UNIT II
REGULAR EXPRESSIONS AND LANGUAGES

Regular Expression – FA and Regular Expressions – Proving languages not to be regular – Closure properties of regular languages – Equivalence and minimization of Automata.

2.1. REGULAR EXPRESSION

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

2.1.1. Definition:

Let Σ be an alphabet. The regular expressions over Σ and the regular sets are defined as follows,

1. ϕ is a regular expression, and the regular set is denoted as empty set { }.
2. ε is a regular expression and the regular set is denoted as {ε}.
3. For each 'a' in Σ. 'a' is a regular expression and the regular set is denoted as {a}.
4. If 'r' and 's' are regular expressions denoting the languages R and S then r+s, rs and r* are regular expressions that denotes the set R∪S, RS and R* respectively.

   Languages associated with the regular expressions 'r' is denoted as L(r). If 'r_1' and 'r_2' are regular expressions, then

   $L(r_1 + r_2) = L(r_1) + L(r_2)$

   $L(r_1 . r_2) = L(r_1) . L(r_2)$

   $L(r_1)^* = (L(r_1))^*$

2.1.2. The Operators of Regular Expressions:

Before describing the regular expression notation, we need to learn the three operations on languages that the operators of regular expressions represent. These operations are,

1. Union(∪):

If L and M are regular expressions then L∪M is the set of strings that are in either L or M or both.

Ex: If L = {001, 10, 111} and M = {ε, 001} then,

L∪M = {ε, 10, 001, 111}.

2. Concatenation(.):

If L and M are regular expressions then, the set of strings that can be formed by taking any stings in L and concatenating it with any string in M. We denote the concatenation operator is frequently called 'dot'.

Ex: If L = {001, 10, 111} and M = {ε, 001} then,
L.M (or) LM = {001, 10, 111, 001001, 10001, 111001}.

3. Closure(*):

A language L is denoted L* and represents the set of those strings that can be formed by taking any number of strings from L, possibly with repetitions and concatenating all of them.

2.1.3. Building Regular Expressions:

Basis:

It consists of three parts,

1. The constants ε and ϕ are regular expressions, denoting the languages {ε} and φ respectively. That is, L(ε) = {ε} and L(ϕ) = ϕ.
2. If 'a' is any symbol, then 'a' is a regular expression. It denotes the language {a}. That is, L(a) = {a}.
3. A variable usually capitalized such as L, is a variable representing any language.

Induction:

It consists of four parts,

1. If E and F are regular expressions, then E+F is regular expression denoting the union of L(E) and L(F). That is, L(E+F) = L(E) ∪ L(F).
2. If E and F are regular expressions, then EF is a regular expression denoting the concatenation of L(E) and L(F). That is, L(E.F) = L(E).L(F)
3. If E is a regular expression, then E* is a regular expression denoting the closure of L(E). That is , L(E)*=(L(E))*.
4. If E is a regular expression, then (E), a parenthesized E, is also a regular expression denoting the same language as E. That is, L((E)) = L(E).

2.1.4. Precedence of Regular Expression Operators:

The regular expression operators have an assumed order or "precedence", which means that operators are associated with their operands in a particular order. For regular expression the following is the order of precedence for the operators,

1. The star(*) operator is of highest precedence.
2. Next in precedence comes the concatenation or dot(.) operator.
3. Finally use the '+' and '–' operators. This '+' and '-' are lowest precedence than other two.

PROBLEMS:

Ex.1:

Let Σ = {a, b, c, d}, check whether (a+b)*(cd) is a regular expression?

Soln:

Let r = (a+b)*cd

Let r_1 = a and r_2 = b

$\therefore r_3 = r_1 + r_2$

r_3 = a + b is a regular expression.

$(r_3)^*$ = (a + b)* is also a regular expression.

Let r_4 = c, r_5 = d

$\therefore r_6$ = cd is also a regular expression. [$\therefore r_6 = r_4.r_5$]

Hence, (a + b)*(cd) is regular expression.

Ex.2:

Describe the following sets by regular expressions,

(a) The set of all strings of 0's and 1's ending in 00.
(b) Set of all strings of 0's and 1's beginning with 11 and ending with '0'.
(c) The set of all strings over {a, b} with three consecutive b's.
(d) The set of all strings with atleast one pair of consecutive 0's and atleast one pair of consecutive 1's.
(e) All strings that end with '1' and does not contains the substring '00'.

Soln:

(a) r = (0+1)*00
(b) r = 11(0+1)*0
(c) r = (a+b)*bbb(a+b)*
(d) r = (1+01)*00(1+01)*(0+10)*11(0+10)*
(e) r = (1+01)*(10+11)*1

2.2. FA AND REGULAR EXPRESSIONS

The regular expressions define the same class, it shows that,

1. Every language defined by one of these automata is also defined by a regular expression. For this proof, we can assume the language is accepted by some DFA.
2. Every language defined by one of these automata. For this part of the proof, the easiest is to show that there is an NFA with ε – transitions accepting the same language.

2.2.1. From DFA's to Regular Expressions:

Theorem:

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

Basis:

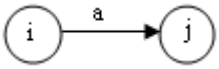
The basis is k=0. Then the regular expression is $R_{ij}^{(0)}$.

Case 1:

If there is not such symbol 'a', then $R_{ij}^{(0)} = \phi$, That is,

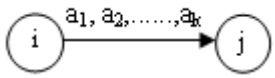
Case 2:

If there is exactly one such symbol 'a', then $R_{ij}^{(0)} = a$, That is,



Case 3:

If there are symbols $a_1, a_2, \ldots\ldots, a_k$ that label arcs from state 'i' to state 'j', then $R_{ij}^{(0)} = a_1+a_2+\ldots..+a_k$, That is,



Case 4:

If i = j then the legal paths are the path of length '0' and all loops from 'i' to itself. The path of length '0' is represented by the regular expression 'ε', since that path has no symbols along it.

