UNIT-III

Regular Expressions

Overview:-

- Regular languages, Expressions and sets
- Identity rules
- properties of regular sets
- Finite automata from Regular expressions
- conversion of FA to Regular expression
- Pumping lemma of Regular sets

Regular languages:-

- A language is called a regular language if some finite automaton recognizes it.
- smallest class of languages which contains all finite languages and closed with respect to union, concatenation and kleen closure.
- Any regular language can be denoted by a regular expressions.

Regular expressions:-

- A regular expression can be described as a sequence of pattern that defines a string.
- Regular expressions are used to match character combinations in strings. string searching algorithm used this pattern to find the operations on a string.
- pattern formed with alphabets and operators like union, Concatenation and kleen.

Definition of Regular expression:-

- Any element of Σ, ε and ϕ are regular expressions. if $a \in \Sigma$, then it can be viewed as a regular expression which is denoted by a.
- union of two regular expressions R and S written as R+S is also a regular expression.

- Concatenation of two regular expressions R and S written as RS is also a regular expression. (2)
- Iteration (or closure) of a regular expression R, written as R^* is also a regular expression.
- If R is a regular expression, then (R) is also a regular expression.
- Regular expression over Σ are precisely those obtained recursively by the application of the above rules once or several times.

Examples:-

- $(a+b) \rightarrow \{a, b\}$
- $(a.b) \rightarrow \{ab\}$
- $(a+b).c \rightarrow \{ac, bc\}$
- $(a+b)(c+d) \rightarrow \{ac, ad, bc, bd\}$
- $a^* \rightarrow \{\varepsilon, a, aa, aaa, aaaa \ldots\}$
- $(a+b)^* \rightarrow \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, \ldots\}$
- $(a+b)^* a \rightarrow \{a, aa, ba, aaa, aba, baa, bba, aaba \ldots\}$
- $ab^* a^* \rightarrow \{a, ab, aba, aaaaa, abbbb, ababa, \ldots\}$
- $(a.b)^* \rightarrow \{\varepsilon, ab, abab, ababab, \ldots\}$

Regular set:-

- Any set that represents the value of the Regular Expression is called a Regular set

Regular Expressions	Regular set
$(0+10^*)$	$L = \{0, 1, 10, 100, 1000, 10000, \ldots\}$
$(0^* 10^*)$	$L = \{1, 01, 10, 010, 0010, \ldots\}$
$(0+\varepsilon)(1+\varepsilon)$	$L = \{\varepsilon, 0, 1, 01\}$

$(a+b)^*$	set of strings of a's and b's of any length including the null string. so, $L = \{\varepsilon, a, b, aa, ab, bb, ba, aaa \ldots\}$
$(a+b)^*abb$	Set of Strings of a's and b's ending with the string abb. so $L = \{abb, aabb, babb, aaabb, ababb, \ldots\}$
$(11)^*$	Set consiting of even number of 1's including empty string, so $L = \{\varepsilon, 11, 1111, 111111, \ldots\}$
$(aa)^*(bb)^*b$	Set of strings consisting of even number of a's followed by odd number of b's, so $L = \{b, abb, aabbb, aabbbbb, aaaab, aaaabbb, \ldots\}$
$(aa+ab+ba+bb)^*$	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 $L = \{aa, ab, ba, bb, aaab, aaba, \ldots\}$

Operations:-

- Let X is a Regular Expression denoting the language L(X) and Y is a Regular Expression denoting the language L(Y), then
- union: $X+Y$ is a Regular Expression corresponding to the language $L(X) \cup L(Y)$ where $L(X+Y) = L(X) \cup L(Y)$
- Concatenation: $X \cdot Y$ is a Regular Expression corresponding to the language $L(X) \cdot L(Y)$ where $L(X \cdot Y) = L(X) \cdot L(Y)$
- Kleen closure: X^* is a Regular Expression corresponding to the language $L(X^*)$ where $L(X^*) = (L(X))^*$

Identity rules:-

Let R, P, L, Q are regular expressions, the following identities hold-

- $\phi^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $RR^* = R^*R$
- $R^*R^* = R^*$
- $(R^*)^* = R^*$
- $(PQ)^*P = P(QP)^*$
- $(a+b)^* = (a^*b^*)^* = (a^*+b^*)^* = (a+b^*)^* = a^*(ba^*)^*$
- $R+\phi = \phi+R = R$ (The identity for union)
- $R\varepsilon = \varepsilon R = R$ (The identity for concatenation)
- $R+R = R$ (Idempotent law)
- $L(M+N) = LM+LN$ (Left distributive law)
- $(M+N)L = ML+NL$ (Right distributive law)
- $\varepsilon + RR^* = \varepsilon + R^*R = R^*$

Regular Expression to Finite Automata:-

- Every language L defined by a regular expression R is also defined by finite automata M

  i.e, $L = L(R) \rightarrow L = L(M)$
- Design a transition diagram for given regular expression, using NFA with ε moves.
- Convert this ε-NFA with to NFA without ε.
- Convert the obtained NFA to equivalent DFA.

properties of Regular sets :-

• property 1 :- The union of two regular set is regular.

Ex:- $RE1 = a(aa)^*$ and $RE_2 = (aa)^*$

$L_1 = \{a, aaa, aaaaa, ----\}$

$L_2 = \{\varepsilon, aa, aaaa, aaaaaa, ----\}$

$L_1 \cup L_2 = \{\varepsilon, a, aa, aaa, aaaa, aaaaa, aaaaaa, -----\}$

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

• property 2 :- The intersection of two regular set is regular.

