# 1. Regular language & Regular Expression.

**Regular Language :-**

A Regular Language over an alphabet ∑ is one that can be obtained from this basic language using the operation of Union, Concatenation and Kleene (*).

**Regular Expression :-**

- Regular Language can be described by an explicit formula.
- It is common to simplify the formula slightly by leaving out the set brackets {} or replacing them with parenthesis () and replace U by +. The result is called Regular Expression.

| | Language | Regular Expression |
|---|---|---|
| 1. | {^} | ^ |
| 2. | {0} | 0 |
| 3. | {001} | 001 |
| 4. | {0,1} | 0+1 |
| 5. | {0,10} | 0+10 |
| 6. | {1,0}110 | (1+0)110 |
| 7. | {110}*{0,1} | (110)*(0+1) |
| 8. | {1}*{10} | (1)*(10) |
| 9. | {10,111,11010}* | (10+111+11010)* |
| 10. | {0,10}*({11}*U{001,^}) | (0+10)*((11)*+(001+^)) |

**Table 2.1. Regular Expression**

**More Examples of regular Expression**

1. 0 or 1
   0+1
2. 0 or 11 or 111
   0+11+111
3. Regular expression over ∑={a,b,c} that represent all string of length 3.
   (a+b+c)(a+b+c)(a+b+c)
4. String having zero or more a.
   a*
5. String having one or more a.
   $a^+$
6. All binary string.
   (0+1)*
7. 0 or more occurrence of either a or b or both
   (a+b)*
8. 1 or more occurrence of either a or b or both
   $(a+b)^+$
9. Binary no. ends with 0
   (0+1)*0
10. Binary no. ends with 1
    (0+1)*1
11. Binary no. starts and ends with 1.

1(0+1)*1

12. String starts and ends with same character.
    0(0+1)*0 or a(a+b)*a
    1(0+1)*1 b(a+b)*b
13. All string of a and b starting with a
    a(a/b)*
14. String of 0 and 1 ends with 00.
    (0+1)*00
15. String ends with abb.
    (a+b)*abb
16. String starts with 1 and ends with 0.
    1(0+1)*0
17. All binary string with at least 3 characters and 3rd character should be zero.
    (0+1)(0+1)0(0+1)*
18. Language which consist of exactly two b's over the set ∑={a,b}
    a*ba*ba*
19. ∑={a,b} such that 3rd character from right end of the string is always a.
    (a+b)*a(a+b)(a+b)
20. Any no. of a followed by any no. of b followed by any no. of c.
    a*b*c*
21. String should contain at least 3 one.
    (0+1)*1(0+1)*1(0+1)*1(0+1)*
22. String should contain exactly two 1's
    0*10*10*
23. Length of string should be at least 1 and at most 3.
    (0+1) + (0+1) (0+1) + (0+1) (0+1) (0+1)
24. No.of zero should be multiple of 3
    (1*01*01*01*)*
25. ∑={a,b,c} where a should be multiple of 3.
    ((b+c)*a (b+c)*a (b+c)*a (b+c)*)*
26. Even no. of 0.
    (1*01*01*)*
27. Odd no. of 1.
    0*(10*10*)*10*
28. String should have odd length.
    (0+1)((0+1)(0+1))*
29. String should have even length.
    ((0+1)(0+1))*
30. String start with 0 and has odd length.
    0((0+1)(0+1))*
31. String start with 1 and has even length.
    1(0+1)((0+1)(0+1))*
32. Even no of 1

(0*10*10*)*

33. String of length 6 or less
    $(0+1+\wedge)^6$
34. String ending with 1 and not contain 00.
    $(1+01)^+$
35. All string begins or ends with 00 or 11.
    (00+11)(0+1)*+(0+1)*(00+11)
36. Language of all string containing both 11 and 00 as substring.
    ((0+1)*00(0+1)*11(0+1)*)+ ((0+1)*11(0+1)*00(0+1)*)
37. Language of C identifier.
    (_+L)(_+L+D)*

## 2. Definition of Finite Automata.

A finite Automata or finite state machine is a 5-tuple$(Q,\Sigma,q_0,A,\delta)$ where,

Q is finite set of states;
$\Sigma$ is finite alphabet of input symbols;
$q_0 \in Q$ (Initial state);
A (set of accepting states);
$\delta$ is a function from $Q\times\Sigma \rightarrow Q$(Transition function);

For any element q of Q and any symbol $a \in \Sigma$, we interpret $\delta$ (q, a) as the state to which the Finite Automata moves, if it is in state q and receives the input a.

**Application of finite automata**

A finite automaton is used to solve several common types of computer algorithm. Some of them are:

1. Design of digital circuit.
2. String matching.
3. Communication protocols for information exchange.
4. Lexical analysis phase of a compiler.

## 3. The Extended transition function δ* for FA.

Let M= $(Q,\Sigma,q_0,A,\delta)$ be an Finite Automata. We define the function $\delta^*: Q\times\Sigma^* \rightarrow Q$ as follow:

1) For any $q \in Q$, $\delta^*(q,\wedge)=q$
2) For any $q \in Q$, $y \in \Sigma^*$, and $a \in \Sigma$
   $\delta^*(q, ya)= \delta(\delta^*(q, y),a)$

Example:

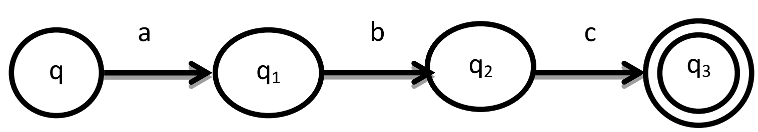


**Fig 2.1 Finite Automata**

$\delta^*(q,abc)$
$\delta(\delta^*(q,ab),c)$

$\delta(\delta^*(\delta^*(q,a),b),c)$
$\delta(\delta(\delta^*(q,\wedge a),b),c)$
$\delta(\delta(\delta(\delta^*(q,\wedge),a),b),c)$
$\delta(\delta(\delta(q,a),b),c)$
$\delta(\delta(q1,b),c)$
$\delta(q_2,c)$
$q_3$

## 4. Acceptance by an Finite Automata.

Let M= ($Q_1$, ∑, $q_0$, A, δ) be an FA. A string x ∈ ∑* is accepted by M if $\delta^*(q_0, x) \in A$. If string is not accepted, we can say it is rejected by M. The language accepted by M, or the language recognized by M, is the set L(M) = {x ∈ ∑*/x is accepted by M} If L is any Language over ∑, L is accepted or recognized by M if and only if L=L(M).