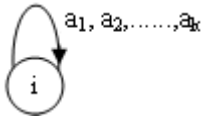
If there is no such symbol 'a', the expression becomes ε, then $R_{ii}^{(0)} = \varepsilon + \phi$, That is,

Case 5:

If there is a symbol 'a', the expression becomes ε, then $R_{ii}^{(0)} = \varepsilon + a$, That is,

(or)

$R_{ii}^{(0)} = \varepsilon + a_1 + a_2 + \ldots\ldots + a_k$, That is,



Induction:

Suppose there is a path from state 'i' to state 'j' that goes through no state higher than 'k'. There are two possible cases to consider,

Case 1:

The path does not go through state 'k' at all. In this case, the label of the path is in the language of $R_{ij}^{(k-1)}$.

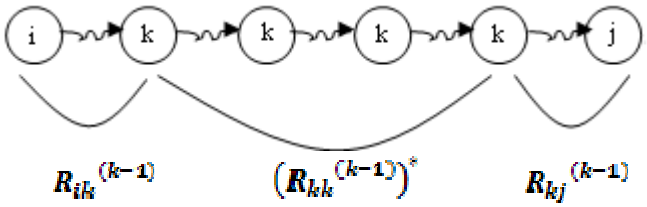
Case 2:

The path goes through state 'k' atleast once. The we can break the path into several pieces. That is,

(i) The first goes from state 'i' to state 'k' without passing through 'k'. That is, $R_{ik}^{(k-1)}$.

(ii) The last piece goes from 'k' to 'j' without passing through 'k'. That is, $R_{kj}^{(k-1)}$.

(iii) All the pieces in the middle go from 'k' to itself without passing through 'k'. That is, $\left(R_{kk}^{(k-1)}\right)^*$.



$\therefore R_{ij}^{(k)}$ = case1 + case2

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

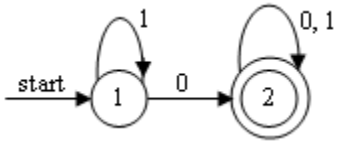
2.2.2. Minimization Rules for Regular Expression:

1. ϕ + R = R
2. ϕR = Rϕ = ϕ
3. εR = Rε = R
4. ε* = ε and ϕ* = ε
5. R + R = R
6. R*R* = R*
7. RR* = R*R
8. (R*)* = R*
9. ε + RR* = R* = ε + R*R
10. R* + ε = R*
11. (R + ε)* = R*
12. R.R = R
13. ε + R = R
14. R*(a + b) + (a + b) = R*(a + b)
15. ϕ + ε = ε
16. R*R + R = R*R
17. (R + ε) (R + ε)* (R + ε) = R*
18. (R + ε) R* = R*(R + ε) = R*
19. ϕ(ε + R)* = ϕ
20. (ε + R) (ε + R)* = R*
21. ε + R* = R*

PROBLEMS:

Ex.1:

Convert the following DFA to regular expression.



Soln:

Regular Expression is denoted by, R.E. = $R_{ij}^{(k)}$

where i = start state, j = final state

k = total number of states

In this, R.E = $R_{12}^{(2)}$; i = 1, j = 2, k = 2

Basis: Assume k = 0

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

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

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

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

Induction:

Formula is,

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

Now, substitute the i, j and k values;

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

Here, Find Regular Expressions $R_{12}^{(1)}$ and $R_{22}^{(1)}$

First Find $R_{12}^{(1)}$

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

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

$= 0 + 1^*0$ [$\therefore (\varepsilon + R)(\varepsilon + R)^* = R^*$ by 20]

$R_{12}^{(1)} = 1^*0$ [$\therefore R^*R + R = R^*R$ by 16]

Now Find $R_{22}^{(1)}$

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

$= \varepsilon + 0 + 1 + \phi\,(\varepsilon + 1)^*0$ [$\therefore \phi R = \phi$]

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

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

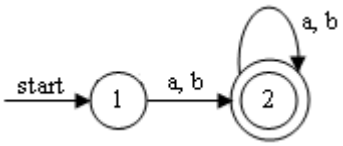
$= 1^*0 + 1^*0(\varepsilon + 0 + 1)^* \varepsilon + 0 + 1$

$= 1^*0 + 1^*0(0 + 1)^*$ [$\therefore (\varepsilon + R)(\varepsilon + R)^* = R^*$ by 20]

$R_{12}^{(2)} = 1^*0(0 + 1)^*$ [$\therefore R^*R + R = R^*R$ by 16]

Ex.2:

Convert the following DFA to Regular Expression.



Soln:

Basis: Assume k = 0

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

$R_{12}^{(0)} = a + b$

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

$R_{22}^{(0)} = \varepsilon + a + b$

Induction:

Formula is,

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

Now, substitute the i, j and k values;

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

Here, Find Regular Expressions $R_{12}^{(1)}$ and $R_{22}^{(1)}$

First Find $R_{12}^{(1)}$

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

$= a + b + \varepsilon + \phi\ (\varepsilon + \phi)^*\ a + b$

$= a + b + \varepsilon(\varepsilon)^*\ a + b$ [$\therefore \phi + \varepsilon = \varepsilon$]

$= a + b + a + b$

$R_{12}^{(1)} = a + b$ [$\therefore R + R = R$]

Now Find $R_{22}^{(1)}$

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

$= \varepsilon + a + b + \phi\ (\varepsilon + \phi)^* a + b$

$R_{22}^{(1)} = \varepsilon + a + b$ [$\therefore \phi\ R = \phi$]

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

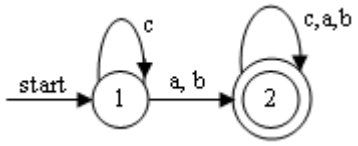
$= a + b + a + b(\varepsilon + a + b)^*\ \varepsilon + a + b$

$= a + b + a + b(a + b)^*$ [$\therefore (\varepsilon + R)\ (\varepsilon + R)^* = R^*$ by 20]

$R_{12}^{(2)} = a + b(a + b)^*$ [$\therefore R^*R + R = R^*R$ by 16]

Ex.3:

Convert the following DFA to Regular expression,



Soln:

Basis: Assume k = 0

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

$R_{12}^{(0)} = a + b$

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

$R_{22}^{(0)} = \varepsilon + c + a + b$

Induction:

Formula is,

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

Now, substitute the i, j and k values;

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

Here, Find Regular Expressions $R_{12}^{(1)}$ and $R_{22}^{(1)}$

First Find $R_{12}^{(1)}$

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

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

$= a + b + c^* a + b$ [$\therefore(\varepsilon + R)(\varepsilon + R)^* = R^*$ by 20]

$R_{12}^{(1)} = c^*(a + b)$ [$\therefore R^*R + R = R^*R$ by 16]

Now Find $R_{22}^{(1)}$

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

$= \varepsilon + c + a + b + \phi (\varepsilon + c)^* a + b$

$R_{22}^{(1)} = \varepsilon + c + a + b$ [$\therefore \phi R = \phi$]

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

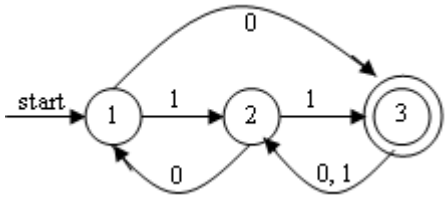
$= c^*(a + b) + c^*(a + b)(\varepsilon + c + a + b)^* \varepsilon + c + a + b$

$= c^*(a + b) + c^*(a + b)(c + a + b)^*$ [$\therefore (\varepsilon + R)(\varepsilon + R)^* = R^*$ by 20]

$R_{12}^{(2)} = c^*(a + b)(c + a + b)^*$ [$\therefore R^*R + R = R^*R$ by 16]

Ex.4:

Find the Regular expression for the set of all strings denoted by $R_{23}^{(2)}$ from the DFA given below,



Soln:

Basis: Assume k = 0

$R_{11}^{(0)} = \varepsilon + \phi$ $\quad R_{21}^{(0)} = 0$ $\quad R_{31}^{(0)} = \phi$

$R_{12}^{(0)} = 1$ $\quad R_{22}^{(0)} = \varepsilon + \phi$ $\quad R_{32}^{(0)} = 0 + 1$

$R_{13}^{(0)} = 0$ $\quad R_{23}^{(0)} = 1$ $\quad R_{33}^{(0)} = \varepsilon + \phi$

Induction:

Formula is,

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

Now, substitute the i, j and k values;

$$R_{23}^{(2)} = R_{23}^{(1)} + R_{22}^{(1)} \left(R_{22}^{(1)}\right)^{*} R_{23}^{(1)}$$

Here, Find Regular Expressions $R_{23}^{(1)}$ and $R_{22}^{(1)}$

First Find $R_{23}^{(1)}$

$$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^{*} R_{13}^{(0)}$$

$= 1 + 0(\varepsilon + \phi)^{*} 0$

$= 1 + 0(\varepsilon)^{*}0$ $\quad [\therefore \varepsilon + \phi = \varepsilon]$

$= 1 + \varepsilon .00$ $\quad [\therefore \varepsilon^{*} = \varepsilon]$

$R_{23}^{(1)} = 1 + 00$ $\quad [\therefore \varepsilon R = R]$

Now Find $R_{22}^{(1)}$

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

$= \varepsilon + \phi + 0(\varepsilon + \phi)^{*} 1$

$= \varepsilon + 0(\varepsilon)^{*} 1$ $\quad [\therefore \varepsilon + \phi = \varepsilon]$

$= \varepsilon + \varepsilon .01$ [$\therefore \varepsilon^* = \varepsilon$]

$R_{22}^{(1)} = \varepsilon + 01$ [$\therefore \varepsilon R = R$]

$R_{23}^{(2)} = R_{23}^{(1)} + R_{22}^{(1)} \left(R_{22}^{(1)}\right)^* R_{23}^{(1)}$

$= 1 + 00 + \varepsilon + 01(\varepsilon + 01)^* 1 + 00$

$= 1 + 00 + (01)^* 1 + 00$ [$\therefore (\varepsilon + R)\,(\varepsilon + R)^* = R^*$ by 20]

$= 1 + 00(\varepsilon + (01)^*)$

$R_{23}^{(2)} = 1 + 00\,(01)^*$ [$\therefore \varepsilon + R^* = R^*$]

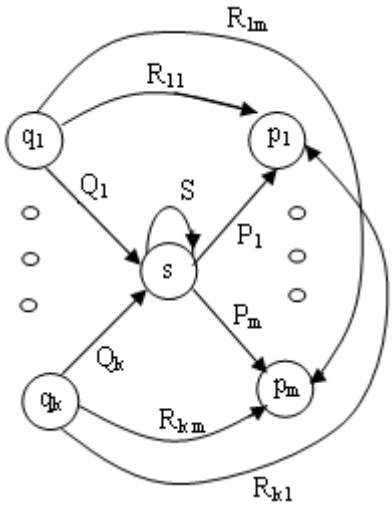
2.2.3. Converting DFA's to Regular Expressions by Eliminating States:

For converting DFA's to Regular Expression by avoids duplicating work at some points.

PROBLEMS:

Ex.1:

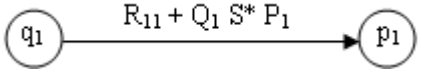
Convert the following DFA to regular expression by eliminating states.



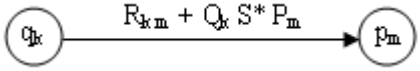
Soln:

To eliminate state 's'. So all arcs involving state 's' are deleted.

