3. REGULAR EXPRESSIONS (RE)

CHAPTER - 3
REGULAR EXPRESSIONS (RE)

In this Chapter, we first define regular expressions as a means of representing certain subsets of strings over Σ and prove that regular sets are precisely those accepted by finite automata or transition systems. We use pumping lemma for regular sets to prove that certain sets are not regular.

A method of representing language as expression is known as **"regular expression"** that is the languages accepted by finite state automata are easily described by simple expressions. (Regular Expressions)

3.1 REGULAR EXPRESSIONS

3.1.1 Introduction

$\bigcup_{i=1}^{\infty} L^i$

Let Σ be a finite set of symbols. Let L_1, L_2 be set of strings in Σ^*. The concatenation of L_1 and L_2 denoted by $L_1 L_2$ is the set of all strings of the form xy, where $x \in L_1$ and $y \in L_2$. Define $L^0 = \{\varepsilon\}$ and $L^i = LL^{i-1}$ for $i \geq 1$.

The *Kleene closure* or *closure of* L, denoted by L^* is the set, defined as :

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

The *Positive closure of* L, denoted by L^+ is the set, defined as:

$$L^+ = \quad = L^* - L^0 = L^* - \{\varepsilon\}$$

- L^* - Constructed by concatenating any number of words from L.
- L^+ - Constructed by concatenating any number of words from L excluding ε . L^+ may contain ε if and only if L does.

Example 3.1

Let $L_1 = \{10, 01\}$, $L_2 = \{11, 00\}$

Then $L_1 L_2 = \{1011, 1000, 0111, 0100\}$

Example 3.2

Let L = {10, 11}

Then L^* = $L^0 \cup L^1 \cup L^2 \cup$.....

= {ε} ∪ {10, 11} ∪ {1011, 1010, 1110, 1111} ∪....

= {ε, 10, 11, 1011, 1010, 1110, 1111, ...}

3.1.2 Definition of Regular Expression

Let ∑ be an alphabet. The regular expressions over ∑ and the sets that they denote are defined recursively as follows:

(i) ϕ is a regular expression and denotes the empty set.

(ii) ε is a regular expression and denotes the set {ε}

(iii) For each $a \in \sum$, '*a*' is a regular expression and denotes the set {*a*}.

(iv) If *r* and *s* are regular expressions denoting the languages *R* and *S* respectively then (*r* + *s*), (*rs*), $(r)^*$ are regular expressions that denotes the sets $R \cup S$, *RS* and R^* respectively.

Example 3.3

(i) $(0/1)^*$ = {ε,0,1,00, 01, 10, 11.....} = $(0+1)^*$ (i.e.) all strings of 0 and 1

(ii) 01^* = {0, 01, 011, 0111, ...}

(iii) 0^* = {ε, 0, 00, 000, ...}

(iv) $1(1)^*$ = {1, 11, 111, 1111, ...) = 1^+

3.1.3 Identities for Regular Expressions

I_1 $\phi + R = R$

I_2 $\phi R = R\phi = \phi$

I_3 $\lambda R = R\lambda = R$

I_4 $\lambda^* = \lambda$

I_5 $R + R = R$

I_6 $R^*R^* = R^*$

I_7 $RR^* = R^*R$

I_8 $(R^*)^* = R^*$

I_9 $\lambda + RR^* = R^* = \lambda + R^*R$

I_{10} $(PQ)^*P = P(QP)^*$

I_{11} $(P + Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

I_{12} $(P + Q)R = PR + QR$ and $R(P + Q) = RP + RQ$

3.2 FINITE AUTOMATA AND REGULAR EXPRESSIONS

NFA - Non deterministic Finite Automata.

DFA - Deterministic Finite Automata.

3.2.1 Conversion of Regular Expression to NFA with ε-transitions (Thompson's Construction)

(a) $\phi = \{\ \}$ is a RE

start → (q_0) (q_f) ε–NFA for ϕ

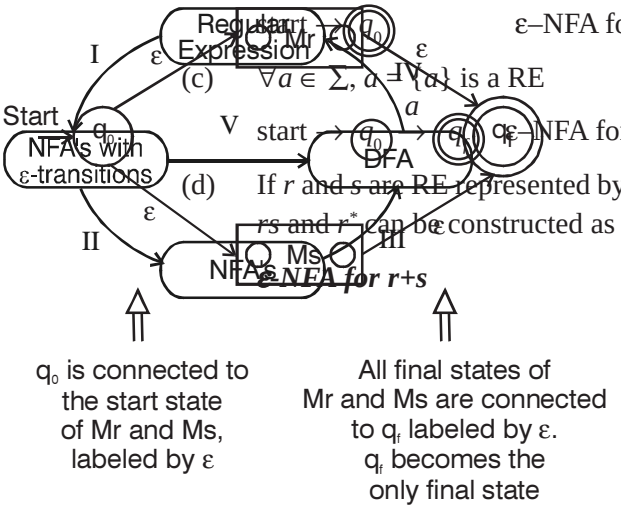
(b) $\varepsilon = \{\varepsilon\}$ is a RE

start → (q_0) (q_f) ε–NFA for ε

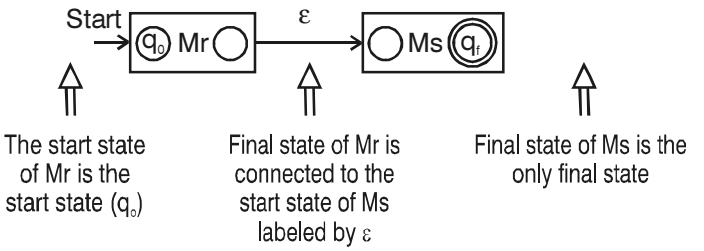
(c) $\forall a \in \Sigma$, $a = \{a\}$ is a RE

start → (q_0) a (q_f) ε–NFA for a

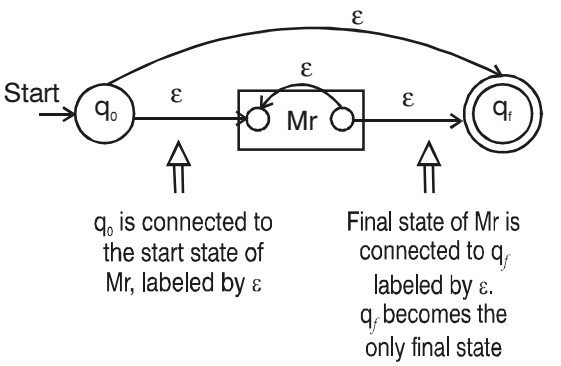
(d) If r and s are RE represented by ε NFA, Mr and Ms respectively, the ε NFA for $r + s$, rs and r^* can be constructed as :



ε-NFA for rs

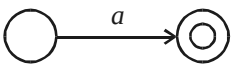


ε-NFA for r*



Example 3.4 Construct the ε-NFA for the given regular expression using Thompson's construction - $(a+b)^* a.b$.

Step 1 : a



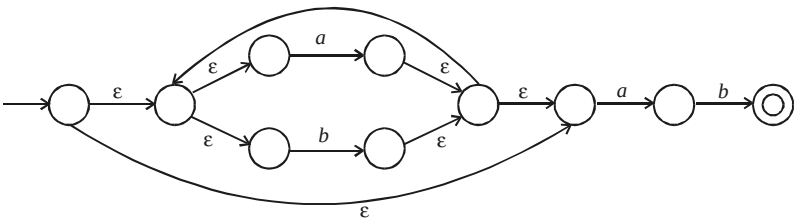
Step 2 : b

Step 3 : $a+b$

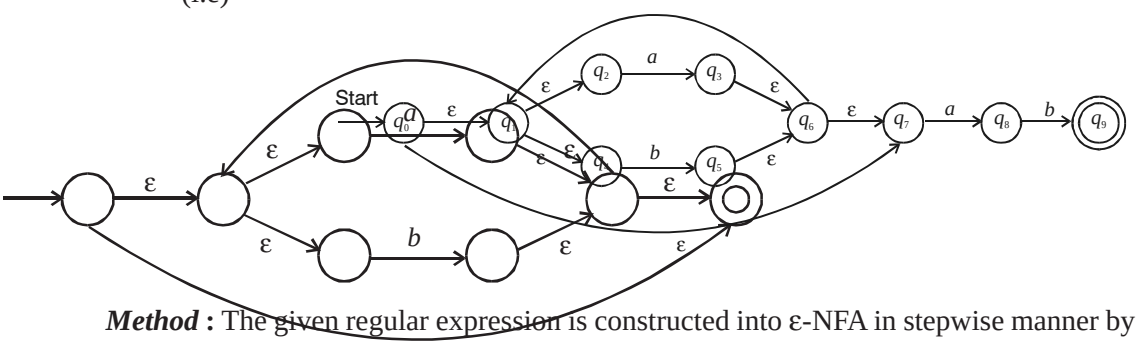
Note : $a+b = (a+b)$

Step 4 : $(a+b)^*$

Step 5 : $(a+b)^* \cdot a \cdot b$



(i.e)



***Method* :** The given regular expression is constructed into ε-NFA in stepwise manner by splitting the expression as per Thompson's rule to derive a final ε–NFA of the regular expression.

Theorem (Conversion of R.E. to FSA)

For every regular expression *r* there exists a NFA with ε-transitions that accepts L(*r*)

Proof

We prove by induction on the number of operators in the regular expression *r* that there is an NFA M with ε transistons, having one final state and no transistions out of this final state such that L(M) = L(*r*).

Basis step (Zero operators)

Suppose *r* is ε, ϕ or *a* for some $a \in \sum$. Then the equivalent NFA's are:

(i) $r = \varepsilon$

(ii) $r = \phi$

(iii) $r = a$

Induction (One or more operators)

Assume the theorem is true for r having fewer than i operators, $i \geq 1$. Let r have i operators. We discuss three cases depending on the form of r.

Case 1 :

Let $r = r_1 + r_2$. Both r_1 and r_2 must have fewer than i operators. Thus there are NFA's $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_2 \{f_2\})$ with $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$. Assume Q_1 and Q_2 are disjoint.

Let q_0, f_0 be a new initial and final state respectively.

$\therefore$ $M = (Q_1 \cup Q_2 \cup \{q_0, f_0\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{f_0\})$ where δ is defined by

(i) $\delta(q_0, \varepsilon) = \{q_1, q_2\}$

(ii) $\delta(q, a) = \delta_1(q, a)$ if $q \in Q_1 - \{f_1\}$, $a \in \Sigma_1 \cup \{\varepsilon\}$

(iii) $\delta(q, a) = \delta_2(q, a)$ if $q \in Q_2 - \{f_2\}$, $a \in \Sigma_2 \cup \{\varepsilon\}$

(iv) $\delta_1(f_1, \varepsilon) = \delta_2(f_2, \varepsilon) = \{f_0\}$

All the moves of M_1 and M_2 are present in M. Any path in the transition diagram of M from q_0 to f_0 must begin by going to either q_1 or q_2 on ε. If the path goes to q_1, it may follow any path in M_1 to f_1 and then goto f_0 on ε.

Similarly paths that begin by going to q_2 may follow any path in M_2 to f_2 and then go to f_0 on ε. These are the only paths from q_0 to f_0. That is, if there is a path labeled x in M_1 from q_1 to f_1 or a path in M_2 from q_2 to f_2. Hence $L(M) = L(M_1) \cup L(M_2)$

Case 2 :

Let $r = r_1 r_2$. Let M_1 and M_2 be as in case 1. Construct $M = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, \{q_1\}, \{f_2\})$, where δ is given by

(i) $\delta(q,a) = \delta_1(q,a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\varepsilon\}$

(ii) $\delta(f_1,\varepsilon) = \{q_2\}$

(iii) $\delta(q,a) = \delta_2(q,a)$ for q in Q_2 and a in $\Sigma_2 \cup \{\varepsilon\}$

Every path in M from q_1 to f_2 is a path labeled by some string x from q_1 to f_1, followed by the edge from f_1 to q_2 labeled ε, followed by a path labeled by some string y from q_2 to f_2.

