UNIT-2(Part-1)

Regular Expressions

1. **RE to RL**
2. **RL to RE**
3. **FA to RE**
4. **RE to FA**
5. **Pumping lemma for RL**
6. **Closure Properties of RL**

Regular Expressions(RE)

→ A regular expression is an algebraic notation that describes exactly the language described by FA.

→ Provide a declarative way to express the strings in the regular language.

→ Any set represented by a RE is called Regular language(RL) or regular set(RS)

→ A **Regular Expression** can be recursively defined as follows:

- **φ** is a Regular Expression denoting an empty language. **{ }**
- **ε** is a Regular Expression indicates the language containing an empty string. {**ε**}
- **X** is a Regular Expression indicating the language**{ X}**
- If **X** is a Regular Expression denoting the language **L(X)** and **Y** is a Regular Expression denoting the language **L(Y)**, then
  - **X+Y** is a Regular Expression corresponding to the language **L(X) ∪ L(Y)** where **L(X+Y) = L(X) ∪ L(Y)**.
  - **X.Y** is a Regular Expression corresponding to the language **L(X) . L(Y)** where **L(X.Y) = L(X) . L(Y)**
  - **X*** is a Regular Expression corresponding to the language **L(X*)**where **L(X*) = (L(X))***
- If we apply any of the rules several times from 1 to 4, they are Regular Expressions.

Note:

- FA accepts RL
- RG generates
- RE denotes RL

Some RE Examples

Regular Expressions	Regular Set
a +b	{a,b}
Ab	{ab}
a*	{ ε, a, aa, aaa,aaaa,...}
aa*	{ a, aa, aaa,aaaa,...}
a*b	{ b, ab, aab, aaab,aaaab,...}
ab*	{ a, ab, abb, abbb,abbbb,...}
a*b*	{ ε ,a,b, ab, aab, aabb,aaaab,...} = $\{a^n b^m$ /m,n>=0}
(a+b)(a+b)	{aa,ab,ba,bb}
(a+b)*	Set of strings of a's and b's of any length including the null string. So L = { ε, a, b, aa , ab , bb , ba, aaa…….}

(ab)*	{ ε,ab,abab,ababab,abababab,.....}
	= $\{(ab)^n$ / n>=0}

(a+b)b	{ab,bb}
a(a + b)	{aa,ab}
ab + a	{ab,a}
(a +ba)b	{ab,bab}
(a + ε)b	{ab,b}
(a+ba)(b+a)	{ab,aa,bab,baa}
(0 + 10*)	L = { 0, 1, 10, 100, 1000, 10000, … }
(0*10*)	L = {1, 01, 10, 010, 0010, …}
(0 + ε)(1 + ε)	L = {ε, 0, 1, 01}

(a+b)*abb	Set of strings of a’s and b’s ending with the string abb. So L = {abb, aabb, babb, aaabb, ababb, …………..}
(11)*	Set consisting of even number of 1’s including empty string, So L= {ε, 11, 1111, 111111, ……….}
(aa)*(bb)*b	Set of strings consisting of even number of a’s followed by odd number of b’s , so L = {b, aab, aabbb, aabbbbb, aaaab, aaaabbb, …………..}
(aa + ab + ba + bb)*	String 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 = {aa, ab, ba, bb, aaab, aaba, …………..}

Find languages of following REs?

1) ab*a

2) ba*b

3) (a+b)b*

4) a*(a+b)

5) (a+b)*(a+b)

Describe the strings that are represented by the regular expression (0+1)*.0.(0+1).(0+1).

Valid Strings={0000,1000, 1010, and many other similar strings}

RL to RE

- **Language =** language of all those strings starting with a, defined over ∑=(a,b)
  R.E =a(a+b)*
- **Language=** language of all those strings starting with b, defined over ∑=(a,b)
  Rejected strings: a, aa, ab, aba,……..
  R.E = b(a+b)*
- ***A regular expression for the language of all those strings in which NO bab occurs in the string. defined over {a,b}***
  solution: R.E= a*b*a
- **Regular expression for the set of all strings of a’s and b’s that have at least 2 a’s.**
  (a+b)* a (a+b)* a (a+b)*

Valid strings={aa, aaa, baa,aab, aba, abba,…}
InValid strings={a, b, baa,abb, abb, abbb,…}

- **Regular expression for the set of all strings whose first symbol from the right end is a 0.**
  L = (0+1)*0
- **Regular expression for the set of all strings whose second symbol from the right end is a 0.**
  L = (0+1)*.0.(0+1)
- **Regular expression for the set of all strings whose 3rd symbol from the right end is a 0.**
  L = (0+1)*.0.(0+1).(0+1)
- **Regular expression for the strings that do not contain a as a string defined over {a,b}**
  (b)*
- A regular expression for the language of all those strings having even length strings and starting with a or odd length strings starting with b
  RE = a(aa+bb+ab+ba)*(a+b) + b(aa+bb+ab+ba)*
- Regular expression for the strings that do not contain single a as a string defined over {a,b}
  (aa+b)*
- Regular expression for the set of all strings of a’s and b’s that have at least 2 a’s.

  (a+b)* a (a+b)* a (a+b)*
- Regular Expression for the Language of all strings with an even number of 0’s or even number of 1’s
  Regular Expression : **(1*01*01*)* + (0*10*10*)***
- **A Regular Expression of all strings divisible by 4 defined over {a,b}.**
  Regular Expression**: {(1+0) (1+0) (1+0) (1+0)}***
- Regular Expression of all those Strings that do not contain the substring 110.
  Regular Expression: **(0+10) * 1***

- A regular expression for the language of all those strings end with abb.
  Regular expression **: (a+b)*abb**
- Determine the regular expression for all strings containing exactly one ‘a’ over ∑ = {a, b, c}.
  (b + c)* a (b + c)*
- Find the regular expression for the set of all strings over input alphabet ∑ = {a, b} with three consecutive b’s.
  (a + b)* bbb (a +b)*
- Find the regular expression for the set of all strings over {0, 1} beginning with 00.
  00 (0 + 1)*

- Find the regular expression for the set of all strings over {0, 1} ending with 00 and beginning with 1.
  1 (0 + 1)* 00
- Find the regular expression for all strings over input alphabets ∑ = {0, 1} ending in either 010 or 0010.
  (0 + 1)*(010 + 0010)
- Find the regular expression for all string containing no more than three a's over input alphabets ∑ = {a, b, c}

  (b + c)*+ (b + c)*a(b + c)* +(b+c)*a(b+c)*a(b+c)* +(b+c)*a(b+c)*a(b+c)*a(b+c)*

  = (b + c)* (ε + a) (b + c)* (ε + a) (b + c)* (ε + a) (b + c)*
- Write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over ∑ = {0, 1}.
  Solution:
  In a regular expression, the first symbol should be 1, and the last symbol should be 0. The r.e. is as follows:
  R = 1 (0+1)* 0
- Write the regular expression for the language starting and ending with a and having any having any combination of b's in between.
  Solution:
  The regular expression will be:
  R = a b* b
- Write the regular expression for the language starting with a but not having consecutive b's.
  Solution: The regular expression has to be built for the language:
  L = {a, aba, aab, aba, aaa, abab, ...}
  The regular expression for the above language is:
  R = {a + ab}*
- Write the regular expression for the language accepting all the string in which any number of a's is followed by any number of b's is followed by any number of c's.
  Solution: As we know, any number of a's means a* any number of b's means b*, any number of c's means c*. Since as given in problem statement, b's appear after a's and c's appear after b's. So the regular expression could be:
  R = a* b* c*
- Write the regular expression for the language over ∑ = {0} having even length of the string.
  Solution:
  The regular expression has to be built for the language:
  L = {ε, 00, 0000, 000000, ...}

The regular expression for the above language is:
R = (00)*

- Write the regular expression for the language having a string which should have atleast one 0 and alteast one 1.
  Solution:
  The regular expression will be:
  R = [(0 + 1)* 0 (0 + 1)* 1 (0 + 1)*] + [(0 + 1)* 1 (0 + 1)* 0 (0 + 1)*]