## 5. Draw Finite Automata for following:

1. **The string with next to last symbol as 0.**

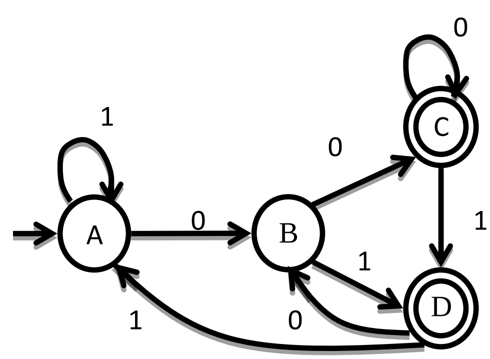


**Fig 2.2 Finite Automata for next to last symbol as 0**

2. **The string with number of 0s and number of 1s are odd**

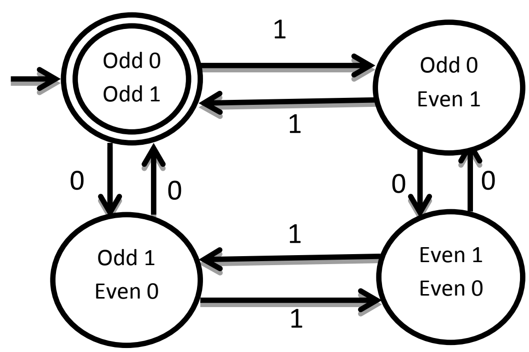


**Fig 2.3 Finite Automata for number of 0s and number of 1s are odd**

3. **The string ending in 10 or 11.**

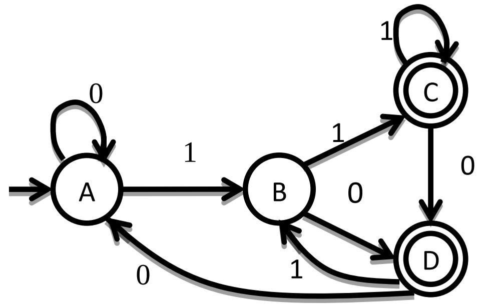


**Fig 2.4 Finite Automata for string ending in 10 or 11**

4. **The string corresponding to Regular expression {00}*{11}***

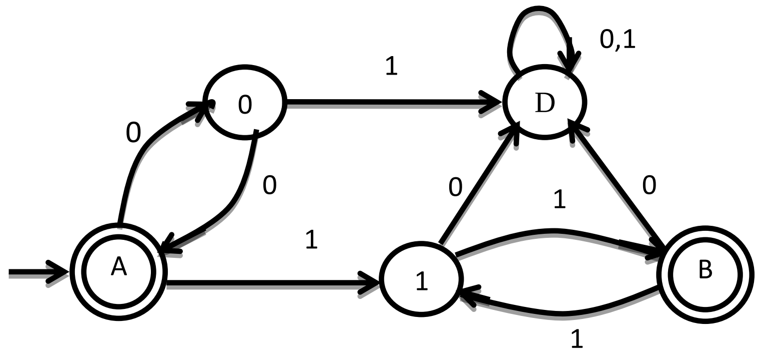


**Fig 2.5 Finite Automata for {00}*{11}***

5. **(a+b)*baaa**

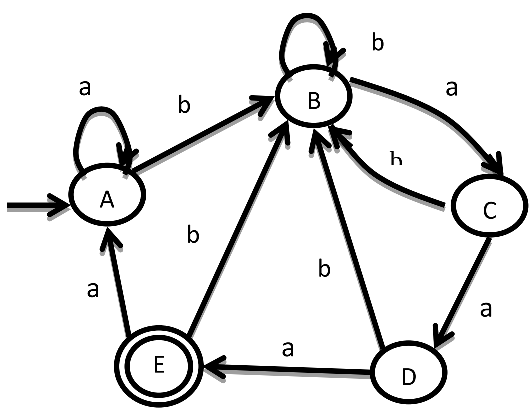


**Fig 2.6 Finite Automata for (a+b)*baaa**

## 6. Find a string of minimum length in {0,1}* not in the language corresponding to the regular expression: 1*(0+10)*1*

The smallest string = 0110

## 7. Define Dead end state.

Dead state are those non accepting states whose transitions for every input symbols terminate on themselves.

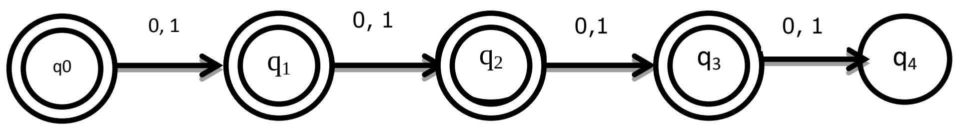


**Fig 2.7 Dead end state**

Ex: all strings of length at most 3. The string of length >3 should be rejected through a dead state or a failure state. In above example dead state is $q_4$.

## 8. Union, Intersection & Compliment operation on Finite Automata

Suppose $M1=(Q_1,\sum,q_1,A_1,\delta_1)$

$M2=(Q_2,\sum,q_2,A_2,\delta_2)$

Accept languages L1 and L2 respectively. Let M be an Finite Automata defined by

$M=(Q,\sum,q_0,A,\delta)$ where,

$Q=Q1\times Q2$

$q_0=(q_1, q_2)$ and transition function δ is defined by the formula

$\delta((p,q),a)=(\delta1(p,a), \delta2(q,a))$ for any $p \in Q1$ and $q \in Q2$ and $a \in \sum$ then

1) if $A=\{(p,q) / p \in A1 \text{ or } q \in A2\}$, M accept the language L1 U L2;
2) if $A=\{(p,q) / p \in A1 \text{ and } q \in A2\}$, M accept the language $L1 \cap L2$;
3) if $A=\{(p,q) / p \in A1 \text{ and } q \notin A2\}$, M accept the language L1 - L2.

## 9. Draw Finite Automata for following languages: L1={x/x 00 is not substring of x, x ∈ {0,1}*} L2={x/x ends with 01, x ∈ {0,1}*} Draw finite Automata for L1 U L2, L1∩ L2 and L1-L2.

**M1**

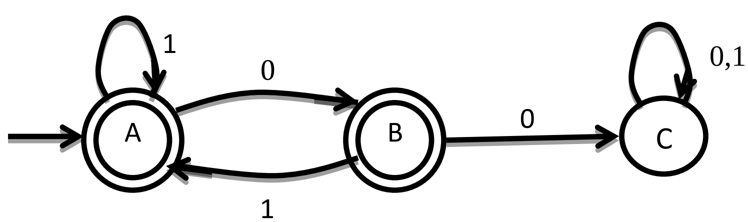


**Fig 2.8 Finite Automata for 00 is not substring**

**M2**

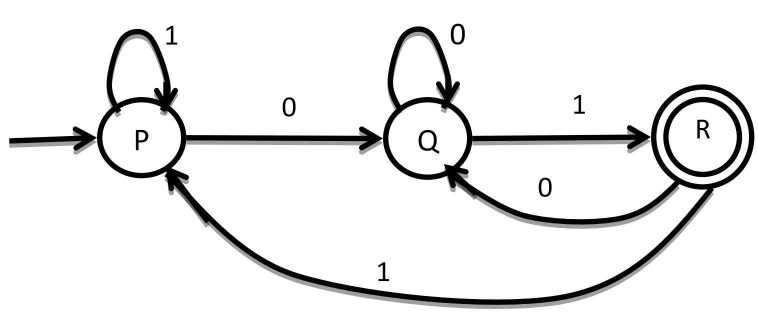

**Fig 2.9 Finite Automata for string ending in 01**

Here Q1={A,B,C} and Q2={P,Q,R}
So,Q = Q1×Q2 ={AP,AQ,AR,BP,BQ,BR,CP,CQ,CR}
$q_0=(q_1,q_2)=(A,P)$