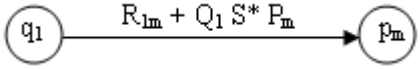
1. Find $q_1 \rightarrow p_1$



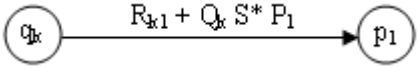
2. Find $q_k \rightarrow p_m$



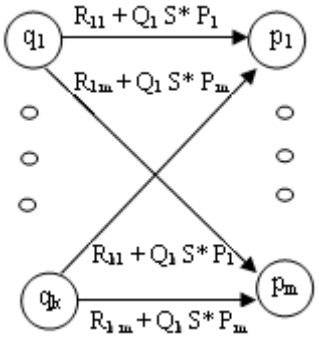
3. Find $q_l \rightarrow p_m$



4. Find $q_k \rightarrow p_l$

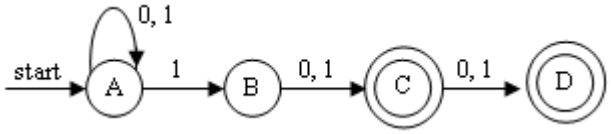


Result of Eliminating State 's' is;



Ex.2:

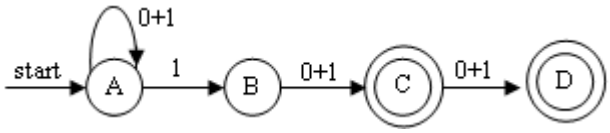
Covert the following NFA to regular expression by eliminating states,



Soln:

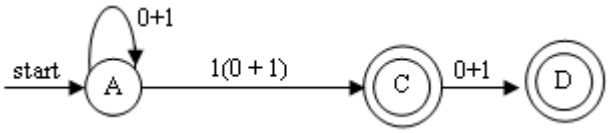
1. To convert it to an automaton with regular expression labels;

Since no state elimination has been performed all we have to do is replace the labels "0,1" with the equivalent regular expression 0 +1. The result is,



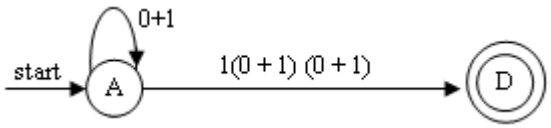
(Fig.1)

2. Let us first eliminate state B. State B has one predecessor A, and one successor C. After eliminate state B the result is,



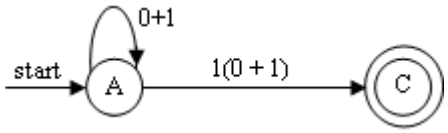
(Fig.2)

3. Now, we must branch eliminating states C and D in separate reductions. To eliminate state C, the mechanics are similar to those we performed above to eliminate state B, and the resulting automaton is,



(Fig.3)

4. Now, we must start again at Fig.2 and eliminate state D instead of C. Since D has no successors. The resulting automaton is,



(Fig.4)

5. Finally, to sum the two expressions to get the expression for the entire automaton. The expression is,

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

2.2.4. <u>Converting Regular Expressions to Automata:</u>

All of the automata we construct are ε – NFA's with a single accepting state.

Theorem:

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

Proof:

Suppose L = L(R) for a regular expression R. We show that L = L(E) for some ε – NFA E with;

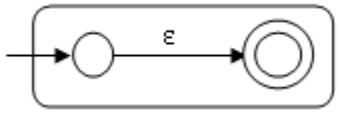
1. Exactly one accepting state.
2. No arcs into the initial state.
3. No arcs out of the accepting state.

Basis:

There are three parts to the basis,

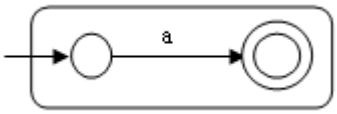
a. How to handle the expression ε.

The language of the automaton is easily seen to be {ε}, since the only path from the start state to an accepting state is labeled ε.



b. It shows the construction for ϕ. Clearly there are no paths from start state to accepting state. So ϕ is the language of this automaton.

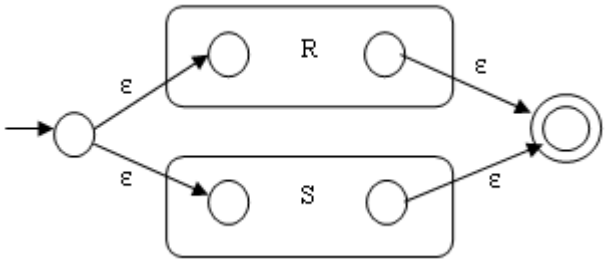
c. The language of this automaton evidently consists of the one string 'a', which is also L(a).



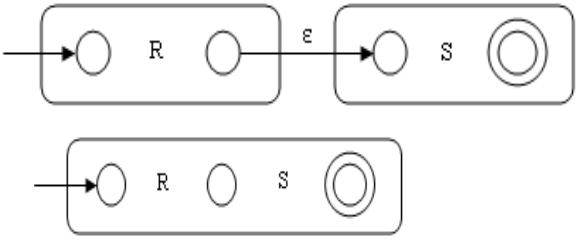
Induction:

There are three parts to the induction,

1. The expression is R + S for some smaller expressions R and s. Thus the language of the automaton is L(R) ∪ L(S). The R + S equivalent ε – NFA is,

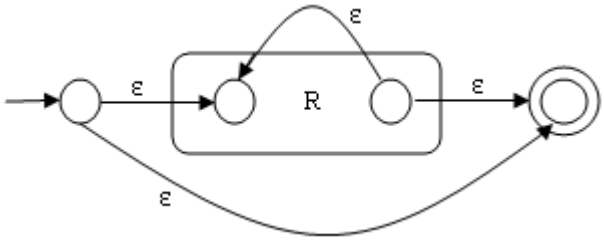


2. The expression is R.S for some smaller expressions R and S. Thus the language of automaton is L(R)L(S). The automaton for the concatenation is shown in fig.



(or)

3. The expression is R* for some smaller expression R. The R* equivalent ε – NFA is,



PROBLEMS:

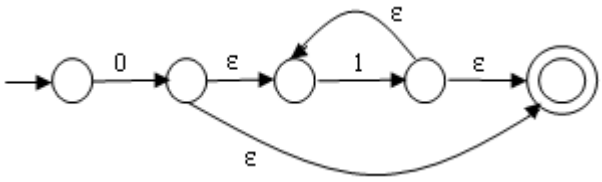
Ex.1:

Convert the following regular expressions to NFA's with ε – transitions.