- **Describe the language denoted by following regular expression**
  r.e. = (b* (aaa)* b*)*
  Solution:
  The language can be predicted from the regular expression by finding the meaning of it. We will first split the regular expression as:
  r.e. = (any combination of b's) (aaa)* (any combination of b's)
  L = {The language consists of the string in which a's appear triples, there is no restriction on the number of b's}

- Write the regular expression for the language L over ∑ = {0, 1} such that all the string do not contain the substring 01.
  Solution:
  The Language is as follows:
  L = {ε, 0, 1, 00, 11, 10, 100, ...}
  The regular expression for the above language is as follows:
  R = (1* 0*)

- Write the regular expression for the language containing the string over {0, 1} in which there are at least two occurrences of 1's between any two occurrences of 0's.
  Solution: At least two 1's between two occurrences of 0's can be denoted by (0111*0)*.
  Similarly, if there is no occurrence of 0's, then any number of 1's are also allowed. Hence the r.e. for required language is:
  R = (1 + (0111*0))*

- Write the regular expression for the language containing the string in which every 0 is immediately followed by 11.
  Solution:
  The regular expectation will be:
  R = (011 + 1)*

Identities Related to Regular Expressions

Given R, P, Q as regular expressions, the following identities hold −

- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $R + \emptyset = \emptyset + R = R$ (The identity for union)
- $R\,\varepsilon = \varepsilon\, R = R$ (The identity for concatenation)
- $\emptyset\, R = R\emptyset = \emptyset$ (The annihilator for concatenation)
- $R + R = R$ (Idempotent law)
- **$\varepsilon + RR^* = \varepsilon + R^*R = R^*$**
- $p + q = q + p$
- $p+(q+r) = (p+q)+r$
- $(pq)r = p(qr)$
- $P(Q + R) = PQ + PR$ (Left distributive law)
- $(P + Q) R = PR + QR$ (Right distributive law)
- $RR^* = R^*R$
- $R^*R^* = R^*$
- $R^* + R^* = R^*$
- $(R^*)^* = R^*$
- $(R + \varepsilon)^* = R^*$
- $(PQ)^*P = P(QP)^*$
- $(P+Q)^*Q = (P^*Q)^*$
- **$(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^* = (P+Q^*)^* = P^*(QP^*)^*$**

Arden's Theorem

In order to find out a regular expression of a Finite Automaton, we use Arden’s Theorem along with the properties of regular expressions.

Statement :

Let **P** and **Q** be two regular expressions.If **P** does not contain null string, then **R = Q + RP** has a unique solution that is **R = QP***

Proof:
$R = Q + (Q + RP)P$ [After putting the value $R = Q + RP$]
$= Q + QP + RPP$
When we put the value of **R** recursively again and again, we get the following equation −
$R = Q + QP + QP^2 + QP^3$…..
$R = Q (\varepsilon + P + P^2 + P^3 + \ldots.)$
R = QP* [As P* represents (ε + P + P2 + P3 + ….)]
Hence, proved.

Assumptions for Applying Arden's Theorem

- The transition diagram must not have NULL transitions
- It must have only one initial state

Method

Step 1 − Create equations as the following form for all the states of the DFA having n states with initial state q_1.

$q_1 = q_1a_{11} + q_2a_{21} + \ldots + q_na_{n1} + \varepsilon$

$q_2 = q_1a_{12} + q_2a_{22} + \ldots + q_na_{n2}$

…………………………

…………………………

…………………………

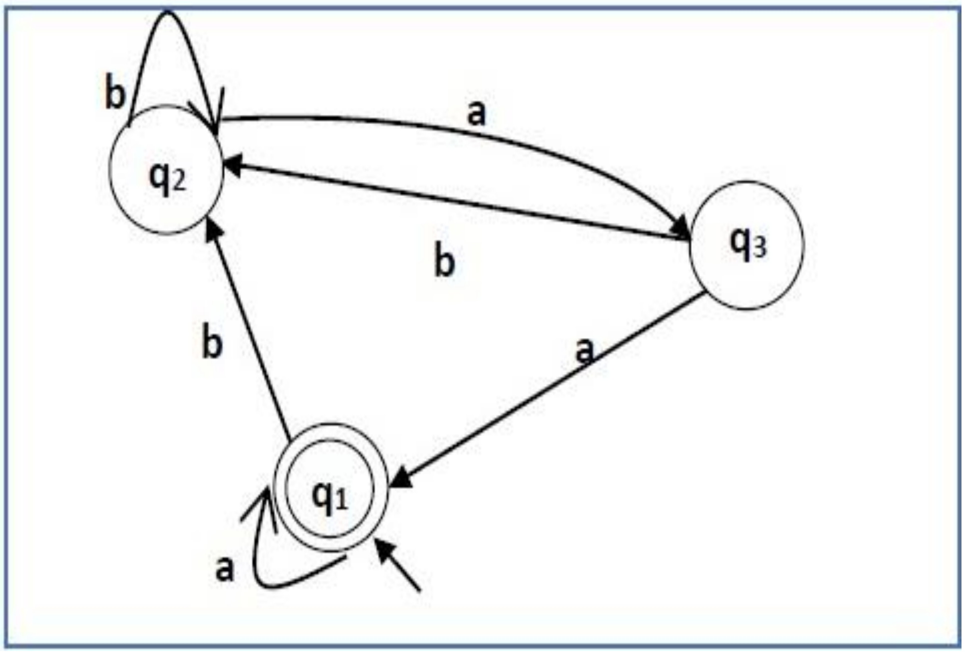
…………………………

$q_n = q_1a_{1n} + q_2a_{2n} + \ldots + q_na_{nn}$

Step 2 − Solve these equations to get the equation for the final state in terms of $\mathbf{a_{ij}}$

Problem

Construct a regular expression corresponding to the automata given below −



Solution:

Here the initial state and final state is $\mathbf{q_1}$.

The equations for the three states q1, q2, and q3 are as follows −

$q_1 = q_1a + q_3a + \varepsilon$ (ε move is because q1 is the initial state)

$q_2 = q_1b + q_2b + q_3b$

$q_3 = q_2a$

Now, we will solve these three equations −

$q_2 = q_1b + q_2b + q_3b$

$= q_1b + q_2b + (q_2a)b$ (Substituting value of q_3)

$= q_1b + q_2(b + ab)$

$= q_1b (b + ab)^*$ (Applying Arden's Theorem)

$q_1 = q_1a + q_3a + \varepsilon$

$= q_1a + q_2aa + \varepsilon$ (Substituting value of q_3)

$= q_1a + q_1b(b + ab^*)aa + \varepsilon$ (Substituting value of q_2)

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

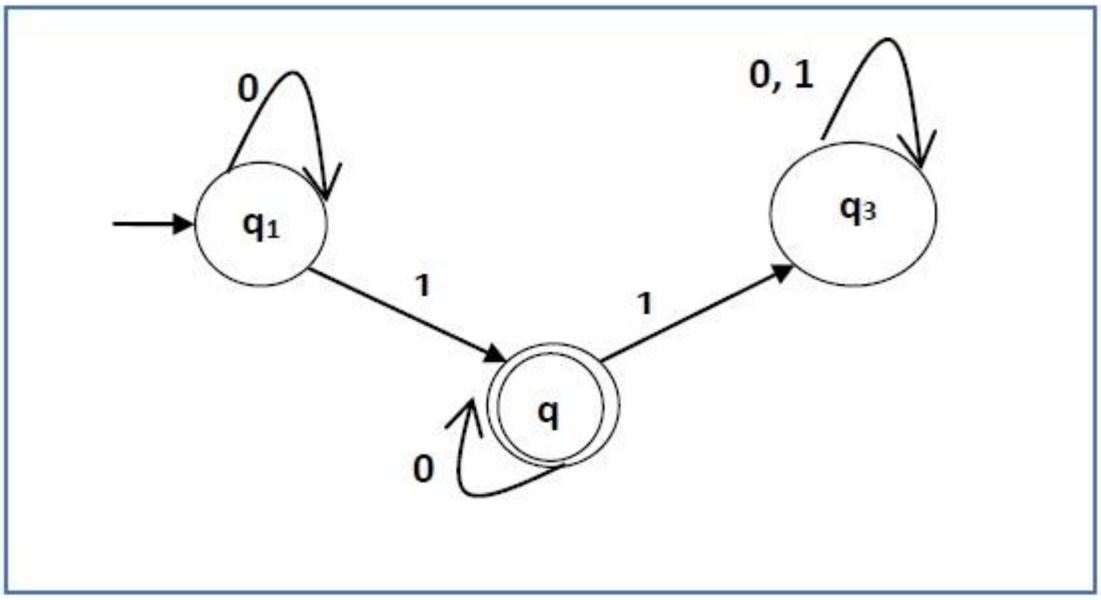
$= \varepsilon (a+ b(b + ab)^*aa)^*$

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

Hence, the regular expression is $(a + b(b + ab)^*aa)^*$.