Ex:- $RE_1 = a(a^*)$ and $RE_2 = (aa)^*$

$L_1 = \{a, aa, aaa, aaaa, ----\}$

$L_2 = \{\varepsilon, aa, aaaa, aaaaaa, -----\}$

$L_1 \cap L_2 = \{aa, aaaa, aaaaa, ----\}$

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

• property 3 :- The complement of a regular set is regular.

Ex:- $RE = (aa)^*$

$L = \{\varepsilon, aa, aaaa, aaaaaa, ----\}$ (str of even length)

complement of L is all the strings that is not in L.

$L' = \{a, aaa, aaaaa, ---\}$ (str of odd length)

$RE(L') = \varepsilon\, a(aa)^*$ which is a regular expression itself.

• property 4 :- The difference of two regular set is regular.

Ex:- $RE_1 = a(a^*)$ and $RE_2 = (aa)^*$

$L_1 = \{a, aa, aaa, aaaa, ----\}$ (string of All length)

$L_2 = \{\varepsilon, aa, aaaa, aaaaaa, ----\}$ (string of even length)

$L_1 - L_2 = \{a, aaa, aaaaa, aaaaaaa, ----\}$ (strings of all odd Lengths excluding Null)

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

• property 5 :- The reversal of a regular set is regular.

Ex:- 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.

- Property 6:- The closure of a regular set is regular.

Ex:- $L = \{a, aaa, aaaaa, ----\}$ i.e, $RE(L) = a(aa)^*$

$L^* = \{a, aa, aaa, aaaa, aaaaa, ------\}$ (strings of all length excluding NULL)

$RE(L^*) = a(a)^*$ which is regular.

- Property 7:- The concatenation of two regular sets is regular.

Ex:- $RE_1 = (0+1)^*0$ and $RE_2 = 01(0+1)^*$

$L_1 = \{0, 00, 10, 000, 010, ---\}$ (set of strings ending in 0)

$L_2 = \{01, 010, 011, ----\}$ (set of strings beginning with 01)

$L_1 L_2 = \{001, 0010, 0011, 0001, 00010, 00011, 1001, 10010, ----\}$

set of strings containing 001 as a substring which can be represented by an $RE = (0+1)^* 001 (0+1)^*$.

Problems:-

1) write the regular expression for the language accepting all combinations of a's, over the set $\Sigma = \{a\}$.

Sol:- $RE = a^*$

2) write the regular expression for the language accepting all combinations of a's except the null string, over the set $\Sigma = \{a\}$

Sol:- $RE = a(a)^*$ (or) $RE = a^+$

3) write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over $\Sigma = \{0, 1\}$

Sol:- $RE = 1(0+1)^*0$

4) write the regular expression for the language starting with a but not having consecutive b's.

Sol:- $RE = (a+ab)^*$

5) 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.

Sol:- $RE = a^* b^* c^*$

6) write the regular expression for the language over $\Sigma = \{0,1\}$ having even length of the string.

Sol:- $RE = (00+01+10+11)^*$

7) write the regular expression for the language having a string which should have atleast one 0 and atleast one 1

Sol:- $RE = [(0+1)^* 0 (0+1)^* 1 (0+1)^*] + [(0+1)^* 1 (0+1)^* 0 (0+1)^*]$

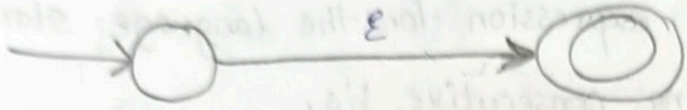
8) write the regular expression for the language L over $\Sigma = (0,1)$ such that all the string do not contain the substring 01.

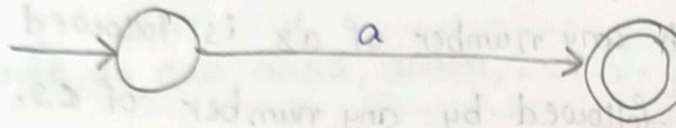
Sol:- $RE = 1^* 0^*$

Regular Expression to Finite Automata:- (1)

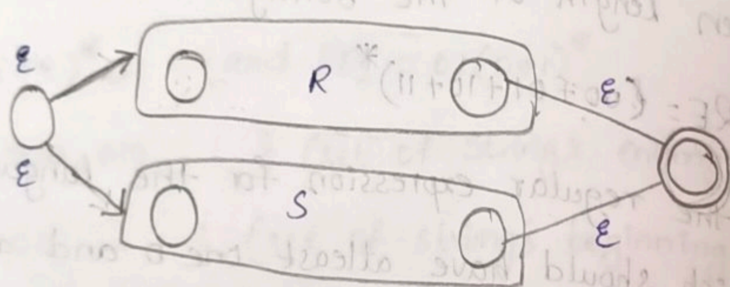
- Every language L defined by a regular expression R is also defined by finite automata M
  i.e., $L = L(R) \rightarrow L = L(M)$
- Design a transition diagram for given regular expression using NFA with ε moves.
- Convert this ε-NFA with to NFA without ε.
- Convert the obtained NFA to equivalent DFA.

