_3 = q_2 \cdot a \longrightarrow (3)$

Substitute (3) in (2) we get

$q_2 = q_1 \cdot b + q_2 \cdot b + q_2 \cdot a \cdot b$

$q_2 = q_1 \cdot b + q_2 (b + ab) \Rightarrow R = Q + RP.$

Solution is $q_2 = q_1 b (b+ab)^*$

$q_3 = q_1 b (b+ab)^* a \longrightarrow (4)$

Substitute (4) in (1) we get

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

$q_1 = \epsilon + q_1 (a + b(b+ab)^* aa) \Rightarrow R = Q + RP.$

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

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

This is required RE.

Constructing FA from RE:-

RE is the language format of the FA. DFA can be constructed from a given RE in 2 ways.

1) Conversion of an RE to NFA with ϵ-transition
2) and then converting it to DFA
3) Direct conversion of RE to DFA

1) Conversion of an RE to NFA with ϵ-transition:-

For making an FA from RE, here are 2 steps as given in the following:

step1:- From the given RE, an equivalent transitional system (FA) with ϵ move has to be constructed

step2:- From the transitional system with ϵ moves, a DFA needs to be constructed by the ϵ-closure method.

For constructing step1 again there are 2 methods:

a) Top down approach
b) Bottom up approach

a) Top down approach, the FA construction starts with from the given RE and ends by reaching a single element for each transition

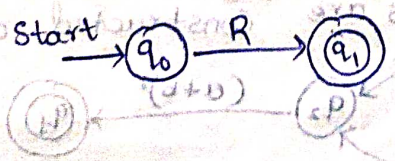
b) In the bottom up approach, the construction starts from the basic element and ends on reacting the RE

a) Top-Down approach:-

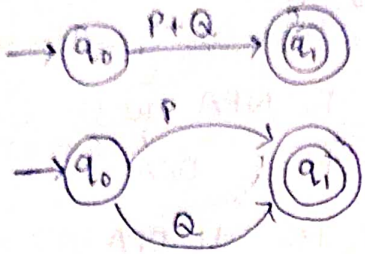
This is divided into several steps as given into the following

→ Take two states, one is the beginning state and another is the final state

→ Make a transition from the beginning state to the final state and place the RE in between beginning and final state.

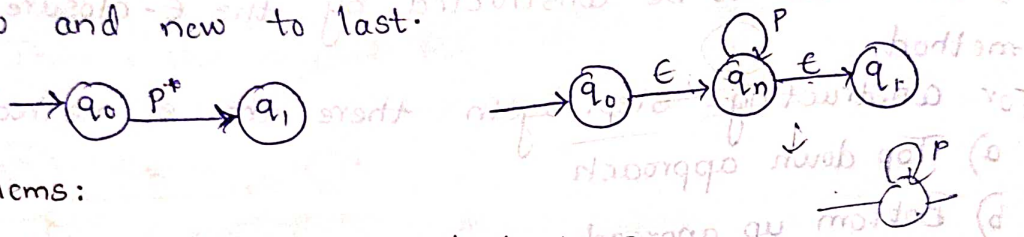


→ If in the RE there is a '+' (union) sign, then there are parallel paths b/w two states



→ If in the RE there is a •(dot) sign, then one extra state is added b/w two states.

→ If in the RE there is a '*' sign, then a new state is added in between. A loop is added on the new states and the label ∈ is put b/w the first to new and new to last.



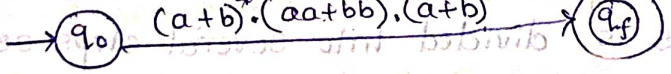
Problems:

1) Construct an FA equivalent to RE

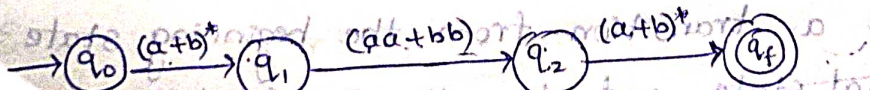
$$L = (a+b)^*(aa+bb)(a+b)^*$$

Sol: two states are taken:

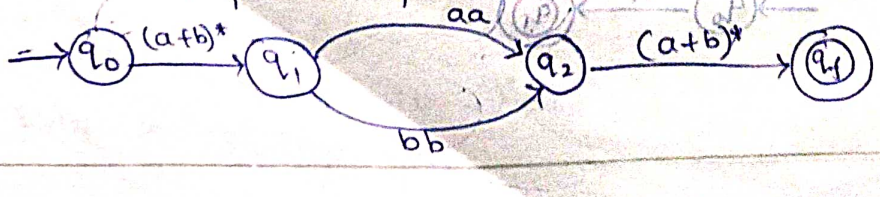
One is beggining and other is final state. The RE is placed b/w the two states with a transition from the beginning state to the final state.



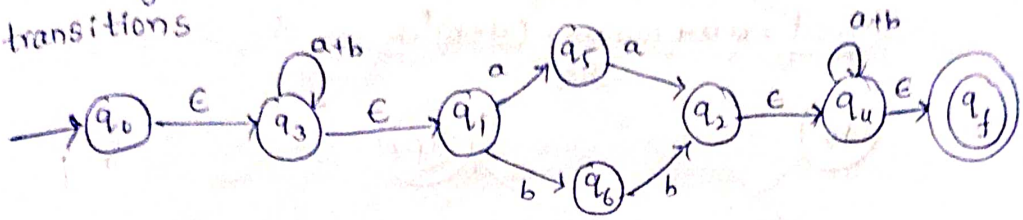
→ In the RE, there are two concatenations, b/w $(a+b)^*$ and $(aa+bb)$ and b/w $(aa+bb)$ and $(a+b)^*$. The extra states are added b/w q_0 and q_f



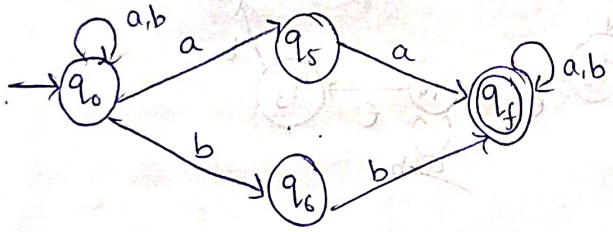
→ In aa and bb, there is a '+' (union) sign. So b/w q_1 and q_2, parallel paths are constructed as given:



→ Removing (·) concatenation of the final FA with ϵ transitions

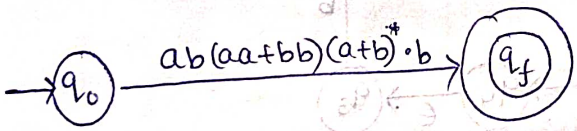


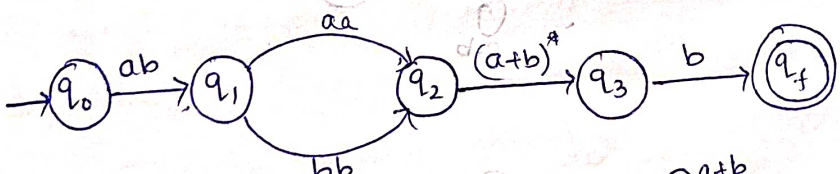
→ B/w q_0, q_3 and q_4, q_f there is a label ϵ. It means that from q_0 by getting no i/p it can reach to q_3 and the same thing is for q_4 and q_f. So, q_0 and q_3 may be treated as the same state and the same for q_4 and q_f. the transitions from q_3 and q_f are added with q_0 and q_4 respectively with same labels.

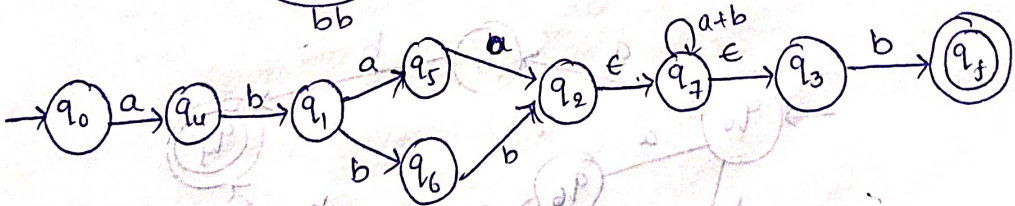


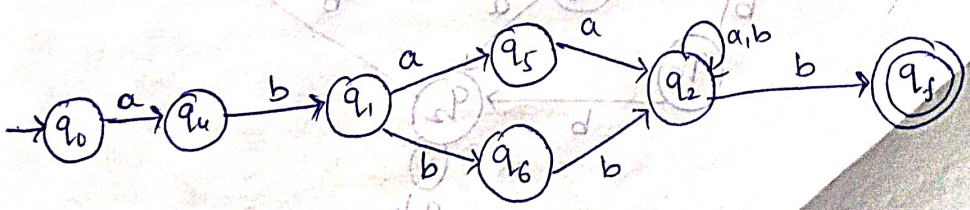
8) Construct FA equivalent to the RE

$$L = ab(aa+bb)(a+b)^*.b$$



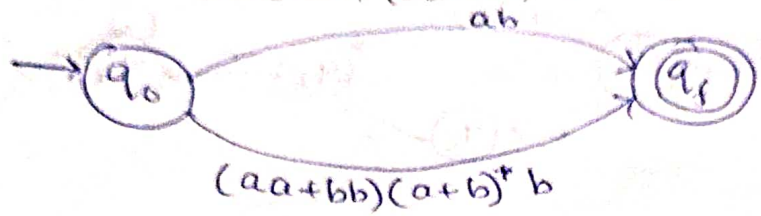


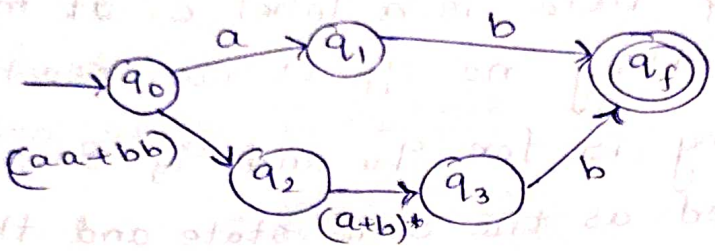


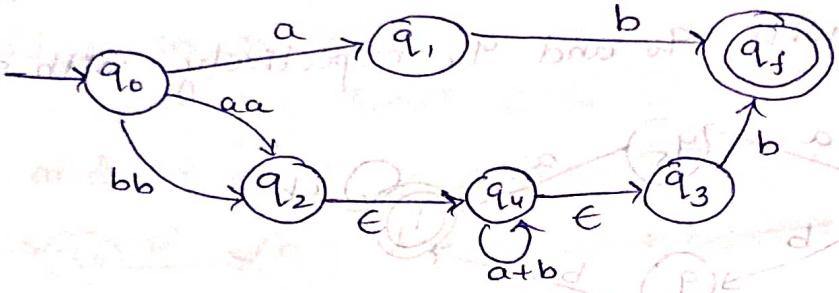


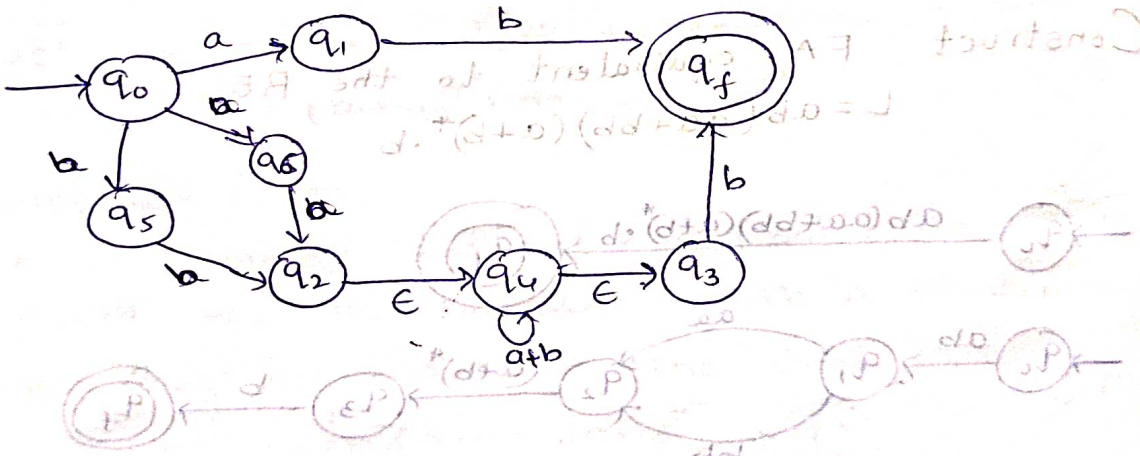
3) Construct an FA equivalent to the RE

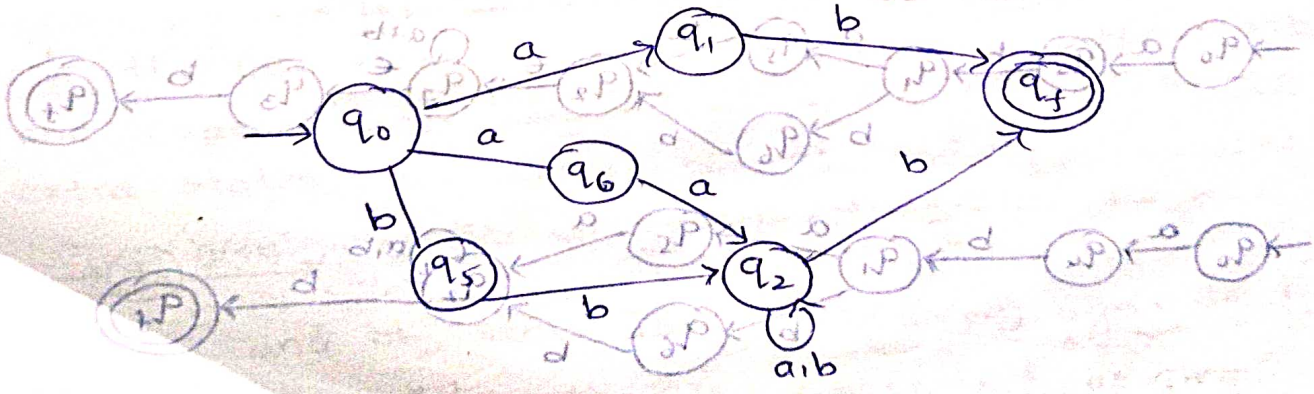
$L = ab + (aa+bb)(a+b)^*.b$





