**RE to FA, Equivalence of RE, FA to RG, RG to FA**

**REGULAR EXPRESSIONS (RE)**

**The regular expressions are used to represent certain sets of strings in an algebraic fashion. It describes the languages accepted by finite state automata.**

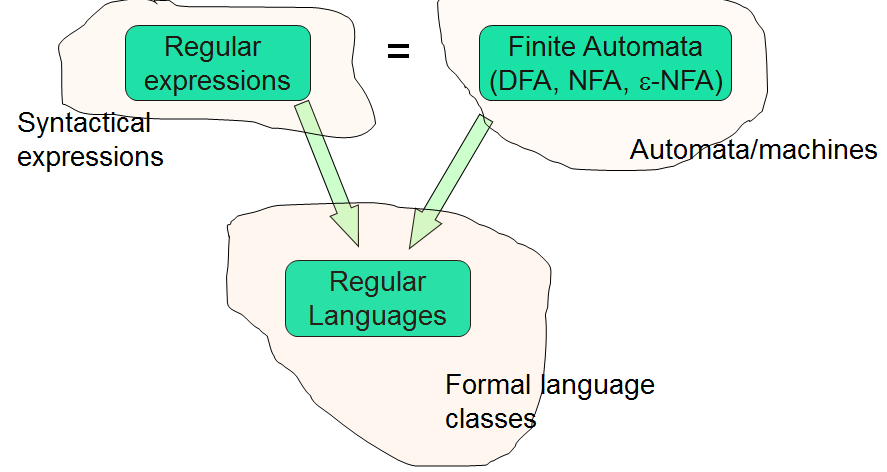
**Formal recursive definition of regular expressions over Ʃ is as follows:**

**1. Any terminal symbol (i.e. an element of Ʃ), Ʌ and ϕ are regular expressions. Input a in Ʃ as a regular expression, is denoted by a.**

**2. The union of two RE R1 and R2, written as R1 + R2, is also a regular expression.**

**3. The concatenation of two regular expressions R1 and R2, written as R1R2, is also a regular expression.**

**4. The iteration (or closure) of a regular expression R written as R\*, is also a regular expression.**

****

**Regular Set:- Any set represented by a regular expression is called a regular set.**

**If for example, a, b Ʃ L. then (i) a denotes the set { a }, (ii) ab denotes { ab }, (iii) a + b + c denotes { a, b, c }, (iv) a\* denotes the set { Ʌ, a, aa, aaa, ... } (v) a+ denotes the set { a, aa, aaa, ... }, (vi) (a + b)\* denotes { a, b }\*.**

**The set represented by R is denoted by L(R).**

**Let R1 and R2 denote any two regular expressions. Then a string in L( R1 + R2 ) is a string either from R1 or a string from R2; L(R1 + R2) = L(R1) U L(R2), { ab, bc } ab + bc**

**(ii) a string in L( R1R2 ) is a string from R1 followed by a string from R2,**

**L(R1R2) = L(R1)L(R2) abbc**

**(iii) a string in L(R\*) is a string, generated by concatenating n elements for some n ≥ 0.**

**L(R\*) = (L(R))\* Also, L(R\*) = (L(R)\* = L(ϕ) = ϕ, L(a) = {a}.**

**CONSTRUCTION OF FINITE AUTOMATA EQUIVALENT TO A REGULAR EXPRESSION:-**

The method is called the **subset method** for constructing a finite automaton equivalent to a given regular expression which involves two steps.

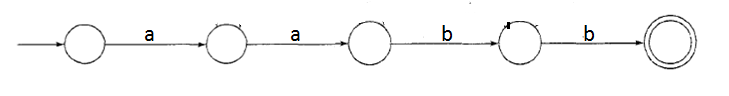
**Step 1:- Construct a transition graph (transition system) equivalent to the given regular expression using Ʌ - moves.**

**Step 2 :- Construct the transition table for the transition graph obtained in step1 and construct the equivalent DFA**.