Thus $L(M) = \{xy \mid x \text{ is in } L(M_1) \text{ and } y \text{ is in } L(M_2)\}$

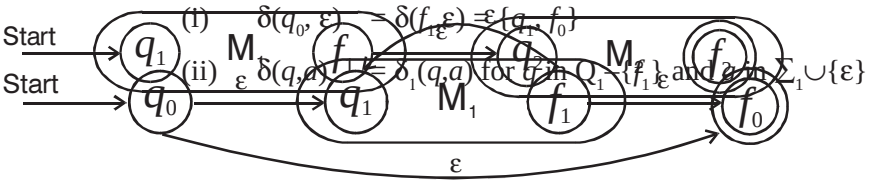
and $L(M) = L(M_1) \cdot L(M_2)$.

Case 3 :

Let $r = r_1^*$. Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ and $L(M_1) = r_1$. Construct $M = (Q_1 \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, \{f_0\})$, where δ is given by

(i) $\delta(q_0, \varepsilon) = \delta(f_1, \varepsilon) = \{q_1, f_0\}$

(ii) $\delta(q,a) = \delta_1(q,a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\varepsilon\}$



Any path from q_0 to f_0 consists either of a path from q_0 to f_0 on ε or a path from q_0 to q_1 on ε followed by some number of paths from q_1 to f_1, then back to q_1 on ε. There is a path in M from q_0 to f_0 labeled x if and only if we write $x = x_1 x_2 \ldots\ldots x_j$ for some $j \geq 0$ such that each $x_i \in L(M_1)$. Hence $L(M) = L(M_1)^*$

3.2.2 Conversion of ε-NFA to DFA

Method

(i) Find the ε-CLOSURE of the state q_0 from the constructed ε-NFA (i.e) from state q_0, ε transition to other states are identified as well as ε transitions from other states are also identified and combined as one set (new state).

(ii) Perform the following steps until there are no more new states as been constructed.

(a) Find the transition of the given regular expression symbols over $\sum$ from the new state (i.e) move (new state, symbol)

(b) Find the ε-CLOSURE of move (new state, symbol).

Example 3.5

Construct the DFA for the given ε-NFA. (Refer the page No.3.6 for Figure).

Step 1 :

ε closure $(q_0) = \{q_0, q_1, q_2, q_4, q_7\} \rightarrow$ A

A - new state

Step 2 :

move (A, *a*) = $\{q_3, q_8\}$

ε closure [move (A,*a*)] = $\{q_1, q_2, q_3, q_4, q_6, q_7, q_8\} \rightarrow$ B

move (A, *b*) = $\{q_5\}$

ε closure [move (A, *b*)] = $\{q_1, q_2, q_4, q_5, q_6, q_7\} \rightarrow$ C

B, C - new states

Step 3 (for the new state B) :

move (B,*a*) = $\{q_3, q_8\}$

ε closure [move(B,*a*)] = $\{q_1, q_2, q_3, q_4, q_6, q_7, q_8\} \rightarrow$ B

move (B, *b*) = $\{q_5, q_9\}$

ε closure [move(B,*b*)] = $\{q_1, q_2, q_4, q_5, q_6, q_7, q_9\} \rightarrow$ D

D – new state

Step 4 (for the new state C) :

move (C, *a*) = $\{q_3, q_8\}$

ε closure [move(C,*a*)] = $\{q_1, q_2, q_3, q_4, q_6, q_7, q_8\} \rightarrow$ B

move (C, *b*) = $\{q_5\}$

ε closure [move(C,*b*)] = $\{q_1, q_2, q_4, q_5, q_6, q_7\} \rightarrow$ C

* no new states

Step 5 (for the new state D) :

move (D, *a*) = $\{q_3, q_8\}$

ε closure [move(D,*a*)] = $\{q_1, q_2, q_3, q_4, q_6, q_7, q_8\} \rightarrow$ B

move (D, *b*) = $\{q_5\}$

ε closure [move(D,*b*)] = $\{q_1, q_2, q_4, q_5, q_6, q_7\} \rightarrow$ C

* no new states

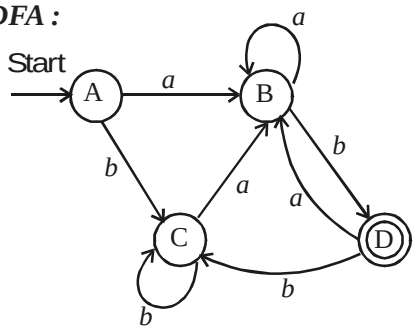
Step 6 :

- ★ Construction terminates, because no new states are generated from step 4 and step 5
- ★ From the above steps (1 to 5) transition table & transition diagram of DFA are constructed.

Transition table of DFA :

New States	Symbols	
	a	*b*
→ A	B	C
B	B	D
C	B	C
(D)	B	C

Transition diagram of DFA :



3.2.3 Conversion of DFA to RE

The conversion of FA to RE is possible with three different methods. They are :

(i) Regular Expression equation method - $R_{ij}^{(k)}$

(ii) Arden's Theorem.

(iii) State elimination technique.

3.2.3.1 Regular Expression equation method –$R_{ij}^{(k)}$

Theorem

For every DFA A = (Q, ∑, δ, S, F), there is a regular expression R, such that L(R) = L(A).

Proof

Let L be the set accepted by the DFA

A = ($\{q_1, q_2, ...q_n\}$, ∑, δ, q_1, F) with q_1 being the start state.

Let $R_{ij}^{(K)}$ be the regular expression describing the set of all strings *x* such that $\delta(q_i, x) = q_j$ going through intermediate states $\{q_1, q_2, ...q_K\}$ only

$R_{ij}^{(K)}$ will be defined inductively. Note that

$$L\left(\bigcup_{j\in F} R_{1j}^{(n)}\right) = L(A)$$

Basis

K = 0, i.e., no intermediate states.

$R_{ij}^{(0)}$ denotes a set of strings which is either ε(or) single symbol.

Case 1 : $i \neq j$

$R_{ij}^{(0)} = \{a \mid \delta(q_i, a) = q_j\}$ denotes set of symbols a such that $\delta(q_i, a) = q_j$

Case 2 : $i = j$

$R_{ij}^{(0)} = R_{ii}^{(0)} = (\{a|\delta(q_i,a) = q_j\} \cup \{\varepsilon\})$ denotes set of all symbols a such that a (or) ε.

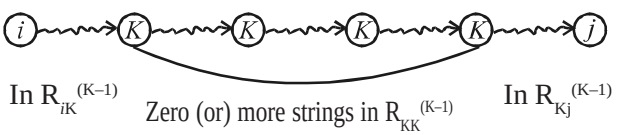
$R_{ii}^{(0)} = a+\varepsilon$

Induction

It involves regular expression operations : union, concatenation and closure.

$$R_{ij}^{(K)} = R_{ij}^{(K-1)} + R_{iK}^{(K-1)} (R_{KK}^{(K-1)})^* R_{Kj}^{(K-1)}$$

ie



The observation of this proof is that regular expression

$L(A) = \bigcup_{q_j \text{ in } F} R_{1j}^{(n)}$

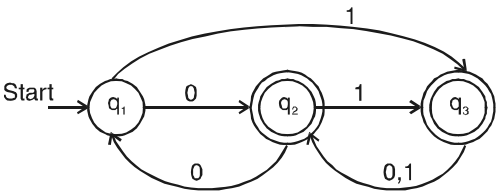
where $R_{1j}^{(n)}$ denotes the labels of all paths from q_1 to q_j

where $F = \{q_{j1}, q_{j2}, \ldots\ldots q_{jp}\}$,

so $L(A) = R_{1j1}^{(n)} + R_{1j2}^{(n)} + \ldots\ldots R_{1jp}^{(n)}$

Examples 3.6

1. Find R for finite automaton given below :



Solution :

We will find $R_{ij}^{(0)}$ where **K = 0**

$R_{11}^{(0)} = \varepsilon$ because from state q_1 to q_1 can be achieved only by ε transition.

$R_{12}^{(0)} = 0$ because from q_1 we can reach q_2 by '0' input.

$R_{13}^{(0)} = 1$ because from q_1 we can reach q_3 by '1' input $\delta(q_1, 1) = q_3$

$R_{21}^{(0)} = 0$ because $\delta(q_2, 0) = q_1$

$R_{22}^{(0)} = \varepsilon$ because from q_2 we can reach q_2 only on ε transition

$R_{23}^{(0)} = 1$ because $\delta(q_2, 1) = q_3$

$R_{31}^{(0)} = \phi$ because from q_3 to reach q_1 no such path exists

$R_{32}^{(0)} = 0+1$ because from q_3 to reach q_2 we can give either '0' or '1' as input symbol.

$R_{33}^{(0)} = \varepsilon$ to remain in q_3 it needs ε transition.

Tabulation is as follows for K=0:

$R_{11}^{(0)}$	ε
$R_{12}^{(0)}$	0
$R_{13}^{(0)}$	1
$R_{21}^{(0)}$	0
$R_{22}^{(0)}$	ε
$R_{23}^{(0)}$	1
$R_{31}^{(0)}$	ϕ
$R_{32}^{(0)}$	0+1
$R_{33}^{(0)}$	ε

we need to apply the following simplification.

- $(\varepsilon + R)^* = R^*$
- $R + RS^* = RS^*$
- $\phi R = R\phi = \phi$ (Annihilation)
- $\phi + R = R + \phi = R$ (Identity)

Now we will go for **K = 1**

n =3 (number of states is 3 $\{q_1, q_2, q_3\}$)

$R_{ij}^{(K)} = R_{ij}^{(K-1)} + R_{iK}^{(K-1)} (R_{KK}^{(K-1)})^* R_{Kj}^{(K-1)}$

Therefore for K = 1

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

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

$$= \varepsilon$$

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

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

$$= 0 + 0 = 0$$

$$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* R_{13}^{(0)}$$

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

$$= 1 + 1 = 1$$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} (R_{11}^{(0)})^* R_{11}^{(0)}$$

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

$$= 0 + 0 = 0$$

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

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

$$= \varepsilon + 00$$

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

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

$$= 1 + 01$$

$$R_{31}^{(1)} = R_{31}^{(0)} + R_{31}^{(0)} (R_{11}^{(0)})^* R_{11}^{(0)}$$

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

$$= \phi + \phi$$

$$= \phi$$

$$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)} (R_{11}^{(0)})^* R_{12}^{(0)}$$

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

$$= 0 + 1 + \phi$$

$$= 0 + 1$$

$$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)} (R_{11}^{(0)})^* R_{13}^{(0)}$$

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

$$= \varepsilon$$

Tabulation for K = 1 is :

$R_{11}^{(1)}$	ε
$R_{12}^{(1)}$	0
$R_{13}^{(1)}$	1
$R_{21}^{(1)}$	0
$R_{22}^{(1)}$	ε + 00
$R_{23}^{(1)}$	1 + 01
$R_{31}^{(1)}$	ϕ
$R_{32}^{(1)}$	0 + 1
$R_{33}^{(1)}$	ε

Now for **K = 2**

$$R_{ij}^{(2)} = R_{ij}^{(2-1)} + R_{i2}^{(2-1)}(R_{22}^{(2-1)})^* R_{2j}^{(2-1)}$$

$$= R_{ij}^{(1)} + R_{i2}^{(1)} (R_{22}^{(1)})^* R_{2j}^{(1)}$$

Therefore

$$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)} = \varepsilon + 0\ (\varepsilon+00)^*\ 0$$

$$= \varepsilon + 0(00)^*\ 0 = (00)^*$$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)} = 0 + 0\ (\varepsilon + 00)^*\ (\varepsilon + 00)$$

$$= 0+0(00)^* = 0(00)^* \quad (\therefore R + RS^* = RS^*)$$

$$R_{13}^{(2)} = R_{13}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{23}^{(1)} = 1 + 0(\varepsilon+00)^*\ (1 + 01)$$