Problem

Construct a regular expression corresponding to the automata given below −



Solution:

Here the initial state is q_1 and the final state is q_2

Now we write down the equations −

$q_1 = q_1 0 + \varepsilon$

$q_2 = q_1 1 + q_2 0$

$q_3 = q_2 1 + q_3 0 + q_3 1$

Now, we will solve these three equations −

$q_1 = \varepsilon 0^*$ [As, $\varepsilon R = R$]

So, $q_1 = 0^*$

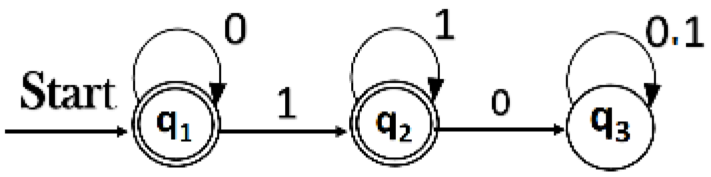
$q_2 = 0^*1 + q_2 0$

So, $q_2 = 0^*1(0)^*$ [By Arden's theorem]

Hence, the regular expression is 0^*10^*.

Problem:

Construct the regular expression for the given DFA



Solution:

Let us write down the equations

$q1 = q1\ 0 + \varepsilon$

Since q1 is the start state, so ε will be added, and the input 0 is coming to q1 from q1 hence we write

State = source state of input × input coming to it

Similarly,

$q2 = q1\ 1 + q2\ 1$

$q3 = q2\ 0 + q3\ (0+1)$

Since the final states are q1 and q2, we are interested in solving q1 and q2 only.

Let us see q1 first

$q1 = q1\ 0 + \varepsilon$

We can re-write it as

$q1 = \varepsilon + q1\ 0$

Which is similar to R = Q + RP, and gets reduced to R = OP*.

Assuming R = q1, Q = ε, P = 0

We get

$q1 = \varepsilon.(0)^*$

$q1 = 0^*$ $(\varepsilon.R^* = R^*)$

Substituting the value into q2, we will get

$q2 = 0^*\ 1 + q2\ 1$

$q2 = 0^*\ 1\ (1)^*$ $(R = Q + RP \rightarrow Q\ P^*)$

The regular expression is given by

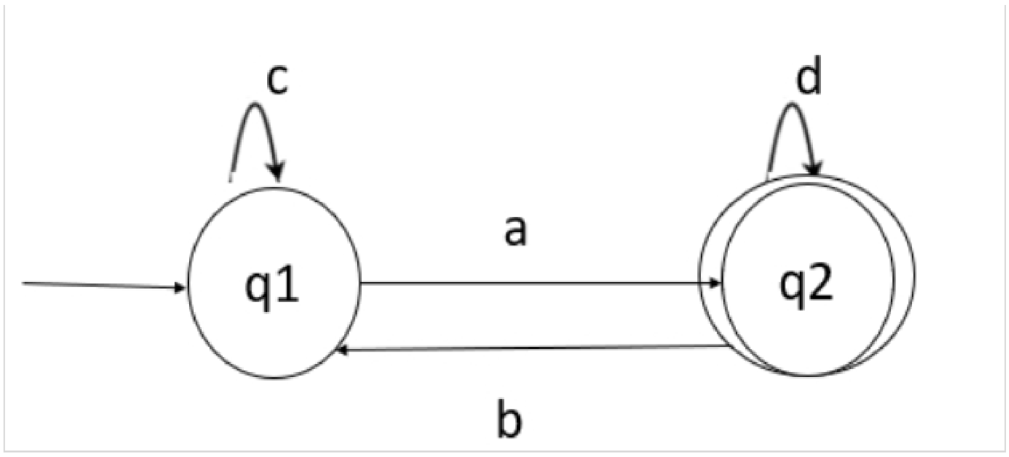
$r = q1 + q2$

$= 0^* + 0^*\ 1.1^*$

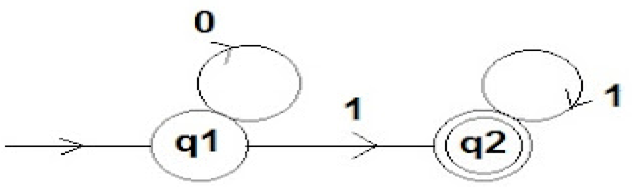
$r = 0^* + 0^*\ 1^+$ $(1.1^* = 1^+)$

Find REs for given FAs

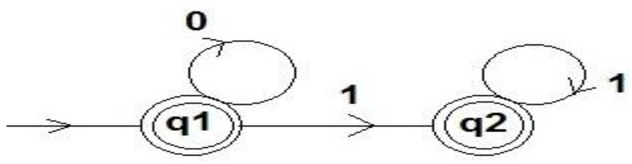
1.



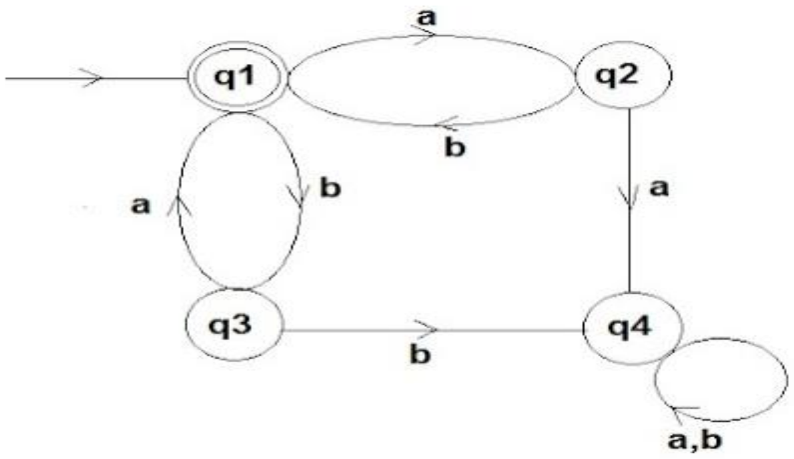
2.



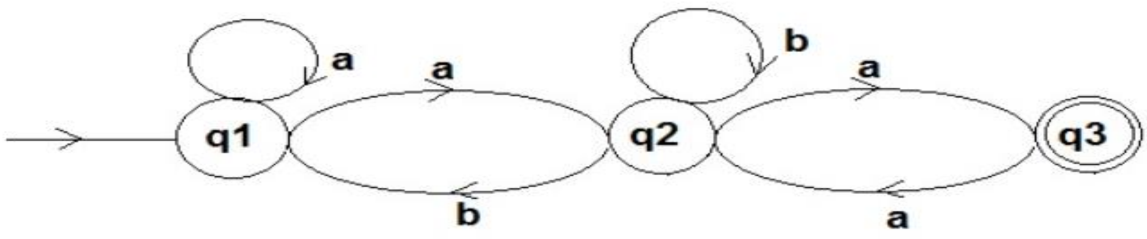
3.



