Unit-II Regular Expressions and Finite Automata

Regular Expressions:

Regular Expressions are an algebraic way to describe languages. Regular Expressions describe the languages accepted by finite automata i.e., they describe exactly the regular languages.

We define a regular expression over an alphabet Σ recursively as follows.

1. Any terminal symbol (i.e., an element of Σ), ϵ and $\emptyset$ are regular expressions.

2. The union of two regular expressions R_1 and R_2 written as R_1+R_2 ,is also a regular expression.

3. The concatenation of two regular expressions R_1 and R_2 written as $R_1 R_2$, is also a regular expression.

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

5. If R is a regular expression, then (R) is also a regular expression.

6. Any expression over Σ obtained recursively by the application of the above rules (1) – (5) is also a regular expression.

Regular Sets:

Any set represented by a regular expression is called a *regular set (or) regular Language*. The set represented by *R* is denoted by L(R), Let a, b ∈ Σ. Then

1. **a** denotes the set {a}
2. **a** + **b** denotes the set {a,b}
3. **ab** denotes the set {ab}
4. **a*** denotes the set { ε,a,aa,aaa,...........}
5. {**a** + **b**}* denotes {a,b}*

Note 1:

$$L(\mathbf{R}_1 + \mathbf{R}_2) = L(\mathbf{R}_1) \cup L(\mathbf{R}_2), \qquad L(\mathbf{R}_1\mathbf{R}_2) = L(\mathbf{R}_1)L(\mathbf{R}_2)$$

$$L(\mathbf{R}^*) = (L(\mathbf{R}))^*$$

Note 2:

$$L(\mathbf{R}^*) = (L(\mathbf{R}))^* = \bigcup_{n=0}^{\infty} L(\mathbf{R})^n$$

$$L(\emptyset) = \emptyset, \qquad L(\mathbf{a}) = \{a\}.$$

Note 3:

By the definition of regular expressions, the class of regular sets over Σ is closed under union, concatenation and closure (iteration)

Note 4:

In the absence of parentheses, we have the hierarchy of operations as follows: iteration (closure). Concatenation and union. That is, in evaluating a regular expression involving various operations, we perform iteration first, then concatenation, and finally union.

Ex:

1. Describe the following sets by regular expressions.

(a) {01, 10} (b) {ϵ, 0, 00, 000, 0000,...............}

(c) {1, 11, 111, 1111,............} (d) {ϵ, 11, 1111, 111111, ,...............}

Sol:

(a) As {01, 10} is the union of {01} and {10}, we have {01,10} is represented by **01 + 10**

(b) {ϵ, 0, 00, 000, 0000,...............} is represented as **0***

(c) As {1, 11, 111, 1111, ...} is obtained by concatenating 1 and any element of {1}*, the regular expression is **1(1)* = 1**$^{+}$

(d) Any element of {ϵ, 11, 1111, 111111, ,...............} is either ϵ or a string of even number of 1's. So the corresponding regular expression is **(11)***

Ex:

Describe the following by RE

a) L_1 = the set of all strings of 0's and 1's ending in 00.

b) L_2 = the set of all strings of 0's and 1's beginning with 0 and ending with 1

Sol:

a) $R_1 = (0+1)^*\,00$

b) $R_2 = 0(0+1)^*\,1$

Ex:

Write a RE which denotes a language, L, over the set, Σ = {1}, having odd length of strings

Sol:

The Language L is given as

L = {1, 111, 11111, 1111111,....}

The corresponding regular expression is

R = (11)*1

Ex:

Describe the following by the RE over the set $\Sigma = \{0,1\}$.

a) The language of all strings containing exactly two 0's

b) The language of all strings containing at least two 0's

c) The language of all string that do not end with 01

sol:

a) $1^* 0\, 1^* 0\, 1^*$ b) $1^* 0\, 1^* 0\, (0+1)^*$ c) $(0+1)^*(00+11+10)$

Identity Rules:

The Identity rules for simplifying the regular expressions are given below.

I_1 $\emptyset + \mathbf{R} = \mathbf{R}$

I_2 $\emptyset\mathbf{R} = \mathbf{R}\emptyset = \emptyset$

I_3 $\Lambda\mathbf{R} = \mathbf{R}\Lambda = \mathbf{R}$

I_4 $\Lambda^* = \Lambda$ and $\emptyset^* = \Lambda$

I_5 $\mathbf{R} + \mathbf{R} = \mathbf{R}$

I_6 $\mathbf{R}^*\mathbf{R}^* = \mathbf{R}^*$

I_7 $\mathbf{R}\mathbf{R}^* = \mathbf{R}^*\mathbf{R}$

I_8 $(\mathbf{R}^*)^* = \mathbf{R}^*$

I_9 $\Lambda + \mathbf{R}\mathbf{R}^* = \mathbf{R}^* = \Lambda + \mathbf{R}^*\mathbf{R}$

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

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

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

Arden's theorem:

Let P and Q be two regular expressions over Σ. If P does not contain ϵ, then the following equation in R, namely

$$\mathbf{R = Q + RP}$$

has a unique solution (i.e. one and only one solution) given by

$$\mathbf{R = QP^*.}$$

Proof:

The given equation is

$$R = Q + RP \rightarrow (1)$$

We have

$$Q + (QP^*) P = Q (\epsilon + P^*P)$$
$$= QP^* \quad (\text{since } \epsilon + R^*R = R^*)$$

Therefore, QP^* is a solution of the equation $\mathbf{R = Q + RP}$

Uniqueness:

Replacing R by Q + RP on the R.H.S of equation (1), we get

$$\begin{aligned}
Q+RP &= Q + (Q + RP) P \\
&= Q + QP + RPP \\
&= Q + QP + RP^2 \\
&= Q + QP + QP^2 + \ldots\ldots\ldots\ldots + QP^i + RP^{i+1} \\
&= Q (\epsilon + P + P^2 + \ldots\ldots\ldots\ldots + P^i) + RP^{i+1} \rightarrow (2)
\end{aligned}$$

From (1) and (2)

$$R = Q (\epsilon + P + P^2 + \ldots\ldots\ldots\ldots + P^i) + RP^{i+1} \text{ for } i \geq 0 \rightarrow (3)$$

We now show that any solution of equ.(1) is equivalent to QP^*.