**Designing Finite Automata from Regular Expression:-**

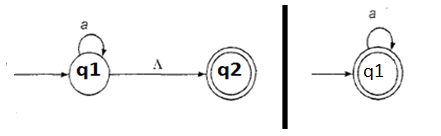
L1 = {aabb}

Finite automaton will be as follows-

****

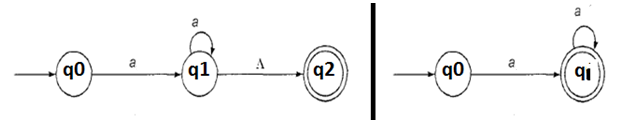
L2 = {an | n ≥ 0}

The language of the given RE is-{ ε , a, aa, aaa, ..........}

****

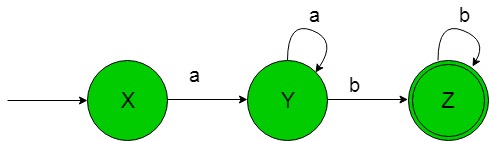
L3 = {an | n ≥ 1}

The language of the given RE is-{a, aa, aaa, ..........}



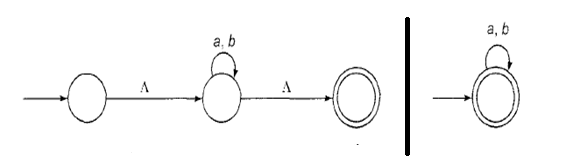
L4 = {anbm | n, m ≥ 1}

The language of the given RE is- {ab, aab, abb, aaaabb, ..........}



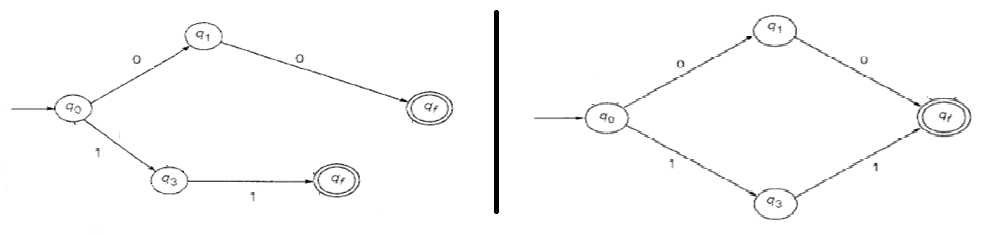
L5 = (a+b)\*

The language is-{ε, a, aa, aaa, aabbb, ......} and its finite automata will be as follows-

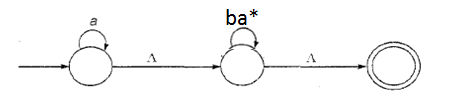


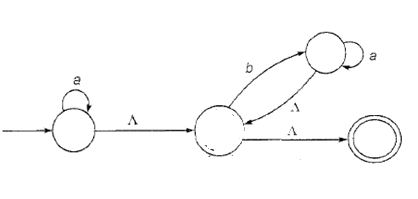
L6 = ab + ba

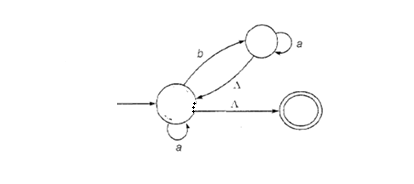
The language of the given RE is -{ ab, ba } and its finite automata will be as follows-

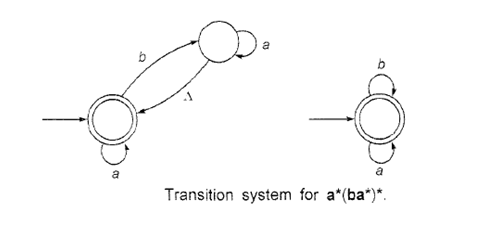


L7 = a\*(ba\*)\* , Its finite automata will be as follows-









**Example 1 :- Construct the finite automaton equivalent to the regular expression**

**a\*b(a + b)\***

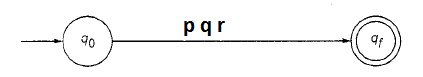
**Sol:-**

**Steps for construction of finite automaton**

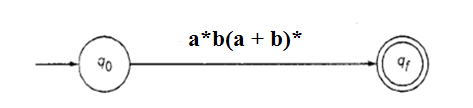
**(1) Assume given r.e. is = pqr, where p,q,r, are r.e. and defined as**