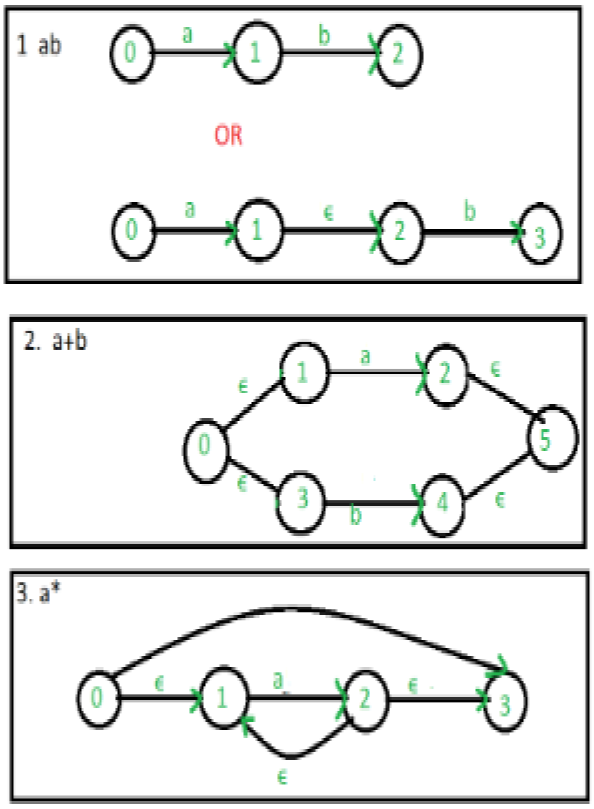
4.



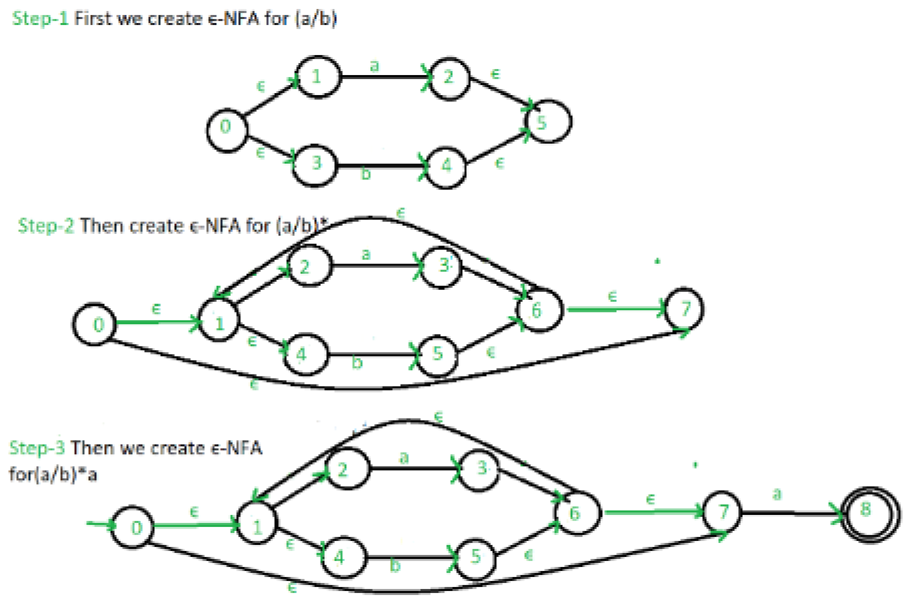
5.



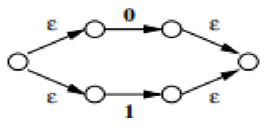
<u>Common regular expressions used in make ∈-NFA:</u>

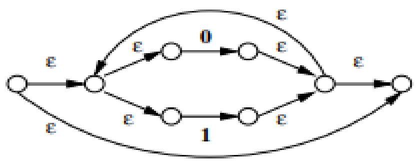


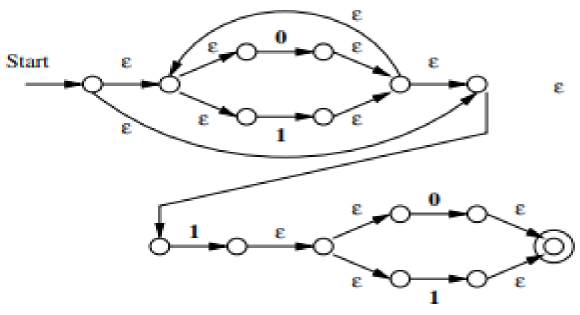
Example: Create a ∈-NFA for regular expression: (a+b)*a


Step-1 First we create ϵ-NFA for (a/b)
Step-2 Then create ϵ-NFA for (a/b)*
Step-3 Then we create ϵ-NFA for(a/b)*a

Example: Create a ∈-NFA for regular expression: (0+1)*1(0+1)


(a)


(b)



(c)

Conversion of RE to DFA

To convert the RE to FA, we are going to use a method called the subset method. This method is used to obtain FA from the given regular expression. This method is given below:

Step 1: Design a transition diagram for given regular expression, using NFA with ε moves.

Step 2: Convert this NFA with ε to NFA without ε.

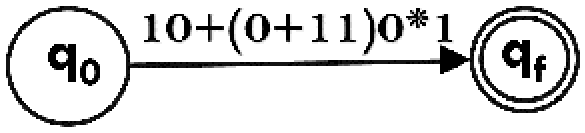
Step 3: Convert the obtained NFA to equivalent DFA.

Example 1:

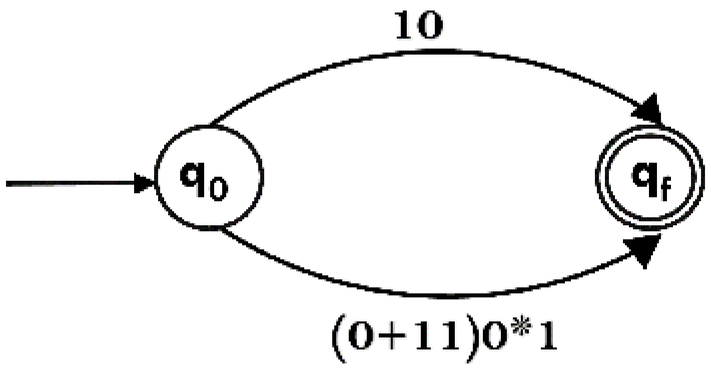
Design a FA from given regular expression 10 + (0 + 11)0* 1.

Solution: First we will construct the transition diagram for a given regular expression.

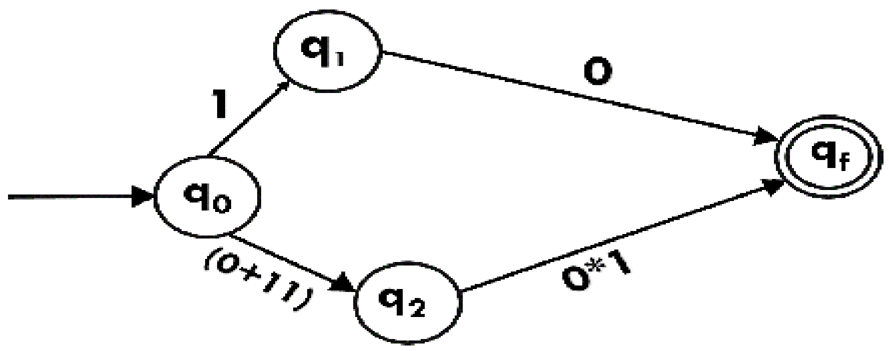
Step 1:



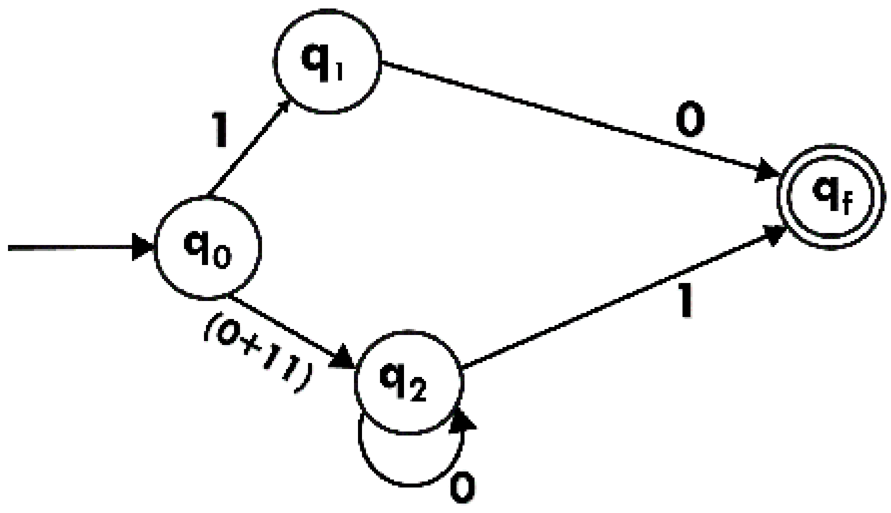
Step 2:



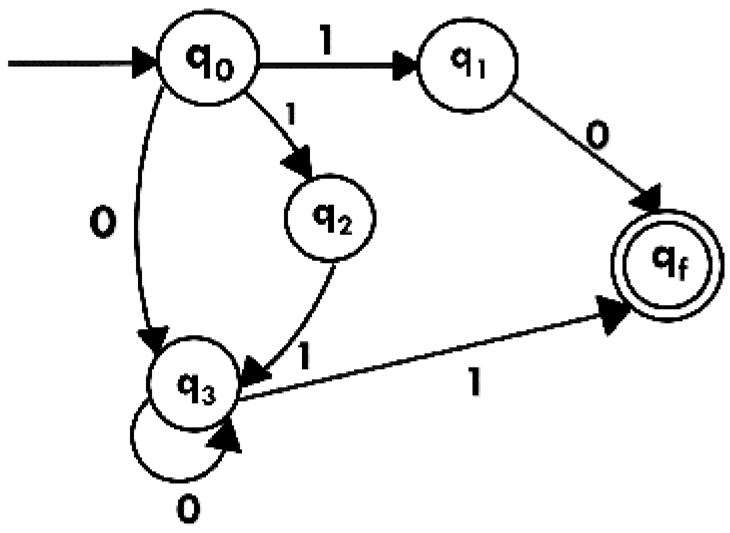
Step 3:



Step 4:



Step 5:



State	0	1
→q0	q3	{q1, q2}
q1	Qf	Φ
q2	Φ	q3

q3	q3	Qf
*qf	Φ	Φ

Now we have got NFA without ε. Now we will convert it into required DFA for that, we will first write a transition table for this NFA.

The equivalent DFA will be:

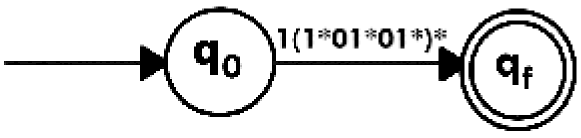
State	**0**	**1**
→[q0]	[q3]	[q1, q2]
[q1]	[qf]	Φ
[q2]	Φ	[q3]
[q3]	[q3]	[qf]
[q1, q2]	[qf]	[qf]
*[qf]	Φ	Φ

Example 2:

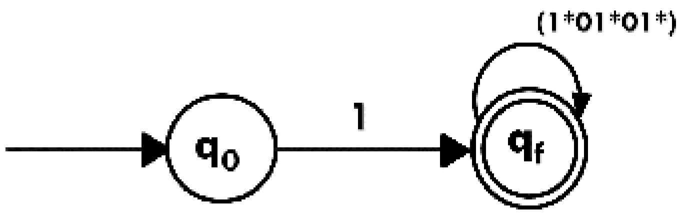
Design a NFA from given regular expression 1 (1* 01* 01*)*.

Solution: The NFA for the given regular expression is as follows:

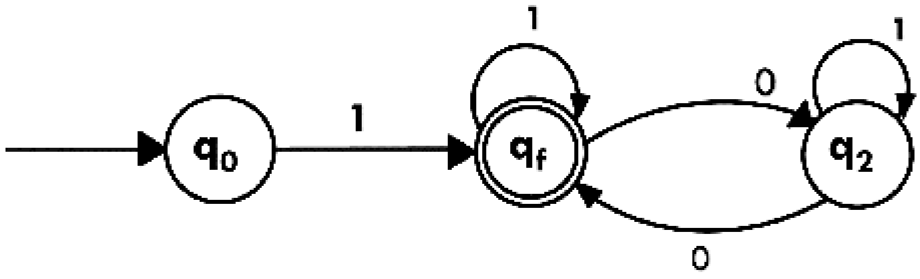
Step 1:



Step 2:



Step 3:



Example 3:

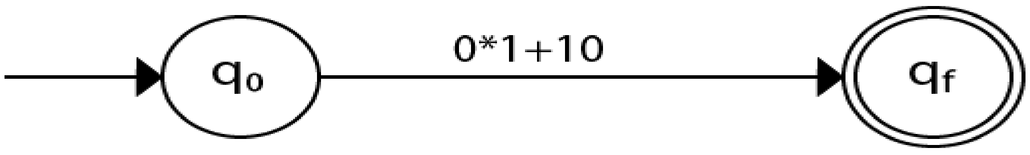
Construct the FA for regular expression 0*1 + 10.

Solution:

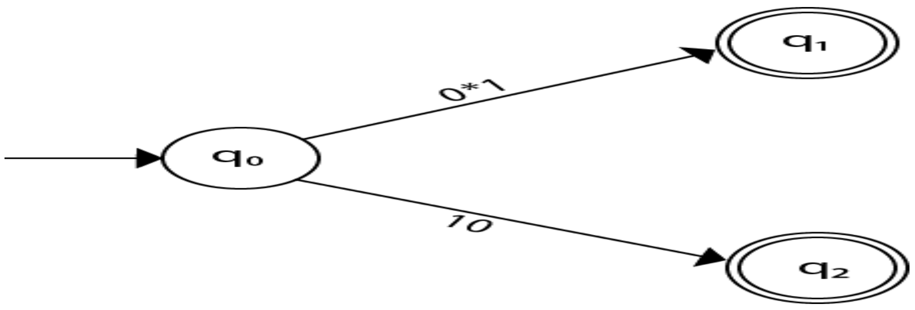
We will first construct FA for R = 0*1 + 10 as follows:

Step 1:

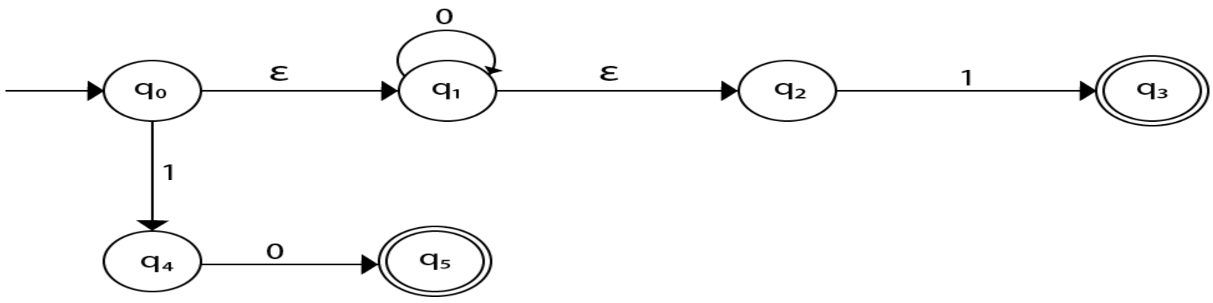
AD



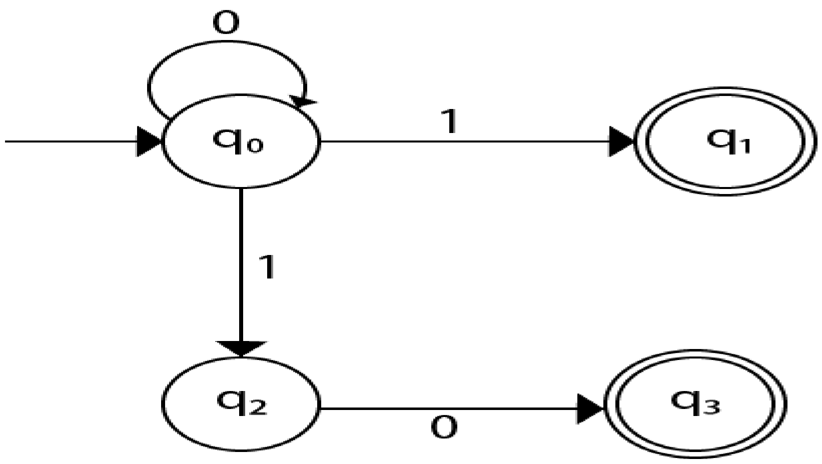
Step 2:



Step 3:

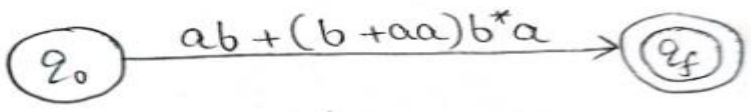


Step 4:

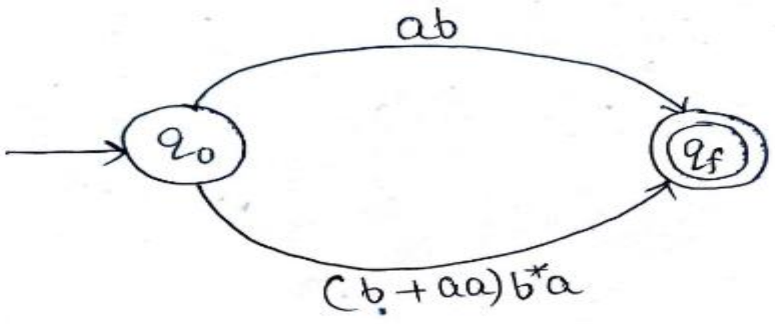


Example: Design a Finite Automata from the given RE **ab + (b + aa)b*a** .
Solution.

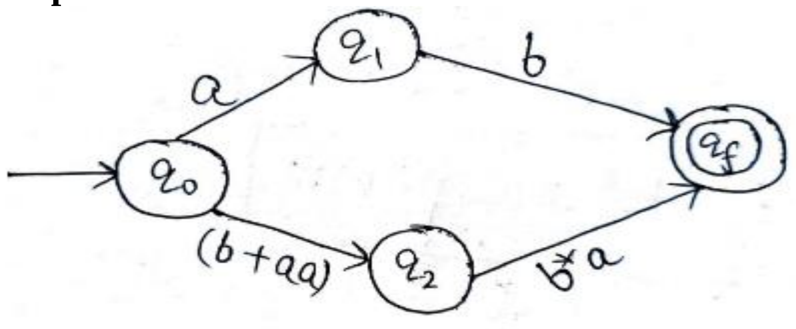
Step 1:



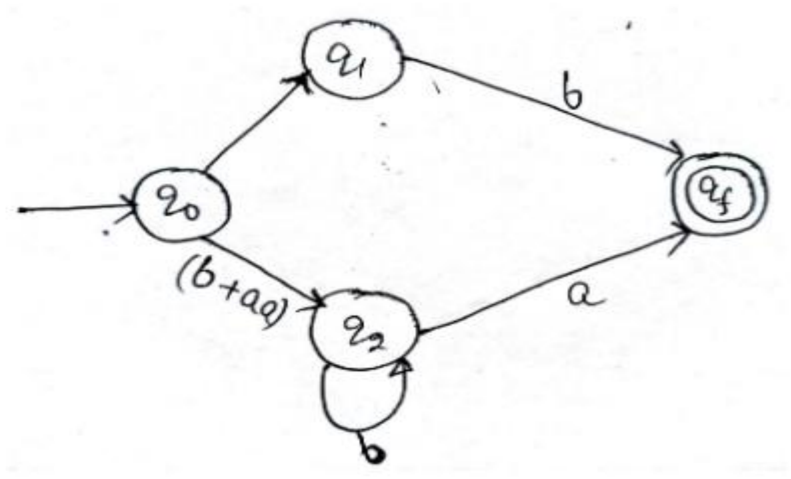
Step 2:



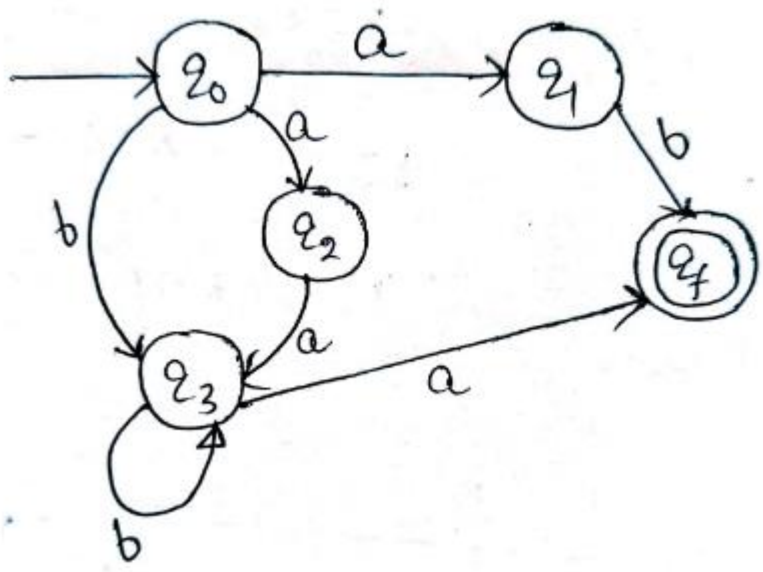
Step 3



Step 4:

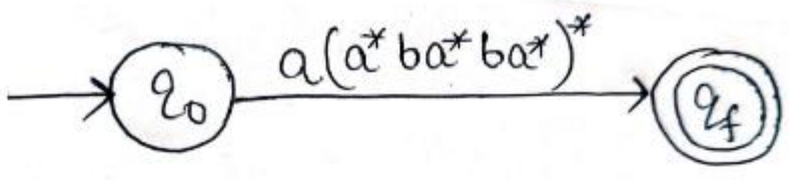


Step 5:

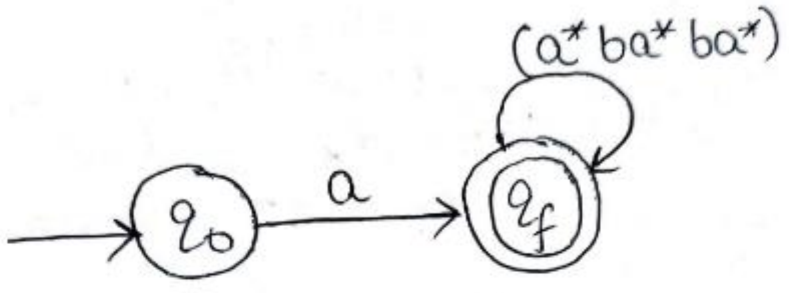


Example: Design a Finite Automata from the given RE a(a*ba*ba*)*

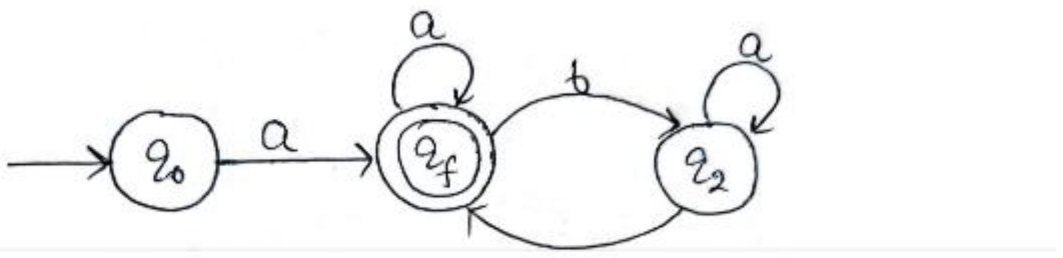
Step 1:



Step 2:



Step 3:



<u>Pumping lemma for Regular languages</u>

- It gives a method for pumping (generating) many substrings from a given string.
- In other words, we say it provides means to break a given long input string into several substrings.
- It gives necessary condition(s) to prove a set of strings is not regular.

Theorem

For any regular language L, there exists an integer n, such that for all w in L

|w|>=n. We can break w into three strings, w=xyz such that.

(1) lxyl < n

(2) lyl > 1

(3) for all k>= 0: the string xy^kz is also in L

Application 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 the Pumping lemma.
- If L does not satisfy the Pumping Lemma, it is not regular.