Suppose that R satisfies equ.(1) then it satisfies equ.(2). Let w be a string of length i in the set R. Then w belongs to the set $Q (\epsilon + P + P^2 + \ldots\ldots\ldots\ldots + P^i) + RP^{i+1}$. As P does not contain ϵ, RP^{i+1} has no string of length less than i+1 and so w is not in the set RP^{i+1}. This means that w belongs to the set $Q (\epsilon + P + P^2 + \ldots\ldots\ldots\ldots + P^i)$ and hence QP^*.

Consider a string w in the set QP^*. Then w is in the set QP^k for some $k \geq 0$ and hence $Q (\epsilon + P + P^2 + \ldots\ldots\ldots\ldots + P^i)$. Therefore, w is in R.

Thus R and QP^* represent the same set. This proves the uniqueness of the solution of equ.(1)

Manipulation of Regular Expressions:

Show that $R = \epsilon + 1^*(011)^*(1^*(011)^*)^* = (1+011)^*$

Sol:

Given Regular expression is

$$R = \epsilon + 1^*(011)^*(1^*(011)^*)^*$$

$$= \epsilon + P_1P_1^* \text{ where } P1 = 1^*(011)^*$$

$$= P_1^* \quad (\text{since } \epsilon + PP^* = P^* \text{ by known identity})$$

$$= (1^*(011)^*)^*$$

$$= (P^*Q^*)^* \quad \text{where } P = 1 \text{ and } Q = 011$$

$$= (P + Q)^* \quad (\text{by known identity})$$

$$= (1+011)^*$$

Prove that $(1 + 00^*1) + (1+ 00^*1)(0 + 10^*1)^*(0 + 10^*1) = 0^*1(0 + 10^*1)^*$

Sol:

$$\text{L.H.S.} = (1 + 00^*1) + (1+ 00^*1)(0 + 10^*1)^*(0 + 10^*1)$$

$$= (1 + 00^*1)\,(\epsilon + (0 + 10^*1)^*(0 + 10^*1))$$

$$= (1 + 00^*1)\,(\epsilon + P^*P) \text{ where } P = (0 + 10^*1)$$

$$= (1 + 00^*1)\,(0 + 10^*1)^* \quad (\text{since } \epsilon + P^*P = P^*)$$

$$= (\epsilon + 00^*)1(0 + 10^*1)^*$$

$$= 0^*1(0 + 10^*1)^* \quad (\text{since } \epsilon + PP^* = P^*)$$

$$= \text{R.H.S.}$$

Finite Automata and Regular Expressions:

Conversion of FiniteAutomata to Regular Expression:

To find the regular expression recognized by the transition system, the following assumptions are made

i) The transition graph does not have ϵ - moves

ii) It has only one initial state, say , V_i

iii) Its vertices are $V_1, V_2, \ldots\ldots\ldots, Vn$

iv) V_i the r.e. represents the set of strings accepted by the system even though V_i is a final state.

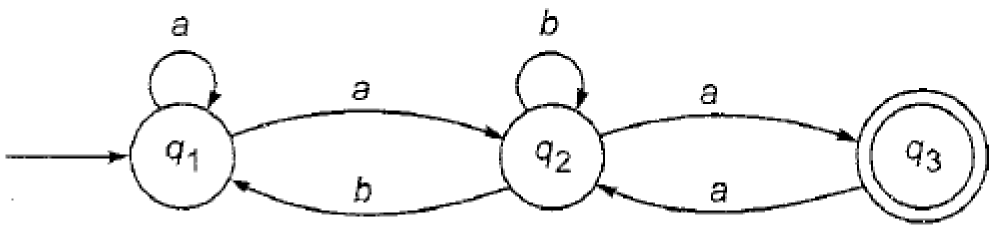
(v) α_{ij} denotes the r.e. representing the set of labels of edges from v_i to v_j. When there is no such edge, $\alpha_{ij} = \emptyset$. Consequently, we can get the following set of equations in $V_1 \ldots V_n$:

$$\begin{aligned} V_1 &= V_1\alpha_{11} + V_2\alpha_{21} + \cdots + V_n\alpha_{n1} + \Lambda \\ V_2 &= V_1\alpha_{12} + V_2\alpha_{22} + \cdots + V_n\alpha_{n2} \\ &\vdots \\ V_n &= V_1\alpha_{1n} + V_2\alpha_{2n} + \cdots + V_n\alpha_{nn} \end{aligned}$$

By repeatedly applying substitutions and Arden's theorem, we can express V_i in terms of α_{ij}'s.

For getting the set of strings recognized by the transition system, we have to **take the 'union" of all V_i's corresponding to final states**.

Ex: Find the regular expression recognized by the following Transition system



Sol:

The three equations for q_1, q_2 and q_3 can be written as

$$q_1 = q_1a + q_2b + \Lambda, \quad q_2 = q_1a + q_2b + q_3a, \quad q_3 = q_2a$$

Substituting q_3 in q_2 and by using Arden's theorem, we get

$$\begin{aligned} q_2 &= q_1a + q_2b + q_2aa \\ &= q_1a + q_2(b + aa) \\ &= q_1a(b + aa)^* \end{aligned}$$

Substituting q_2 in q_1, we get

$$\begin{aligned} q_1 &= q_1a + q_1a(b + aa)^*b + \Lambda \\ &= q_1(a + a(b + aa)^*b) + \Lambda \end{aligned}$$