Here $(\varepsilon + 00)^* = (00)^*$

and $(1 + 01) = (\varepsilon + 0)\ 1$

and so $R_{13}^{(2)} = 1 + 0\ (00)^*\ (\varepsilon + 0)1$

Here $(00)^*\ (\varepsilon + 0) = 0^*$

Hence $0(00)^*\ (\varepsilon + 0)1 = 0(0^*)1$

$\therefore R_{13}^{(2)} = 1 + 00^*1 = 0^*1$

$$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{21}^{(1)}$$

$$= 0 + (\varepsilon + 00)\ (\varepsilon + 00)^*\ 0$$

$$= 0 + 00(00)^*0$$

$$= 0(00)^*$$

$$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)}$$

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

$= (00)^*$

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

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

$= (\varepsilon + 0)1 + (00)^*(1 + 01)$

$= (\varepsilon + 0) 1 + (00)^* (\varepsilon + 0)1$

$= (\varepsilon + 0) 1 + 0^*1$ (because $(00)^*(\varepsilon + 0)1=0^*1$)

$= 0^*1$

$R_{31}^{(2)} = R_{31}^{(1)} + R_{32}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)}$

$= \phi + (0 + 1) (\varepsilon + 00)^*0$

$= \phi + (0 + 1)(00)^*0$

$= (0 + 1)(00)^*0$ (because $(\varepsilon + 00)^* = (00)^*$)

$R_{32}^{(2)} = R_{32}^{(1)} + R_{32}^{(1)}(R_{22}^{(1)})^*R_{22}^{(1)}$

$= (0 + 1) + (0 + 1)(\varepsilon + 00)^* (\varepsilon + 00)$

$= (0 + 1) + (0 + 1)(00)^*$

$= (0 + 1)(00)^*$ (because $R + RS^* = RS^*$)

$R_{33}^{(2)} = R_{33}^{(1)} + R_{32}^{(1)} (R_{22}^{(1)})^* R_{23}^{(1)}$

$= \varepsilon + (0 + 1) (\varepsilon + 00)^* (1 + 01)$

$= \varepsilon + (0 + 1)(00)^*(\varepsilon + 0)1$

(because $(\varepsilon + 00)^* = (00)^*$ and $(1 + 01)=((\varepsilon + 0)1)$)

$= \varepsilon + (0 + 1)0^*1$ (because $(00)^*(\varepsilon + 0)=0^*$)

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

Table for K=2 :

$R_{11}^{(2)}$	$(00)^*$
$R_{12}^{(2)}$	$0(00)^*$
$R_{13}^{(2)}$	0^*1
$R_{21}^{(2)}$	$0(00)^*$
$R_{22}^{(2)}$	$(00)^*$
$R_{23}^{(2)}$	0^*1
$R_{31}^{(2)}$	$(0 + 1)(00)^*0$
$R_{32}^{(2)}$	$(0 + 1)(00)^*$
$R_{33}^{(2)}$	$\varepsilon + (0 + 1)0^*1$

No we will find regular expression L(M)

L(M) = $\bigcup_{q_j \text{ in } F} R_{1j}^{(n)}$

ie $R_{1j1}^{(n)} + R_{1j2}^{(n)} + ... + R_{1jP}^{(n)}$

Where F is set of final sets

F=$\{q_{j1}, q_{j2}, ...q_{jp}\}$

In our example, the set of final states

F = $\{q_2, q_3\}$

$\therefore$ L(M) = $R_{12}^{(3)} + R_{13}^{(3)}$ (where n=3 states)

$\therefore$ $R_{12}^{(3)} = R_{12}^{(2)} + R_{13}^{(2)} (R_{33}^{(2)})^* R_{32}^{(2)}$

$= 0(00)^* + 0^*1 (\varepsilon + (0 + 1)0^*1)^* (0 + 1)(00)^*$

$= 0(00)^* + 0^*1((0 + 1)0^*1)^*(0 + 1)(00)^*$

(because $(\varepsilon+R)^* = R^*$)

Similarly $R_{13}^{(3)} = R_{13}^{(2)} + R_{13}^{(2)} (R_{33}^{(2)})^* R_{33}^{(2)}$

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

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

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

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

Hence regular expression is

$R_{12}^{(3)}+R_{13}^{(3)} = 0^*1((0 + 1)0^*1)^* (0 + 1)(00)^* + 0(00)^* + 0^*1((0 + 1)0^*1)^*$

$= 0^*1((0 + 1)0^*1)^*(0 + 1)(00)^* + 0(00)^*$

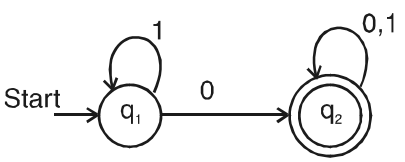
(because $R + RS^* = RS^*$)

$\therefore$ The regular expression for the finite automata is

L(M) = $0^*1 ((0 + 1)0^*1)^*(0 + 1)(00)^* + 0(00)^*$

Example 3.7 Find R for A where

L(A) = $\{x.y: x \in \{1\}^* \text{ and } y \in \{0,1\}^*\}$



(Nov/Dec 2003)

Solution :

Let K = 0

$R_{11}^{(0)} = \varepsilon + 1$ [$\because$ from q_1 to q_1 can be achieved from ε (or) 1]

$R_{12}^{(0)} = 0$ [from q_1 to q_2 can be achieved by '0' input]

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

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

Let K = 1

$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^* R_{1j}^{(0)}$

	Direct Substitution	Simplified
$R_{11}^{(1)}$	$(\varepsilon+1)+(\varepsilon+1)(\varepsilon+1)^*(\varepsilon+1)$	11^* ($\varepsilon+1=1$)
$R_{12}^{(1)}$	$0 + (\varepsilon+1)(\varepsilon+1)^* 0$	1^*0 ($R+RS^* = RS^*$)
$R_{21}^{(1)}$	$\phi + \phi(\varepsilon+1)^* (\varepsilon+1)$	ϕ ($R\phi=\phi$)
$R_{22}^{(1)}$	$(\varepsilon+0+1) + \phi (\varepsilon+1)^* 0$	$(\varepsilon+0+1)$

Similarly for **K= 2**

$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)}(R_{22}^{(1)})^* R_{2j}^{(1)}$

	Direct Substitution	Simplified
$R_{11}^{(2)}$	$11^* + 1^*0(\varepsilon+0+1)^* \phi$	11^*
$R_{12}^{(2)}$	$1^*0+1^*0(\varepsilon+0+1)^*(\varepsilon+0+1)$	$1^*0(0+1)^*$
$R_{21}^{(2)}$	$\phi + (\varepsilon+0+1)(\varepsilon+0+1)^* \phi$	ϕ
$R_{22}^{(2)}$	$(\varepsilon+0+1)+(\varepsilon+0+1)(\varepsilon+0+1)^* (\varepsilon+0+1)$	$(0+1)^*$

The final regular expression of A is

L(M) =

Here n = 2 (number of states)

F = $\{q_2\}$

$\therefore$ L(M) = $R_{12}^{(2)} = 1^*0(0+1)^*$

3.2.3.2 Arden's Theorem

1. It is used to find the regular expression for the given finite state automata.
2. Let P & Q be two regular expressions over $\sum$. If P does not contain ε, then the equation in R = Q + RP has a solution (i.e.,) R=QP*

Proof :

$Q + RP = Q + (QP^*)P$ because $R=QP^*$

$= Q(\varepsilon + P^*P)$

$= QP^* = R$

3. The principle of this theorem is:

   (i) The finite automata should not have ε moves

   (ii) The FA should have only one start state say q_1

   (iii) Its states are $q_1, q_2 ..... q_n$

   (iv) R is the regular expression (regex) representing the set of strings accepted by the FA.

   (v) α_{ij} denotes the set of labels of edges from q_i to q_j.

   If there is no edge $\alpha_{ij} = \phi$. We will get

   $q_1 = q_1\alpha_{11} + q_2\alpha_{21} + ...+ q_n \alpha_{n1} + \varepsilon$

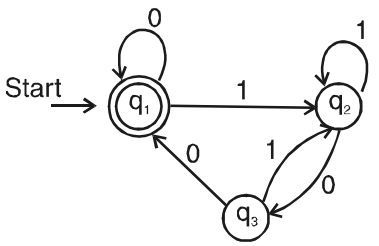
   $q_2 = q_1 \alpha_{12} + q_2 \alpha_{22}+ .....+ q_n\alpha_{n2}$

   $q_n=q_1 \alpha_{1n}+q_2\alpha_{2n}+ .....+ q_n\alpha_{nn}$

   Now by applying repeated substitution we can express RE in terms of α_{ij}'s.

Example 3.8

Construct regular expression to the given FA (using Arden's Theorem)



(April/May 2004)

Solution :

We can obtain regex (or) regular expression by applying Arden's theorem.

Step 1 : (i) Check whether FA does not have ε-moves

(ii) It has only one start state

Step 2 : Express states interms of transitions

The transitions that required to reach q_1 from other states is

$q_1 = q_1 0 + q_3 0+\varepsilon.....$ (1)

Similarly for

$q_2 = q_2 1 + q_1 1 + q_3 1 .... (2)$

$q_3 = q_2 0 ..... (3)$

Now substitute (3) in (2)

$q_2 = q_2 1 + q_1 1 + q_2 01$

$q_2 = q_2(1 + 01) + q_1 1$

Since q_2 is in LHS & RHS we can write it has

$q_2 = q_1 1(1 + 01)^*$

Now $q_1 = q_1 0 + q_2 00 + \varepsilon$ (because $q_3 = q_2 0$)

$= q_1 0 + q_1 1(1 + 01)^* 00 + \varepsilon$

$= q_1(0 + 1(1+01)^*00) + \varepsilon$

By applying Arden's theorem

$q_1 = \varepsilon(0 + 1(1 + 01)^*00)^*$

$= (0 + 1(1 + 01)^*00)^*$

As q_1 is the only final state, the regular expression corresponding to given FA is

RE = (0+1(1+01)*00)*

3.2.3.3 State elimination technique

The following Figure (a) shows a generic state *s* about to be eliminated. We suppose that the automaton of which *s* is a state has predecessor states q_1, q_2, ... q_k for s and successor states p_1, p_2,..... p_m for *s*. It is possible that some of that *q*'s are also *p*'s, but we assume that *s* is not among the *q*'s or *p*'s, even if there is a loop from *s* to itself. We also show a regular expression on each arc from one of the *q*'s to *s*; expression Q_i labels the arc from q_i. Likewise, we show a regular expression P_j labeling the arc from *s* to p_i, for all *i*. We show a loop on *s* with label *S*. Finally, there is a regular expression R_{ij} on the arc from q_i to p_j, for all *i* and *j*. Note that some of these arcs may not exist in the automaton, in which case we take the expression on that arc to be ϕ.

The Figure (b) shows what happens when we eliminate state *s*. All arcs involving state *s* are deleted. To compensate, we introduce, for each predecessor q_i of *s* and each successor p_j of *s*, a regular expression that represents all the paths that start at q_i, go to *s*, perhaps loop around *s* zero or more times, and finally go to p_j. The expression for these paths is $Q_i S^* P_j$. This expression is added (with the union operator) to the arc from q_i to p_j. If there was no arc $q_i \rightarrow p_j$, then first introduce one with regular expression ϕ.