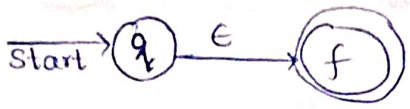




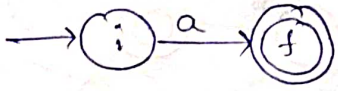


Bottom up approach (Thomson Construction):-

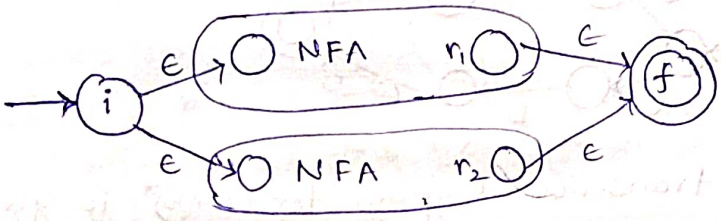
→ For i/p ϵ, the transition diagram is constructed as constructed as



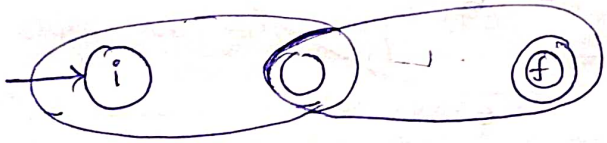
→ For i/p $a \in \Sigma$, the transition diagram is



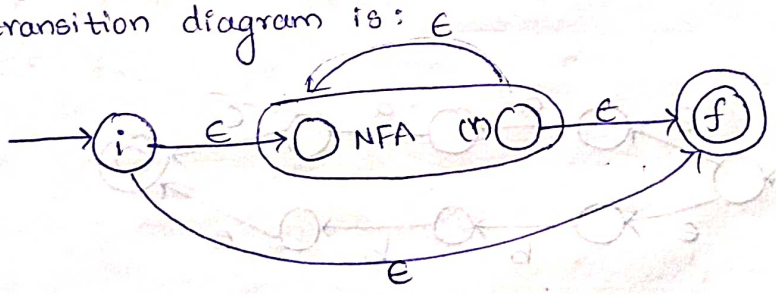
→ If NFA (r_1) is the NFA for RE r_1 and NFA (r_2) is the NFA for RE r_2, then for RE r_1+r_2 the transition diagram is:



→ If NFA(r_1) is the NFA for RE r_1 and NFA (r_2) is the NFA for RE r_2, then for the RE $r_1 \cdot r_2$ the transition diagram is:



→ If NFA(r) is the NFA for RE r, then for the RE r^*, the transition diagram is:



Problems:

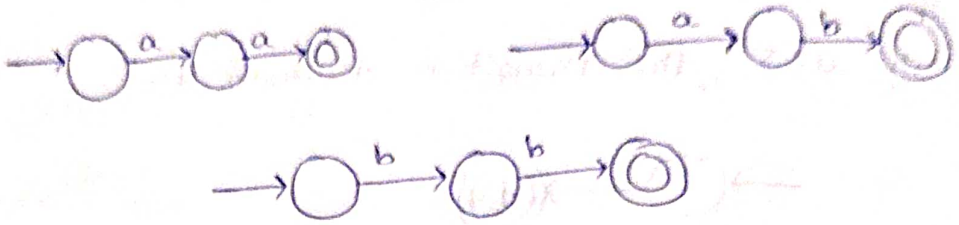
1) Construct FA equivalent to the RE

$$L = ab(aa+bb)(a+b)^* a$$

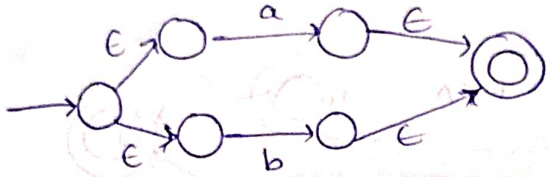
Sol: Step1:- The terminal symbols in L are 'a' and 'b'.