Hence,

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

$$q_2 = (a + a(b + aa)^*b)^* a(b + aa)^*$$

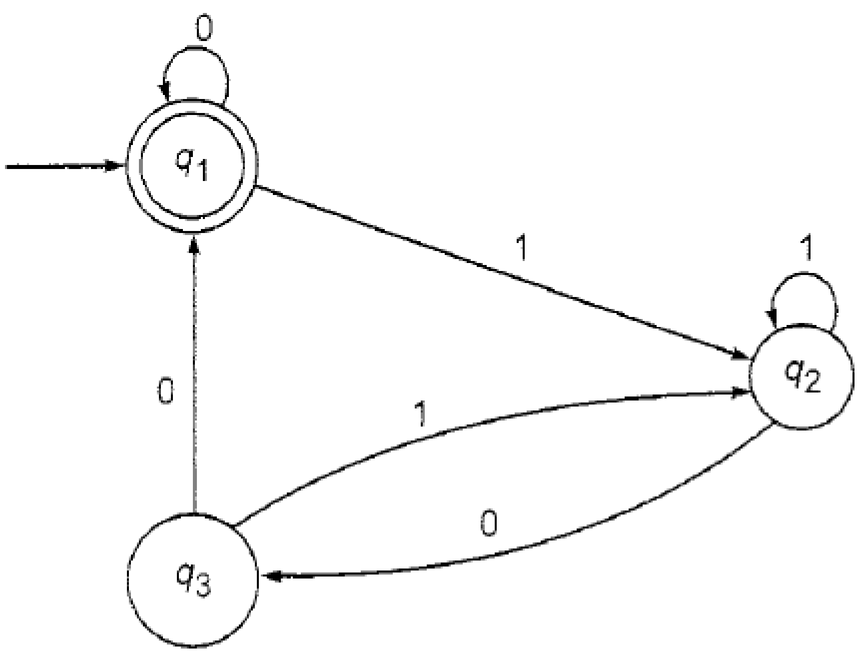
$$q_3 = (a + a(b + aa)^*b)^* a(b + aa)^*a$$

Since q_3 is a final state, the set of strings recognized by the graph is given by

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

Ex:

Find the regular expression recognized by the following Transition system



Sol:

The equations for the states q_1, q_2, q_3 are

$$q_1 = q_1 0 + q_3 0 + \Lambda$$

$$q_2 = q_1 1 + q_2 1 + q_3 1$$

$$q_3 = q_2 0$$

So,

$$q_2 = q_1 1 + q_2 1 + (q_2 0)1 = q_1 1 + q_2(1 + 01)$$

By using Arden's Theorem, we get

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

Also,

$$q_1 = q_1 0 + q_3 0 + \Lambda = q_1 0 + q_2 00 + \Lambda$$

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

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

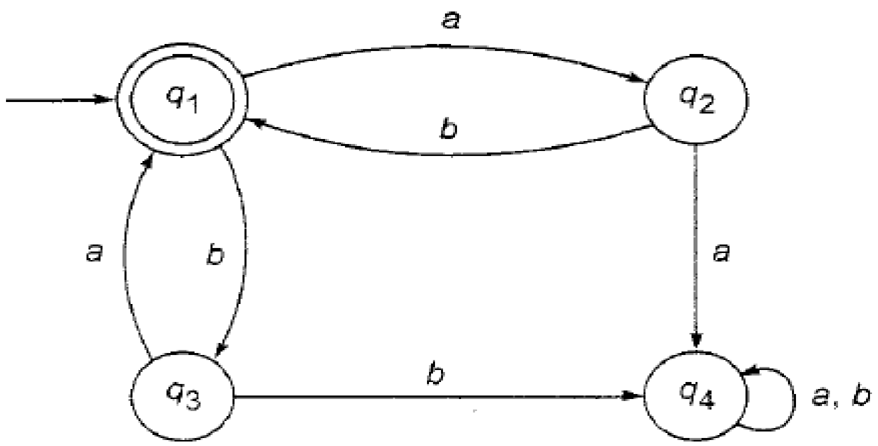
Again by using Arden's theorem, we get

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

As q_1 is the only final state, the regular expression corresponding to the given diagram is $(0 + 1(1 + 01)^* 00)^*$.

Ex:

Find the regular expression recognized by the following Transition system



Sol: The equations for q1, q2, q3, q4 are

$$q_1 = q_2b + q_3a + \Lambda$$

$$q_2 = q_1a$$

$$q_3 = q_1b$$

$$q_4 = q_2a + q_3b + q_4a + q_4b$$

As q1 is the final state and the equation for q1 involves q2 and q3, we use only q2 and q3-equations.(we can neglect q4).

Substituting equations for q2 and q3 in q1-equation , we get

$$q_1 = q_1ab + q_1ba + \Lambda = q_1(ab + ba) + \Lambda$$

By using Arden's Theorem, we get

$$q_1 = \Lambda(ab + ba)^* = (ab + ba)^*$$

As q1 is the final state, the regular expression corresponding to the given transition diagram is

$$(ab + ba)^*$$

Converting Regular Expressions to ε- NFA:

Theorem:

If R is a regular expression over Σ representing $L=L(R) \subseteq \Sigma^*$, then there exists an NFA A with ε-moves such that $L = L(A)$.

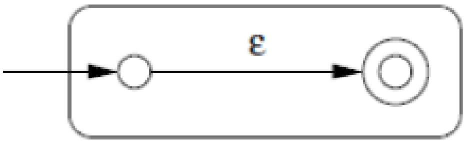
Note:

The proof is by structural induction on R following the recursive definition of regular expressions.