$\delta((A,P),0) = (\delta(A,0), \delta(P,0)) = BQ$

$\delta((A,P),1) = (\delta(A,1), \delta(P,1)) = AP$

$\delta((B,Q),0) = (\delta(B,0), \delta(Q,0)) = CQ$

$\delta((B,Q),1) = (\delta(B,1), \delta(Q,1)) = AR$

$\delta((C,Q),0) = (\delta(C,0), \delta(Q,0)) = CQ$

$\delta((C,Q),1) = (\delta(C,1), \delta(Q,1)) = CR$

$\delta((A,R),0) = (\delta(A,0), \delta(R,0)) = BQ$

$\delta((A,R),1) = (\delta(A,1), \delta(R,1)) = AP$

$\delta((C,R),0) = (\delta(C,0), \delta(R,0)) = CQ$

$\delta((C,R),1) = (\delta(C,1), \delta(R,1)) = CP$

$\delta((C,P),0) = (\delta(C,0), \delta(P,0)) = CQ$

$\delta((C,P),1) = (\delta(C,1), \delta(P,1)) = CP$

**L1-L2**

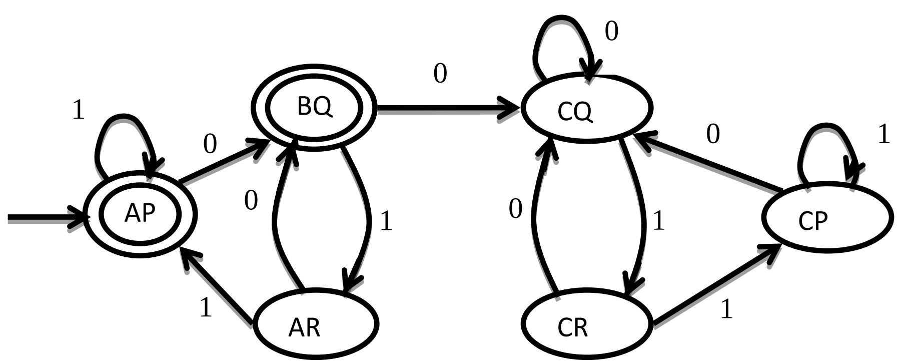


**Fig 2.10 Finite Automata for L1-L2**

**L1 U L2**

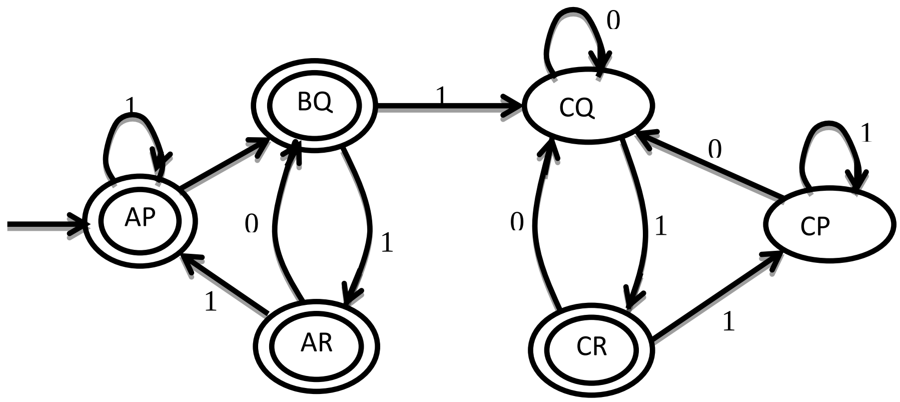


**Fig. 2.11 Finite Automata for L1 U L2**

**L1∩ L2**

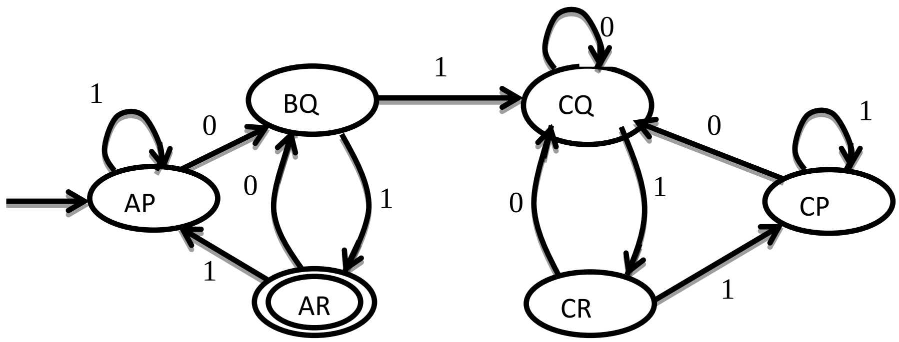


**Fig. 2.12 Finite Automata for L1∩ L2**

## 10. Draw Finite Automata for following languages: L1={x/x 11 is not substring of x, x ∈ {0,1}*} L2={x/x ends with 10, x ∈ {0,1}*} Draw finite Automata for L1∩ L2 and L1-L2. (Most IMP)

**M1**

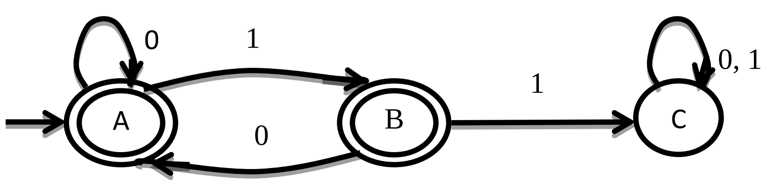


**Fig. 2.13 Finite Automata for 11 is not substring**

**M2**

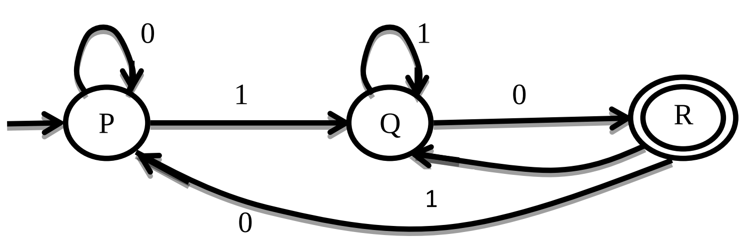

**Fig. 2.14 Finite Automata for string ending in 10**

Here $Q1=\{A,B,C\}$, $Q2=\{P,Q,R\}$

So, $Q = Q1 \times Q2 = \{AP,AQ,AR,BP,BQ,BR,CP,CQ,CR\}$

$q_0=(q_1,q_2)$

$q_0=(A,P)$

$\delta((A,P),0) = (\delta(A,0), \delta(P,0))$
$= AP$

$\delta((A,P),1) = (\delta(A,1), \delta(P,1))$
$= BQ$

$\delta((B,Q),0) = (\delta(B,0), \delta(Q,0))$
$= AR$

$\delta((B,Q),1) = (\delta(B,1), \delta(Q,1))$
$= CQ$

$\delta((A,R),0) = (\delta(A,0), \delta(R,0))$
$= AP$

$\delta((A,R),1) = (\delta(A,1), \delta(R,1))$
$= BQ$

$\delta((C,Q),0) = (\delta(C,0), \delta(Q,0))$
$= CR$

$\delta((C,Q),1) = (\delta(C,1), \delta(Q,1))$
$= CQ$

$\delta((C,R),0) = (\delta(C,0), \delta(R,0))$
$= CP$

$\delta((C,R),1) = (\delta(C,1), \delta(R,1))$
$= CQ$

$\delta((C,P),0) = (\delta(C,0), \delta(P,0))$
$= CP$

$\delta((C,P),1) = (\delta(C,1), \delta(P,1))$
$= CQ$

**L1 - L2**

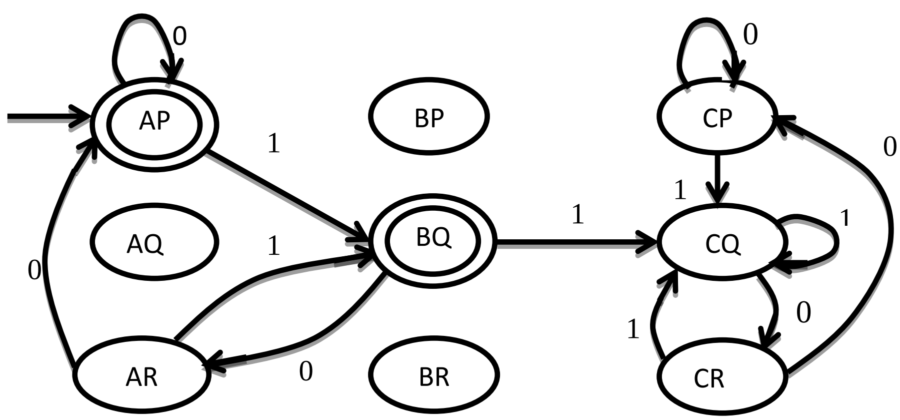


**Fig. 2.15 Finite Automata for L1-L2**

**L1 ∩ L2**

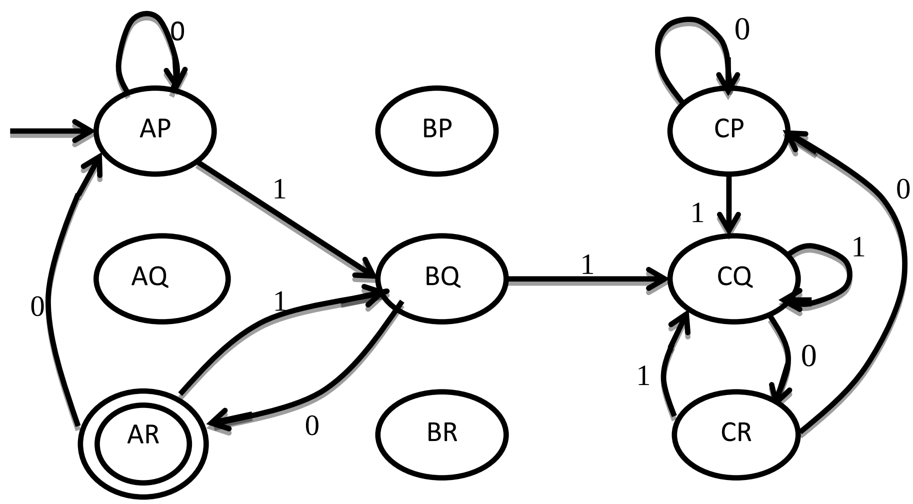


**Fig. 2.16 Finite Automata for L1∩L2**

## 11. Draw the Finite Automata recognizing the following language

- **L1 ∩ L2**
- **L1 - L2**

**L1**

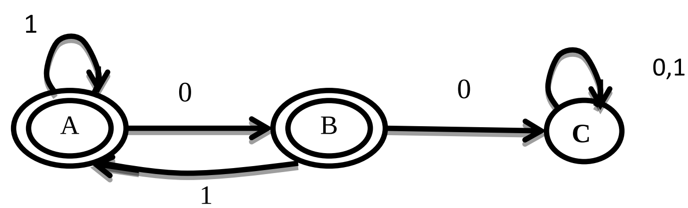


**Fig. 2.17 Finite Automata**

**L2**

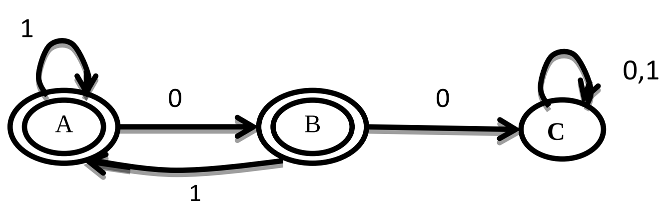


**Fig. 2.18 Finite Automata**

Q1={A,B,C},Q2={A,B,C}

Q=Q1×Q2= {AA,AB,AC,BA,BB,BC,CA,CB,CC}

$q_0=(q_1,q_2)$ $q_0$=(A,A)