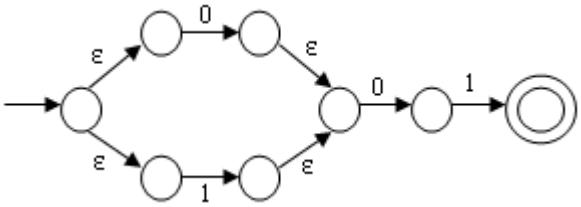
(i) 01*
(ii) (0 + 1) 01
(iii) 00 (0 + 1)*
(iv) (0 + 1)* 1 (0 + 1)

Soln:

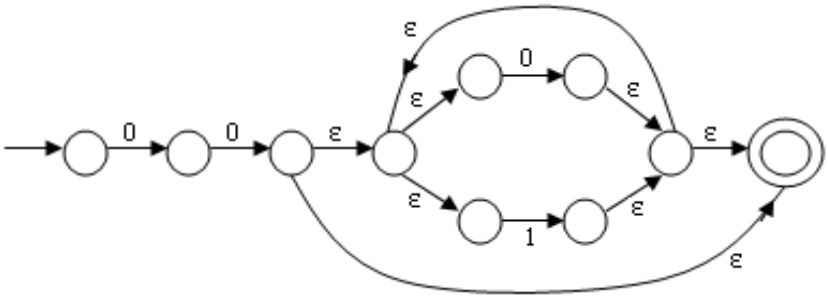
(i) 01*



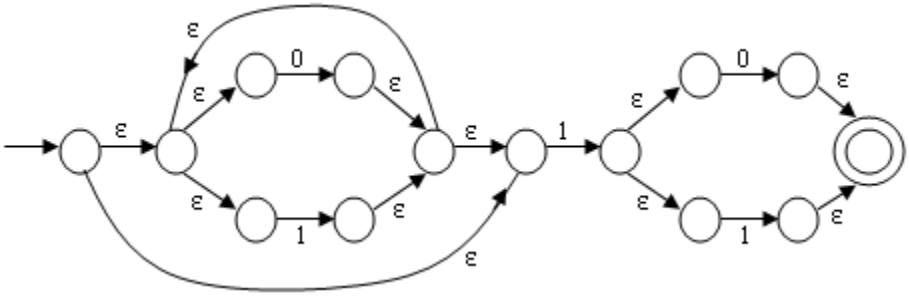
(ii) (0 + 1) 01



(iii) 00 (0 + 1)*



(iv) (0 + 1)* 1 (0 + 1)



2.3. PROVING LANGUAGES NOT TO BE REGULAR

2.3.1. Pumping Lemma:

It is a powerful technique, which is used to prove that certain languages are not regular.

Principle:

- For a string of length > n accepted by DFA (n, number of states) the walk through of a DFA must contain a cycle.
- Repeating the cycle an arbitrary number of times, it should yield another string accepted by the DFA.

2.3.2. Theorem:

Let L be a regular language. 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 three strings, w = xyz, such that;

(1) $| y | \geq 1$

(2) $| xy | \leq n$

(3) For all $k \geq 0$, the string xy^kz is also in L.

Proof:

Suppose L is regular. Then L = L(A) for some DFA A. Suppose A has 'n' states. Now, consider any string 'w' of length 'n' or more, say $w = a_1a_2 \ldots a_m$, where $m \geq n$ and each 'a_i' is an input symbol. For i=0,1,….n define state P_i to be,

$$\hat{\delta}(q_0, a_1a_2 \ldots a_i)$$

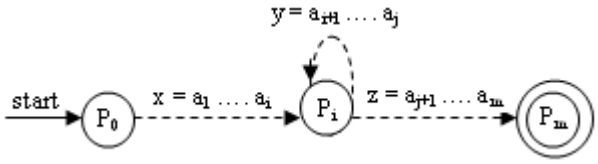
Where δ is the transition function of A, and q_0 is the start state of A. That is P_i is the state A is in after reading the first 'i' symbols of 'w'. Note that $P_0 = q_0$.

By the Pigeonhole principle, it is not possible for the n+1 different P_i's for i = 0,1, …. n to be distinct, since there are only 'n' different states. Thus we can find two different integers 'i' and 'j', with $0 \leq i \leq j \leq n$, such that $P_i = P_j$. Now, we can break w = xyz as follows,

(1) $x = a_1a_2 \ldots a_i$

(2) $y = a_{i+1}\, a_{i+2} \ldots a_j$

(3) $z = a_{j+1}\, a_{j+2} \ldots a_m$



That is, A receives the input xy^kz for any $k \geq 0$.

- If $k = 0$, then the automaton goes from the start state q_0 to P_i on input 'x'. Since, P_i is also P_j, it must be that A goes from P_i to the accepting state on input 'z'. Thus, A accepts xz.
- If $k > 0$, then A goes from q_0 to P_i on input 'x', circles from P_i to P_{ik} times on input y_k, and then goes to the accepting state on input z. Thus for any $k \geq 0$, xy^kz is also accepted by A, that is xy^kz is in L.

2.3.3. Applications of the Pumping Lemma:

To prove certain language is not regular.

1. Assume language L is regular.
2. Let 'n' be the constant of pumping lemma and is finite.
3. Select a string 'w' in L with $|w| \geq n$.
4. Show that for every decomposition of 'w' into 'xyz'(such that $|y| \geq 1$, $|xy| \leq n$) there exists $k \geq 0$, such that $xy^kz \notin L$.
5. Conclude the assumption in (1) is false, that is, the language is not regular.

PROBLEMS:

Ex.1:

Show that $L = \{0^n1^n \mid n \geq 1\}$ is not regular.

Soln:

Assume L is regular.

L = {01, 0011, 000111, 00001111, …….. }

Let w = xyz.

Take a string in L = 000111 [Take any string in L]

To prove, 000111 is not regular.

Case 1:

w = 000111 ; n = 6

Now, divide 'w' into xyz.