There are 3 base cases.

a) Regular Expression R = ε — **L(ε) = {ε}**

NFA A:



L(A) = {ε}

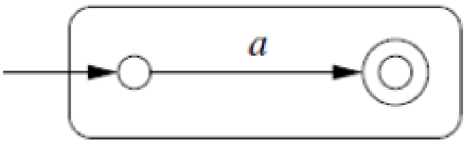
b) Regular Expression R = ϕ — **L(ϕ) = {}**

NFA A:

L(A) = {}

c) Regular Expression R = a ∈ Σ — **L(a) = {a}**

NFA A:

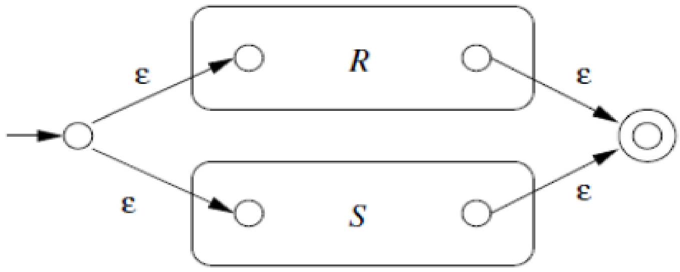


L(A) = {a}

Induction Case: R + S

Regular Expression: R + S **L(R+S) = L(R) ∪ L(S)**

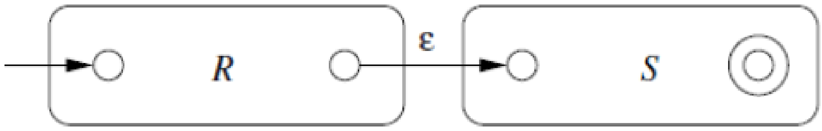
NFA A:



Induction Case: R S

Regular Expression: R S **L(RS) = L(R) L(S)**

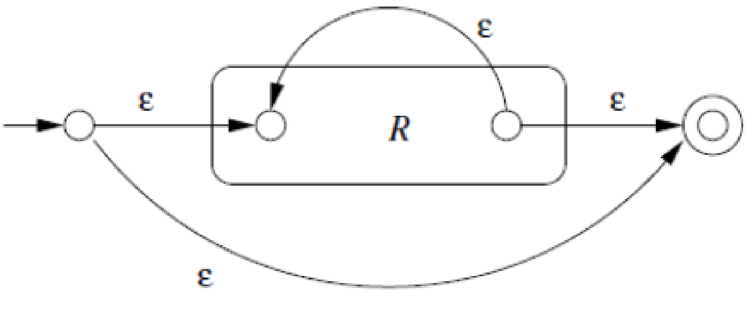
NFA A:



Induction Case: R*

Regular Expression: R* **L(R*) = (L(R))***

NFA A:



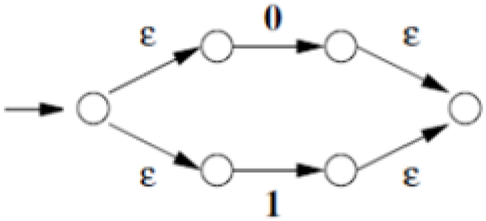
Convert the Regular Expression (0+1)*1 to ε- NFA

Sol:

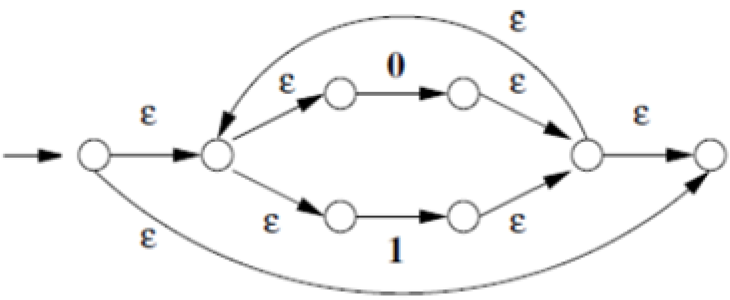
Automata for 0:

Automata for 1:

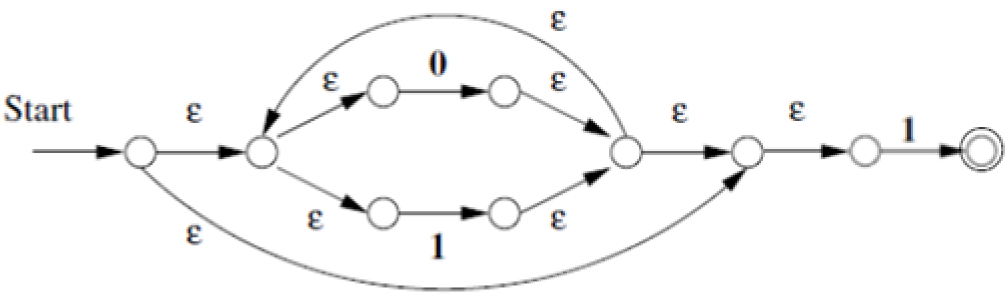
Automata for (0+1):



Automata for (0+1)*:



Automata for (0+1)*1:



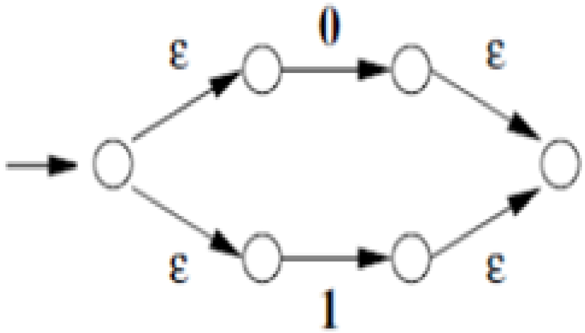
Convert the Regular Expression (0+1)*1(0+1) to ε- NFA

Sol:

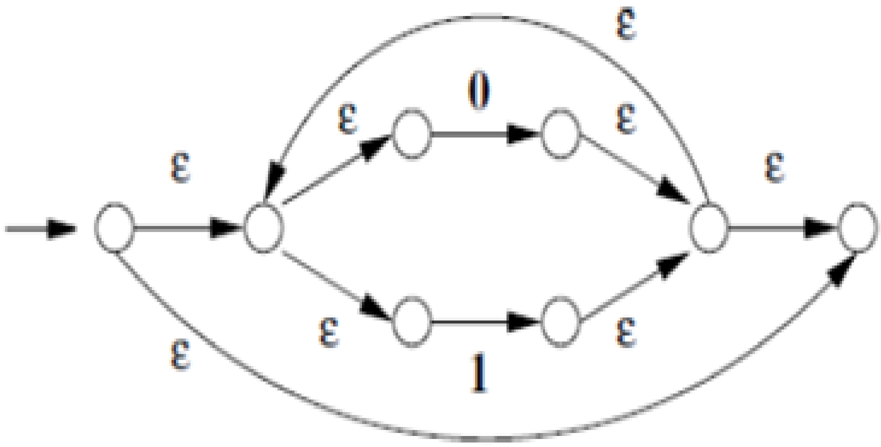
Automata for 0:

Automata for 1:

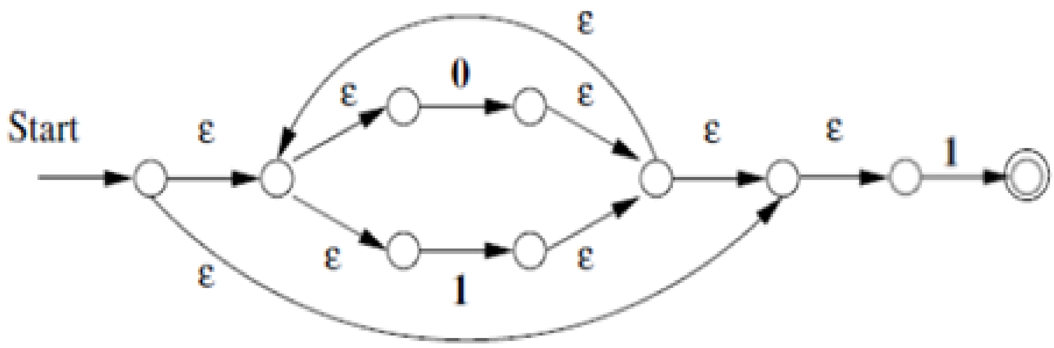
Automata for (0+1):



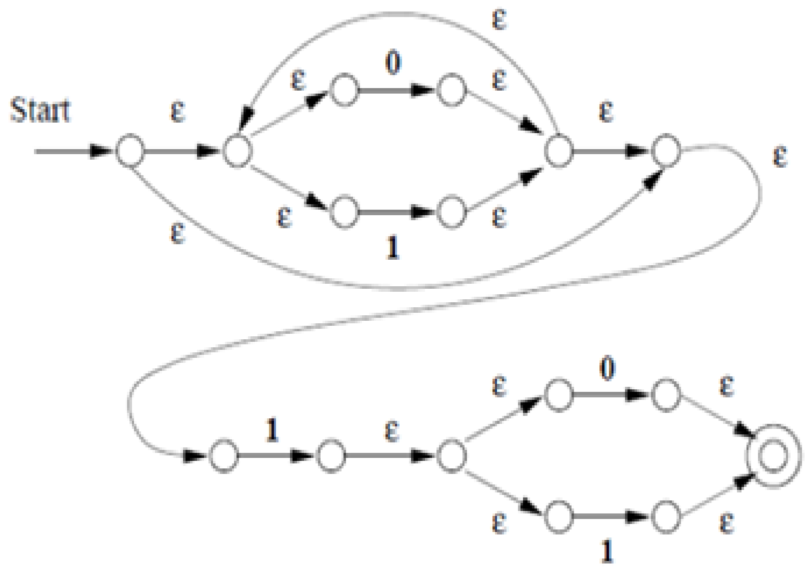
Automata for (0+1)*:



Automata for (0+1)*1:



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

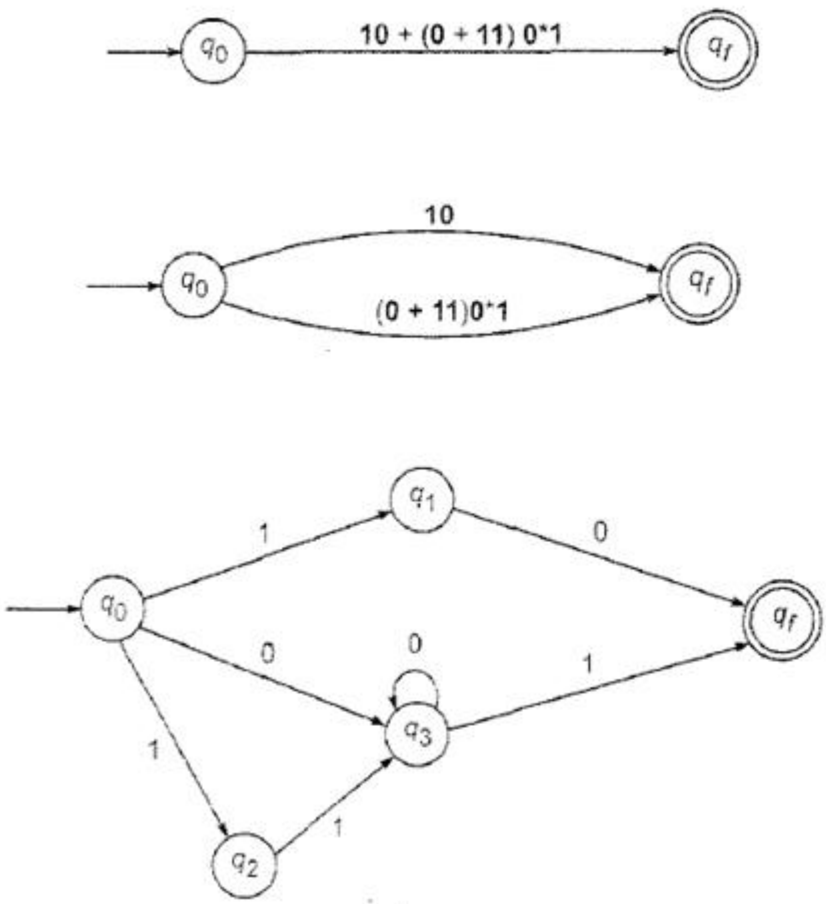


Ex: construct an NFA without ε- moves for the following regular Expression

$$\mathbf{10 + (0 + 11))0^*1}.$$

Sol:

The NFA is constructed by eliminating the operation +, concatenation and *, and the ε-moves in successive steps. The step-by-step construction is shown in the following figure.