$\delta((A,A),0) = (\delta(A,0), \delta(A,0)) = BB$
$\delta((A,A),1) = (\delta(A,1), \delta(A,1)) = AA$
$\delta((B,B),0) = (\delta(B,0), \delta(B,0)) = CC$
$\delta((B,B),1) = (\delta(B,1), \delta(B,1)) = AA$
$\delta((C,C),0) = (\delta(C,0), \delta(C,0)) = CC$
$\delta((C,C),1) = (\delta(C,1), \delta(C,1)) = CC$

**L1 - L2**

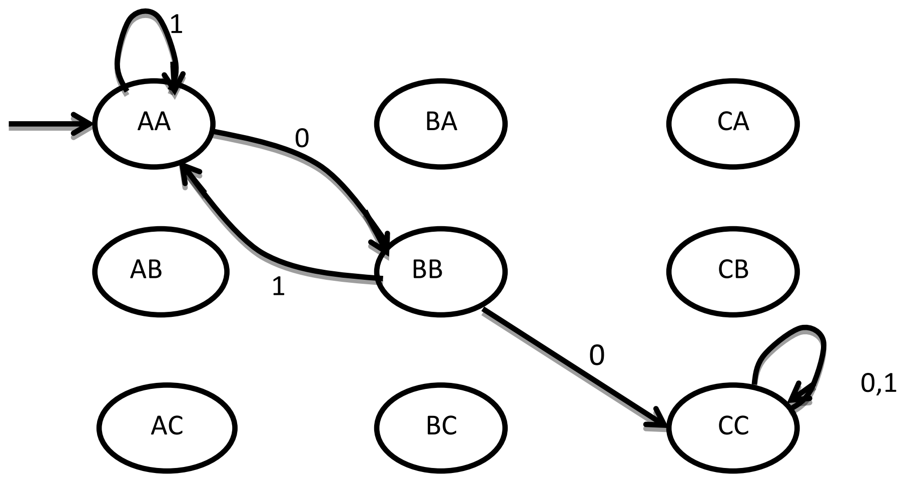


**Fig. 2.19 Finite Automata for L1-L2**

**L1 ∩ L2**

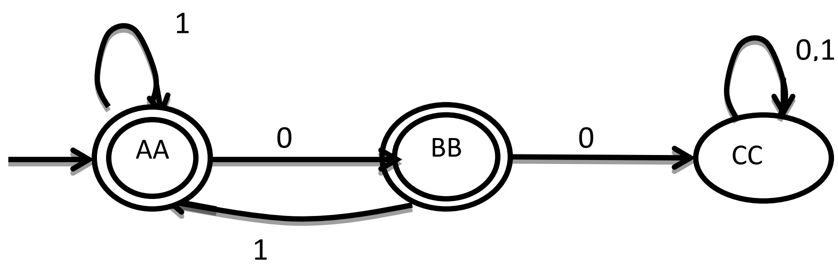


**Fig. 2.20 Finite Automata for L1∩L2**

## 12. Definition: Nondeterministic Finite Automata.

A nondeterministic finite automaton (NFA) is a 5-tuple $(Q,\Sigma,q_0,A,\delta)$ Where Q and Σ are nonempty finite sets, $q_0 \in Q$, $A \subseteq Q$, and $\delta : Q \times \Sigma \rightarrow 2^Q$

Q is a finite set of states;
Σ is a finite set of input alphabets;
$q_0 \in Q$ is the initial state;
$A \subseteq Q$ is the set of accepting states;
$\delta : Q \times (\Sigma \cup \{\Lambda\}) \rightarrow 2^Q$ is the transition function.

## 13. Definition: Non recursive Definition of δ* for an NFA.

For an NFA $M=(Q,\Sigma, q_0,A, \delta)$, and any $p \in Q$, $\delta^*(p, \Lambda)= \{p\}$. For any $p \in Q$ and $x= a_1a_2....a_n \in \Sigma^*$ (with $n\geq1$), $\delta^*(p ,x)$ is the set of all states q for which there is a sequence of states $p=p_0,p_1,....p_{n-1},p_n = q$ satisfying

$P_i \in\delta (p_{i-1} , a_i)$ for each i with$1 \leq i \leq n$

## 14. Definition: Recursive Definition of δ* for an NFA

Let $M=(Q,\Sigma, q0,A, \delta)$, be an NFA. The function $\delta^*: Q \times \Sigma^* \rightarrow 2^Q$ is defined as follows.

1) For any $q \in Q$, $\delta^*(q, \Lambda) = \{q\}$.
2) For any $q \in Q$, $y \in\Sigma^*$, and $a \in\Sigma$,

$$\delta^*(q, ya)= \bigcup_{r \in\delta^*(q, y)} \delta(r, a)$$

## 15. Definition: Acceptance by an NFA

Let $M= (Q,\Sigma,q_0,A,\delta)$ be an NFA. The string $x \in\Sigma^*$ is accepted by M if $\delta^*(q_0, x) \cap A\neq\emptyset$. The language recognized, or accepted, by M is the set L(M) of all string accepted by M. For any language $L \subseteq \Sigma^*$, L is recognized by M if L= L(M).

## 16. Using the Recursive Definition of δ* in an NFA.

Let $M=(Q,\Sigma, q_0,A, \delta)$, where $Q= \{q_0, q_1, q_2, q_3\}$, $\Sigma=\{0,1\}$, $A= \{q_3\}$, and δ is given by the following table:

| **q** | **δ(q,0)** | **δ(q,1)** |
|---|---|---|
| $q_0$ | $\{ q_0\}$ | $\{ q_0, q_1\}$ |
| $q_1$ | $\{ q_2\}$ | $\{ q_2\}$ |
| $q_2$ | $\{ q_3\}$ | $\{ q_3\}$ |
| $q_3$ | $\emptyset$ | $\emptyset$ |

**Table 2.2 Transition Table**

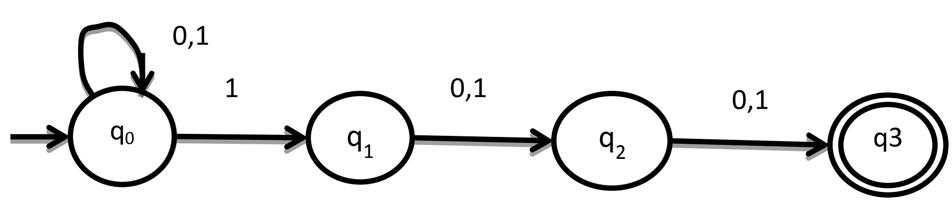


**Fig. 2.21 NFA**

Let us try to determine L(M) by calculating $\delta^*( q_0,x)$ for a few string x of increasing length. First observe that form the non recursive definition of δ* it is almost obvious that δ and δ* agree for string of length 1. We see from the table that $\delta^*( q_0,0)=\{ q_0\}$ and $\delta^*( q_0,1)=\{ q_0, q_1\}$;

$\delta^*( q_0,11) \quad = \quad \bigcup \quad \delta( r,1)$

$r \in \delta^*\{q_0,1\}$

$= \quad \cup \quad \delta(r,1)$
$r \in \{q_0, q1\}$
$= \delta\{q_0,1\} \cup \delta(q_1,1)$
$= \{q_0, q_1\} \cup \{q_2\}$
$= \{q_0, q_1, q_2\}$

$\delta^*(q_0,01) \quad = \quad \cup \quad \delta(r,1)$
$r \in \delta^*(q_0,0)$

$= \quad \cup \quad \delta(r,1)$
$r \in (q0)$
$= \delta(q_0,1)$
$= \{q_0, q_1\}$

$\delta^*(q_0,111) \quad = \quad \cup \quad \delta(r,1)$
$r \in \delta^*(q_0,11)$
$= \quad \cup \quad \delta(r,1)$
$r \in (q_0, q_1, q_2)$