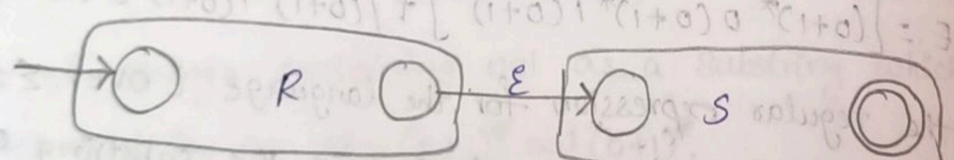
Basis for construction of FA from RE :-

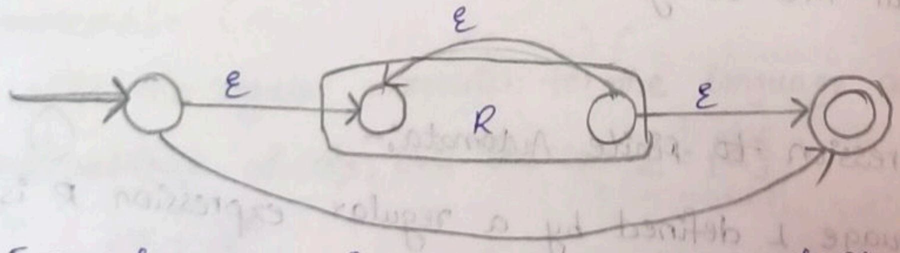




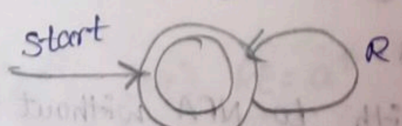
1. Expression ε : the language of the FSA is {ε}
2. Expression φ : φ is the language of FSA
3. Expression a; the language of the FSA is {a}.





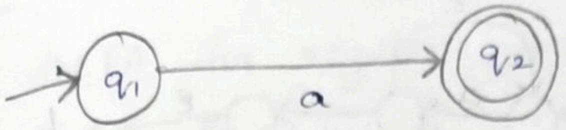


- Expression R+S for two regular expressions R and S.
- Expression RS for two regular expressions R and S.
- Expression R* for regular expression R



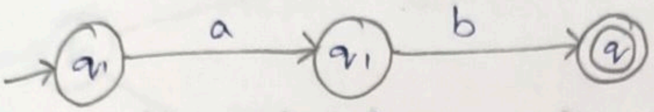
Some basic Regular expression:-

case1:- For a regular expression 'a', we can construct the following FA -



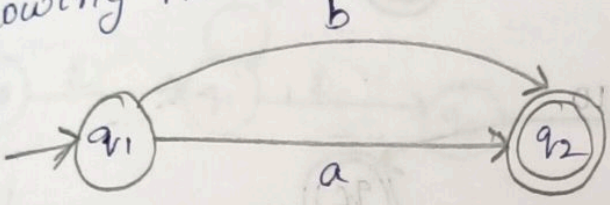
Finite automata for RE = a

case 2:- For a regular expression 'ab' we can construct the following FA -



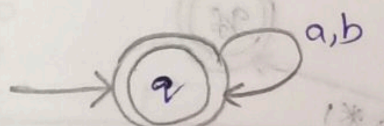
Finite automata for RE = ab

case 3:- For a regular expression (a+b), we can construct the following FA -



Finite automata for RE = (a+b)

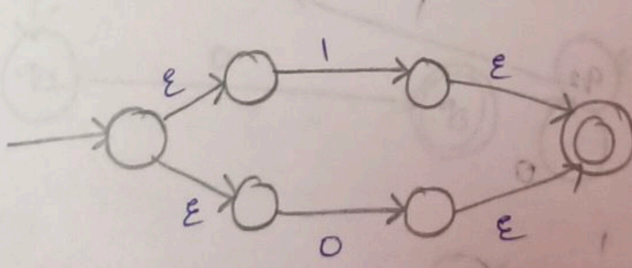
case 4:- For a regulare expression $(a+b)^*$, we construct the following FA -



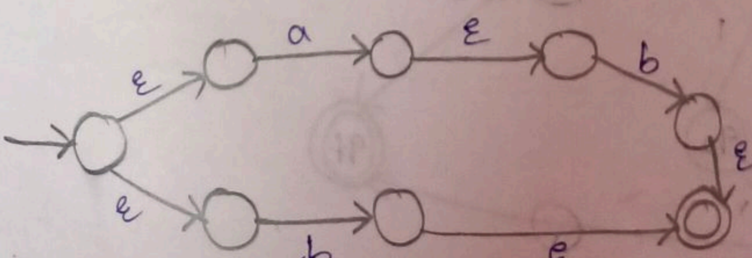
Finite automata for RE = $(a+b)^*$

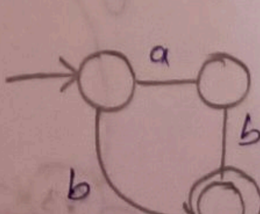
Problems:-

1) 1+0

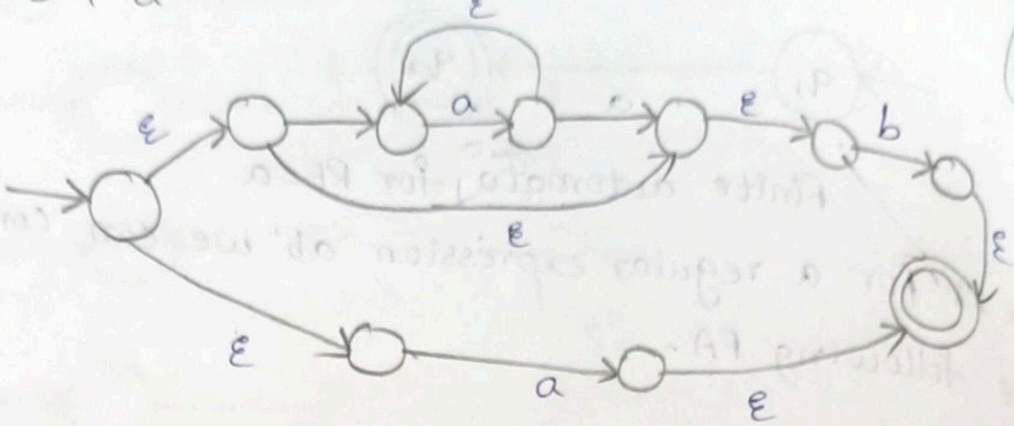


2) ab+b





3) $a^*b + a$

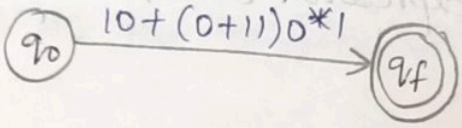


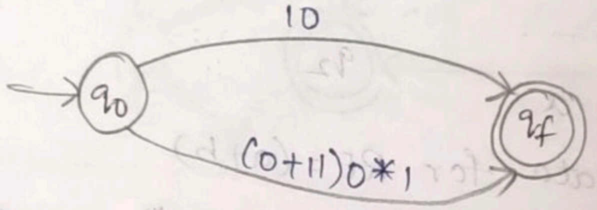
(4)