Let x = 00, y = 0, z = 111 [$\therefore |y| \geq 1, |xy| \leq n$]

Find, xy^kz. When k = 2,

$xy^2z = 0000111$

$\therefore 0000111 \notin L$.

So L is not regular.

Case 2:

$w = 000111$; $n = 6$

Now, divide 'w' into xyz.

Let $x = 0, y = 00, z = 111$ [$\therefore |y| \geq 1, |xy| \leq n$]

Find, xy^kz. When k = 2,

$xy^2z = 00000111$

$\therefore 00000111 \notin L$.

So L is not regular.

Ex.2:

Show that $L = \{0^i1^j \mid i > j\}$ is not regular.

Soln:

Assume L is regular.

L = {001, 0001, 00011, 000011, …….. }

Let w = xyz.

Take a string in L = 00011 [Take any string in L]

To prove, 00011 is not regular.

Case 1:

$w = 00011$; $n = 5$

Now, divide 'w' into xyz.

Let $x = 000, y = 1, z = 1$ [$\therefore |y| \geq 1, |xy| \leq n$]

Find, xy^kz. When k = 2,

$xy^2z = 000111$

$\therefore 000111 \notin L$.

So L is not regular.

Case 2:

$w = 00011$; $n = 5$

Now, divide 'w' into xyz.

Let $x = 000, y = 11, z = \varepsilon$ [$\therefore |y| \geq 1, |xy| \leq n$]

Find, xy^kz. When k = 2,

$xy^2z = 0001111$

$\therefore 0001111 \notin L$.

So L is not regular.

2.4. CLOSURE PROPERTIES OF REGULAR LANGUAGES

If certain languages are regular, and a language L is formed from them by certain operations, then L is also regular. These theorems are often called closure properties of the regular languages. Some of the closure properties fro regular languages are,

(1) The Union of two regular languages is regular.
(2) The Intersection of two regular languages is regular.
(3) The Complement of a regular language is regular.
(4) The Difference of two regular languages is regular.
(5) The Reversal of a regular language is regular.
(6) The Closure(star) of a regular language is regular.
(7) The Concatenation of regular language is regular.
(8) The Homomorphism of a regular language is regular.

1. Closure Under Union:

Let L and M be languages over alphabet Σ. Then L∪M is the language that contains all strings that are in either or both or L and M.

Theorem:

If L and M are regular languages then L∪M is also regular.

Proof:

Since L and M are regular languages the regular expressions say,

$L = L(R), \quad M = L(S)$

Then $L \cup M = L(R + S)$

For ex: Consider two languages L and M,

$L = \{0^n 1^n \mid n \geq 1\}$ and $M = \{0^i 1^j \mid i \geq j\}$

ie) $L = \{01, 0011, 000111, 00001111, \ldots\ldots\}$ and $M = \{01, 001, 0011, 00011, \ldots\ldots\}$

$\therefore L \cup M = \{01, 0011, 000111, 00001111, 001, 00011, \ldots\}$

This is present in both the languages L and M. So the Union of two regular languages is regular.

2. Closure Under Intersection:

Let L and M be languages over alphabet Σ. Then L∩M is the language that contains all strings in both the languages L and M.

Theorem:

If L and M be regular languages, then L∩M is regular.

Proof:

Since L and M are regular languages the regular expressions say,

L = L(R), M = L(S)

Then L∩M = L(R . S)

For ex: Consider two languages L and M,

$L = \{0^n1^n \mid n \geq 1\}$ and $M = \{0^i1^j \mid i \geq j\}$

ie) L = {01, 0011, 000111, 00001111, …….. } and M = {01, 001, 0011, 00011, …….. }

∴ L∩M = {01, 0011, ……}

This is present in both the languages L and M. So the Intersection of two regular languages is regular.

3. <u>Closure Under Complementation:</u>

Steps for converting a regular expression to its complement is,

(i) Convert the regular expression to NFA.

(ii)Convert NFA to a DFA by subset construction.

(iii) Complement the accepting states of that DFA.

Theorem

If L is a regular language over Σ then $\bar{L} = \Sigma^* - L$ is also regular.

Proof:

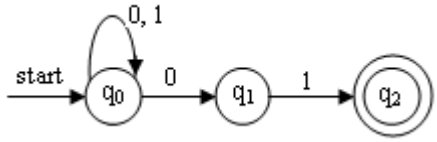
Let L = L(A) for DFA, A = (Q, Σ, δ, q_0, F) then, $\bar{L}$ = L(B) where B = (Q, Σ, δ, q_0, Q-F). B is similar to A but accepting states of A have become non accepting states of B and accepting states of B have become accepting states of A. Then 'w' is in L(B) iff $\hat{\delta}(q_0, w)$ is in Q-F which occurs iff 'w' is not in L(A).

Ex.

Find the complement of (0+1)*01.

Soln:

Step 1: Convert the regular expression to NFA.



Step 2: Convert NFA to a DFA by subset construction:

$P(N) = 2^3 = 8$, To find 8 subsets of the set of states.

$P(N) = (\{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\})$

$\{q_0\}$ ---------- (A)

$\delta_D (\{q_0\}, 0) = \{q_0, q_1\}$ ---------- (B)

$\delta_D (\{q_0\}, 1) = \{q_0\}$ ---------- (A)

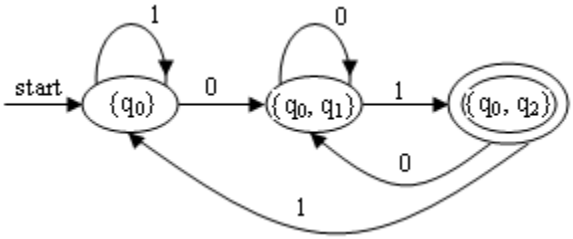
$\delta_D (\{q_0, q_1\}, 0) = \{q_0, q_1\}$ ---------- (B)

$\delta_D (\{q_0, q_1\}, 1) = \{q_0, q_2\}$ ---------- (C)

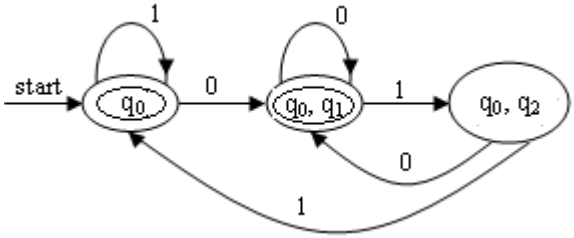
$\delta_D (\{q_0, q_2\}, 0) = \{q_0, q_1\}$ ---------- (B)

$\delta_D (\{q_0, q_2\}, 1) = \{q_0\}$ ---------- (A)

Transition Diagram:



Step 3: Complement the accepting states of that DFA.