The strategy for constructing a regular expression from a finite automaton is as follows:

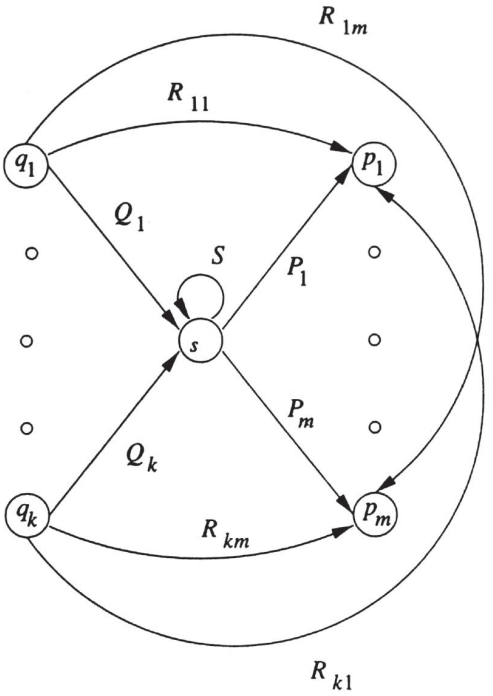


Figure (a) A state s about to be elimiated

1. For each accepting state q, apply the above reduction process to produce an equivalent automaton with regular-expression labels on the arcs. Eliminate all states except q and the start state q_0.

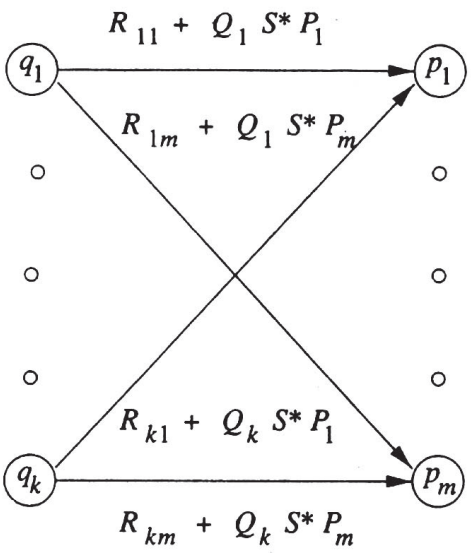


Figure (b) Result of eliminating state s from Figure (a)

2. If $q \neq q_0$, then we shall be left with a two-state automaton that looks like Figure (c). The regular expression for the accepted strings can be described in various ways. One is (R + SU*T)*SU*. In explanation, we can go from the start state to itself any number of times, by following a sequence of paths whose labels are in either L(R) or L(SU*T). The expression SU*T represents paths that go to the accepting state via a path in L(S), perhaps return to the accepting state several times using a sequence of paths with labels in L(U), and then return to the start state with a path whose label is in L(T). Then we must go to the accepting state, never to return to the start state, by following a path with a label in L(S). Once in the accepting state, we can return to it as many times as we like, by following a path whose label is in L(U).

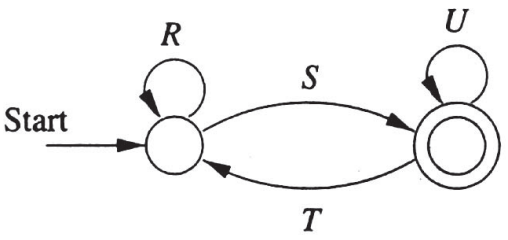


Figure (c) A generic two-state automaton

3. If the start state is also an accepting state, then we must also perform a state-elimination from the original automaton that gets rid of every state but the start state. When we do so, we are left with a one-state automaton that looks like Figure (d). The regular expression denoting the strings that it accepts is R*.

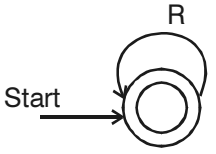


Figure (d). A generic one-state automaton

4. The desired regular expression is the sum (union) of all the expressions derived from the reduced automata for each accepting state, by rules (2) and (3).

Examples 3.9

Find the regular expression for the given FA using state elimination technique.

$w = \{x \mid bc, \quad x \in \{0,1\}^*, \{b, c\} \subseteq \{0, 1\}\}$

Solution :

The finite automata has two end states {C,D}

We apply state elimination technique for the end state D.

Step 1:

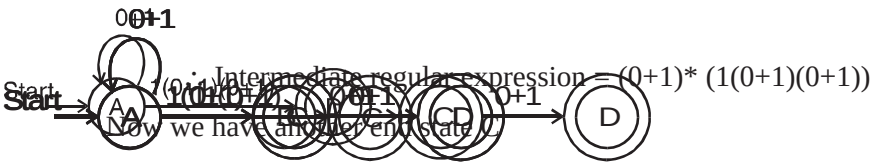
The transition on 0, 1 can be written has 0 + 1.

Step 2:

Let's eliminate state B.

Step 3:

Now eliminate state C to reach one of the final state D



; Intermediate regular expression = (0+1)* (1(0+1)(0+1))

Now we have another end state D

Step 1 :

Let's eliminate state B

The intermediate state B is eliminated and directly we can reach C. So transition is 1(0+1)

Step 2 :

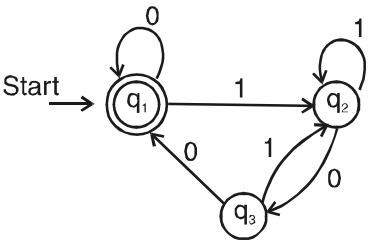
Eliminate D to obtain AC

$\therefore$ Intermediate regular expression is (0+1)* (1(0+1))

Therefore the final Regular Expression is the sum of both

L(M) = (0+1)* (1(0+1)(0+1)) + (0+1)* (1(0+1))

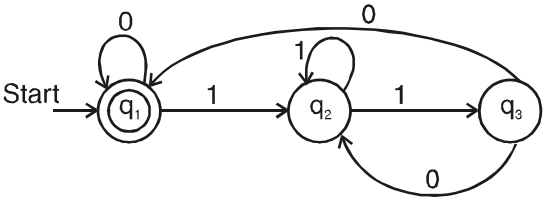
Example 3.10 Find the regular expression for the given FA using state elimination technique.



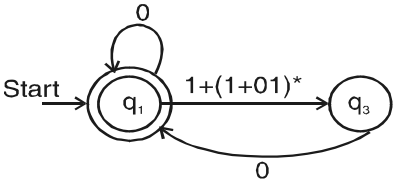
Solution :

Step 1 :

Eliminate q_2



Therefore it becomes



Step 2 :

Eliminate q_3

$\therefore$ R **.** E = (0+1+(1+01)* 00)*

3.3 PUMPING LEMMA FOR REGULAR SETS

In this section we give a necessary condition for an input string to belong to a regular set. The result is called *pumping lemma* as it gives a method of pumping (generating) many input strings from a given string. As pumping lemma gives a necessary condition, it can be used to show that certain sets are not regular.

Theorem (Pumping Lemma)

Let M = *(*Q, ∑, δ, q_0, F*)* be a finite automaton with *n* states. Let L be the regular set accepted by M. Let $w \in$ L and $|w| \geq m$. If $m \geq n$, then there exists *x, y,* z such that $w = xyz$, $y \neq \lambda$ and $xy^iz \in$ L for each $i \geq 0$.

Proof

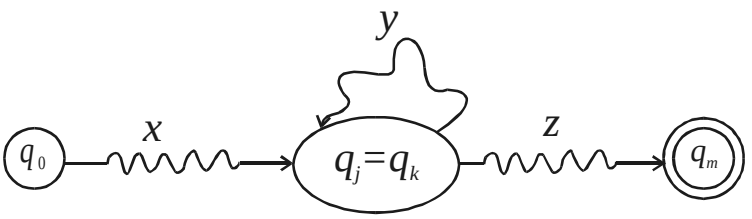
Let $w = a_1a_2 \ldots a_m$, $m \geq n$

$\delta(q_0, a_1a_2 \ldots a_i) = q_i$ for $i = 1, 2, \ldots m$; $Q_1=(q_0, q_1, \ldots q_m)$

That is, Q_1 is the sequence of states in the path with path value $w = a_1a_2 \ldots a_m$. As there are only *n* distinct states, at least two states in Q_1 must coincide. Among various pairs of repeated states, we take the first pair. Let us take them as q_j and q_k $(q_j=q_k)$. Then *j* and *k* satisfy the condition $0 \leq j < k \leq n$.

The string *w* can be decomposed into three substrings $a_1a_2 \ldots a_j$, $a_{j+1} \ldots a_k$ and $a_{k+1} \ldots a_m$. Let *x, y, z* denote these strings $a_1a_2 \ldots a_j$, $a_{j+1} \ldots a_k$, $a_{k+1} \ldots a_m$ respectively. As $k \leq n$, $|xy| \leq n$ and $w = xyz$. The path with path value *w* in the transition diagram of M is shown in the following Figure.

The automaton M starts from the initial state q_0. On applying the string *x*, ir reaches $q_j(= q_k)$. On applying the string *y*, it comes back to $q_j(= q_k)$. So after application of y_i for each $i \geq 0$, the automaton is in the same state q_j. On applying z, it reaches q_m, a final state. Hence $xy^iz \in$ L. As every state in Q_1 *is* obtained by applying an input symbol $y \neq \lambda$



Note :

The decomposition is valid only for strings of length greater than or equal to the number of states. For such a string $w = xyz$, we can iterate the substring *y* in *xyz* as many times as we like and get strings of the form xy^iz which are longer than *xyz* and are in L. By considering the path from q_0 to q_k and then the path from q_k to q_m (without going through the loop), we get a path ending in a final state with path value *xz*. (This corresponds to the case when $i = 0$.)

Application of Pumping Lemma

This theorem can be used to prove that certain sets are not regular. We now give the steps needed for proving that a given set is not regular.

Step 1:

Assume L *is* regular. Let n be the number of states in the corresponding FA.

Step 2 :

Choose a string w such that $|w| \geq n$. Use pumping lemma to write $w = xyz$, with $|xy| \leq n$ and $|y| > 0$.

Step 3 :

Find a suitable integer i such that $xy^iz \notin L$. This contradicts our assumption. Hence L *is* not regular.

Note :

The crucial part of the procedure is to find i such that $xyz \notin L$. In some cases we prove $xy^iz \notin$ L by considering $| xy^iz |$. In some cases we may have to use the 'structure' of strings in L.

3.4 CLOSURE PROPERTIES OF REGULAR SETS

If a class of languages is closed under a particular operation we call that fact a closure property of the class of languages.

In this section we discuss the closure properties of regular sets under (a) union (b) concatenation (c) closure (d) complementation (e) intersection (f) transpose (g) substitution (h) homomorphism.

Theorem

The regular sets are closed under union, concatenation and closure.

Proof

Let L_1 and L_2 be regular sets such that

$L_1 = T(A_1)$ and $L_2 = T(A_2)$ where

$A_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ and $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$

we shall assume that $Q_1 \cap Q_2 = \phi$.

(i) **Union** : Let $A = (Q, \Sigma, \delta, q_0, F)$, where

$$Q = Q_1 \cup Q_2 \cup \{q_0\}, \; q_0 \notin Q_1 \cup Q_2$$

q_0 is the start state, $\Sigma = \Sigma_1 \cup \Sigma_2$, $F = F_1 \cup F_2$ and δ is defined as follows:

$\delta (q_0, \lambda) = \{q_1, q_2\}$

If $q \in \delta_1(p, a)$ then $q \in \delta (p, a)$

If $q \in \delta_2(p, a)$ then $q \in \delta (p, a)$

It is clear that $T(A) = (L_1 \cup L_2)$

(ii) **Concatenation** : Let $A = (Q, \Sigma, \delta, q_0, F)$ where

$$Q = Q_1 \cup Q_2, \Sigma = \Sigma_1 \cup \Sigma_2, q_0 = q_1, F = F_2 \text{ and}$$

δ is defined as follows :

$$\delta(q, a) = \{\delta_1 (q, a)\} \text{ for each } q \text{ in } Q_1 - F_1$$

$$\delta(q, a) = \{\delta_1 (q, a)\} \text{ for each } q \text{ in } F_1$$

and $\delta(q, a) = \{\delta_2 (q, a)\}$ for each q in Q_2.