The transition diagram for 'a' and 'b' are given in fig

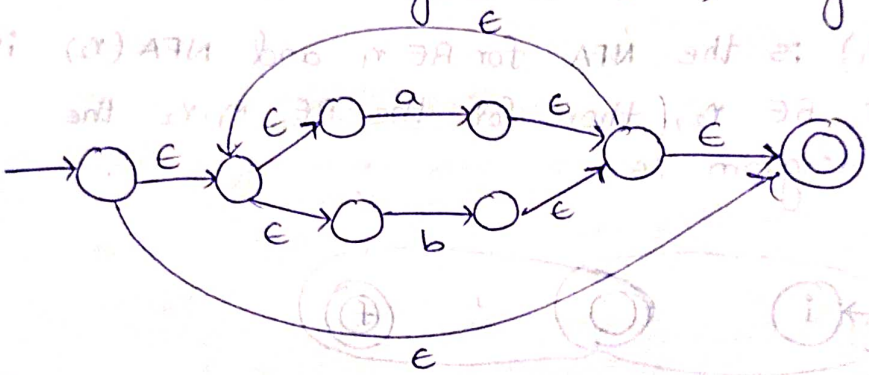
Step 2: The transition diagrams for 'aa', 'ab', 'bb' are given in fig:



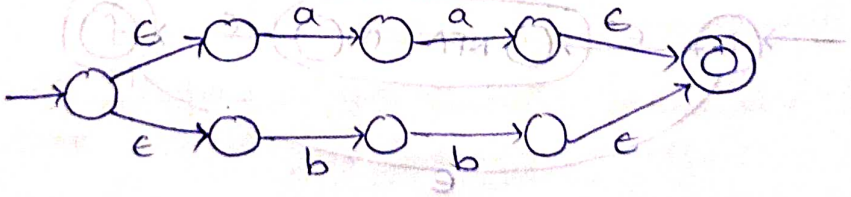
Step 3: the transition diagram for (a+b) is given in fig



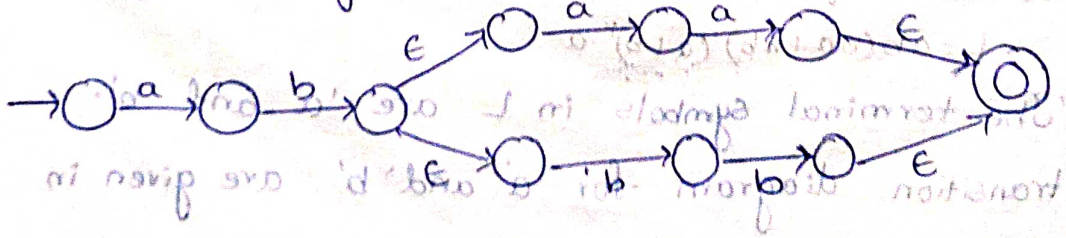
Step 4: the transition diagram for $(a+b)^*$ is given in fig



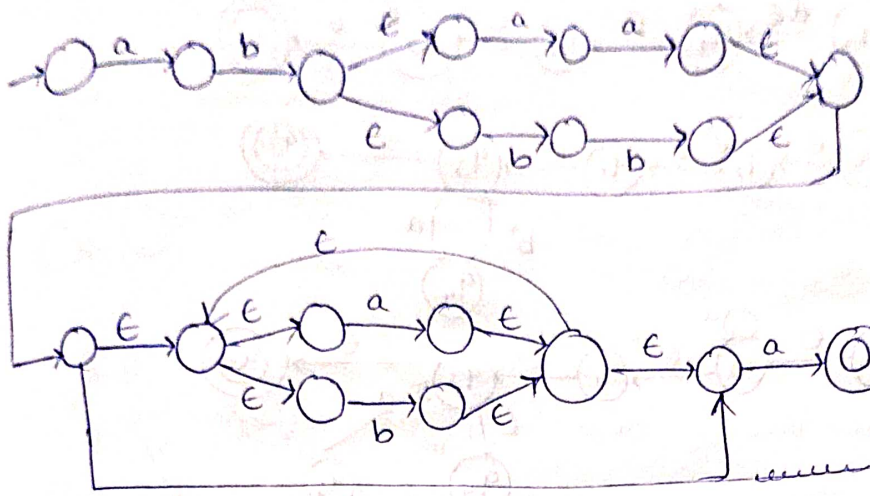
Step 5: For (aa+bb) the transitional diagram is given in fig:



Step 6: the constructed transitional diagram for ab(aa+bb) is given in fig:



Step 1:- The constructed transition diagram for $ab(aa+bb)^*(a+b)^*a$ is given by



This can be simplified to fig by removing ε transitions

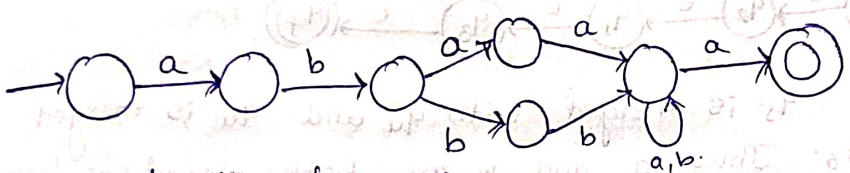


Fig: Construction of FA from RE by Thomson Construction

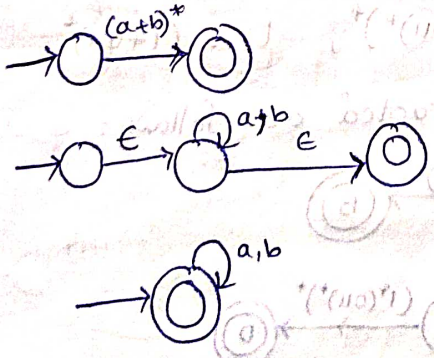
Equivalence of Two Regular Expressions:-

For every RE, there is an accepting FA. If the FA is constructed from both of the REs are same, then we can say that two RE's are equivalent

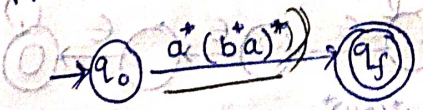
Ex:- Prove that the following RE's are equivalent