**p = a\*, q = b, r = (a + b)\***

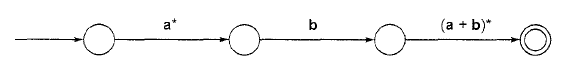
**(2) Draw a transition system using r.e. = pqr, which is as follows**

****

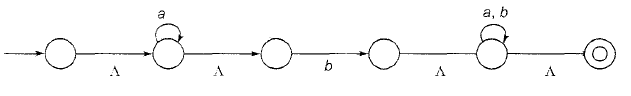
**(3) Substitute the value of pqr on the labels (input)**

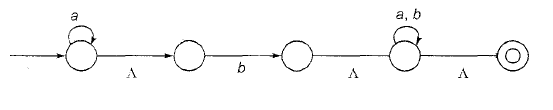
****

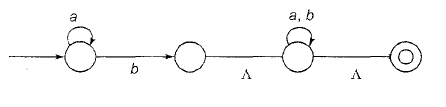
**(4) Write only one r.e. corresponding to one label. Since here we have three r.e (p, q, r), so we expand states for making three labels (inputs).**

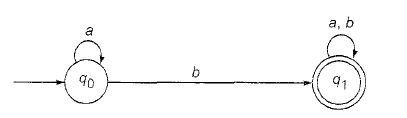
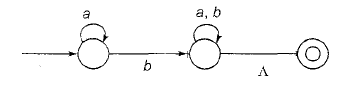
****

**(5) Repeat the steps from (1) to (4) for each labelled r.e.**

****

****

****

****

**Fig. finite automaton equivalent to a\*b (a+b)\***

**Transition Table for Example**

|  |  |  |
| --- | --- | --- |
| **State / Ʃ** | **Input a** | **Input b** |
| **q0** | **q0** | **q1** |
|  | **q1** | **q1** |

Q. Construct a transition system corresponding to the regular expressions

**(i) a\*ab (ii) a\*bb\*a (iii) a + bb (iv) ab+c+ bab\*a**

**(v) a(a + b)\* + ab (vi) a\* + ab(a+ b)\* (vii) (ab + c\*)\*b**

**Example 2 :- Construct the finite automaton equivalent to the regular expression**

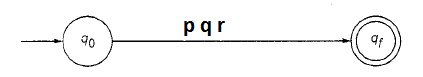
**(0 + 1)\*(00 + 11)(0 + 1)\***

**Sol:- Steps for construction of finite automaton**

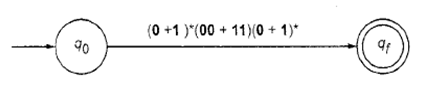
**(1) Assume given r.e. is = pqr, where p,q,r, are r.e. and defined as**

**p = (0 + 1)\*, q = (00 + 11), r = (0 + 1)\***

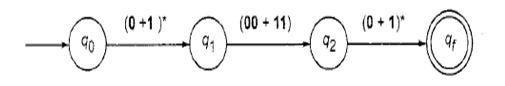
**(2) Draw a transition system using r.e. = pqr, which is as follows**

****

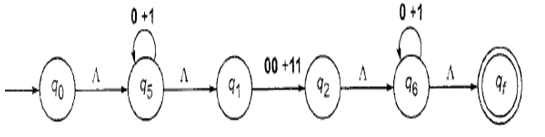
**(3) Substitute the value of pqr on the labels (input)**

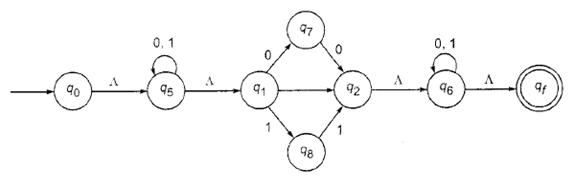
****

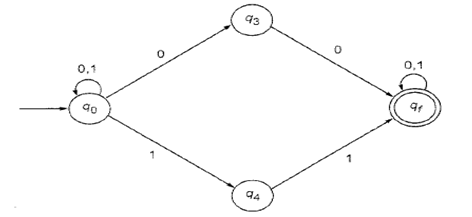
**(4) Write only one r.e. corresponding to one label. Since here we have three r.e (p, q, r), so we expand states for making three labels (inputs).**