Steps to prove that a language is not regular by using PL are as follows−

- step 1: We have to assume that L is regular. So, the pumping lemma should hold for L. It has to have a pumping length (say n).All strings longer that n can be pumped |w|>=n.
- step 2: Now find a string 'w' in L such that |w|>=n

  Divide w into xyz such that PL conditions satisfied.

  Select **w** such that **$|w| \geq n$**

  Select **y** such that **$|y| \geq 1$**

  Select **x** such that **$|xy| \leq k$**

  Assign the remaining string to **z.**
- step 3: Find suitable integer i such that $xy^iz \notin L$ for some i.

  = CONTRADICTION.

Example:

Prove that **L = {a^ib^i | $i \geq 0$}** is not regular.

Solution −

- At first, we assume that **L** is regular and n is the number of states.
- Let w = a^nb^n. Thus $|w| = 2n \geq n$.
- By pumping lemma, let w = xyz, where $|xy| \leq n$.
- Let $x = a^p$, $y = a^q$, and $z = a^rb^n$, where $p + q + r = n$, $p \neq 0$, $q \neq 0$, $r \neq 0$. Thus $|y| \neq 0$.
- Let k = 2. Then $xy^2z = a^pa^{2q}a^rb^n$.
- Number of as = $(p + 2q + r) = (p + q + r) + q = n + q$
- Hence, $xy^2z = a^{n+q}\,b^n$. Since $q \neq 0$, xy^2z is not of the form a^nb^n.
- Thus, xy^2z is not in L. Hence L is not regular.

Closure Properties of Regular Sets

- Union
- Intersection
- concatenation
- Kleene closure(or) Star closure (or) Closure
- Complement
- Difference
- Reversal

- Homomorphism
- Inverse homomorphism

1. Union:

The union of two regular set is regular.
If L and M are regular languages, so is L UM.

Proof :

Let us take two regular expressions

$RE_1 = a(aa)^*$ and $RE_2 = (aa)^*$

So, L_1 = {a, aaa, aaaaa,.....} (Strings of odd length excluding Null)

and L_2 ={ ε, aa, aaaa, aaaaaa,.......} (Strings of even length including Null)

$L_1 \cup L_2$ = { ε, a, aa, aaa, aaaa, aaaaa, aaaaaa,.......}

(Strings of all possible lengths including Null)

RE $(L_1 \cup L_2) = a^*$ (which is a regular expression itself)

Hence, proved.

2. Intersection:

The intersection of two regular set is regular.
If L and M are regular languages, so is $L \cap M$

Proof1 :

Let us take two regular expressions

$RE_1 = a(a^*)$ and $RE_2 = (aa)^*$

So, L_1 = { a,aa, aaa, aaaa, ...} (Strings of all possible lengths excluding Null)

L_2 = { ε, aa, aaaa, aaaaaa,.......} (Strings of even length including Null)

$L_1 \cap L_2$ = { aa, aaaa, aaaaaa,.......} (Strings of even length excluding Null)

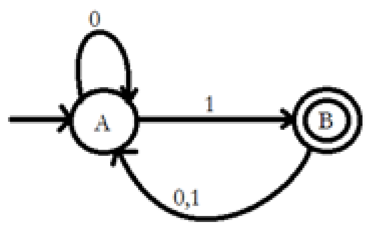
RE $(L_1 \cap L_2) = aa(aa)^*$ which is a regular expression itself.

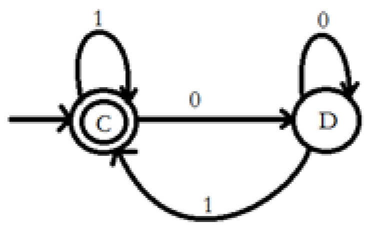
Hence, proved.

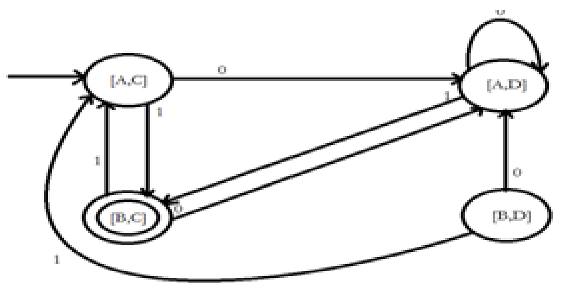
Proof2 :

Let A and B be two DFA's whose regular languages are L and M respectively.

Now, construct C, the product automation of A and B. Make the final states of C be the pairs consisting of final states of both A and B.







<u>3. Complement:</u>

The complement of a regular set is regular.

If L is regular, then L'= Σ* – L is also regular

The complement of a language L (with respect to an alphabet Σ such that Σ* contains L) is Σ* – L Since Σ* is surely regular, the complement of a regular language is always regular

Proof:

Let us take a regular expression −

RE = (aa)*

So, L = {ε, aa, aaaa, aaaaaa, ...} (Strings of even length including Null)

Complement of **L** is all the strings that is not in **L**.

So, L' = {a, aaa, aaaaa, ...} (Strings of odd length excluding Null)

RE (L') = a(aa)* which is a regular expression itself.

Hence, proved.

<u>4. Difference:</u>

The difference of two regular set is regular.

If L and M are regular languages, so is L-M, which means all the strings that are in L , but not in M.

Proof :

Let us take two regular expressions −

RE_1 = a (a*) and RE_2 = (aa)*

So, L_1 = {a, aa, aaa, aaaa, ...} (Strings of all possible lengths excluding Null)

L_2 = { ε, aa, aaaa, aaaaaa,.......} (Strings of even length including Null)

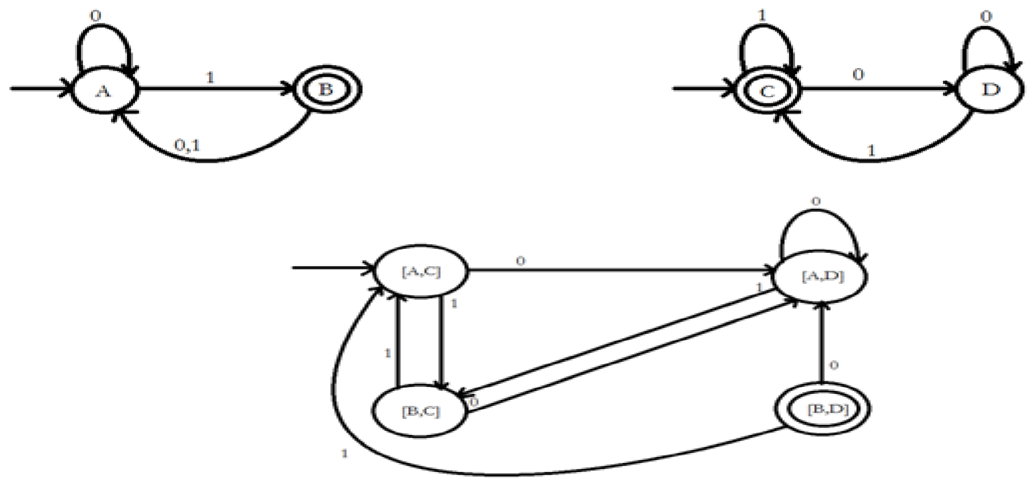
$L_1 – L_2$ = {a, aaa, aaaaa, aaaaaaa, ...}

(Strings of all odd lengths excluding Null)

RE ($L_1 – L_2$) = a (aa)* which is a regular expression.

Hence, proved.

Proof 2: Let A and B be two DFA’s whose regular languages are L and M respectively. Now, construct C, the product automation of A and B. Make the final states of C be the pairs consisting of final states of A, but not of B. The DFA’s A-B and C-D remain unchanged, but the final DFA varies as follows:



<u>5. Reversal:</u>

The reversal of a regular set is regular.

If L is regular then $\mathbf{L^R}$ is also regular

Proof −

We have to prove $\mathbf{L^R}$ is also regular if **L** is a regular set.

Let, L = {01, 10, 11, 10}

RE (L) = 01 + 10 + 11 + 10

L^R = {10, 01, 11, 01}

RE (L^R) = 01 + 10 + 11 + 10 which is regular

Hence, proved.