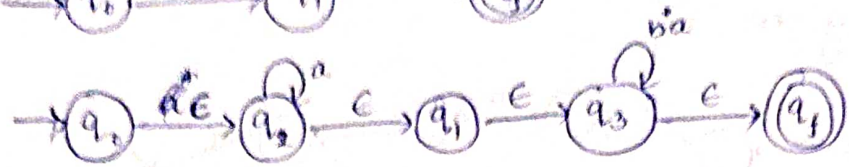
$L_1 = (a+b)^*$ $L_2 = a^*(b^*a^*)$

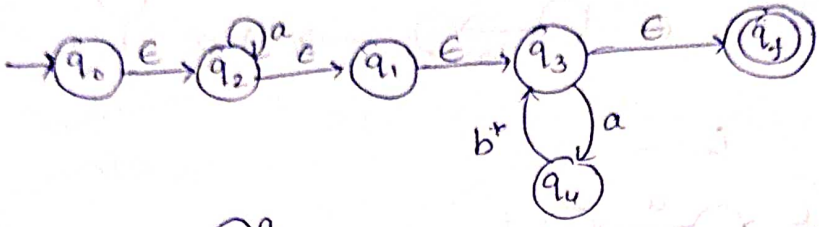
Sol: The FA for L_1 is constructed as follows:

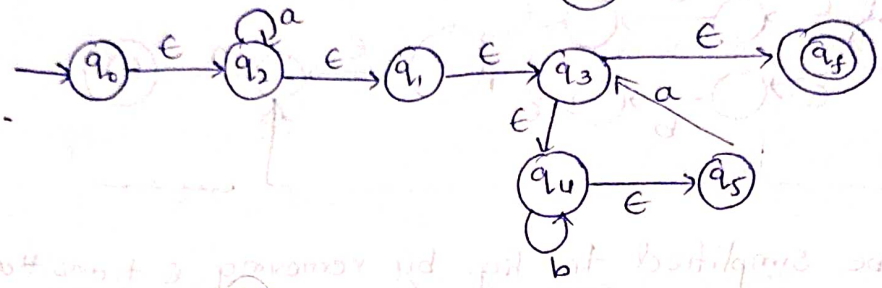


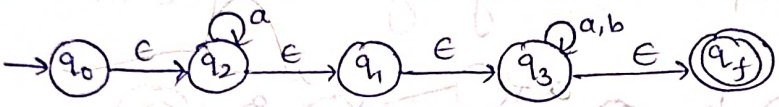
The FA for L_2 is constructed as follows:-







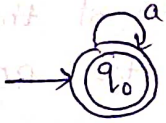




q_5 is merged with q_4 and q_4 is merged with q_3. Thus a and b are both placed as loop on q_3.

~~q_2 is~~

q_2 is merged with q_0 and q_3 is merged with q_f.

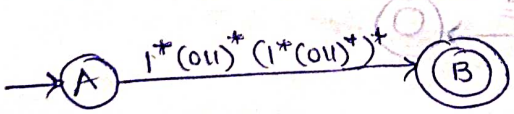


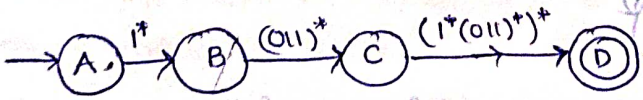
we are seeing that FA generated by two RE's L_1 and L_2 are the same. So we can decide that two FA are equivalent

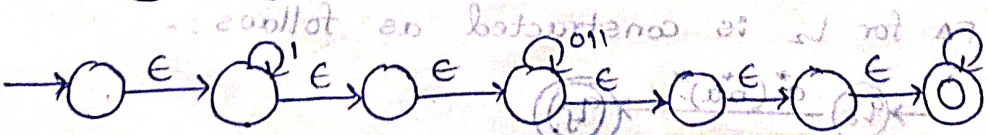
Ex:- Prove that the following RE's are equivalent

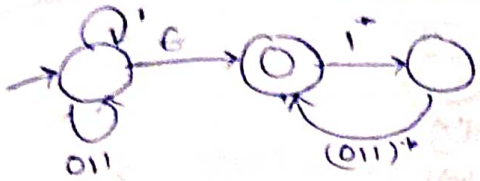
$$L_1 = 1^*(011)^*(1^*(011)^*)^*, \quad L_2 = (1+011)^*$$

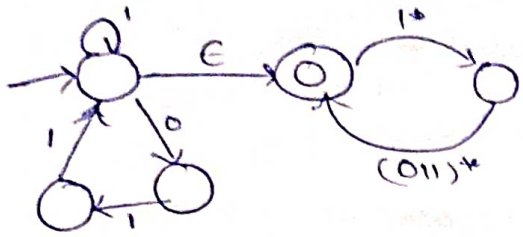
Sol:- The FA for L_1 is constructed as follows:

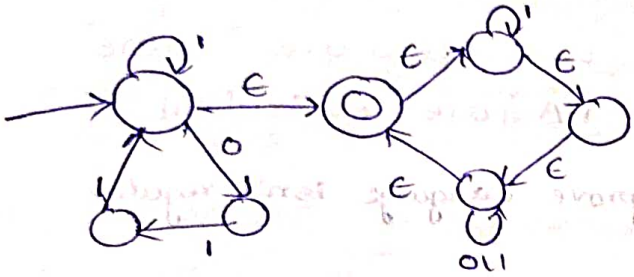


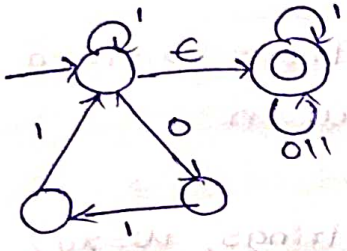


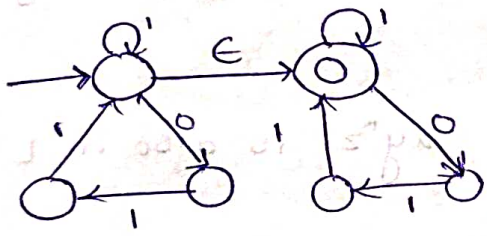


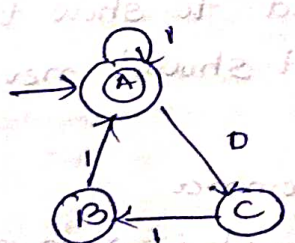




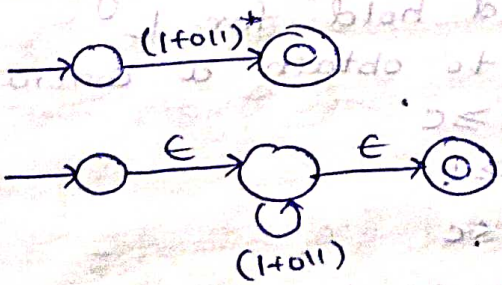


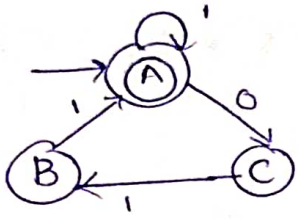






The FA for L_2 is constructed as follows:-





The FA generated by two RE's are same. So we can decide that two FA are equivalent

* **Pumping Lemma :-** used to prove Language isn't regular

Theorem :-

Let L be a regular language then there exists a Constant 'c' such that for every string w in

$L \rightarrow |w| \geq c$

we can break w into 3 strings, $w = xyz$ such that
- $|y| > 0$
- $|xy| \leq c$
- $\forall\ k \geq 0$, the string xy^kz is also in L

Applications of Pumping Lemma:

Pumping Lemma is to be applied to show that certain languages are not regular. It should never be used to show a language is regular
- If L is regular, it satisfies pumping lemma
- If L does not satisfy pumping lemma, it is non-regular

Method to prove that a Language L is not regular :-

→ At 1st, we have to assume that L is regular
→ So the pumping lemma should hold for L
→ Use the pumping lemma to obtain a contradiction
- Select w such that $|w| \geq c$
- select y such that $|y| \geq 1$
- select x such that $|xy| \leq c$
- Assign the remaining string to x
- select k such that the resulting string is not in L

Hence L is not regular

Problem:

1) Prove that $L = \{ a^i b^i \mid i \geq 0 \}$ is not regular

Sol: At first, we assume that L is regular and n is the no of states

Let $w = a^n b^n$. Thus $|w| = 2n \geq n$

By pumping lemma, let $w = xyz$ where $|xy| \leq n$

let $x = a^p$, $y = a^q$ and $z = a^r b^n$ where $p+q+r = n$

$p \neq 0$, $q \neq 0$, $r \neq 0$. thus $y \neq 0$

let $k = 2$, then $xy^2z = a^p a^{2q} a^r b^n$

No of a's $= (p + 2q + r) = (p+q+r) + q = n+q$

Hence $xy^2z = a^{n+q} \cdot b^n$ Since $q \neq 0$, xy^2z is not of form $a^n b^n$.

Thus xy^2z is not L. Hence L is not regular.

Closure properties of Regular Set :-

* A set is closed (under an operation) iff the operation on two elements of set produces another element of the set. If an element outside the set is produced then the operation is not closed.
* closure is a property which describes when we combine any two elements of the set, the result is also included in the set.

1) The integer numbers are closed under the operation of multiplication

2) Two RE's L_1 and L_2 over Σ are closed under union operation

3) The complement of an RE is also regular $RE^* = RE$

4) If L is regular and L is subset of Σ^* is regular $\Sigma^* - L$ is also a regular set. L is subset of Σ^* $\Sigma^* - L$ is

5) RE's are closed under intersection operation.

Decision problems of RE :-

* Decision problems are the problems which can be answered in "Yes" or "No"
* Decision problems which require only a finite amount of

memory

* the decision problems related to RE are:

1) Whether a string 'x' belongs to RE R?
   ($x \in R$?)
2) Whether the language set of an FA is empty?
   ($L(M) = \phi$?)
3) Whether the language set of an FA is finite?
   ($L(M)$ is finite?)
4) Whether there exists any string accepted by two FA, M_1 and M_2?
5) Whether the language set of an FA M is a subset of the language set of another FA M_2?
   ($L(M_1) \subseteq L(M_2)$?)
6) Whether two FA M_1 and M_2 accept the same language?
   ($L(M_1) = L(M_2)$)
7) Whether two RE's R_1 and R_2 are from the same language set?
8) Whether an FA is the min state FA for a language?

Applications of RE's:-

RE is mainly used in "lexical analysis of Compiler design". In the programming language, we need to declare a variable or identifier. that identifier must start with a letter followed by any number of letters of digits. the structure of identifier is represented by RE. the definition of an identifier in a programming language is

letter $\longrightarrow$ A|B| ... |Z|a|b| ... |Z

digit $\longrightarrow$ 0|1| ... |9

id $\longrightarrow$ letter (letter|digit)*

The definition of an unsigned number in a programming language is

digit $\longrightarrow$ 0|1|...|9

digit $\longrightarrow$ digit+

opt-fraction $\longrightarrow$ (.digits)*

opt-exponent $\longrightarrow$ (E(+|-)* digits)*

unsigned-num $\longrightarrow$ digit opt-fraction opt-exponent

The other application of RE is pattern searching from a given text

Construction of FA from Regular Grammar:

The process of constructing FA from RG

Step1:- If the grammar does not produce any null string, then the no.of states of F.A is equal to the no.of non-terminals of RG +1. → if no $A \to \epsilon$

Each state of FA represents each non-terminal state and the extra state is the final state of FA.