A word w is in $L_1 L_2$ if $w = w_1 w_2$ where $w_1 \in L_1$ and $w_2 \in L_2$. This is equivalent to $\delta_1 (q_1, w_1)$ being in F_1 and $\delta_2 (q_2, w_2)$ being in F_2 which is equivalent to $w_1 w_2$ being in $T(A)$. Thus $L_1 L_2 = T(A)$.

(iii) **Closure** : Let $L = T(A)$ be a regular set where

$$A = (Q, \Sigma, \delta, q_0, F)$$

Let $$B = (Q, \Sigma, \delta', q_0, F)$$

where δ' is defined as follows:

$$\delta' (q, a) = \{\delta(q, a)\} \text{ if } q \text{ is in } Q - F \text{ and}$$

$$\delta' (q,a) = \{\delta(q, a), \delta(q_0, a)\} \text{ if } q \text{ is in } F$$

It is easy to see that $L^* = T(B)$. If λ is in L, then it is accepted by B. Otherwise, we can modify B to accept λ.

Theorem

The class of regular sets is closed under complementation. That is if X is a regular set and $X \subseteq \Sigma^* - L$ is a regular set.

Proof

Let X be $X(M)$ for DFA $M = (Q, \Sigma_1, \delta, q_0, F)$ and let $X \subseteq \Sigma^*$. First we assume $\Sigma_1 = \Sigma$, if there are symbols in Σ_1 not in Σ, we may delete all transitions of M on symbols not in Σ. The fact that $X \subseteq \Sigma^*$ assures us that we shall not thereby change the language of M. If there are symobls in Σ not in Σ_1, then none of these symbols appear in words of X. We may therefore introduce a dead state d into M with $\delta(d, a)=d$ for all a in Σ and $\delta(q, a) = d$ for all q in Q and a in $\Sigma - \Sigma_1$. Now to accept $\Sigma^*–X$, complement the final states of M. That is let $M' = (Q, \Sigma, \delta, q_0, Q - F)$.

The M' accepts a word w if and only if $\delta(q_0, w)$ is in $Q–F$, that is, w is in $\Sigma^*–X$. Note that it is essential to the proof that M is deterministic and without y moves.

Theorem

If X and Y are regular sets over Σ, then $X \cap Y$ is also regular.

Proof

$X \cap Y = \overline{X} \cap \overline{Y}$ by Demorgon's law where-denotes complementation with respect to an alphabet including the alphabets of X and Y. Closure under intersection then follows from closure under ~~union~~ and complementation.

Theorem

If L is regular then L^R is also regular.

Proof

As L is regular, we can construct a FA

M = $(Q, \sum_1, \delta, q_0, F)$ such that $T(M) = L$.

We can construct a transition system (or transition diagram or transition graph) M' by starting with the state diagram of M, and reversing the direction of the directed edges. The set of initial states of M' is defined as the set F, and q_0 defined as the (only) final state of M' (i.e) $M' = (Q, \sum, \delta, F, \{q_0\})$.

If $w \in T(M)$, we have a path from q_0, to some final state in F with path value w. By reversing the edegs, we get a pair in M from some final state in F to q_0. Its path value is w^R. So $w^R \in$ T(M). In a similar way, we can see that if $w_1 \in$ T(M'), then $w_1^R \in$ T(M). Thus from the state diagram it is easy to see that $T(M') = T(M)^R$. We can prove that $w \in T(M)$ if $w^R \in T(M')$ by induction on $|w|$. Since $T(M')$ is regular it follows that $T(M)^R$ is regular.

Theorem

The class or regular sets is closed under substitution.

Proof

Let R $\subseteq \sum^*$ be a regular set and for each a in $\sum$ let Ra $\subseteq \Delta^*$ be a regular set. Let $f: \sum \rightarrow \Delta^*$ be the substitution defined by $f(a) = Ra$. Select regular expressions denoting R and each Ra. Replace each occurrence of the symbol a in the regular expression for R by the regular expression for Ra. To prove that the resulting regular expression denotes $f(R)$, observe that the substitution of a union, product, or closure of the substitution.

A simple induction on the number of operators in the regular expression completes the proof.

Theorem

The class of regular sets is closed under homomorphisms and inverse homomorphisms.

Proof

Closure under homomorphisms follows immediately from closure under substitution, since every homomorphism is a substitution in which *h(a)* has one member.

To show closure under inverse homomorphism, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA accepting L, and let h be a homomorphism from Δ to Σ^*. We construct a DFA M′ that accepts *h(L)* by reading symbol a in Δ and simulating M on *h(a)*. Formally, let $M' = (Q, \Delta, \delta', q_0, F)$ and define $\delta'(q, a)$ for q in Q and a in Δ to be $\delta(q, h(a))$. Note that *h(a)* may be a long string or λ but δ is defined on all strings by extension. It is easy to show by induction on $|x|$ that $\delta'(q_0, x) = \delta(q_0, h(x))$. Therefore M' accepts x if and only if M accepts *h(x)*. That is $L(M') = h^{-1}L(M)$.

Definition

Let L_1 and L_2 be two languages. The quotient of L_1 and L_2 denoted by L_1/L_2 is defined as $L_1/L_2 = \{x \mid$ there exists y in L_2 such that xy is in $L_1\}$.

Theorem

The class of regular sets is closed under quotient with arbitrary sets.

Proof

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton accepting some regular set R, and let L be an arbitrary language. The quotient R/L is accepted by a finite automaton $M' = (Q, \Sigma, \delta, q_0, F')$ which behaves like M except that the final states of M' are all states q of M such that there exists y in L for which $\delta(q, y)$ is in F. Then $\delta(q_0, x)$ is in F' if and only if there exists y such that $\delta(q_0, xy)$ is in F. Thus M' accepts R/L.

3.5 DFA MINIMIZATION ALGORITHMS

3.5.1 Myhill - Nerode Theorem

This minimization algorithm finds a DFA M' equivalent to the DFA $M = (Q, \Sigma, \delta, q_0 F)$, with reduced number of states.

Steps

1. Mark the pair of inequivalent states *(p, q)* with 'X'
   (a) Initially 'X' is placed by considering one final state and one non-final state, where $\delta(p, x) \in$ F and $\delta(q, x) \notin$ F.
   (b) If $\delta(p, a) = r$ and $\delta(q, a) = s$ for input symbol a and the states r, s are already distinguishable by some string x, then p, q are distinguished by ax, otherwise (p, q) is placed in a list associated with (r, s) entry.
2. From the unmarked states, each pair of equivalent states are identified.
3. New states of DFA has, pair of equivalent states and the states which are not in the equivalent pairs are individual states.

4. Construct DFA transition table with reduced number of states.

Algorithm for marking pairs of inequivalent states

begin

for p in F and q in $Q–F$ **do** mark (p, q);

for each pair of distinct states (p, q) in $F \times F$ or $(Q–F) \times (Q–F)$ **do**

if for some input symbol a, $(\delta(p, a), \delta(q, a))$ is marked **then**

begin

mark (p, q);

recursively mark all unmarked pairs on the list for (p, q) and on the lists of other pairs that are marked at this step.

end

else /* no pair $(\delta(p, a), \delta(q, a))$ is marked */

for all input symbols a **do**

put (p, q) on the list for $(\delta(p, a), \delta(q, a))$ unless

$\delta(p, a) = \delta(q, a)$

end

Example 3.11

For the Regular Expression = $(a + b)^* ab$, the DFA is constructed in Section 3.2.1, Example 3.4.

Transition table of DFA :

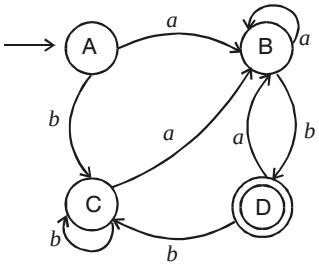
New States	Symbols	
	a	*b*
→ A	B	C
B	B	D
C	B	C
(D)	B	C

Find the minimum number of states for the given DFA and its corresponding transition table.

Solution :

1. The pair of equivalent states are identified by marking ‘X’ in the each pair of inequivalent states.

Transition diagram of DFA :



Marking 'X' in the inequivalcnt states

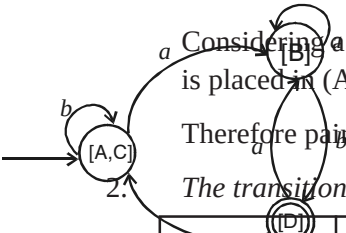
	A	B	C
B	X		
C		X	
D	X	X	X

Initially 'X' is placed by considering *p* in F and *q* in *Q–F.* In the example (D, A) (D, B) and (D, C) are marked 'X', because *p* is distinguishable from *q.* When considering a pair (A, C), (δ(A, *b),* δ(C, *b*)) = (C, C) and (δ(A, *a),* δ(C, *a*)) = (B, B) and hence no string starting with *a* or *b* can distinguish A from C. Therefore it is a equivalent pair.

Considering a pair (A, B), (δ(A, *b),* δ(B, *b*)) = (C, D) is already marked as X, therefore 'X' is placed in (A, B). Similarly (B, C) entry is also marked as 'X'.

Therefore pair of equivalent states are (AC) and the remaining new states are (B) (D).

2. *The transition table with reduced number of states:*



New States	Symbols	
	a	*b*
[A,C]	[B]	[A,C]
[B]	[B]	[D]
[D]	[B]	[A,C]

3. *The equivalent transition diagram is :*

Note :

(i) If $p \equiv q$, we say p is equivalent to q, iff for each input string x, $\delta(p, x)$ is an accepting state, iff $\delta(q, x)$ is an accepting state.

(ii) p is distinguishable from q if there exists an x such that $\delta(p, x)$ is in F and $\delta(q, x)$ is not or vice versa.

3.5.2 Construction of Π_{final} from Π

The algorithm for minimizing the number of states of a DFA, work by finding group of states that can be distinguished by some input string. The states in each group cannot be distinguished from one another, and any pair of states chosen from different groups have been found distinguished by some input.

Steps

1. Initially the states are divided into 2 groups (i.e.) final states and non final states.
2. For each group, repeat the following steps until no more groups can be splitted.
   a) The transition on the input symbols is checked for every state.
   b) If the transition states falls into two or different groups then the group is splitted.

Algorithm for minimizing the number of states of a DFA.

Input : A DFA M with set of states S, set of inputs $\sum$, transitions defined for all states and inputs, start state s_0, and set of accepting states F.

Output : A DFA M' accepting the same language as M and having as few states as possible.

Algorithm

1. Construct an initial partition Π of the set of states with two groups; the accepting states F and the nonaccepting states S–F.
2. Apply the procedure shown below to Π to construct a new partition Π_{new}.

   for each group G of Π do **begin**

   partition G into subgroups such that two states s and t of G are in the same subgroup if and only if for all input symbols a, states s and t have transitions on a to states in the same group of Π;

   /* at worst, a state will be in a subgroup by itself */

   replace G in Π_{new} by the set of all subgroups formed

   end

3. If $\Pi_{new} = \Pi$, let $\Pi_{final} = \Pi$ and continue with setp (4).

   Otherwise, repeat step (2) with $\Pi = \Pi_{new}$.