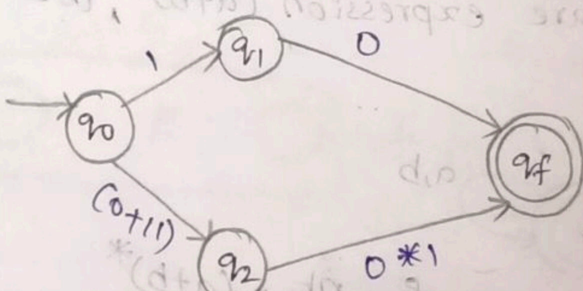
Example :-

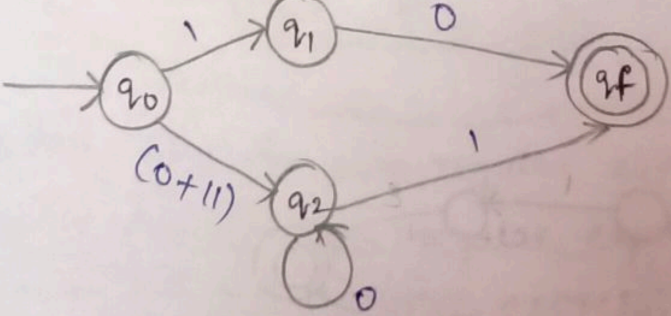
Design a FA from given regular expression $10+(0+11)0^*1$

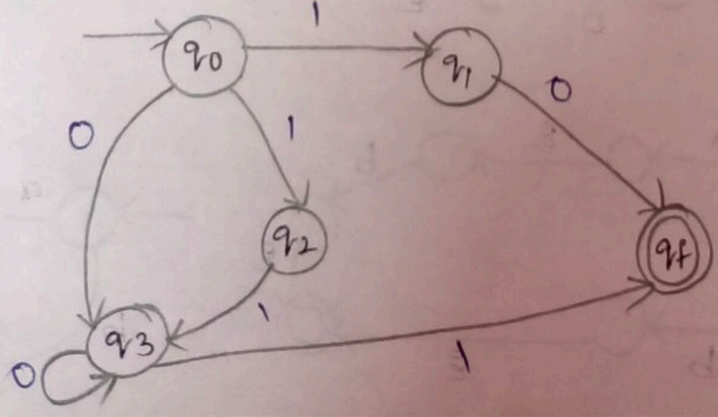
Sol :-







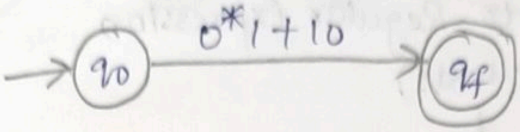


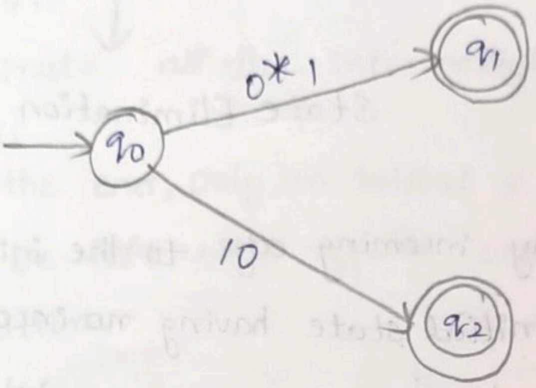


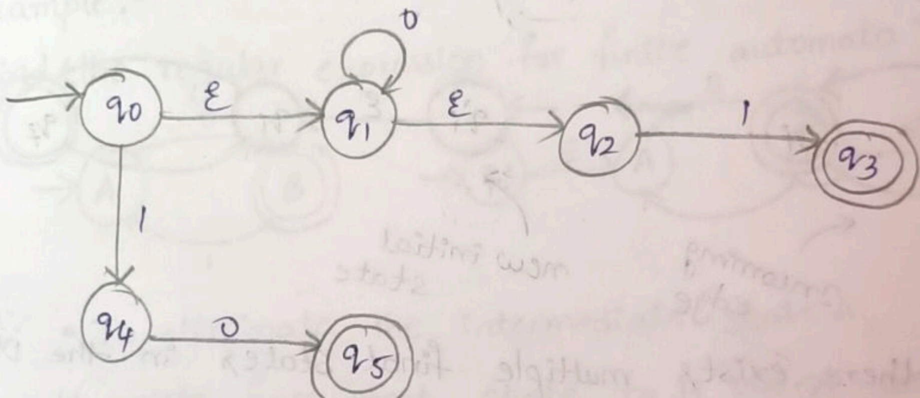
Example:-

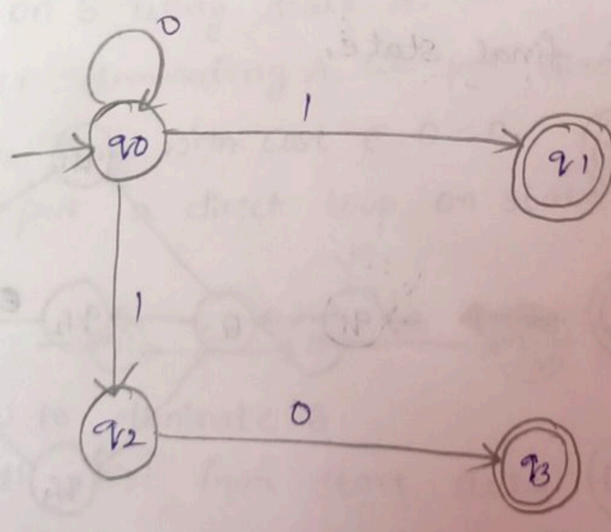
Construct the FA for regular expression 0^*1+10