****

**(5) Repeat the steps from (1) to (4) for each labelled r.e.**

****

****

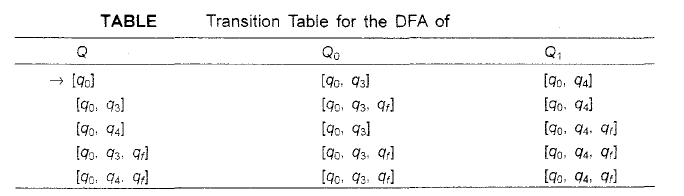
****

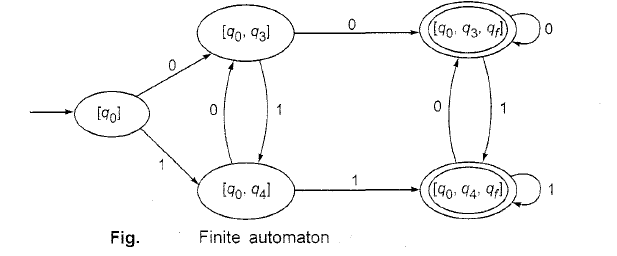
**Fig. finite automaton equivalent to (0 + 1)\*(00 + 11)(0 + 1)\*.**

**Transition Table (NDFA)for Example**

|  |  |  |
| --- | --- | --- |
| **State / Ʃ** | **Input 0** | **Input 1** |
| **q0** | **q0, q3** | **q0, q4** |
| **q3** | **qf** | **-** |
| **q4** | **-** | **qf** |
|  | **qf** | **qf** |

**Since we get NDFA transition table, which needs to convert into equivalent DFA table.**

****

****

**Q. Construct the finite automaton equivalent to the regular expression**

**10 + (0 + 11)0\*1**

**Sol:- ???**

**Equivalence of Two REGULAR EXPRESSIONS**

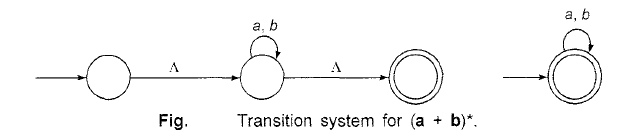
The regular expressions P and Q are equivalent iff they represent the same set or the corresponding finite automata are equivalent. To prove the equivalence of P and Q,

1. Prove that the sets P and Q are the same. (for nonequivalence, find a string in one set but not in the other.)
2. Or use the identities to prove the equivalence of P and Q.
3. Or construct the corresponding Finite automaton M and M' and prove that M and M' are equivalent. (for nonequivalence, prove that M and M’ are not equivalent.)

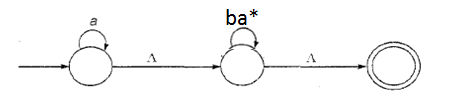
*The method to be chosen depends on the problem*.

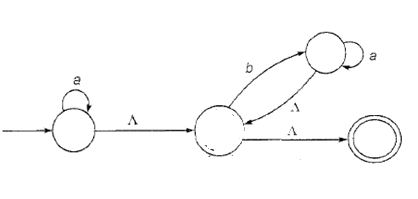
Example:- Check (a + b)\* = a\*(ba\*)\*.

Sol :- Draw FA for both r.e, then check whether both FA’s are same or not)



Transition system for a\*(ba\*)\* is constructed as follows.