4. The representatives will be the states of the reduced DFA M′. Let *s* be a representative state, and suppose on input *a* there is a transition of M from *s* to *t*. Let *r* be the representative of *t*'s group (*r* may be *t).* Then M′ has a transition from *s* to *r* on *a*. Let the start state of M′ be the representative of the group containing the start state s_0 of M, and let the accepting states of M′ be the representatives that are in F. Note that each group of Π_{final} either consists only of states in F or has no states in *F.*

5. If M′ has a dead state, that is, a state *d* that is not accepting and that has transitions to itself on all input symobls, then remove *d* from M′. Also remove any states not reachable from the start state. Any transitions to *d* from other states become undefined.

Example 3.12

Find the minimum number of states for the given DFA and its corresponding transition table.

States	Symbols	
	a	*b*
→ A	B	C
B	B	D
C	B	C
(D)	B	C

1. Initial partition Π consists of two groups:

   (Non-final) (Final) i.e. (ABC) (D)

2. To construct Π_{new}, consider the group (D), this group consists of a single state, it can't be split further, so (D) is placed in Π_{new}.When considering the group (ABC), upon an input symbol *a* & *b*:

   δ (A, *a*) = B δ (A, *b*) = C

   δ (B, *a*) = B δ (B, *b*) = D

   δ (C, *a*) = B δ (C, *b*) = C

   On input '*a*' each of the state has a transition to B. So, all states could remain in one group as for as input '*a*' is concerned. On input '*b*', the state B has a transition to D, that belongs to another group. Therefore the Π_{new} at this stage is : (AC) (B) (D)

3. Considering the group (AC), on input '*a*' both the states goes to B and on input '*b*' both the states goes to C. Therefore these groups cannot be splitted in further.

   $\Pi_{final} = \Pi_{new}$ = (AC) (B) (D)

4. Transition table

New States	Inputs	
	a	*b*
→ [A,C]	[B]	[A,C]
[B]	[B]	[D]
((D))	[B]	[A,C]

3.6 SOLVED PROBLEMS

1. Describe the following sets by regular expressions.

(a) {01, 11}

(b) {101}

(c) {λ, 1, 11, 111,.....}

(d) {1, 11, 111,.....}

(e) The set of all strings over {0, 1} which has atmost two 0's.

(f) $\{a^n : n$ is divisible by 2 or 3 or $n=5\}$

(g) The set of all strings over $\{a, b\}$ beginning and ending with a.

(h) $\{1^{2n+1} : n \geq 0\}$

(i) {0, 1, 2}

(j) $\{w \in \{a,b\}^* \mid w$ has only one $a\}$

(k) $\{a^2, a^5, a^8,.....\}$

Solution (R.E) :

(a) R.E = $01 \cup 11$

(b) R.E = 101

(c) R.E = $(1)^*$

(d) R.E = $1(1)^* = 1^+$

(e) The string w may contain no zero, one zero or atmost two zeros.

$\therefore$ The required regular expression is :

$$1^* \quad + \quad 1^*\,01^* \quad + \quad 1^*\,01^*\,01^*$$

(no zero) (one zero) (two zero's)

(f) R.E = $(aa)^*$ + $(aaa)^*$ + $aaa\ aa$

↓ ↓ ↓

divisible 2 divisible by 3 $n = 5$

(g) R.E = $a(a+b)^*\ a$

(h) R.E = $1(11)^*$

(i) R.E = 0+1+2

(j) w in the given set has only one a which can occur any where in w. So $w=x\ a\ y$, where x and y consist of some b's (or none). Hence the given set is represented by b^*ab^*.

(k) R.E = $aa(aaa)^*$

2. Describe in the English language, the sets represented by the following regular expressions.

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

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

(c) $(aa+b)^*\ (bb+a)^*$

Solution :

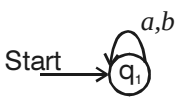
(a) The set of all strings over $\{a,b\}$ starting with a and ending with ab.

(b) The strings are either strings of a's follwed by one b or strings of b's followed by one a.

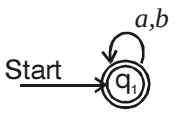
(c) The set of all strings over $\{a,b\}$ of the form vw where a occurs in pairs in v and b occurs in pairs in w.

3. Find the regular expression corresponding to the automaton given below:

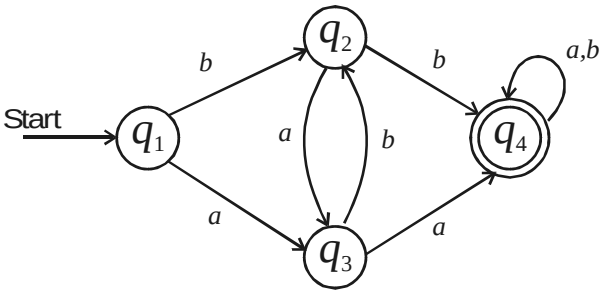
(a)



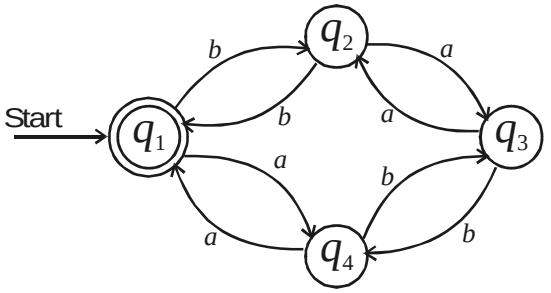
(b)



(c)



(d)



Solution :

(a) R.E = ϕ, because the final state is not mentioned.

(b) R.E = $(a+b)^*$, because the given transition system accepts all the strings over $\{a,b\}$ (i.e) *a,b, ab, ba,.....*

(c) The set of all strings over $\{a,b\}$ containing two successive *a*'s or two successive *b*'s. Some of the accepted strings are *bb, aa, baa, abb,.....*

(d) The set all strings over $\{a,b\}$ containing even number of *a*'s and even number of *b*'s. Some of the accepted strings are *baba, baab, abab,.....*

4. Find regular expressions representing the following sets :

(a) The set of all strings over {0,1} having atmost one pair of 0's or atmost one pair or 1's

(b) The set of all strings over *a* in which the number of occurances of *a* is divisible by 3.

(c) The set of all strings over {0,1} with three consecutive 0's.

(d) The set of all strings over {0,1} beginning with 00.

(e) The set of all strings over {0,1} ending with 00 and beginning with 1.

Solution :

(a) R.E = $(1+01)^* + (1+01)^*00\ (1+01)^* + (0+10)^* + (0+10)^*\ 11(0+10)^*$

$(1 + 01)^*$ - represents the set of all strings containing no pair of 0's.

(1+01)* 00(1+01)* – represents the set of all strings containing exactly one pair of 0's

(0 + 10)* - represents the set of all strings containing no pair of 1's.

(0 + 10)* 11(0 + 10)* - represents the set of all strings containing exactly one pair of 1's.

(b) R.E = *aaa*(*aaa*)*

(c) R.E = (0+1)* 000(0 + 1)*

(d) R.E = 00(0 + 1)*

(e) R.E = 1 (0 + 1)* 00

5. Prove that

(1+00*1)+(1+00*1)(0+10*1)*(0+10*1)=0*1(0+10*1)*

Solution :

L.H.S. =

$$\frac{(1+00*1)}{R}+\frac{(1+00*1)}{R}\frac{(0+10*1)*(0+10*1)}{}$$

= (1+00*1) (λ + (0+10*1)*(0 + 10*1))

using 1_{12}–R(P + Q) = RP + RQ

= (1 + 00*1)(0+10*1)* using I_9–λ + RR*=R*

= (λ + 00*)1(0 + 10*1) using I_{12}–(P+Q)R=PR + QR for 1+00*1

= 0*1(0+10*1)* using I_9–λ + RR*= R*

6. Prove that (*a*+*b*)* = *a**(*ba**)*

Solution :

Let *r* = (*a*+*b*)*

s = *a**(*ba**)*

Hint :

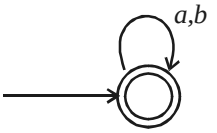
(i) Two regular expressions *r* and *s* are equivalent if and only if the corresponding DFA's are equivalent.

(ii) For every regular expression *r* there exists a NFA with ε-transitions that accepts L(*r*).

(iii) From the NFA with ε transition the ε moves are eliminated to check the equality of two regular expressions.

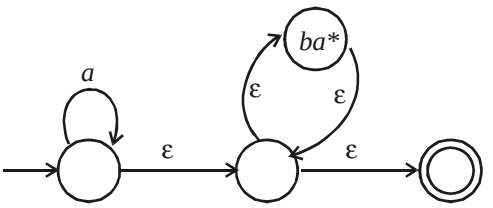
Step 1 : NFA for $r = (a+b)^*$

Step 2 : NFA for $r = (a + b)^*$ (after eliminating ε moves)

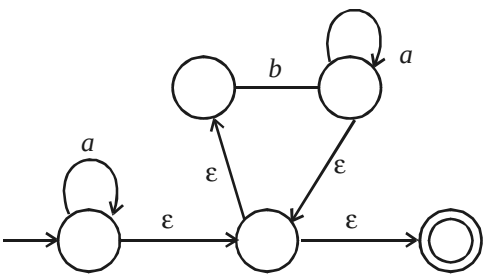


Step 3 : NFA for $s = a^*(ba^*)^*$

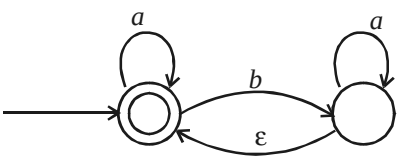
(i)



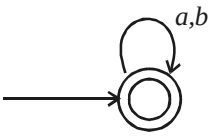
(ii)



(iii)



Step 4: NFA for $s = a^*(ba^*)^*$ (after eliminating ε moves)



Conclusion :

Since the two NFA's (transition systems) are the same, we conclude that $r = s$

7. Construct a DFA with reduced state equivalent to the Regular Expression

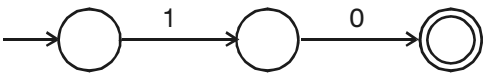
RE = 10 + (0 + 11)0*1.

Solution :

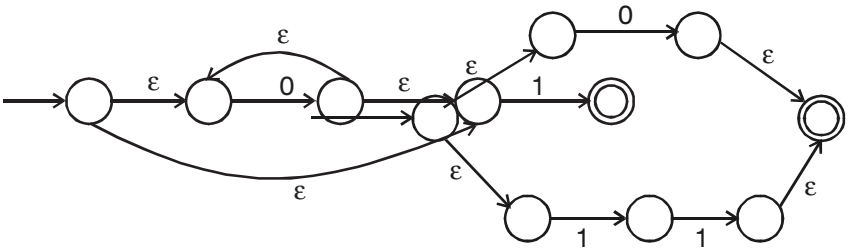
1. Construct a transition system using Thompson's contruction. (NFA with ε transitions)
2. Find the equivalent NFA without ε moves.
3. Construct a transition table of NFA without ε moves.
4. From step 3 construct a transition table of DFA and draw its equivalent transition diagram.