<u>6.Closure:</u>

The closure of a regular set is regular.
Proof −
If L = {a, aaa, aaaaa, ...} (Strings of odd length excluding Null)
i.e., RE (L) = a (aa)*
L* = {a, aa, aaa, aaaa , aaaaa,……………} (Strings of all lengths excluding Null)
RE (L*) = a (a)*
Hence, proved.

7.Concatenation:

The concatenation of two regular sets is regular.

Proof:

Let RE_1 = (0+1)*0 and RE_2 = 01(0+1)*

Here, L_1 = {0, 00, 10, 000, 010, ...} (Set of strings ending in 0)

and L_2 = {01, 010,011,.....} (Set of strings beginning with 01)

Then, L_1 L_2 = {001,0010,0011,0001,00010,00011,1001,10010,.............}

Set of strings containing 001 as a substring which can be represented by an RE − (0 + 1)*001(0 + 1)*

Hence, proved.

8.Homomorphism

A homomorphism on an alphabet is a function that gives a string for each symbol in that alphabet.

If L is a regular language, and h is a homomorphism on its alphabet, then

h(L) = {h(w) | w is in L} is also a regular language.

Proof: Let E be a regular expression for L.

Apply h to each symbol in E.

Language of resulting RE is h(L)

Example:

Let h(0) = ab; h(1) = ε.

Let L be the language of regular expression 01* + 10*.

Then h(L) is the language of regular expression abε* + ε(ab)*

abε* + ε(ab)* can be simplified.

ε* = ε, so abε* = abε.

ε is the identity under concatenation.

That is, εE = Eε= E for any RE E.

Thus, abε* + ε(ab)* = abε + ε(ab)*

= ab + (ab)*.

Finally, L(ab) is contained in L((ab)*), *so a RE for h(L) is (ab)*

9.Inverse Homomorphism

Definition: Given homomorphism $h : \Sigma* \to \Delta*$ and $L \subseteq \Delta*$, h^{-1} (L) = {w ∈ Σ ∗ | h(w) ∈ L}   h −1 (L) consists of strings whose homomorphic images are in L

Example: Let Σ = {a, b}, and Δ = {0, 1}. Let L = (00 ∪ 1)∗ and h(a) = 01 and h(b) = 10.

h^{-1} (1001) = {ba},

h^{-1} (010110) = {aab}

h^{-1} (L) = (ba)∗

Regular languages are closed under inverse homomorphism, i.e., if L is regular and h is a homomorphism then h −1 (L) is regular.

Tutorial Questions:

1. Define regular expression. Give regular expression for the following languages.
   (i) Strings over the alphabet {a, b} ending with ab
   (ii) Strings over the alphabet {0,1} that contain substring 10
   (iii) Strings over the alphabet {0,1} that contain 1 in the 3rd position from right end
2. What is regular expression? Write the regular expression for the following languages over {0, 1}*
   i) The set of all strings such that number of 0's is odd
   ii) The set of all strings that contain exactly three 1's
   iii) The set of all strings that do not contain 1101

3. Write regular expressions for the following language over the alphabet ∑= {0, 1}
   i) Strings with three consecutive 1's
   ii) Strings with three 1's

4. Write the regular expression for the language L over Σ = {0, 1} such that all the strings
   i) do not contain the substring 01.
   ii) should have at least one 0 and at least one 1.

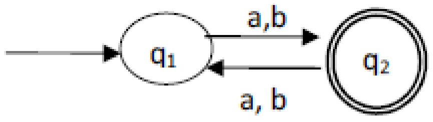
5. Explain pumping lemma for regular languages with the applications of pumping lemma
   (or) State and Explain the pumping lemma for Regular languages..

6. Prove that the language L = $\{(10)^p 1^q \mid p, q \in N, p \geq q\}$ is not regular.

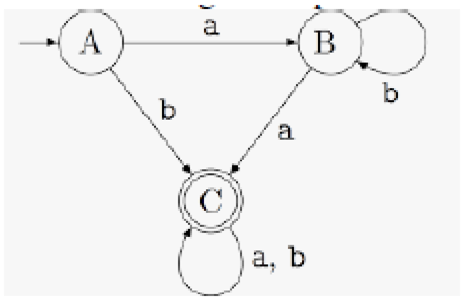
7. Describe the closure properties of Regular sets.
   (or) Summarize the closure properties of regular languages.

8. Explain about the identity rules of Regular Expressions?
   (or) List any ten algebraic laws for regular expressions and explain
   (or) Explain about the Properties of Regular Expressions?

9. Construct a DFA for the Regular expression (0+1)* (00+11) (0+1)* ?
10. Design a NFA for the given regular expression 1 (1* 01* 01*)*.
11. Construct an NFA for r = $(a+bb)^*ba^*$
12. Construct an NFA for the regular expression (a+b)* (aa+bb) (a+b)*
13. Construct a regular expression for the given transition diagram



14. Construct a Regular expression corresponding to the following finite automata.

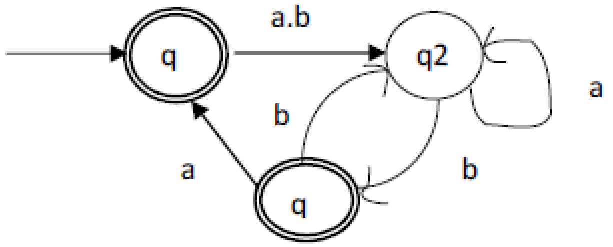


15. Construct a regular expression corresponding to the DFA represented by the below transition table. q1 is both the initial state and final state.

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_1	q_2

16. Construct a NFA equivalent to the regular expression **(10+11)*00.**
17. Construct a NFA equivalent to the regular expression 10(0+11)0*1?
18. Show that L= $\{a^p$ /p is prime$\}$ is Context free?
19. Convert the regular expression (((00)*(11)) + 01)* into an NFA.
20. Prove that the following language L is not regular using pumping lemma
    L = { *w* belongs to {a,b}* | $w = w^R$}

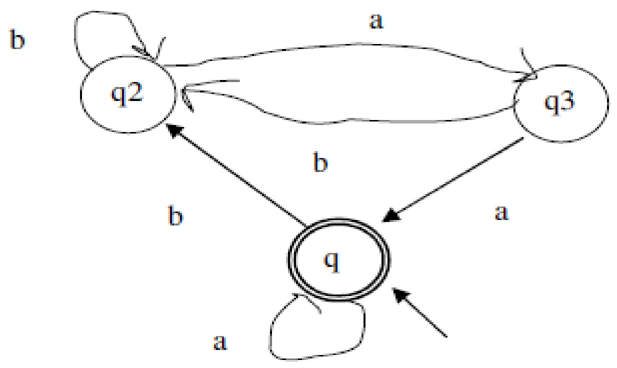
21. Convert the following DFA to a regular expression.



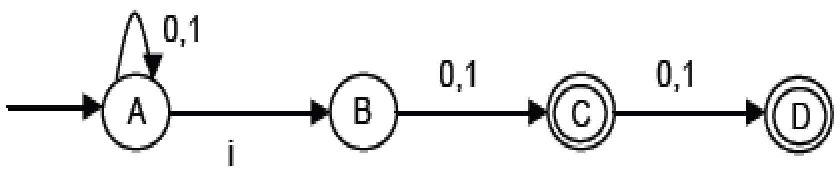
22. Prove that the following language L is not regular using pumping lemma

$L = \{ a^{2n} b^{3n} a^{n} \mid n \geq 0 \}$

23. Construct a Regular expression corresponding to the following finite automata.



24. Convert the following NFA into regular expression.



25. Write the regular expression for the L={w €_{0,1}* | w has no pair of consecutive zeros?

26. (0/1)*011 for this regular expression draw the NFA with ϵ-transitions and convert it into NFA.

27. Give a regular expression that generates the language L over the alphabet Σ={a, b} where each b in the string is followed by exactly one or three a's.

28. Show that $L=\{a^{2n} / n>0\}$ is Regular.

29. What is a regular language? Convert the given regular expression to regular language.

i) (1+ε)(00*1)0*   ii) (0*1*)000(0+1)*   iii) (00+10)*1*(10+00)*

30. What is relationship between finite automata and regular expression? Explain the process of converting DFA to regular expression.