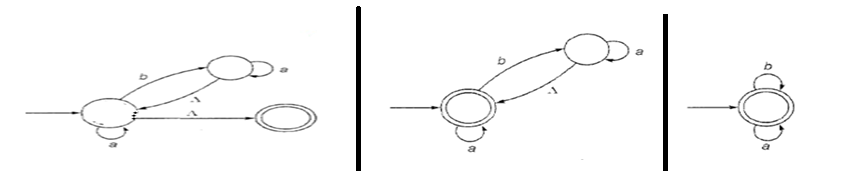


Fig. Transition system for a\*(ba\*)\*

**Q1. Check whether aa\* + bb\* is the same as (a + b)\*? (Hint:- Draw FA for both r.e.)**

**Q2. (0\*1\*)\* is the same as**

**(a) (0 + 1)\* (b) (01)\* (c) (10)\* (d) none of these**

**Q3. (a + a\*)\* is equivalent to**

**(i) a(a\*)\* (ii) a (iii) aa\* (iv) none of these**

**Q4. a\*(a + b)\* is equivalent to**

**(i) a\* + b\* (ii) (ab)\* (iii) a\*b\* (iv) none of these**

**Q5. Check whether the regular expression (a + b)\*c\* is the same as a\*(b + c)\*.**

**Q6. Check which REs are equivalent to RE (a+b)\* ??**

**L1 = a\* + b\* L2 = (a\*+b\*)\* L3 = (a\*b\*)\***

**L4 = (a\*+b)\* L5 = (a+b\*)\* L6 = b\*(ab\*)\***

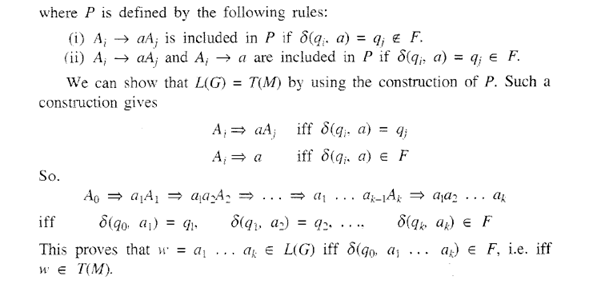
**Q7. Is L = { a2n | n ≥ 1} regular?**

**Sol:- Draw Finite automaton.**

**Construction of a Regular Grammar Generating T(M) for a given DFA M**

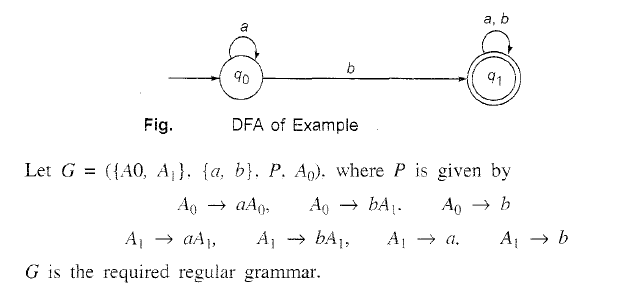
Let M = ({q0, q1,…..qn}, Ʃ, δ, q0, F). If w is in T(M), then it is obtained by concatenating the labels corresponding to several transitions, the first from q0 and the last terminating at some final state.

Corresponding Grammar G is shown by four tuples as, ( {A0,A1,….An}, Ʃ, P, A0 ),



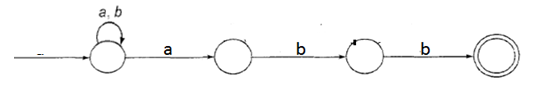
Example 1 :- Construct a regular grammar G generating the regular set represented by **a\*b(a + b)\*.**

Sol:- We construct the DFA corresponding to the expression. Grammar G is shown by four tuples as, ( {A0,A1,….An}, Ʃ, P, A0 ) and following is finite automaton.

****

Q1 :- Construct a regular grammar G generating the regular set for (a + b)\*abb.

* Sol:- We construct the DFA corresponding to r.e.



* Grammar G is shown by four tuples as, ( {A0,A1,A2,A3},{ a, b }, P, A0 ), where P is given by as follows.

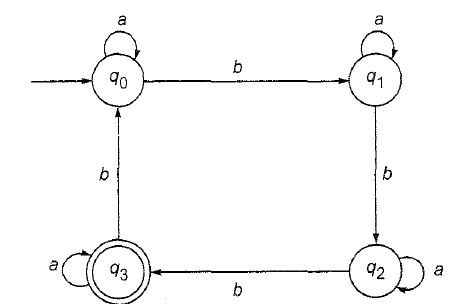
A0 🡪 aA0 | bA0 |aA1

A1 🡪 bA2

A2 🡪 bA3 | b

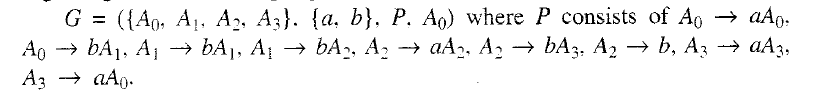
Q2 :- Construct a RG accepting L ={ w Є {a, b}\* | w is a string over {a, b} such that the number of b's is 3 mod 4}.

Sol:- First construct FA accepting L. The symbol a can occur with any no’s in w but b has to occur in form of 4k + 3, where k ≥ 0. Means length of b will follow as 3,7,11,….. (4k+3) and number of a can have any length.