Sol:-









RE from finite Automata :-

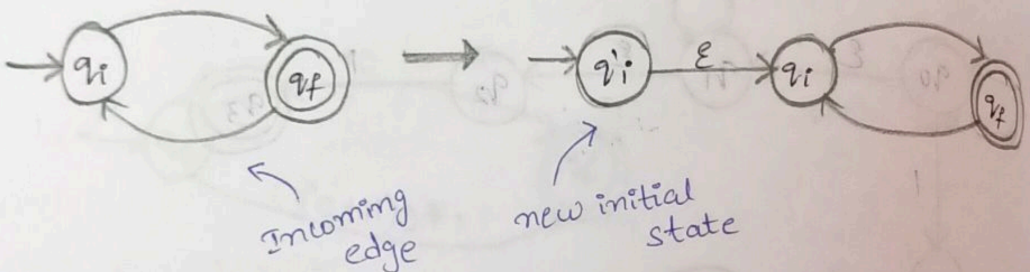
Converting DFA to Regular Expression
(Methods)

Arden's Method

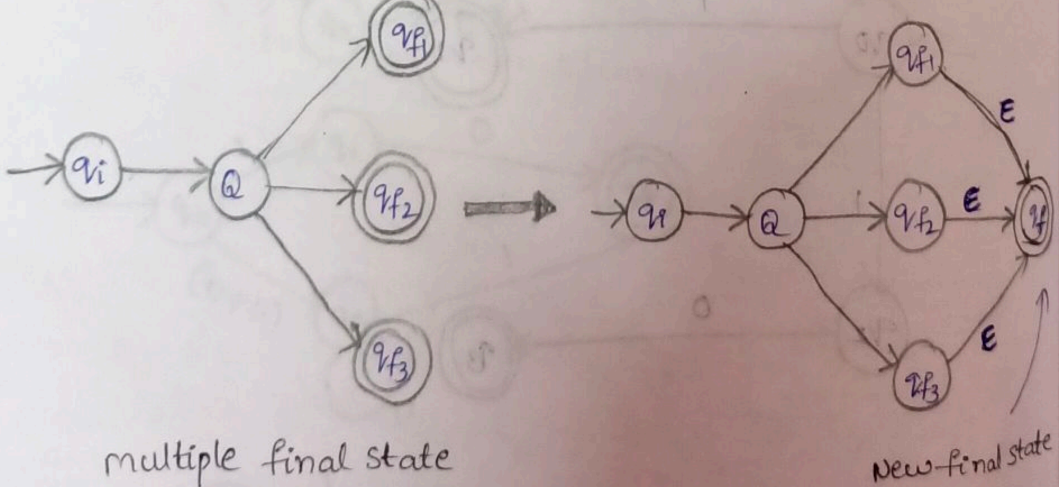
State Elimination Method

State Elimination method :-

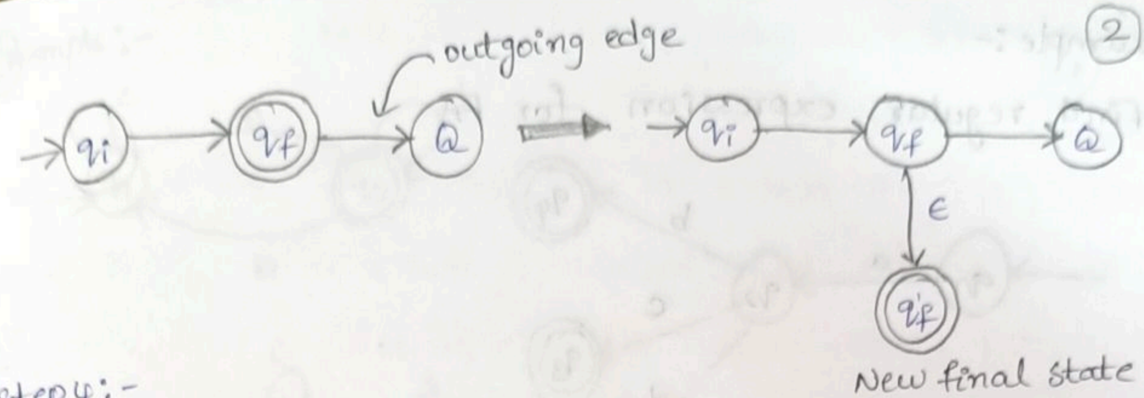
- Step 1: If there exists any incoming edge to the initial state, then create a new initial state having no incoming edge to it.



- Step 2: If there exists multiple final states in the DFA then convert all the final states into non-final states and create a new single final state.



Step 3:- If there exists any outgoing edge from the final state, then create a new final state having no outgoing edge from it.

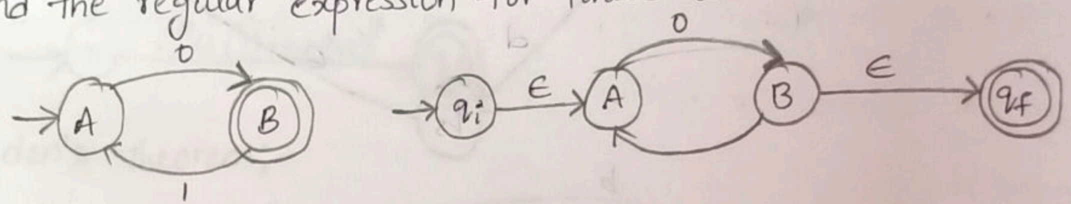


- Step 4:-

- Eliminate all the intermediate states one by one in any order.
- At the end, only an initial state going to the final state will be left with regular expression as cost of this transition.