$= \delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1)$
$= \{q_0, q_1, q_2, q_3\}$

$\delta^*(q_0,011) \quad = \quad \cup \quad \delta(r,1)$
$r \in \delta^*(q_0,01)$
$= \quad \cup \quad \delta(r,1)$
$r \in (q_0, q_1)$

$= \delta(q_0,1) \cup \delta(q_1,1)$
$= \{q_0, q_1, q_2\}$

## 17. Definition: A Nondeterministic Finite Automaton with Λ - Transition.

A nondeterministic finite automata with Λ - Transition is a 5-tuple (Q,Σ, q0,A, δ), where Q and Σ are finite sets, $q_0 \in Q$, $A \subseteq Q$, and $\delta : Q \times (\Sigma \cup \{\Lambda\}) \rightarrow 2^Q$

## 18. Definition: Nondeterministic Definition of δ* for an NFA- Λ.

For an NFA- Λ M= $(Q,\Sigma,q_0,A,\delta)$, states $p,q \in Q$, and a string $x = a_1a_2....a_n \in \Sigma^*$, we will say M moves from p to q by a sequence of transition corresponding to x if there exist an integer $m \geq n$, a sequence $b_1b_2....b_m \in \Sigma \cup \{\Lambda\}$ satisfying $b_1b_2....b_m=x$, and a sequence of states $p= p_0, p_1... p_m=q$ so that for each i with $1 \leq i \leq m$, $pi \in (p_{i-1}, b_i)$.

For $x \in \Sigma^*$ and $p \in \delta^*(p, x)$ is the set of all states $q \in Q$ such that there is a sequence of transition corresponding to x by which M moves from p to q.

## 19. Definition: Λ- Closure of a Set of States.

Let $M=(Q,\Sigma, q_0,A, \delta)$, be an NFA – Λ, and let S ne any subset of Q. The Λ- closure of S is the set Λ(s) defined as follows.

1) Every element of S is an element of Λ(s);
2) For any $q \in \Lambda(s)$, every element of $\delta(q, \Lambda)$ is in Λ(s);
3) No other elements of Q are in Λ(s);

## 20. Definition: Recursive Definition of δ* for an NFA- Λ.

Let $M= (Q, \Sigma, q_0, A, \delta)$ be an NFA – Λ, The extended transition function $\delta^*: Q \times \Sigma^* \rightarrow 2^Q$ is defined as follows.

1) For any $q \in Q$, $\delta^*(q, \Lambda) = \Lambda(\{q\})$.
2) For any $q \in Q$, $y \in \Sigma^*$, and $a \in \Lambda$,

$$\delta^*(q,ya)= \Lambda\left( \bigcup_{r \in \delta^*(q,y)} \delta(r,a) \right)$$

A string x is accepted by M if $\delta^*(q_0, x) \cap A \neq \emptyset$. The language recognized by M is the set L(M) of all strings accepted by M.

## 21. Compare FA, NFA and NFA-^.

| Parameter | NFA | FA | NFA-^ |
|---|---|---|---|
| Transition | Non deterministic | Deterministic | Non deterministic |
| Power | NFA is as powerful as DFA | FA is powerful as an NFA | It’s powerful as FA |
| Design | Easy to design due to non-determinism | More difficult to design | Allow flexibility in handling NFA problem. |
| Acceptance | It is difficult to find whether w ∈ L as there are several paths. Backtracking is required to explore several parallel paths. | It is easy to find whether w ∈ L as transition are deterministic. | ^-transition is useful in constructing a composite FA with respect to union, concatenation, and star. |

**Table. 2.3 Difference between FA, NFA, NFA-^**

## 22. Applying Definitions of ^(S) and δ*

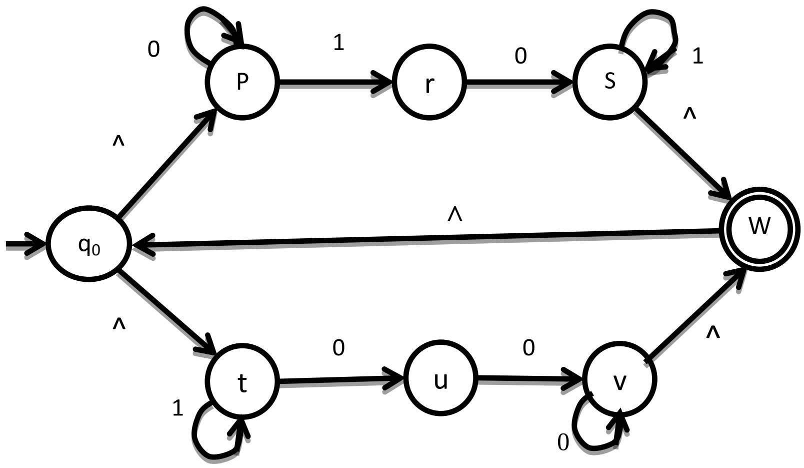


**Fig. 2.22 NFA-^**

$$\delta^*(q_0,\wedge) = \wedge(\{q_0\})$$
$$= \{q_0,p,t\}$$

$$\delta^*(q_0,0) = \wedge\Big(\bigcup_{r\in\delta^*(q0,\wedge)} \delta(r,0)\Big)$$
$$= \wedge(\delta(q_0,0) \cup \delta(p,0) \cup \delta(t,0))$$
$$= \wedge(\emptyset \cup \{p\} \cup \{u\})$$
$$= \wedge(\{p,u\})$$
$$= \{p,u\}$$

$$\delta^*(q_0,01) = \wedge\Big(\bigcup_{r\in\delta^*(q0,0)} \delta(p,1)\Big)$$
$$= \wedge(\delta(p,1) \cup \delta(u,1))$$
$$= \wedge(\{r\})$$
$$= \{r\}$$

$$\delta^*(q_0,010) = \wedge\Big(\bigcup_{r\in\delta^*(q0,01)} \delta(p,0)\Big)$$
$$= \wedge(\delta(r,0)$$
$$= \wedge(\{s\})$$
$$= \{s,w,q_0,p,t\}$$

## 23. Conversion from NFA to FA.

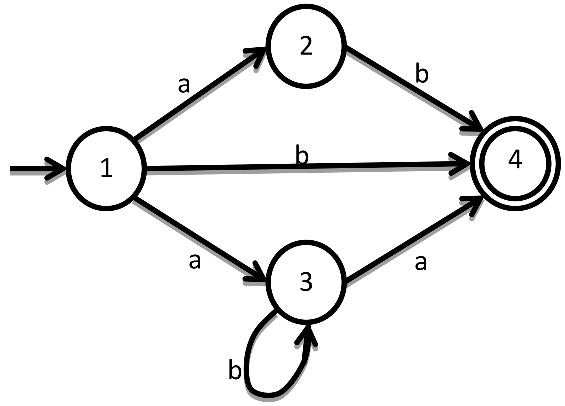


Fig. 2.23 NFA

δ*( 1,a) = {2,3}
δ*( 1,b) = {4}
δ*({2,3},a) = δ(2,a) U δ(3,a)
= {4}
δ*({2,3},b) = δ(2,b) U δ(3,b)
= {3,4}
δ*(4,a) = {Ø},
δ*(4,b) = {Ø}
δ*({3,4},a) = δ(3,a) U δ(4,a)
= {4}
δ*({3,4},b) = δ(3,b) U δ(4,b)
= {3}
δ*(3,a) = {4}
δ*(3,b) = {3}

| q | δ(q,a) | δ(q,b) |
|---|---|---|
| 1 | {2,3} | {4} |
| 2 | { Ø } | {4} |
| 3 | {4} | {3} |
| 4 | {Ø} | {Ø} |

Table. 2.4 Transition Table

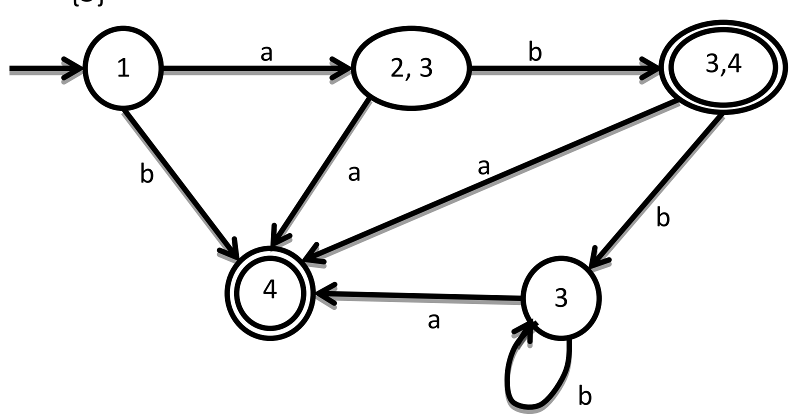


Fig. 2.24 Finite Automata

## 24. Convert NFA - ^ to FA

| q | δ (q,^) | δ (q,0) | δ(q,1) |
|---|---|---|---|
| A | {B} | {A} | Ø |
| B | {D} | {C} | Ø |
| C | Ø | Ø | {B} |
| D | Ø | {D} | Ø |