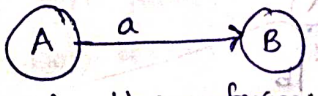
If it produces a null string then the no.of states is the same as the no of non-terminals

→ The initial state of FA is start symbol of the RG

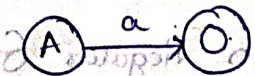
→ If the language generated by RG contains a null string, then the initial state is also the final state of constructing FA

Step2:- For a production in the form $A \longrightarrow aB$, make a δ function $\delta(A,a) \longrightarrow B$

There is an arc from state A to state B with label a



→ For a production in the form $A \longrightarrow a$, make a δ function $\delta(A,a) \longrightarrow$ final state



→ For a production $A \longrightarrow \epsilon$, makes a δ function, $\delta(A,\epsilon) \longrightarrow A$ and A is final state

Problems:-

1) Convert the following regular grammar into FA

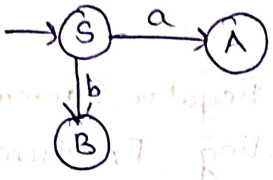
$S \rightarrow aA \mid bB \mid a \mid b$, $A \rightarrow aS \mid bB \mid b$, $B \rightarrow aA \mid bS$

Sol: In the grammar, there are there are three non-terminaly namely S, A and B

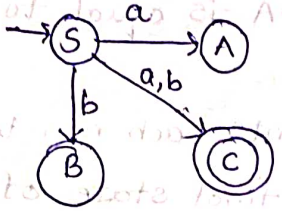
∴ the no of states of F.A is 4

Let us name the final state as C

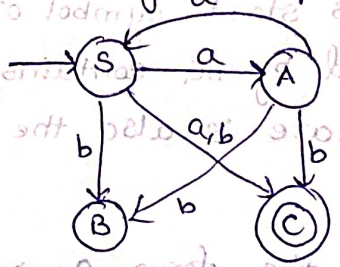
1) For the production $S \rightarrow aA \mid bB$, the transition diagram is



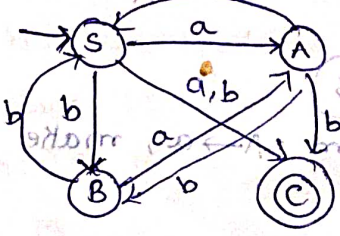
2) For the production $S \rightarrow a \mid b$, the transition diagram is



3) For the production $A \rightarrow aS \mid bB \mid b$ the transition diagram including the previous one is



4) For the production $B \rightarrow aA \mid bS$ the transitional diagram including the previous one is



This is the FA for given Regular Grammar

2) Convert the following RG into FA

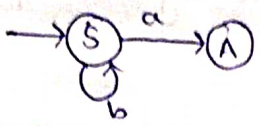
$S \to aA/bS$, $A \to bB/a$, $B \to aS/b$

In the grammar there are 3 non-terminals namely S, A and B

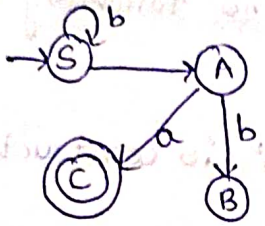
∴ the no of states of FA are 4.

let us name final state as C.

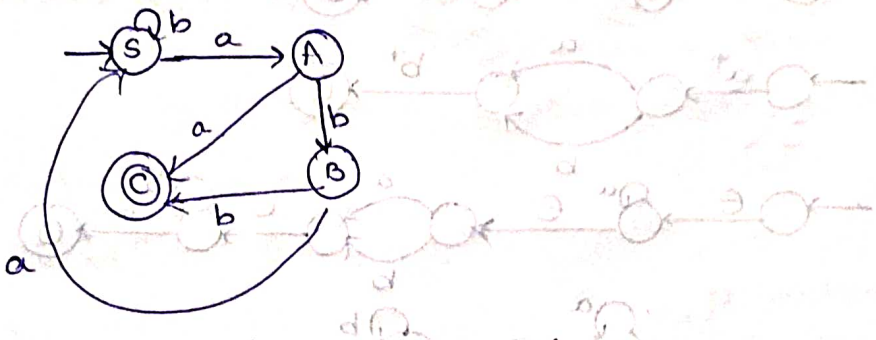
1) For the production of $S \to aA/bS$ the transition diagram is



2) For the production $A \to bB/a$ the transition diagram is



3) For the production $B \to aS/b$ the transition diagram is



This is the finite Automata for RG.

Construction of Regular Grammar from RE :-

The following section describes the process of constructing regular grammar from an RE

step1 :- Construct the equivalent FA for given RE (eliminate all null moves)

step2 :- The no. of non-terminals of the grammar will be equal to the no. of states of FA

step3 :- For all transitional functions of P in the form

$\delta(Q_1, a) \longrightarrow Q_2$

where $Q_1, Q_2 \in Q$ and, $a \in \Sigma$

④ the production rule is in the form

$A \rightarrow aA_1$

If Q_1 is 1.s then for the transitional function

$\delta(Q_1, a) \rightarrow Q_1$

④ The production rules are

$A \rightarrow aA_1$ and $A \rightarrow a$

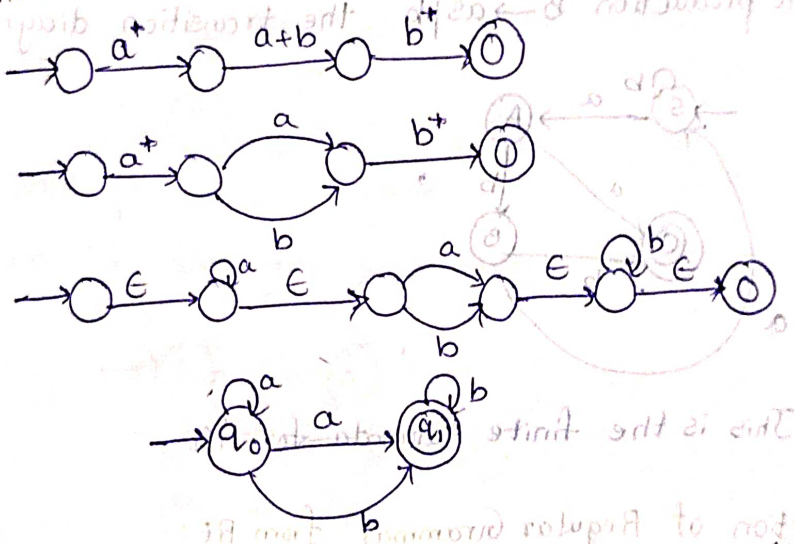
the start symbol is the symbol representing the initial state of the F.A

Problems:-

1) Construct a regular grammar for the following RE's

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

4) The FA for the string $a^+(a+b)b^*$ is constructed as follows:-



the no. of states of FA is 2, which means that there are 2 non-terminals in the grammar for the RE