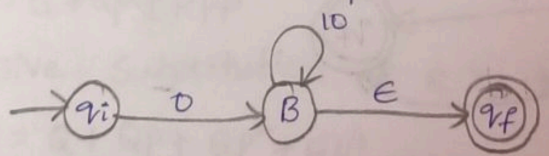
Example:-

Find the regular expression for finite automata?

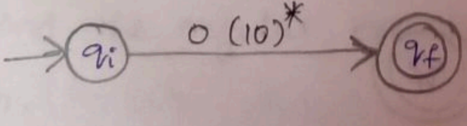


Sol:- • To eliminate the intermediate state A

- path exists from start state to B via A and there is a loop on B using state A.
- After eliminating A, we put direct transition from state start to B with cost $\epsilon . 0 = 0$.
- we put a direct loop on state B having cost $1.0 = 10$



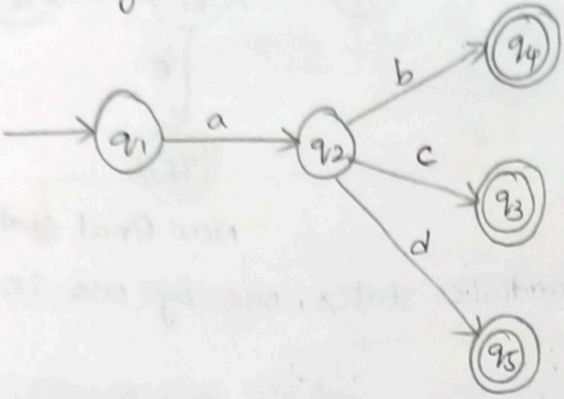
- Now to eliminate B
  - path exist from start state to final state via B
  - After eliminating B, we put direct transition from start to final start with cost $0.(01)^*.\epsilon = 0(01)^*$