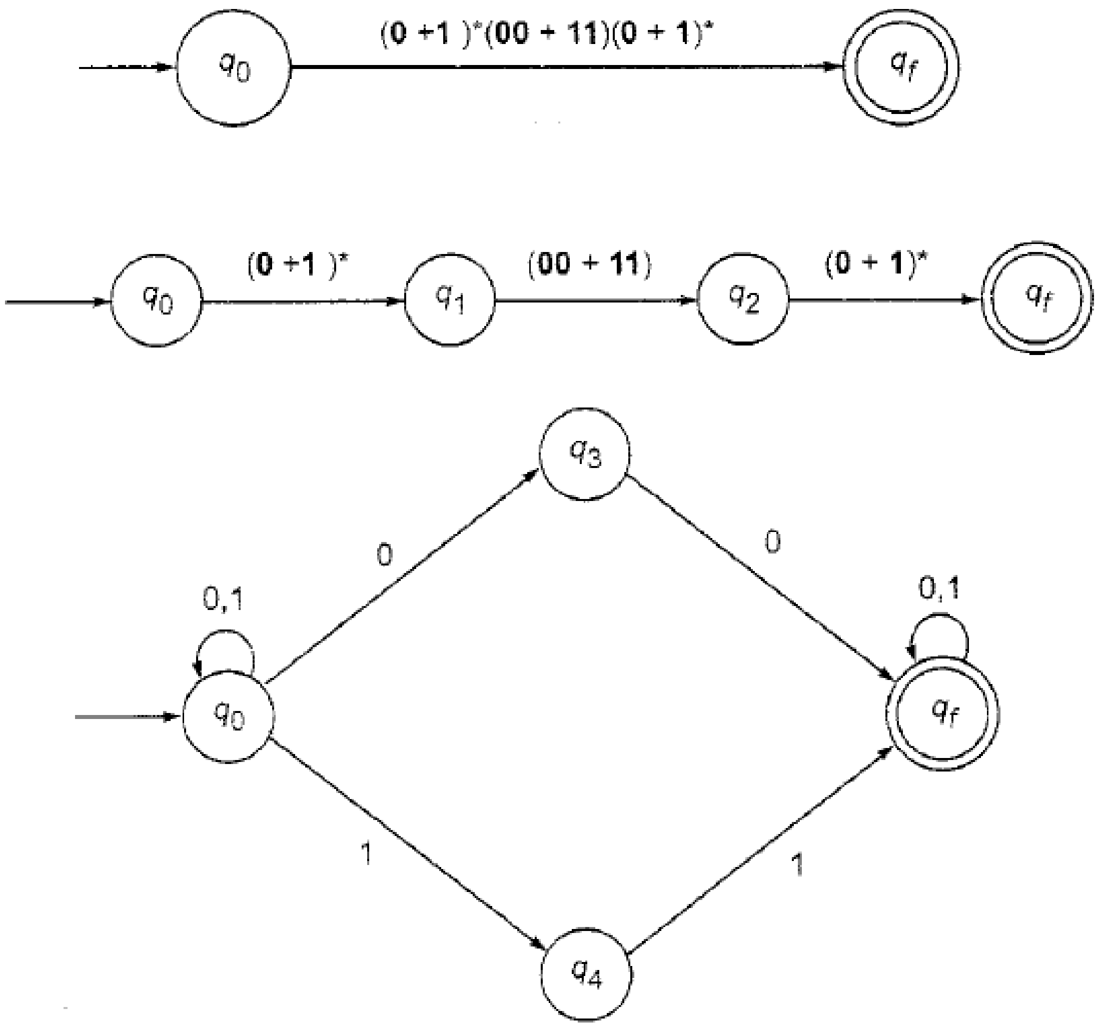


Ex: construct an NFA without ε- moves for the following regular Expression

$$(0 + 1)^*(00 + 11)(0 + 1)^*$$

Sol:

The NFA is constructed by eliminating the operation +, concatenation and *, and the ε-moves in successive steps. The step-by-step construction is shown in the following figure.



Pumping Lemma:

The class of languages known as the **regular languages** has at least four different descriptions. They **are** the languages **accepted by**

i) DFA's

ii) NFA's and

iii) ε - NFA's

iv) Regular expressions

Not every language is a regular language. We use a powerful technique known as the pumping lemma for showing certain languages not to be regular.

Pumping Lemma for Regular Languages:

Let L be the regular language. Then there exists a constant n (which depends on L) such that for every string $w \in L$ such that $|w| \geq n$, w can be expressed as $w = xyz$ for some strings x, y , z and

i) $y \neq \varepsilon$

ii) $|xy| \leq n$

iii) $xy^kz \in L$ for each $k \geq 0$.

Application of Pumping Lemma:

This lemma is used to show that certain languages are not regular. The steps needed to show that certain language is not regular are given below.

Step 1: Assume that L is regular. Let n be the number of states in the corresponding finite automata.

Step 2: choose a string w such that $|w| \geq n$. Use pumping lemma to write w=xyz with $|xy| \leq n$ and $|y| \geq 0$.

Step 3: find a suitable integer k such that $xy^kz \notin L$. This contradicts our assumption that L is regular. Hence L is not a regular language.

Show that the language $L=\{ a^p \mid p \text{ is a prime}\}$ is not regular

Sol:

Suppose that L is regular. Let n be the number of states in the finite automata accepting L.

Let p be a prime number such that $p \geq n$. Let $w = a^p$. By pumping lemma, w can be expressed as w=xyz with $|xy| \leq n$ and $|y| > 0$.

Now $|w| = |xyz| = |a^p| = p$.

Let $y=a^m$ for some $m \geq 1$ (and $\leq n$)

Let $i = p+1$ then by pumping lemma, $xy^iz \in L$.