4. Closure Under Difference:

Let L and M be languages over alphabet Σ. Then L-M is the language that contains all strings in L which is not in M.

Theorem:

If L and M be regular languages, then L-M is regular.

Proof:

Since L and M are regular languages the regular expressions say,

$$L = L(R), \quad M = L(S)$$

Then L-M = $L \cap \bar{M}$

By theorem $\bar{M}$ is regular and also intersection of two regular languages is regular.

For ex: Consider two languages L and M,

$L = \{0^i1^j \mid i \geq j\}$ and $M = \{0^n1^n \mid n \geq 1\}$

ie) L = {01, 001, 0011, 00011, …….. } and M = {01, 0011, 000111, 00001111, …….. }

∴ L-M = {001, 00011, ……}

This is present in the language L. So the Difference of two regular languages is regular.

5. Closure Under Substitution:

For each symbol 'a' in the alphabet of some regular set R, Let Ra be a particular regular set. Suppose that we replace each word a1a2 ….. an in R by set of words of the form w1w2 ……wn, where wi is an arbitrary word in Rai, then the result is always regular set.

A substitution 'f' is a mapping of an alphabet Σ onto subsets of Δ* for some alphabet Δ. Regular sets are closed under substitution.

Homomorphism(h):

It is a substitution such that h(a) contains a single string for each 'a'.

Let h(0) = ab, h(1) = ε and w = 0011

$$h(w) = abab$$

Since homomorphism is also a kind of substitution, it is also closed under regular expression.

2.5. EQUIVALENCE AND MINIMIZATION OF AUTOMATA

2.5.1. Equivalence of two states:

The language generated by a DFA is unique. But, there can exist many DFA's that accept the same language. In such cases, the DFA's are said to be equivalent.

2.5.2. Definition of Equivalent and Inequivalent States:

Equivalent (Indistinguishable) State:

Two states 'p' and 'q' of a DFA are equivalent if and only if $\delta(p, w)$ and $\delta(q, w)$ are final states or both $\delta(p, w)$ and $\delta(q, w)$ are non-final state for all $w \in \Sigma^*$ that is, if $\delta(p, w) \in F$ and $\delta(q, w) \in F$ then the states 'p' and 'q' are equivalent. If $\delta(p, w) \notin F$ and $\delta(q, w) \notin F$ then also the states 'p' and 'q' are equivalent.

Inequivalent (Distinguishable) State:

Two states 'p' and 'q' of a DFA are inequivalent, if there is atleast one string 'w' such that one of $\delta(p, w)$ and $\delta(q, w)$ is final state and the other is non-final state, then the states 'p' and 'q' are called inequivalent states.

The equivalent and inequivalent states can be obtained using **table filling algorithm**(also called mark procedure).

2.5.3. Table Filling Algorithm:

1. For each pair(p, q) where $p \in Q$ and $q \in Q$. Find $q \notin F$ or vice versa then, the pair (p, q) is inequivalent and mark the pair (p, q) [by putting 'X' for the pair (p, q)]
2. For each pair (p, q) and for each $a \in \Sigma$ find $\delta(p, a) = r$ and $\delta(q, a) = s$. If the pair (r, s) is already marked at inequivalent then the pair (p, q) is also inequivalent and mark it as say 'X'.

Ex.1:

Obtain the inquivalent table for the automaton shown below,

State	Input	
	0	1
→A	B	F
B	G	C
*C	A	C
D	C	G
E	B	F
F	C	G
G	G	E
H	G	C

Soln:

Construct a table with an entry for each pair of states. An 'X' is placed in the table each time we discover a pair of state that cannot be equivalent. Initially an 'X' is placed in each entry corresponding to one final state and one non-final state.

B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G	X	X	X	X	X	X	
H	X		X	X	X	X	X
	A	B	C	D	E	F	G

2.5.4. <u>Minimization of Automata:</u>

Once we have found the distinguishable and indistinguishable pairs we can easily minimize the number of states of a DFA and accepting the same language which is accepted by original DFA. It is required to reduce the number of states for storage efficiency.

<u>PROBLEMS:</u>

Ex.1:

Construct minimized DFA for the following table,

State	Input	
	A	b
→A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

Soln:

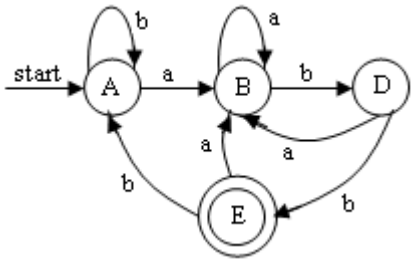
Groups [ABCD] [E]

Find equivalent for each group, here A ≡ C

Now, choose the representative for states A and C.

If the representative is A then, eliminate state C and substitute all C for A inside the table.

State	Input	
	a	b
→A	B	A
B	B	D
D	B	E
*E	B	A



Ex.2:

Construct minimized DFA for the following table,

State	Input	
	A	b
→A	B	C
B	B	D
C	B	C
D	B	E
*E	F	G
*F	F	H
*G	F	G
*H	F	I
*I	F	G

Soln:

Groups [ABCD] [EFGHI]

Find equivalent for each group, here group [ABCD], A ≡ C

Now, choose the representative for states A and C.