**Table. 2.5 Transition Table**

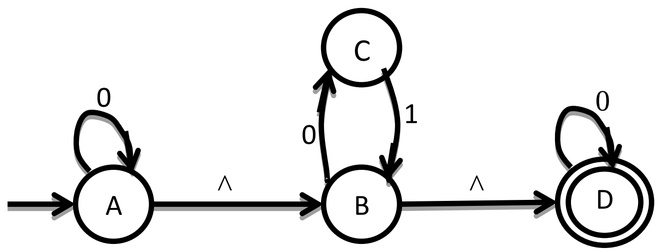


**Fig. 2.25 NFA-^**

$\delta^*(A, \wedge) = \{A,B,D\}$

$\delta^*(A,0) = \wedge\left(\bigcup_{r \in \delta^*(A,\wedge)} \delta(r,0)\right)$

$= \wedge\left(\bigcup_{r \in (A,B,D)} \delta(r,0)\right)$

$= \wedge(\delta(A,0) \cup \delta(B,0) \cup \delta(D,0))$

$= \wedge\{A,C,D\}$

$=\{A,B,C,D\}$

$\delta^*(A,1) = \wedge\left(\bigcup_{r \in \delta^*(A,\wedge)} \delta(r,1)\right)$

$= \wedge\left(\bigcup_{r \in (A,B,D)} \delta(r,1)\right)$

$= \wedge(\delta(A,1) \cup \delta(B,1) \cup \delta(D,1))$

$= \emptyset$

$\delta^*(B,\wedge) = \{B,D\}$

$\delta^*(B,0) = \wedge\left(\bigcup_{r \in \delta^*(B,\wedge)} \delta(r,0)\right)$

$= \wedge\left(\bigcup_{r \in \delta(B,D)} \delta(r,0)\right)$

$= \wedge(\delta(B,0) \cup \delta(D,0))$

$= \wedge\{C,D\}$

$= \{C,D\}$

$\delta^*(B,1) = \wedge\left(\bigcup_{r \in \delta^*(B,\wedge)} \delta(r,1)\right)$

$= \wedge\left(\bigcup_{r \in (B,D)} \delta(r,1)\right)$

$= \wedge(\delta(B,1) \cup \delta(D,1))$

$= \emptyset$

$\delta^*(C,\wedge) = \{C\}$

$\delta^*(C,0) = \wedge\left(\bigcup_{r \in \delta^*(C,\wedge)} \delta(r,0)\right)$

$= \wedge\left(\bigcup_{r \in (C)} \delta(r,0)\right)$

$= \wedge(\delta(C,0))$

$= \emptyset$

$\delta^*(C,1) = \wedge\left(\bigcup_{r \in \delta^*(C,\wedge)} \delta(r,1)\right)$

$= \wedge\left(\bigcup_{r \in (C)} \delta(r,1)\right)$

$= \wedge(\delta(C,1))$

$= \wedge\{B\}$

$= \{B,D\}$

$\delta^*(D,\wedge) = \{D\}$

$\delta^*(D,0) = \wedge\left(\bigcup_{r \in \delta^*(D,\wedge)} \delta(r,0)\right)$

$= \wedge\left(\bigcup_{r \in (D)} \delta(r,0)\right)$

$= \wedge(\delta(D,0))$

$= \{D\}$

$\delta^*(D,1) = \wedge\left(\bigcup_{r \in \delta^*(D,\wedge)} \delta(r,1)\right)$

$= \wedge\left(\bigcup_{r \in (D)} \delta(r,1)\right)$

$= \wedge(\delta(D,1))$

$= \emptyset$

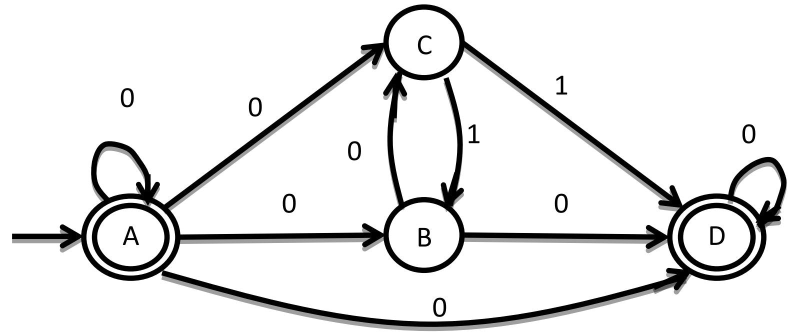

**Fig. 2.26 NFA**

| | |
|---|---|
| δ({A},0) | = {A,B,C,D} |
| δ({A},1) | = Ø |
| δ({A,B,C,D},0) | = (δ(A,0) U δ(B,0) U δ(C,0) U δ(D,0)) |
| | = {A,B,C,D} |
| δ({A,B,C,D},1) | = (δ(A,1)U δ(B,1)U δ(C,1)U δ(D,1)) |
| | = {B,D} |
| δ({B,D},0) | = (δ(B,0)U δ(D,0)) |
| | = {C,D} |
| δ({B,D},1) | = (δ(B,1)U δ(D,1)) |
| | = Ø |
| δ({C,D},0) | = (δ(C,0)U δ(D,0)) |
| | = {D} |
| δ({C,D},1) | = (δ(C,1)U δ(D,1)) |
| | = {B,D} |
| δ({D},0) | = {D} |
| δ({D},1) | = Ø |

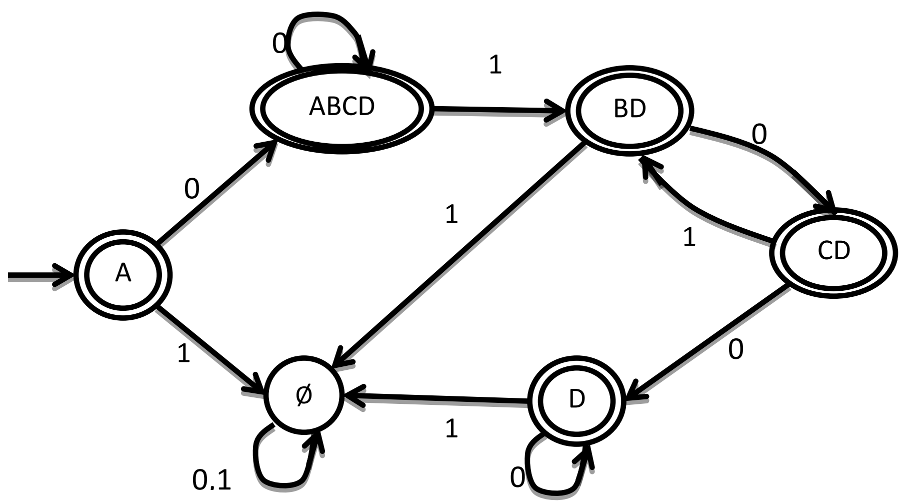


**Fig. 1.27 Finite Automata**

## 25. Convert NFA - ^ to FA

| q | δ (q,^) | δ (q,0) | δ(q,1) |
|---|---|---|---|
| A | {B,D} | {A} | Ø |
| B | Ø | {C} | {E} |
| C | Ø | Ø | {B} |
| D | Ø | {E} | {D} |
| E | Ø | Ø | Ø |