Now $|xy^iz| = |xyz| + |y^{i-1}|$

$$= p + (i-1)m$$

$$= p + pm$$

$$= p(1+ m)$$

Since $p(1+m)$ is not a prime, $xy^iz \notin L$ which is a contradiction which arises because of our assumption that L is regular.

Hence L is not regular.

Show that the language

$$L = \{a^{i^2} \mid i \geq 1\}$$

is not regular.

Sol:

Suppose that L is regular. Let n be the number of states in the finite automata accepting L.

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

Now $|w| = |xyz| = n^2$

Consider the string xy^2z . by pumping lemma, $xy^2z \in L$.

Now $|xy^2z| = |x| + 2|y| + |z|$

$> |x| + |y| + |z|$ as $|y| > 0$.

$= |xyz| = n^2$

Thus $n^2 < |xy^2z|$ → (1)

Also, $|xy^2z| = |x| + 2|y| + |z|$

$= |xyz| + |y|$

$\leq n^2 + n$ (since $|y| \leq n$)

$< n^2 + n + n + 1$

$= (n+1)^2$

Thus $|xy^2z| < (n+1)^2$ →(2)

From equs.(1) and (2) we have $n^2 < |xy^2z| < (n+1)^2$

Thus $|xy^2z|$ lies between n^2 and $(n+1)^2$ and hence is not a perfect square.

So $xy^iz \notin L$ which is a contradiction which arises because of our assumption that L is regular.

Hence L is not regular.

Closure Properties of Regular Languages:

Closure Properties of Regular Languages are the theorems indicate that the regular languages are closed under certain operations.

A closure property of regular languages say that

–**If a language is created from regular languages using the operation mentioned in the theorem, it is also a regular language**.

The closure properties for regular languages are given below.

1. The union of two regular languages is regular i.e., If L and M are regular languages, then $L \cup M$ is regular.
2. The concatenation of regular languages is regular i.e., If L and M are regular languages, then LM is regular.
3. The closure (star) of a regular language is regular i.e., If L is a regular language, then L^* is regular.
4. The complement of a regular language is regular i.e., If L is a regular language over alphabet Σ, then its complement $\bar{L} = \Sigma^*$-L is also a regular language.
5. The intersection of two regular languages is regular i.e., If L and M are regular languages, then $L \cap M$ is regular.
6. The difference of two regular languages is regular i.e., If L and M are regular languages, then L-M is regular.
7. The reversal of a regular language is regular i.e., If L is a regular language, then its reversal L^R is regular.

Application of Regular Expressions:

Some of the applications of Regular Expressions are given below.

1. Regular Expressions provide a way for matching patterns of strings in a very efficient manner. They can be used in applications that search for patterns in text.

2. Regular Expressions can be applied in specifying the component of a compiler called a lexical analyzer. This component scans the source program and recognizes all tokens.

Finite Automata and Regular Grammars:

Converting DFA to Regular Gramar:

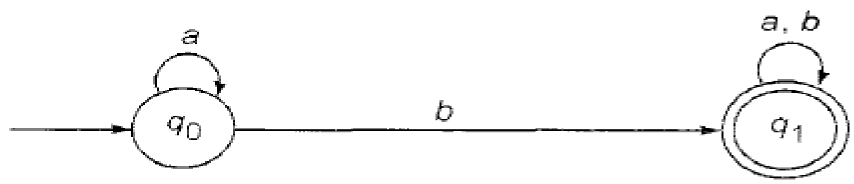
Let M = ($\{q_0, q_1, \ldots\ldots\ldots q_n\}$, Σ , δ , q_0 ,F) be the given DFA M. We construct a regular grammar G generating L(M) as follows.

G = ($\{A_0, A_1, \ldots\ldots\ldots, A_n\}$, Σ , P , A_0) where P is defined by the following rules

i) $A_i \rightarrow aA_j$ is included in P if $\delta(q_i,a) = q_j \notin F$.

ii) $A_i \rightarrow aA_j$ and $A_i \rightarrow a$ are included in P if $\delta(q_i,a) = q_j \in F$.

Ex: construct a regular grammar for the following DFA.



Sol: The regular grammar G generating the same language as that of given DFA is as follows.

G = ($\{A_0, A_1\}$, {a,b} , P , A_0) where P consists of following productions.

Since $\delta(q_0,a) = q_0 \notin F$, include the productions $A_0 \rightarrow aA_0$

Since $\delta(q_0,b) = q_1 \in F$, include the productions $A_0 \rightarrow bA_1$ and $A_0 \rightarrow b$

Since $\delta(q_1,a) = q_1 \in F$, include the productions $A_1 \rightarrow aA_1$ and $A_1 \rightarrow a$

Since $\delta(q_1,b) = q_1 \in F$, include the productions $A_1 \rightarrow bA_1$ and $A_1 \rightarrow b$

Therefore, the required regular grammar is

G = ($\{A_0, A_1\}$, {a,b} , P , A_0) where P consists of

$A_0 \rightarrow aA_0 \mid bA_1 \mid b$

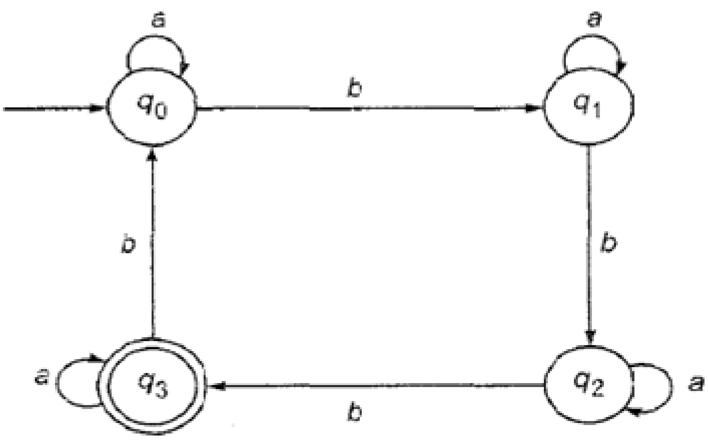
$A_1 \rightarrow aA_1 \mid bA_1 \mid a \mid b$

Ex:Construct a regular grammar accepting the language

$L = \{ w \in \{a, b\}^* \mid W$ is a string over $\{a, b\}$ such that the number of b's is 3 mod 4$\}$.