If the representative is A then, eliminate state C and substitute all C for A inside the table.

ie)

State	Input	
	A	b
→A	B	A
B	B	D
D	B	E
*E	F	G
*F	F	H
*G	F	G
*H	F	I
*I	F	G

Now, find equivalent for group [EFGHI], here E ≡ G ≡ I, representative is => E. Eliminate states G and I then Substitute all G and I for E. ie)

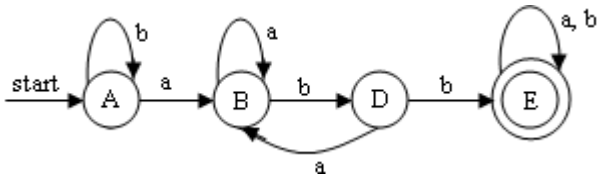
State	Input	
	A	b
→A	B	A
B	B	D
D	B	E
*E	F	E
*F	F	H
*H	F	E

Now E ≡ H, representative is => E. Eliminate state H then Substitute all H for E. ie)

State	Input	
	A	b
→A	B	A
B	B	D
D	B	E
*E	F	E
*F	F	E

Now E ≡ F, representative is => E. Eliminate state F then Substitute all F for E. ie)

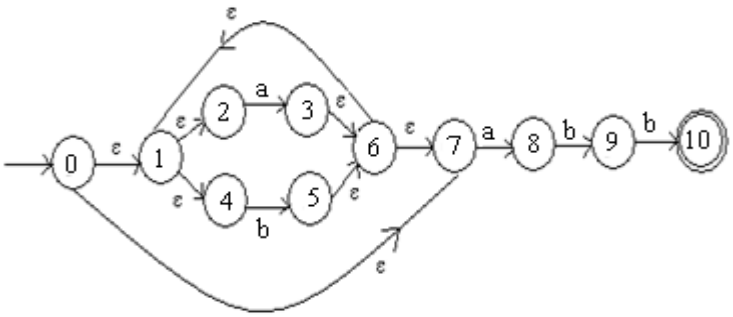
State	Input	
	A	b
→A	B	A
B	B	D
D	B	E
*E	E	E



Ex.3:

Construct the minimal state DFA for the regular expression (a/b)*abb.

Soln:



Step 1: To find ε-closure.

ECLOSE(0) = {0, 1, 2, 4, 7} -- (A)
δ (A, a) = {3, 8} δ (A, b) = {5}
ECLOSE (3,8) = {3, 8, 1, 2, 4, 6, 7} –(B)
δ (B, a) = {3, 8} δ (B, b) = {5, 9}
ECLOSE (5) = {5, 1, 2, 4, 6, 7} --(C)
δ (C, a) = {3, 8} δ (C, b) = {5}
ECLOSE (5,9) = {5, 9, 1, 2, 4, 6, 7} –(D)
δ (D, a) = {3, 8} δ (D, b) = {5, 10}
ECLOSE (5,10)= {5, 10, 1, 2, 4, 6, 7} – (E)
δ (E, a) = {3, 8} δ (E, b) = {5}

Step 2: Construct transition table.

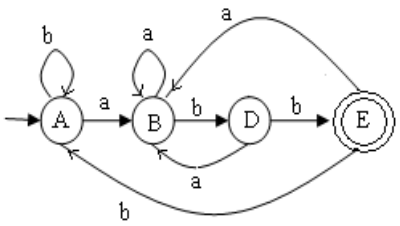
State	a	b
→A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

Step 3: To minimize the table.

Divide two groups i.e., [ABCD] [E]
Here, A is equivalent to C(i.e., A & C are common states)
Choose the representative → A ∴ substitute C for A.

State	A	b
→A	B	A
B	B	D
D	B	E
*E	B	A

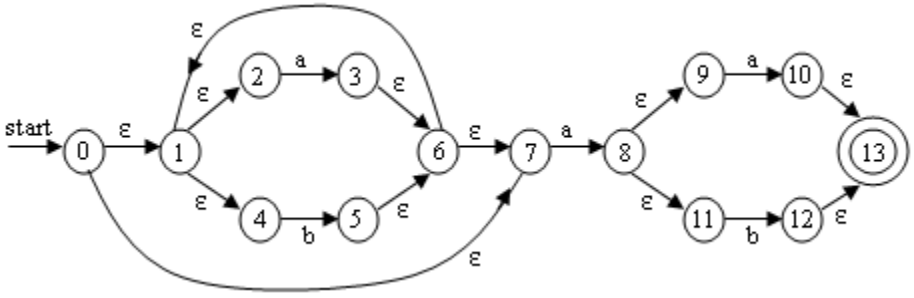
Step 4: Construct Transition diagram for DFA.



Ex.4:

Construct the minimal state DFA for the regular expression (a/b)*a(a/b).

Soln:



Step 1: To find ε-closure.

ECLOSE(0) = {0, 1, 2, 4, 7} ---------- A
δ (A, a) = {3, 8} δ (A, b) = {5}
ECLOSE (3,8) = {3, 8, 6, 7, 1, 2, 4, 9, 11} ------ B
δ (B, a) = {3, 8,10} δ (B, b) = {5, 12}
ECLOSE (5) = {5, 6, 7, 1, 2, 4} ---------- C
δ (C, a) = {3, 8} δ (C, b) = {5}

ECLOSE (3, 8, 10) = {3,8,10,6,7,1,2,4,9,11,13}---D
δ (D, a) = {3, 8, 10} δ (D, b) = {5, 12}
ECLOSE (5,12)= {5, 12, 6, 7, 1, 2, 4, 13} -------- E
δ (E, a) = {3, 8} δ (E, b) = {5}

Step 2: Construct transition table.

State	a	b
→A	B	C
B	D	E
C	B	C
*D	D	E
*E	B	C

Step 3: To minimize the table.

Divide two groups i.e., [ABC] [DE]
Here, A is equivalent to C(i.e., A & C are common states)
Choose the representative → A ∴ substitute C for A.

State	a	b
→A	B	A
B	D	E
*D	D	E
*E	B	A

Step 4: Construct Transition diagram for DFA.