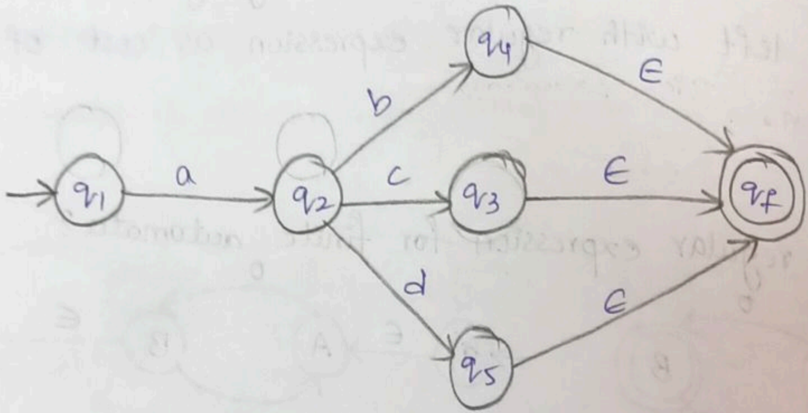
∴ Regular Expression $= 0(10)^*$

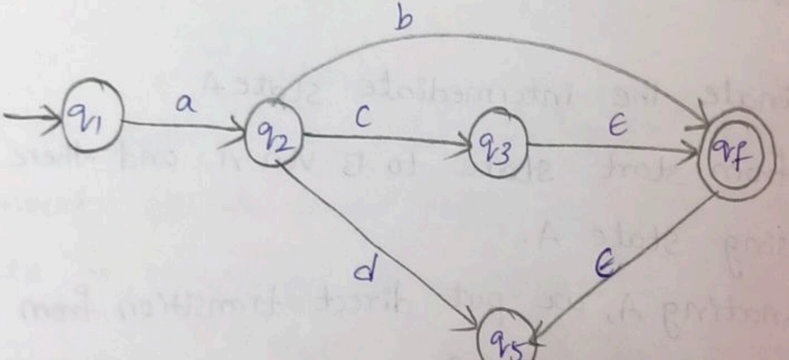
Example :-

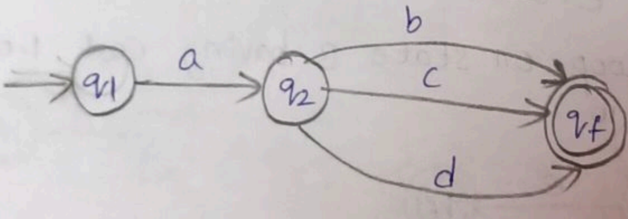
- Find regular expression for FA

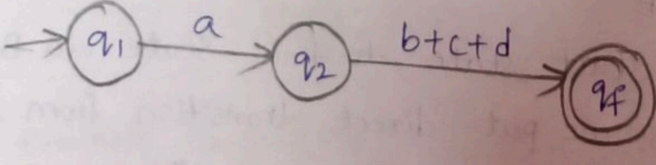


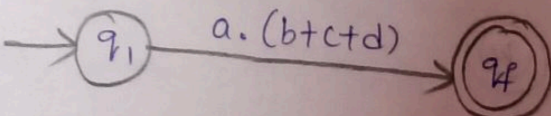
Sol:-





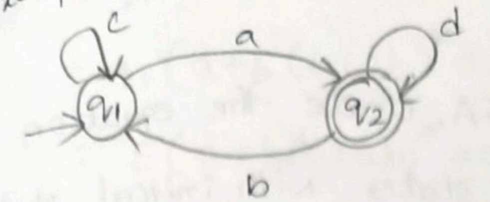




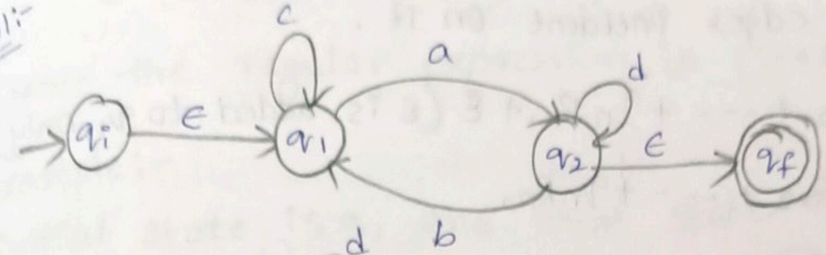


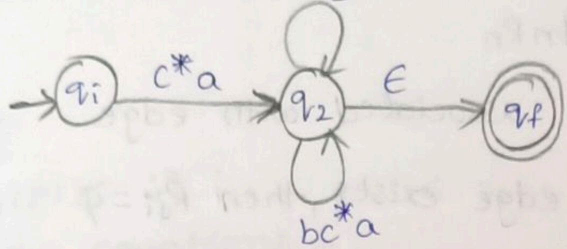
∴ Regular Expression $= a(b+c+d)$

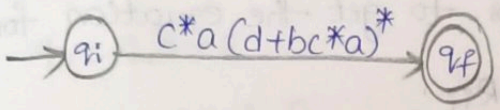
Example:-



sol:-







Arden's Theorem:-

- 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 substituting R

$= Q + QP + RPP$

recursive substitution of R in the equation that results -

$R = Q + QP + QP^2 + QP^3 \ldots$

$R = Q(\varepsilon + P + P^2 + P^3 + \ldots)$

$R = QP^*$ [As P^* represents $(\varepsilon + P_1 + P_2 + P_3 + \ldots)$]

Hence proved.

- To find the regular expression of a given FA using Arden's theorem.
- Assumptions for Applying Arden's Theorem
  - transition diagram must not have NULL transitions.

- It must have only one initial state

Arden's method:-

5

- Step 1: For all states of the FA, create the equation in the following form (assume n states with initial state q_1) based on the edges incident on it.

  - $q_1 = q_1R_{11} + q_2q_{21} + \cdots + q_nR_{n1} + \varepsilon$ (ε is added to q_1 only)
  - $q_2 = q_1R_{12} + q_2R_{22} + \cdots + q_nR_{n2}$

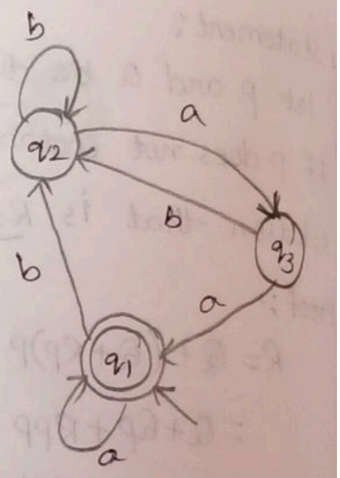
  - $q_n = q_1R_{1n} + q_2R_{2n} + \cdots + q_nR_n$

  R_{ji} represents input symbol associated with edge from q_j to q_i if no such edge exists, then $R_{ji} = \varphi$

- Step 2: Solve these equations to get the equation for the final state in terms of R_{ij}.

Example:-

Initial state and final state is q_1.



Sol:-

Step 1:-

Equations for the three states q_1, q_2 and q_3 are as follows -

- $q_1 = q_1a + q_3a + \varepsilon$
- $q_2 = q_1b + q_2b + q_3b$
- $q_3 = q_2a$

- 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)

$= q_1 a + q_1 b(b+ab^*)aa + \varepsilon$ (Substituting value of q_2)

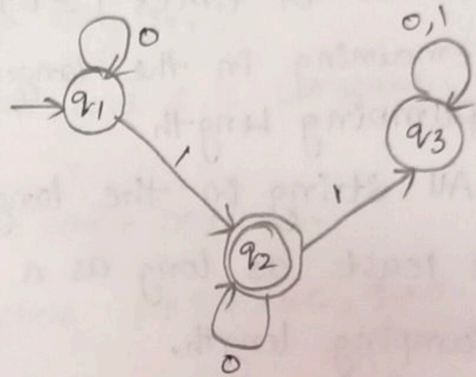
$= q_1(a + b(b+ab)^* aa) + \varepsilon$

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

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

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

Example:-
Initial state is q_1 and final state is q_2



sol:-

step1:-
the equations -

$q_1 = q_1 0 + \varepsilon$

$q_2 = q_1 1 + q_2 0$

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

Solve these three equations -

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

  So, $q_1 = 0^*$

- $q_2 = 0^* 1 + q_2 0$ // substitute q_1)

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

Hence, the regular expression is $0^* 1 0^*$

Write the regular expression?

- $R_1 = \{0^n 1^n : n \geq 0\}$
- $R_2 = \{W : W$ has equal number of 0's and 1's$\}$
- $R_3 = \{W1W : W$ is string over $\Sigma\}$
- $R_4 = \{WW^R : W$ is string over $\Sigma\}$

The Pumping lemma - Idea :-

- a tool to prove a language is NOT Regular

Intuition:-

~~Intuitions~~ There is a property that ALL Regular Languages have. If a Language can be shown to NOT have this property, then that Language is NOT Regular.