Regular grammar G accepting L = T(M) is given as ( {A0, A1, A2, A3 },{ a, b}, P, A0)

Where Production P is given as follows.



Q. Construct a finite automaton recognizing L(G), where G is the grammar

S 🡪 aS | bA | b and A 🡪 aA | bS | a.

**Regular Sets, Arden’s Theorem**

Example 1:- Find a regular expression for each of the following subsets of { a, b }.



1. The set of all strings containing exactly 2a’s.
2. The set of all strings containing at least 2a’s.



1. The set of all strings containing at most 2a's.



1. The set of all strings containing the substring aa.



Solution

(a) b\*ab\*ab\* (b) (a + b)\*a(a + b)\*a(a + b)\* (c) b\*ab\*ab\* + b\*ab\* + b\*

(d) (a + b)\*aa(a + b)\*

EXAMPLE 2:- Find the regular expression representing the set of following strings

1. ambncp where m, n, p ≥ 1 a+b+c+, aa\*bb\*cc\*
2. ambncp where m, n, p ≥ 0 a\*b\*c\*,
3. amb2nc3p where m, n, p ≥ 1 aa\*bb(bb)\*ccc(ccc)\*

EXAMPLE 3:- Find the sets represented by the following regular expressions.

1. (aa)\* + (aaa)\*

aa, aaa, aaaa, …{ x Є {a}\* | x| is divisible by 2 or 3 }

1. ab( a + b )\* { aba, abab, abbaaa,…..}

IDENTITIES FOR REGULAR EXPRESSIONS

Two regular expressions P and Q are equivalent ( P = Q ), if P and Q represent the same set of strings. Following are the identities for regular expressions (useful for simplifying regular expressions).

I3 ɅR = RɅ = R

I5 R + R = R

16 R\*R\* = R\*

17 RR\* = R\*R

I8 (R\*)\* = R\*

19 Ʌ + RR\* = R\* = Ʌ + R\*R

110 (PQ)\*P = P(QP)\*

111 (P + Q)\* = (P\*Q\*)\* = (P\* + Q\*)\*

112 (P + Q)R = PR + QR and R(P + Q) = RP + RQ

Example 1 :- Solve regular expression R = Ʌ + 1\*(011)\*(1\* (011)\*)\*.

Solution:-

R = Ʌ + P1P1\*, where P1 = 1\*(011)\*

= P\* using 19

= (1\*(011)\*)\*

= (P2\*P3\*)\* letting P2 = 1, P3 = 011

= (P2 + P3)\* using I11

= (1 + 011)\*

EXAMPLE 2:- Prove (1 + 00\*1) + (1 + 00\*1)(0 + 10\*1)\* (0 + 10\*1) = 0\*1(0 + 10\*1)\*.

Solution:-

L.H.S. = (1 + 00\*1) (Ʌ + (0 + 10\*1)\* (0 + 10\*1) using I12

= (1 + 00\*1) (0 + 10\*1)\* using 19

= (Ʌ + 00\*)1 (0 + 10\*1)\* using I12 for 1 + 00\*1

=0\*1(0 + 10\*1)\* using 19

**Arden's theorem :- Let P and Q be two regular expressions over I. If P does not contain Ʌ, then the following equation in R, namely**

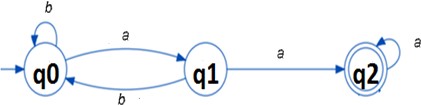
**R = Q + RP has a unique solution given by R = QP\*.**