Step 1 (NFA with ε transitions) :

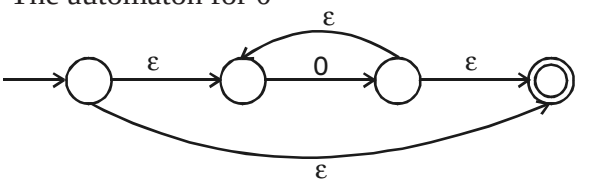
(i) The automaton for 10



(ii) The automaton for 0+11



(iii) The automaton for 0*

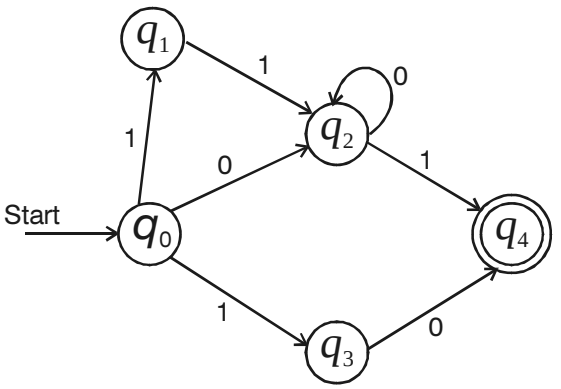


(iv) The automaton for 0*1

(v) The automaton for (0+11)0* 1

(vi) The automaton for 10+(0+11)0*1

Step 2 (NFA without ε moves) :



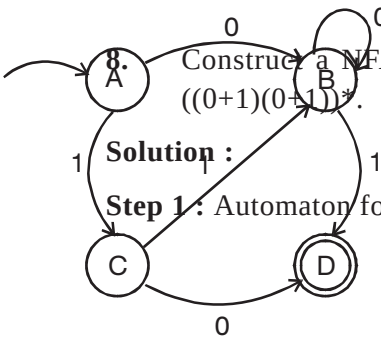
Step 3 (Transition table of NFA) :

States	Inputs	
	0	1
q_0	$\{q_2\}$	$\{q_1, q_3\}$
q_1	–	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_4\}$
q_3	$\{q_4\}$	–
q_4	–	–

Step 4 (Transition table of DFA) :

New states	States	Inputs	
		0	1
A	$[q_0]$	$[q_2]$	$[q_1, q_3]$
B	$[q_2]$	$[q_2]$	$[q_4]$
C	$[q_1, q_3]$	$[q_4]$	$[q_2]$
D	$[q_4]$	–	–

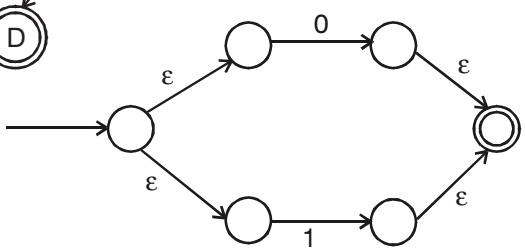
Transition diagram :



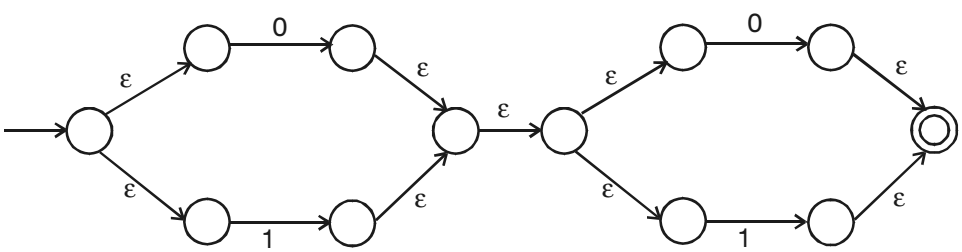
8. Construct a NFA for the given regular expression using Thompson's construction ((0+1)(0+1))*.

Solution :

Step 1 : Automaton for 0+1



Step 2 : Automaton for (0+1) (0+1)



Step 3 : Automaton for ((0+1)(0+1))*

.

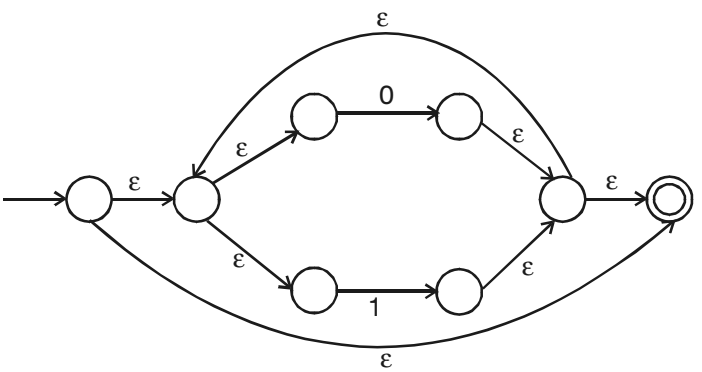
9. Construct an NFA equivalent to (0+1)* (00+11) **(April/May 2004)**

Solution :

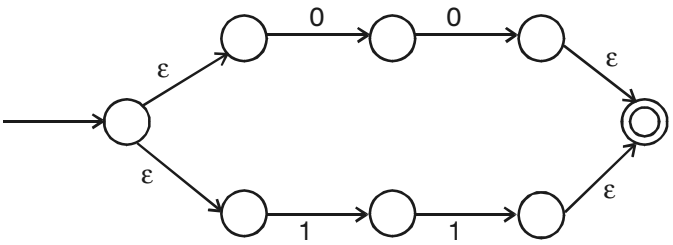
Step 1 (NFA with ε transitions) :

(i) Automaton for (0+1)

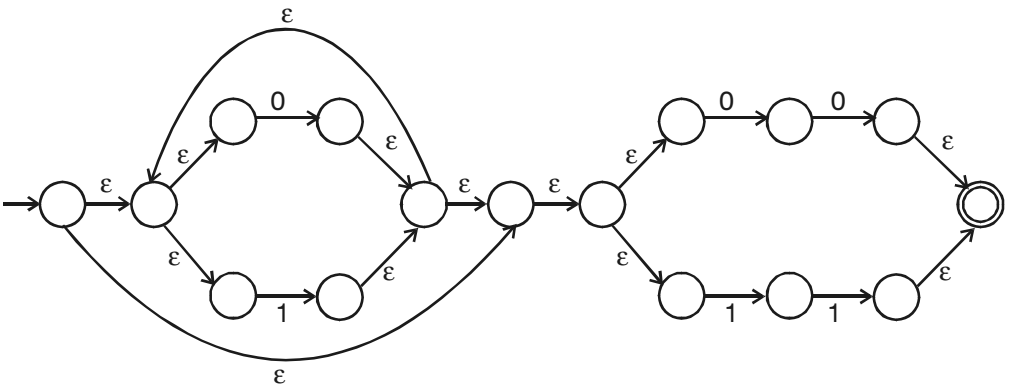
(ii) Automaton for (0+1)*



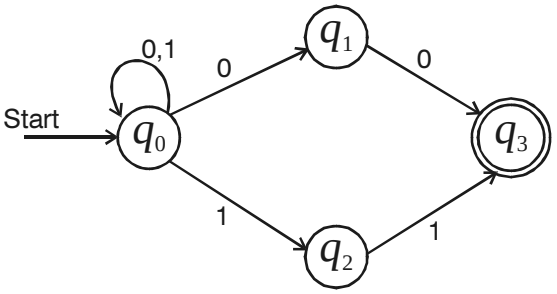
(iii) Automaton for (00+11)



(iv) Automaton for (0+1)* (00+11)



Step 2 (NFA without ε transitions):



10. Show that the set L = $\{a^{i^2} \mid i \geq 1\}$ is not regular.

Solution : **(Nov/Dec 2003)**

Method 1

Step 1 :

Suppose L is regular and we get a contradiction. Let *n* be the number of states in FA accepting L.

Step 2 :

Let $w = a^{n^2}$. Then $| w | = n^2 > n$. By pumping lemma we can write $w = xyz$ with $| xy | \leq n$ and $| y | > 0$.

Step 3 :

Consider xy^2z. $|xy^2z| = |x| + 2|y| + |z| > |x| + |y| + |z|$ as $|y| > 0$.

This means $n^2 = |xyz| = |x| + |y| + |z| < |xy^2z|$. As $|xy| \leq n$, $|y| < n$. Therefore, $|xy^2z| = |x| + 2|y| + |z| \leq n^2 + n$,

i.e., $n^2 < |xy^2z| \leq n^2 + n < n^2 + n + n + 1$

Hence, $|xy^2z|$ strictly lies between n^2 and $(n + 1)^2$, but it is not equal to any one of them. Thus $|xy^2z|$ is not a perfect square and so $xy^2z \notin$ L. But by pumping lemma $xy^2z \in$ L. This is a contradition.

Method 2

Assume L is regular.

Consider z to be any string in L

Let $z = a^{p^2}$

$= a^r$ where $r = p^2$

According to pumping lemma, z can be written as

UVW such that

$|UV| \leq n$, $|V| \geq 1$.

Therefore, a^r can be expressed in the form of UVW, where

$UV = a^m$ where $m < r$

$V = a^q$ where $q < m$

and W = a^{r-m} since $UV = a^m$ $|V| \geq 1$

UV^iW can be expressed as

$UV\,V^{i-1}W$ ..... (1)

Substituting the values of UV, V and W in (1)

$UVV^{i-1}W = (a^m)\,(a^q)^{i-1}\,a^{r-m}$

$= a^{m+q(i-1)+r-m}$

$UV^iW = a^{q(i-1)+r}$

To prove that any language to be regular we have to show

$UV^iW \in$ L $\forall$ i

Since we have assumed the given language to be regular UV^iW, should be in $L \ \forall \ i$

for $i = 1$, $UV^iW = a^{q(i-1)+r}$

$= a^{q(1-1)+r} = a^r = a^{p^2}$

where p^2 is perfect square.

for $i = 2$, $UV^iW = a^{q(i-1)+r}$

$= a^{q(2-1)+r}$

$= a^{q+r}$

$= a^{q+p^2} \notin L$

Since $q + p^2$ cannot be perfect square.

This result leads to contradiction to our assumption that given L is regular.

Therefore, $L = \{a^{i^2} \mid i \geq 1\}$ is not regular.

11. Show that $L = \{a^p \mid p$ is a prime$\}$ is not regular.

Solution :

Method 1

Step 1:

We suppose L is regular and get a contradiction. Let n be the number of states in the FA accepting L.

Step 2 :

Let p be a prime number greater than n. Let $w = a^p$. By pumping lemma, w can be written as $w = xyz$, with $|xy| \leq n$ and $|y| > 0$. x, y, z are simply strings of a's. So, $y = a^m$ for some $m \geq 1$ (and $\leq n$).

Step 3:

Let $i = p + 1$. Then $|xy^iz| = |xyz| + |y^{i-1}| = p + (i - 1)\ m = p + pm$. By pumping lemma, $xy^iz \in L$. But $|xy^iz| = p + pm = p\ (1 + m)$ and $p(1 + m)$ is not a prime. So $xy^iz \notin L$. This is a contradiction. Thus L is not regular.

Method 2

Let $z = a^K \in L$

representing $z = UVW$

such that $|UV| \leq K$

$|V| \geq 1$

Let $UV = a^m$ where $m < K$

$V = a^j$ where $j < m$

$$W = a^{K-m}$$

$$UV^iW = UVV^{i-1}W$$

$$= a^m\, a^{j(i-1)}\, a^{k-m}$$

$$= a^{m+j(i-1)+K-m}$$

$$= a^{j(i-1)+K}$$

For $i = 0$,

$$UV^iW = a^{j(0-1)+K}$$

$$= a^{K-j} \notin L \text{ since } K-j \text{ is not prime}$$

Hence the given language is not regular.

12. Show that L = $\{0^i1^i | i \geq 1\}$is not regular. **(Apr/May 2004)**

Solution :

Method 1

Step 1 :

Suppose L is regular and we get a contradiction. Let n be the number of states in FA accepting L.

Step 2 :

Let $w = 0^n1^n$. Then $|w| = 2n > n$. By pumping lemma we write $w = xyz$ with $|xy| \leq n$ and $|y| \neq 0$.

Step 3 :

We want to find i so that $xy^iz \notin L$ for getting a contradiction. The string y can be in any of the following forms:

Case 1 : y has 0's, i.e. $y = 0^k$ for some $k \geq 1$.

Case 2 : y has only 1's, i.e. $y = 1^l$ for some $l \geq 1$.

Case 3 : y has both 0's, and 1's i.e. $y = 0^k1^j$ for some $k,j \geq 1$.

In case 1, we can take $i = 0$. As $xyz = 0^n1^n$, $xz = 0^{n-k}1^n$. As $k \geq 1$, $n - k \neq n$. So $xz \notin L$.