Let us take them as A for q_0 and B for q_1

For the transition function

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

the production rule is $A \rightarrow aA$

Similarly $\delta(q_0, a) \rightarrow q_1, b$; $A \rightarrow aB|a$

$\delta(q_0, b) \rightarrow q_1 \Rightarrow A \rightarrow bB|b$

$\delta(q_1, b) \rightarrow q_1 \Rightarrow B \rightarrow bB|b$

the start symbol will be A as q_0 is the initial state. the grammar G for the RE $a^*(a+b)b^*$ is

$\{V_N, T, P, S\}$ where

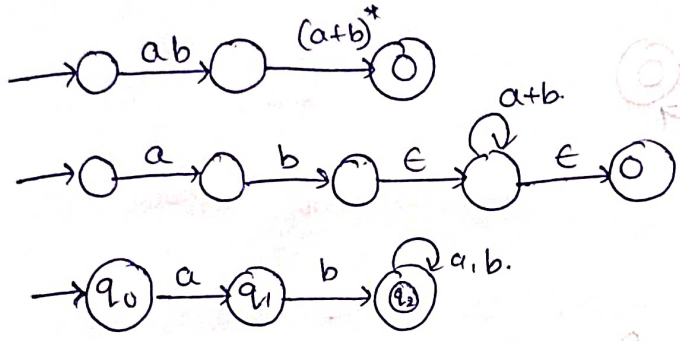
$V_N = \{A, B\}$

$T = \{a, b\}$

$S = \{A\}$

$P = \{A \rightarrow aA|bB|a|b|aB, B \rightarrow bB|b\}$:

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



Replace q_0 as A
q_1 as B
q_2 as C.

$\delta(q_0, a) \rightarrow q_1 \Rightarrow A \rightarrow aB$

$\delta(q_1, b) \rightarrow q_2 \Rightarrow B \rightarrow bc|b$

$\delta(q_2, a) \rightarrow q_2 \Rightarrow C \rightarrow ac|a$

$\delta(q_2, b) \rightarrow q_2 \Rightarrow C \rightarrow bc|b$

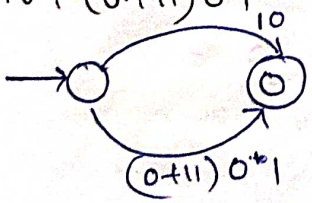
The grammar tuples are $N = \{A, B, C\}$

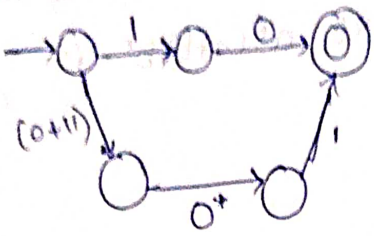
$T = \{a, b\}$

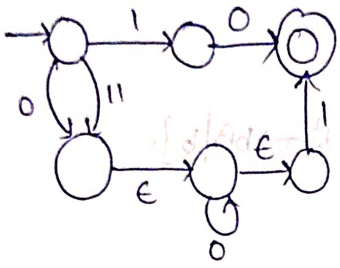
$S = \{A\}$

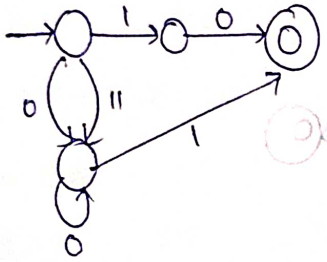
$P = \{A \rightarrow aB, B \rightarrow bc|b, C \rightarrow ac|bc|a|b\}$

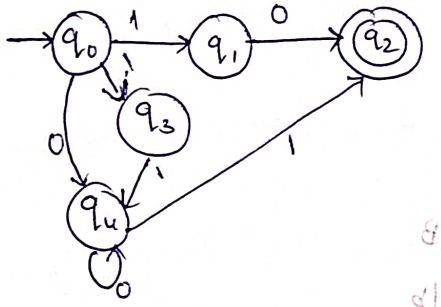
c) $10 + (0+11)0^*1$











Replace q_0 as A

q_1 as B

q_3 as C

q_4 as D

q_2 as ϵ

$\delta(q_0,1) \longrightarrow q_1 \Rightarrow A \rightarrow 1B$

$\delta(q_0,1) \longrightarrow q_3 \Rightarrow A \rightarrow 1C$

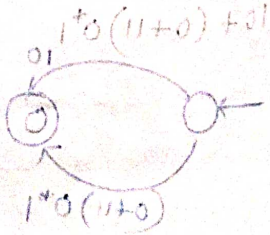
$\delta(q_0,0) \longrightarrow q_4 \Rightarrow A \rightarrow 0D$

$\delta(q_1,0) \longrightarrow q_2 \Rightarrow B \rightarrow 0E|0$

$\delta(q_3,1) \longrightarrow q_4 \Rightarrow C \rightarrow 1D$

$\delta(q_4,0) \longrightarrow q_4 \Rightarrow D \rightarrow 0D$

$\delta(q_4,1) \longrightarrow q_2 \Rightarrow D \rightarrow 1E|1$



The grammar tuples are:

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

$T = \{ 0, 1 \}$

$P = \{ A \rightarrow 1B \mid 1C \mid 0D,$

$B \rightarrow 0E \mid 0,$

$C \rightarrow 1D,$

$D \rightarrow 0D \mid 1E \mid 1$

$\}$

$S = \{ A \}$

Pumping lemma:- If a substring is repeated many times and if resultant string is also available in language L then we can say it is regular

Step 1:- consider language as Regular

2:- Assume a Constant C & select string w such that $|w| \geq c$

3:- Divide w as xyz

Such that $|y| > 0$.

$|xy| \leq c$

for $i \geq 1$. every string of form $xy^iz \in L$

Eg:- $\{a^n b^n / n \geq 0\}$ isn't regular

1:- $L = \{ab, aabbb, aaabbb \ldots\}$

2:- Let Constant $c = 6$. → min length must be 6

consider $w = aaabbb$ here $|w| \geq c$ is true

3:- $w = \underbrace{aa}_{x}\underbrace{ab}_{y}\underbrace{bb}_{z}$

$|y| = 2 > 0$.

$|xy| = |aabb| = 4 \leq 6$

for $i = 2$

$xy^2z = aaababbb$ not in L.