Proof:-

Q + (QP\*)P = Q(Ʌ + P\*P) = QP\* by 19

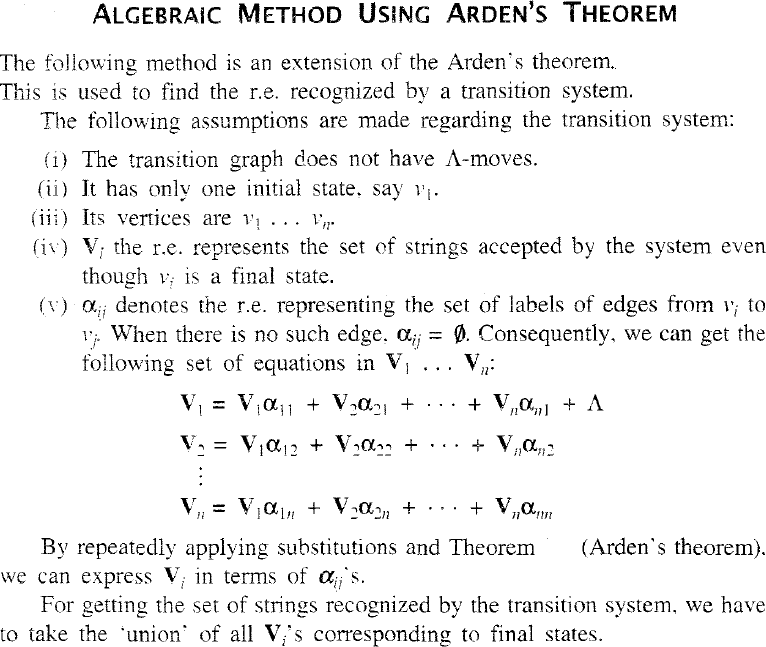
Hence theorem is satisfied when R = QP\*. This means R =QP\* is a solution.

(*Arden's theorem is very much useful in simplifying regular expressions for any given transition system. It replaces a given regular expression P by a simpler regular expression equivalent to P ).*

Example 1:-

Write all incoming edges using ‘+’ sign with input and next transition state. Also since initial state has one incoming edge 🡪 with no input and next transition state, so it is written as Ʌ.

q0 = q0b + q1b + Ʌ

q1 = q0a (write all incoming edges with input and next transition state)

q2 = q1a + q2a ((write all incoming edges with input and next transition state)

Steps:-

* Check whether the transition system contain any Ʌ - move and initial state.
* Since no Ʌ - move and only one initial state, so this method can be applied.
* Regular expression is evaluated for final state.

*( R = Q + RP has a unique solution (i.e. one and only one solution) given by R = QP\*. )*

Let’s take

q2 = q1a + q2a

q2 = q1aa\*

= q0aaa\*

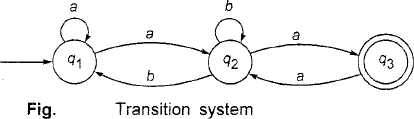
q0 = q0b + q1b + Ʌ = q0b + q0ab + Ʌ

q0 = q0 ( b + ab ) + Ʌ

q0 = Ʌ (b +ab)\*

q2 = q0aaa\* = (b +ab)\*aaa\*

Example 2 :- Consider the transition system given in Fig. Prove that the strings recognized are (a + a(b + aa)\*b)\*a(b + aa)\*a.



Sol:-

* Check whether the graph contain any Ʌ - move and initial state.
* Since no Ʌ - move and only one initial state, so this method can be applied.

The three equations for q1, q2 and q3 can be written as

q1 = q1a + q2b + Ʌ,

q2 = q1a + q2b + q3a,

q3 = q2a

By substituting q3 in the q2-equation,

we get q2 = q1a + q2b + q2aa

= q1a + q2(b + aa) = q1a(b + aa)\*

Substituting q2 in q1, we get q1 = q1a + q1a(b + aa)\*b + Ʌ

= ql(a + a(b + aa)\*b) + Ʌ

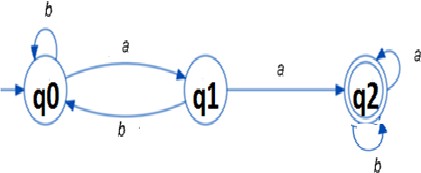
q1 = Ʌ (a + a(b + aa)\*b)\*

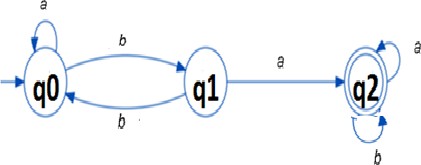
q2 = (a + a(b + aa)\*b)\* a(b + aa)\*

q3 = (a + a(b + aa)\*b)\* a(b + aa)\*a

Since q3 is a final state, the set of strings recognized by the graph is given by (a + a(b + aa)\*b)\*a(b + aa)\*a.

Q. Generate regular expression using Arden’s theorem.



Q. **Generate regular expression using Arden’s theorem.**