**Table 2.6. Transition Table**

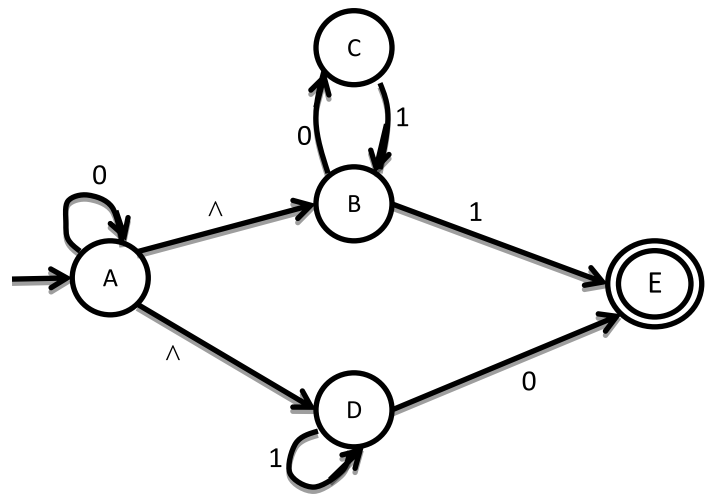


**Fig. 2.28 NFA -^**

$$\delta^*(A,\wedge) = \{A,B,D\}$$

$$\begin{aligned}
\delta^*(A,0) &= \wedge\Big(\bigcup_{r\in\delta^*(A,\wedge)} \delta(r,0)\Big)\\
&= \wedge\Big(\bigcup_{r\in\delta(A,B,D)} \delta(r,0)\Big)\\
&= \wedge(\delta(A,0) \cup \delta(B,0) \cup \delta(D,0))\\
&= \wedge\{A,C,E\}\\
&= \{A,B,C,D,E\}
\end{aligned}$$

$$\begin{aligned}
\delta^*(A,1) &= \wedge\Big(\bigcup_{r\in\delta^*(A,\wedge)} \delta(r,1)\Big)\\
&= \wedge\Big(\bigcup_{r\in(A,B,D)} \delta(r,1)\Big)\\
&= \wedge(\delta(A,1) \cup \delta(B,1) \cup \delta(D,1))\\
&= \wedge(E,D)\\
&= \{ED\}
\end{aligned}$$

$$\delta^*(B,\wedge) = \{B\}$$

$$\begin{aligned}
\delta^*(B,0) &= \wedge\Big(\bigcup_{r\in\delta^*(B,\wedge)} \delta(r,0)\Big)\\
&= \wedge\Big(\bigcup_{r\in\delta(B)} \delta(r,0)\Big)\\
&= \wedge(\delta(B,0))\\
&= \wedge\{C\}\\
&= \{C\}
\end{aligned}$$

$$\delta^*(B,1) = \wedge\Big(\bigcup_{r\in\delta^*(B,\wedge)} \delta(r,1)\Big)$$

$$= \wedge\left(\bigcup_{r \in (B)} \delta(r,1)\right)$$
$$= \wedge(\delta(B,1))$$
$$= \{E\}$$

$$\delta^*(C,\wedge) = \{C\}$$
$$\delta^*(C,0) = \wedge\left(\bigcup_{r \in \delta^*(C,\wedge)} \delta(r,0)\right)$$
$$= \wedge\left(\bigcup_{r \in (C)} \delta(r,0)\right)$$
$$= \wedge(\delta(C,0))$$
$$= \emptyset$$
$$\delta^*(C,1) = \wedge\left(\bigcup_{r \in \delta^*(C,\wedge)} \delta(r,1)\right)$$
$$= \wedge\left(\bigcup_{r \in (C)} \delta(r,1)\right)$$
$$= \wedge(\delta(C,1))$$
$$= \wedge\{B\}$$
$$=\{B\}$$
$$\delta^*(D,\wedge)= \{D\}$$
$$\delta^*(D,0) = \wedge\left(\bigcup_{r \in \delta^*(D,\wedge)} \delta(r,0)\right)$$
$$= \wedge\left(\bigcup_{r \in (D)} \delta(r,0)\right)$$
$$= \wedge(\delta(D,0))$$
$$= \wedge\{E\}$$
$$= \{E\}$$
$$\delta^*(D,1) = \wedge\left(\bigcup_{r \in \delta^*(D,\wedge)} \delta(r,1)\right)$$
$$= \wedge\left(\bigcup_{r \in (D)} \delta(r,1)\right)$$
$$= \wedge(\delta(D,1))$$
$$= \{D\}$$
$$\delta^*(E,\wedge) =\{E\}$$
$$\delta^*(E,0) = \emptyset$$
$$\delta^*(E,1) = \emptyset$$

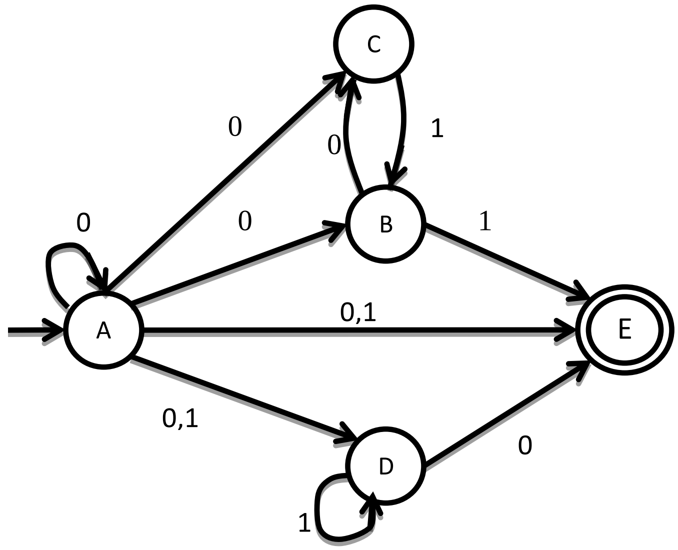


**Fig. 2.29 NFA**

| | |
|---|---|
| $\delta(\{A\},0)$ | $= \{A,B,C,D,E\}$ |
| $\delta(\{A\},1)$ | $= \{ED\}$ |
| $\delta(\{A,B,C,D,E\},0)$ | $= (\delta(A,0) \cup \delta(B,0) \cup \delta(C,0) \cup \delta(D,0) \cup \delta(E,0))$ |
| | $= \{A,B,C,D,E\}$ |
| $\delta(\{A,B,C,D,E\},1)$ | $= (\delta(A,1) \cup \delta(B,1) \cup \delta(C,1) \cup \delta(D,1) \cup \delta(E,0))$ |
| | $= \{E,B,D\}$ |
| $\delta(\{E,D\},0)$ | $= (\delta(E,0) \cup \delta(D,0))$ |
| | $= \{\emptyset\} \cup \{E\}$ |
| | $= \{E\}$ |
| $\delta(\{E,D\},1)$ | $= (\delta(E,1) \cup \delta(D,1))$ |
| | $= \{\emptyset\} \cup \{D\}$ |
| | $= \{D\}$ |
| $\delta(\{B,E,D\},0)$ | $= (\delta(B,0) \cup \delta(E,0) \cup \delta(D,0))$ |
| | $= \{C,E\}$ |
| $\delta(\{B,E,D\},1)$ | $= (\delta(E,1) \cup \delta(D,1))$ |
| | $= \{D,E\}$ |
| $\delta(\{C,E\},0)$ | $= (\delta(C,0) \cup \delta(E,0))$ |
| | $= \emptyset$ |
| $\delta(\{C,E\},1)$ | $= (\delta(C,1) \cup \delta(E,1))$ |
| | $= \{B\}$ |
| $\delta(\{D\},0)$ | $= \{E\}$ |
| $\delta(\{D\},1)$ | $= \{D\}$ |
| $\delta(\{B\},0)$ | $= \{C\}$ |
| $\delta(\{B\},1)$ | $= \{E\}$ |
| $\delta(\{E\},0)$ | $= \emptyset$ |
| $\delta(\{E\},1)$ | $= \emptyset$ |