In case 2, take $i = 0$. As before, xz is 0^n1^{n-1} and $n \neq n-1$. So $xz \notin L$.

In case 3, take $i = 2$. As $xyz = 0^{n-k}\, 0^k\, 1^j\, 1^{n-j}\, 1^{n-j}$, $xy^2z = 0^{n-k}\, 0^k\, 1^j\, 0^k\, 1^j\, 1^{n-j}$. As xy^2z is not of the form 0^i1^i, $xy^2z \notin L$.

Thus in all the cases we get a contradiction. Therefore, L is not regular.

Method 2

Let $z = 0^K\, 1^K \in L$ where $|z| \geq K$

representing $z = UVW$

such that $|UV| \leq K$ & $|V| \geq 1$

Let $UV = 0^m$ where $m \leq K$

$V = 0^j$ where $j < m$

$W = 0^{K-m}\ 1^K$

$UV^iW = UVV^{i-1}W$

$= 0^m\ 0^{j(i-1)}\ 0^{K-m}\ 1^K$

$= 0^{m+j(i-1)+K-m}\ 1^K$

$= 0^{j(i-1)+K}\ 1^K$

For $i = 1$, $UV^iW = 0^{j(1-1)+K}\ 1^K$

$= 0^K\ 1^K \in L$

For $i = 2$ $UV^iW = 0^{j(2-1)+K}\ 1^K$

$= 0^{K+j}\ 1^K \notin L$. Since the given language should contain equal number of 0's & 1's

Hence the given language is not regular.

13. Show that L = $\{ww | w \in \{a, b\}^*\}$ is not regular

Solution :

We prove the result by the method of contradiction.

Step 1 :

Suppose L is regular. Let n be the number of states in the automaton M accepting L.

Step 2 :

Let us consider $ww = a^nba^nb$ in L. $|ww| = 2(n+1) > n$. We can apply pumping lemma to write $ww = xyz$ with $|y| \neq 0$, $|xy| \leq n$.

Step 3 :

We want to find i so that $xy^iz \notin L$ for getting a contradiction. The string y can be in only one of the following forms:

Case 1 : y has no b's, i.e. $y = a^k$ for some $k \geq 1$.

Case 2 : y has only one b.

We may note that y cannot have two b's. If so, $|y| \geq n + 2$. But $|y| \leq |xy| \leq m = n - k < n$ (or a^nbc^mb). We cannot write xz in the form uu with $u \in \{a, b\}^*$, and so $xz \notin L$. In case 2 also, we can take $i = 0$. Then $xy^0z = xz$ has only one b (as one b is removed from xyz,

b being in y). So $xz \notin L$ as any element in L should have even number of a's and even number of b's.

Thus in both the cases we get a contradiction. Therefore L is not regular.

Note :

If a set L of strings over Σ is given and if we have to test whether L is regular or not, we try to write a regular expression representing L using the definition of L. If this is not possible, we use pumping lemma to prove that L is not regular.

14. Is $L = \{a^{2n} | n \geq 1\}$ regular?

Solution :

Method 1

We can write a^{2n} as $a(a^2)^i a$, where $i \geq 0$. Now $\{(a^2)^i \mid i \geq 0\}$ is simply $\{a^2\}^*$. So L is represented by the regular expression $a(P)^*a$, where P represents $\{a^2\}$. The corresponding FA (using the construction) is given below:

Method 2

$L = \{a^{2n} | n \geq 1\}$

Consider a string z in L

Let $z = a^{2m} \in L$ where $m \geq 1$

$= a^{2m} = a^P$ where $p = 2m$

rewriting z in form of UVW, such that $|UV| \leq n$, $|V| \geq 1$

Let $UV = a^r$ where $r \leq p$

$V = a^q$ where $q < r$

$W = a^{p-r}$

UV^iW can be expressed as $UVV^{i-1}W$

$$UVV^{i-1}W = a^r a^{q(i-1)} a^{p-r}$$
$$= a^{r+q(i-1)+p-r}$$
$$= a^{p+q(i-1)}$$

For $i = 1$,

$$UV^iW = a^{p+q(i-1)}$$

$= a^{p+q(1-1)} = a^p = a^{2m} \in L$

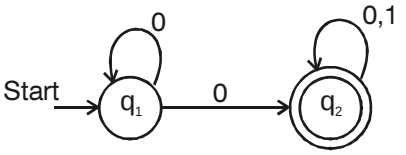
For $i = 2$

$UV^iW = a^{p+q(2-1)} = a^{p+q} \notin L$

Because $p + q = 2\ m+q$ is not in multiple of 2.

Hence $L = \{a^{2n} \mid n \geq 1\}$ is not regular.

15. Find regular expression for the given automata (using Arden's Theorem).



Solution :

We can obtain regex (or) regular expression by applying Arden's theorem.

Step 1 : (i) Check whether FA does not have ε-moves

(ii) It has only one start state

Step 2 : Express states interms of transitions

The transitions that required to reach q_1 from other states is

$q_1 = q_1 1 + \varepsilon$ .....①

Similarly for

$q_2 = q_2 0 + q_2 1 + q_1 0$ .....②

Since q_1 is in LHS & RHS (R = Q +RP*) we can write it has

$q_1 = \varepsilon 1^*$ (R = QP*)

Now $q_2 = q_2(0+1) + q_1 0$

$= q_1 0(0 + 1)^*$ (Since q_2 is in LHS & RHS)

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

Since q_2 is a final state R.E = 1*0 (0+1)*

16. Given $L = \{a^i b^j c^k \mid k \geq i+j,\ i, j \geq 1\}$. Show that L is not regular.

Solution :

Let $z = a^m b^n c^o$ where $o \geq m+n$

Let $UV = a^p$ where $p \leq m$

$V = a^q$ where $q < p$ & $q \geq 1$

$W = a^{m-p}\ b^n\ c^o$

$UV^iW = UV\ V^{i-1}W$

$= a^p\ (a^q)^{i-1}\ a^{m-p}\ b^n\ c^o$

$= a^p\ a^{q(i-1)}\ a^{m-p}\ b^n\ c^o$

$= a^{p+q(i-1)+m-p}\ b^n\ c^o$

$= q^{m+q(i-1)}\ b^n\ c^o$

Consider some random i, for example $i = 12$

$UV^iW \quad = a^m+q^{(12-1)}\ b^n\ c^o$

$= a^{m+11q}\ b^n\ c^o$

Take random values of q, n, o, m

Consider $m = 4$, $n = 3$, $0 = 10$, $q = 1$

Where $o \geq m+n$

Substituting these values in UV^iW for $i = 12$,

we get

$a^{11+4}\ b^3\ c^{10} = a^{15}\ \ b^3\ \ c^{10}$ where the condition $o \geq m+n$ does not hold.

Hence, the given language is not regular.

17. Show that the set $L = \{0^{j^2} \mid$ j is an integer, $j \geq 1\}$ which consists of all strings of 0's and whose length is a perfect square is not regular.

Solution :

Same as problem No. 10.

18. Show that $L = \{0^{2n} \mid n \geq 1\}$ is not regular.

Solution :

Same as problem No.14

19. $L = \{a^n\ b^n \mid n \geq 1\}$. Prove that L is not regular.

Solution :

Same as problem No. 12.

20. Show that the set of palindromes over alphabet $\{a, b\}$ is not regular i.e., $L = \{a^m\ b\ a^m\}$ is not regular.

Solution :

Let $z = a^m\ b\ a^m$ where $|z| \geq m$

$UV = a^j$ where $j \leq m$

$V = a^n$ where $n < j$

$W = a^{m-j}\ b\ a^m$

$UV^iW = UVV^{i-1}W$

$= a^j\ (a^n)^{i-1}\ a^{m-j}\ b\ a^m$

$= a^{j+n(i-1)+m-j}\, b\, a^m$

$= a^{m(i-1)+m}\, b\, a^m$

For $i = 0$,

$UV^iW = a^{n(i-1)+m}\, b\, a^m$

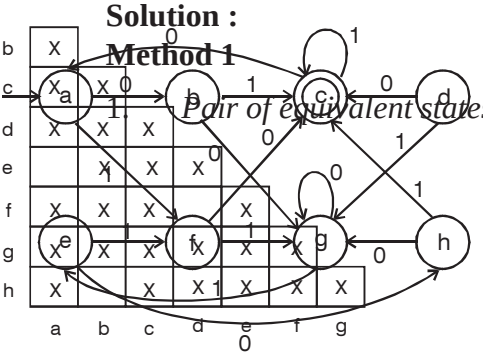
$= a^{n(0-1)+m}\, b\, a^m$

$= a^{m-n}\, b\, a^m \notin$ L, since given language is palindrome. Therefore the language is not regular.

21. Find the minimum number of states and the corresponding transition table for the given DFA.

Solution :

Method 1



Pair of equivalent states are identified by placing 'X' in the inequivalent pairs.

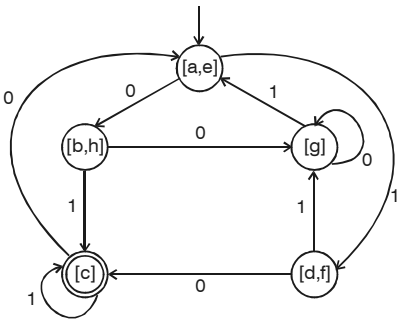
Initially X is placed in each entry corresponding to F and *Q–F*. They are (a, c), (b, c), (c, d), (c, e), (c, f), (c, g) and (c, h). Considering a pair (a, b), X is placed in that entry. Since (δ(b, 1), δ(a, 1)) = (c, f) already has an X. Similarly, the (a, d) entry receives an X, since (δ(a, 0), δ(d, 0)) = (b, c) has an X. The pair (a, e) on input 0 results (b, h) and on input 1 results the same sate f. Therefore it is an equivalent pair. Similarly, other equivalent pairs are identified.

On completion of the table, the equivalent states are a = e, b = h, d = f and the remaining new states are g, c.

2. *DFA transition table with reduced number of states:*

New states	Symbols	
	0	1
→ [a, e]	[b, h]	[d, f]
[b, h]	[g]	[c]
((c))	[a, e]	[c]
[d, f]	[c]	[g]
[g]	[g]	[a, e]

3. *The equivalent transition diagram:*



Method 2

1. Initially the states are divided into two groups :
   (Non-final) (Final)
   (abdefgh) (c)

2. To find $\prod_{new}$ the group (abdefgh) is checked on transition '0' and 1.

   $\delta(a,0) = b$ $\delta(a,1) = f$
   $\delta(b,0) = g$ $\delta(b,1) = c$
   $\delta(d,0) = c$ $\delta(d,1) = g$
   $\delta(e,0) = h$ $\delta(e,1) = f$
   $\delta(f,0) = c$ $\delta(f,1) = g$
   $\delta(g,0) = g$ $\delta(g,1) = e$
   $\delta(h,0) = g$ $\delta(h,1) = c$

   On considering the input '0', the states d and f, goes to another group c. There fore is:
   (abegh) (df) (c)
   On considering the input '1', the states b and h, goes to another group c.

   $\prod_{new}$ = (aeg) (bh) (df) (c)

3. For the gorup (aeg), check transition on input '0' and '1'

$\delta(a,0) = b$ $\delta(a,1) = f$

$\delta(e,0) = h$ $\delta(e,1) = f$

$\delta(g,0) = g$ $\delta(g,0) = e$

On considering the input '0' the states a and e, goes to another group (bh) but the state g remains in the same group. Therefore the group (aeg) is splitted as:

Π_{new} = (ae) (g) (bh) (df) (c).

These groups can't be splitted further.

$\therefore \Pi_{final} = \Pi_{new}$ = (ae) (g) (bh) (df) (c).