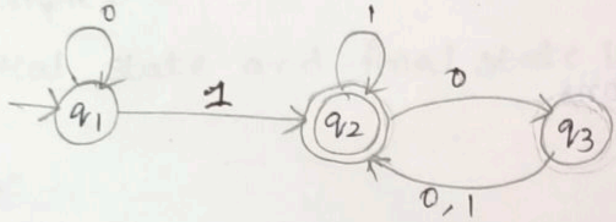
- pumping

-Repeating a section of the string an arbitrary number of times (≥ 0), with the resulting string remaining in the language.

- Pumping Length

- All string in the language can be pumped if they are at least as long as a certain special value, called the Pumping length.

Example :-



Intuitive Examples :-

$101 \Rightarrow 1(01)^*$

$000001 \Rightarrow (0)^* 00001$

$110100 \Rightarrow 1(1)^* 0100$

Pumping Lemma (for Regular languages)

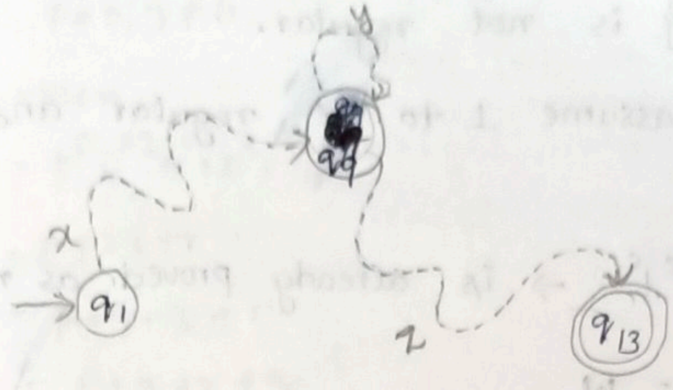
- If L be a Regular language, then there exists a constant c (the pumping length) such that for every string W in L, $|W| \geq c$

then W may be divided into 3 pieces, $W = xyz$ Satisfying the following conditions:

i) $|y| > 0$

ii) $|xy| \leq c$

iii) For all $i \geq 0$, the string xy^iz is also in L. ③



Example :-

Prove that $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

- At first, we assume that L is regular and c is constant.
- Let $W = a^c b^c$. Thus $|W| = 2c \geq c$
- By pumping lemma, let, $W = xyz$, where $|xy| \leq c$.
- Let $x = a^p$, $y = a^q$, and $z = a^r b^c$, where $p+q+r = c$, $q \neq 0$.

Thus $|y| \neq 0$ and $|xy| \leq c$.

- Let $i = 2$. Then $xy^2z = a^p a^{2q} a^r b^c$.
- Number of as $= (p+2q+r) = (p+q+r)+q = c+q$
- Hence, $xy^2z = a^{c+q} b^n$, since $q \neq 0$, xy^2z is not of the form $a^c b^c$.
- Thus, xy^2z is not in L. Hence L is not regular.

Method to prove that a language L is not regular ①

- First assume that L is regular
  so, the pumping lemma must 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 z.
  - select i such that the resulting string is not in L.
- Hence L is not regular.

Q) Prove $L = \{w | w$ is string having equal no. of 0's & 1's $\}$ is not regular. (2)

Sol:- First we assume L to be regular and c is the constant.

Let $w = 0^c 1^c$ → is already proved as not regular.

Let $0(01)^c 1 = w$

Thus $|w| = 2c + 2 \geq c$

$w = xyz$ where $|xy| \leq c$

Let $x = 0(01)^p$

$y = (01)^q$ $\quad |y| \neq 0$

$z = (01)^r 1$

Let $i = 2$

the $xy^2z = 0(01)^p (01)^{2q} (01)^r 1$

$= p + 2q + r = (p+q+r) + q$

$= c + q$ → is also some constant

$= c_1$

So, this still in the form of $0(01)^{c_1} 1$

→ But this is not holding true for that string of L, L is not regular.

Q) $R3 = \{ ww | w$ is string over $\{0,1\} \}$ is not regular.

Sol:- Let $w = 0^c 1 0^c 1$, $|w| \geq c$

$w = xyz$ where $|xy| \leq c$

$|y| > 0$

$x = 0^p$

$y = 0^q$

$z = 0^r 1 0^c 1$

where $p+q+r=c$

$p \neq 0, q \neq 0$

Let $i=2$, then

$xyyz = 0^p 0^{2q} 0^r 1 0^c 1$

here $= p+2q+r$

$= p+q+q+r$

$= p+q+r+q$

$(p+q+r=c)$

$= c+q = c_1$

$xy^2z = 0^{c_1} 1 0^c 1$

Q) $R_4 = \{0^i 1^j \mid i > j\}$ is not regular

Sol:- $w = 0^{c+1} 1^c$

$w = xyz$ where $|xy| \leq c$

$|y| > 0$

$x = 0^p$

$y = 0^q$

$z = 0^r 1^c$

$p+q+r = c+1$

$p \neq 0, q \neq 0$

Let $i=0$ (pump down)

then $xy^0z = xz$

$xz = 0^p 0^r 1^c$

$p+r \leq c$ since $q \neq 0$

Thus, xz, number of 0s is lessthan or equal to the number of 1's

$\therefore$ It is not in the form, so, it is not regular.

Q) $R_5 = \{ww^R \mid w$ is string over $\{0,1\}\}$ is not regular.

Sol:- Let $w = 01^c1^c0$, $|w| \geq c$

$w = xyz$ where $|xy| \leq c$

$|y| > 0$

$x = 01^p$

$y = 1^q$

$z = 1^r1^c0$

where $p+q+r=c$

$p \neq 0, q \neq 0, r \neq 0$

Let $i = 2$

$xyyz = 01^p1^{2q}1^r1^c0$

$= 01^{p+2q+r}1^c0$

$p+2q+r=c$

$[p+q+r]+q = c+q$

$xyyz = 01^{c+q}1^c0$

Thus, it is not in the form, Hence not regular.