Sol:

The required DFA that accepts the given Language L is given as



The regular grammar that generates the L is given as

$G = (\{A_0, A_1, A_2, A_3\}, \{a, b\}, P, A_0)$ where P consists of $A_0 \to aA_0$, $A_0 \to bA_1$, $A_1 \to bA_1$, $A_1 \to bA_2$, $A_2 \to aA_2$, $A_2 \to bA_3$, $A_2 \to b$, $A_3 \to aA_3$, $A_3 \to aA_0$.

Converting Regular Grammar to Automata:

Let $G = (\{A_0, A_1, \ldots\ldots\ldots, A_n\}, \Sigma, P, A_0)$ be a regular grammar. Now we can construct the automata M that accepts L(G) as given below.

$M = (\{q_0, q_1, \ldots\ldots\ldots, q_n, q_f\}, \Sigma, \delta, q_0, \{q_f\})$ where δ is defined as follows.

i) Each production $A_i \to aA_j$ induces a transition from q_i to q_j with label 'a'

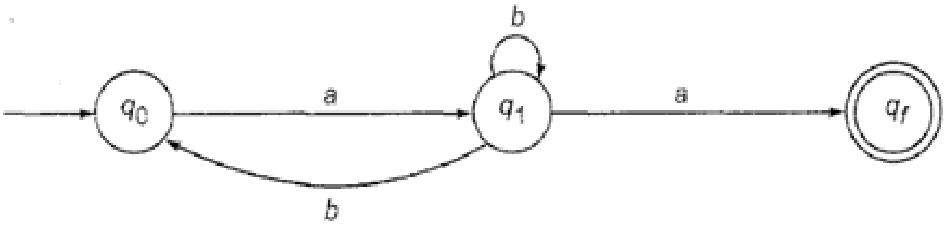
ii) Each production $A_k \to a$ induces a transition from q_k to q_f with label 'a'.

Ex:

Let $G = (\{A_0, A_1\}, \{a, b\}, P, A_0)$, where P consists of $A_0 \rightarrow aA_1$, $A_1 \rightarrow bA_1$, $A_1 \rightarrow a$, $A_1 \rightarrow bA_0$. Construct a transition system M accepting $L(G)$.

Let $M = (\{q_0, q_1, q_f\}, \{a, b\}, \delta, q_0, \{q_f\})$, where q_0 and q_1 correspond to A_0 and A_1, respectively and q_f is the new (final) state introduced. $A_0 \rightarrow aA_1$ induces a transition from q_0 to q_1 with label a. Similarly, $A_1 \rightarrow bA_1$ and $A_1 \rightarrow bA_0$ induce transitions from q_1 to q_1 with label b and from q_1 to q_0 with

label b, respectively. $A_1 \rightarrow a$ induces a transition from q_1 to q_f with label a. M is given in Fig.



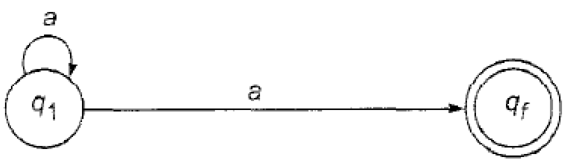
Ex:

If a regular grammar G is given by $S \rightarrow aS \mid a$, find M accepting $L(G)$.

Solution

Let q_0 correspond to S and q_f be the new (final) state. M is given in Fig. Symbolically,

$$M = (\{q_0, q_f\}, \{a\}, \delta, q_0, \{q_f\})$$



Regular Expressions and Regular Grammars:

Construction of Regular grammar from Regular Expression

We can convert the regular expression into its equivalent regular grammar by using the following method.

1. Construct an ε – NFA from the given regular expression

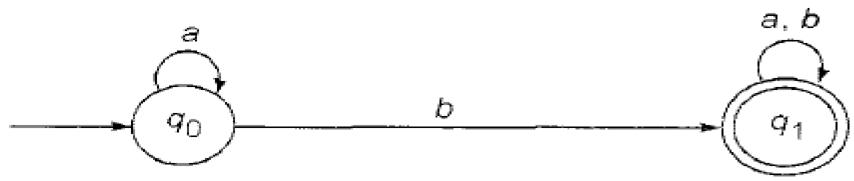
2. Eliminate ε – transitions and construct an equivalent DFA

3. From the DFA, construct the regular grammar (by above known procedure)

Ex: construct a regular grammar for the regular expression a*b(a+b)*

Sol:

The equivalent DFA for the given regular expression is



The regular grammar G generating the same language as that of given DFA is as follows.

$G = (\{A_0, A_1\}, \{a,b\}, P, A_0)$ where P consists of following productions.

Since $\delta(q_0,a) = q_0 \notin F$, include the productions $A_0 \rightarrow aA_0$

Since $\delta(q_0,b) = q_1 \in F$, include the productions $A_0 \rightarrow bA_1$ and $A_0 \rightarrow b$

Since $\delta(q_1,a) = q_1 \in F$, include the productions $A_1 \rightarrow aA_1$ and $A_1 \rightarrow a$

Since $\delta(q_1,b) = q_1 \in F$, include the productions $A_1 \rightarrow bA_1$ and $A_1 \rightarrow b$

Therefore, the required regular grammar is

$G = (\{A_0, A_1\}, \{a,b\}, P, A_0)$ where P consists of

$A_0 \rightarrow aA_0 \mid bA_1 \mid b$

$A_1 \rightarrow aA_1 \mid bA_1 \mid a \mid b$

Constructing regular expression from regular grammar:

We can convert the regular grammar into its equivalent regular expression by using the following method.

1. Construct finite automata that accept the language generated by the regular grammar.

2. Find regular expression equivalent to finite automata using Arden's theorem.