δ({C},0) = Ø
δ({C},1) = {B}

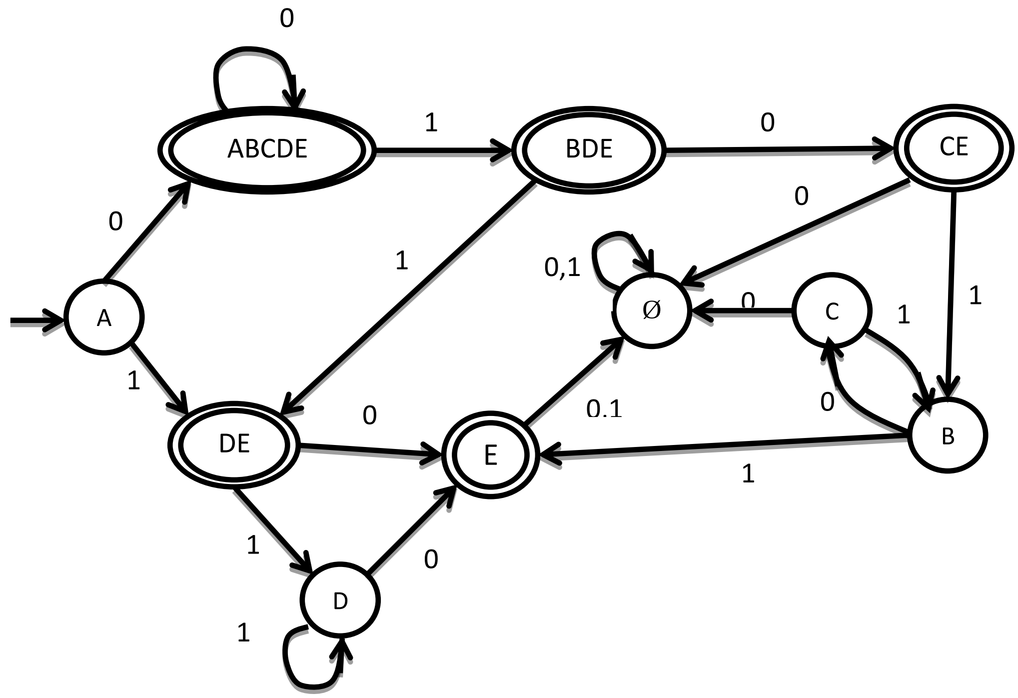


**Fig.2.30 Finite Automata**

## 26. Kleene's Theorem Part-1 or Prove: Any Regular Language can be accepted by finite automata.

**Proof:**

- On the basis of statement L can be recognized by FA, NFA and NFA-^. It is sufficient to so that every regular language can be accepted by NFA- ^.
- Set of regular language over alphabet Σ contains the basic languages. Ø, {^} and {a} (a ∈ Σ) to be closed under operation of union, concatenation, and Kleene*.
- This allows us to prove using structural induction that every regular language over Σ can be accepted by an NFA-^.
- The basis step of the proof is to show that the three basic languages can be accepted by NFA-^s.
- The induction hypothesis is that L1 and L2 are languages that can be accepted by NFA-^s, and the induction step is to show that L1 U L2, L1L2, and $L_1$* can also be accepted by NFA-^s.
- NFA-^ for the three basic languages is shown below.

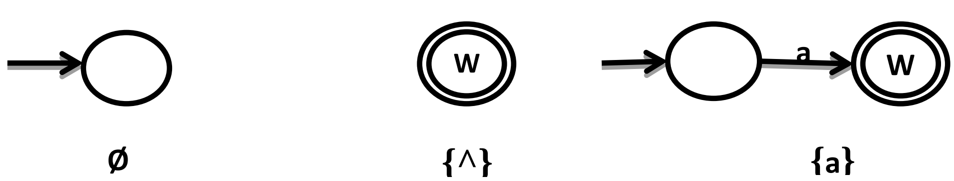

**Fig. 2.31 Basic Languages for NFA-^**

- Now, suppose that L1 and L2 are recognized by the NFA-^s M1 and M2, respectively, where for both i=1 and i=2,

  $M_i=(Q_i,\varepsilon,q_i,A_i,\delta_i)$
- By renaming state if necessary, we may assume that Q1 ∩ Q2 = Ø. We will construct NFA-^s $M_u$, $M_c$, and $M_k$ recognizing the language L1 U L2, L1L2, and $L_1$*, respectively.

**Construction Of $M_u$**

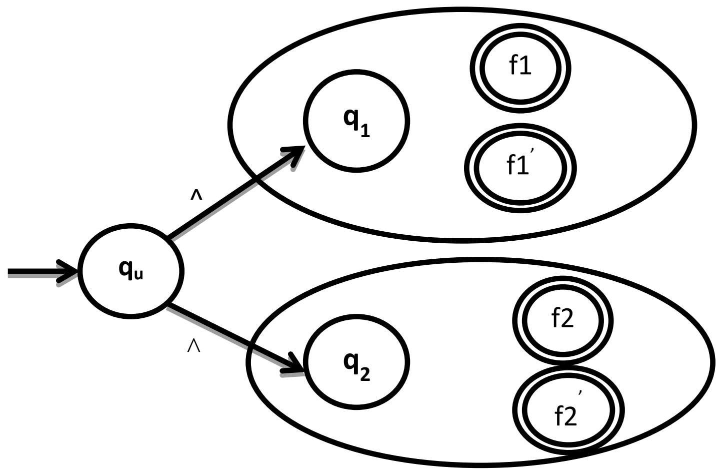


**Fig. 2.32 Construction Of $M_u$**

- Construction of $M_u = (Q_u, \varepsilon, q_u, A_u, \delta_u)$. Let $q_u$ be a new state, not in either Q1 or Q2 and let

  $Q_u$ = Q1 U Q2 U { $q_u$ }

  $A_u$ = A1 U A2
- Now, we define $\delta_u$ so that $M_u$ can move from its initial state to either q1 or q2 by a ^ transition, and then make exactly the same moves that the respective $M_i$ would. Normally we define:

  $\delta_u(q_u,\text{^}) = \{q1,q2\}$

  $\delta_u(q_u,a) = \emptyset$ for every a ∈ ε

  And for each q∈ Q1 U Q2 and a ∈ ε U {^},

  $\delta_u(q_u,a)$={ $\delta_1(q,a)$ if q ∈ Q1} and { $\delta_2(q,a)$ if q ∈ Q2}
- For either value of i, if x∈ $L_i$, then $M_u$ can process x by moving to $q_i$ on a ^-transition and then executing the moves that cause $M_i$ to accept x, on the other hand, if x is accepted by $M_u$, there is a sequence of transition corresponding to x, starting at $q_u$ and ending at an element of A1 or A2. The first of these transition must be a ^-transition from $q_u$ to either q1 or q2, since there are no other transition from $q_u$. therefore, since Q1 ∩ Q2 = Ø, either all the transition are between of Q1 or all are between elements of $Q_2$. It follow that x must be accepted by either M1 or M2.

**Construction Of $M_c$**

- Construction of $M_c = (Q_c, \varepsilon, q_c, A_c, \delta_c)$. In this case we do not need any new states, Let $Q_c$ = Q1 U Q2, $q_c=q_1$, and $A_c = A_2$. The transition will include all those of M1 and M2 as well as a ε-transition from each state in $A_1$ to $q_2$.
- In other words, for any q not in A1, and α ∈ ε U {^}, $\delta_c(q,a)$ is defined to be either $\delta_1(q,a)$ or $\delta_2(q,a)$, depending on whether q is in Q1 and Q2, for q ∈ A1.

$\delta_c(q,a) = \delta_1(q,a)$ for every $a \in \varepsilon$,
And $\delta_c(q,\wedge) = \delta_1(q,\wedge) \cup \{q2\}$

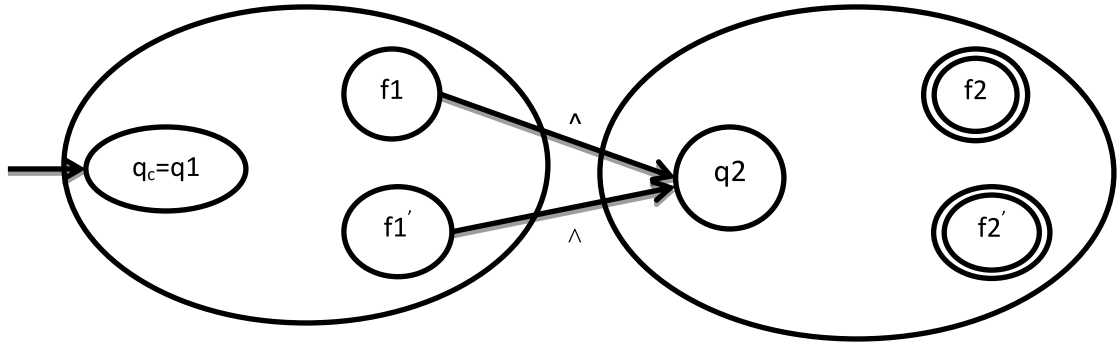


**Fig. 2.33 Construction Of $M_c$**

- On an input string x1x2, where $x_i \in L_i$ for both value of i, $M_c$ can process x1, arriving at a state A1; jump from this state to q2 by a ^-transition; and then process x2 the way M2 would, So that x1x2 is accepted. Conversely, if x is accepted by $M_c$, there is a sequence of transition corresponding to x that begins at q1 and ends at an element of A2. One of them must therefore be from an element of Q1 to an element Q2, and according to the definition of $\delta_c$, this can only be a ^- transition from an element of A1 to q2. Because $Q1 \cap Q2 = \emptyset$, all the previous transition are between elements of Q1 and all the subsequent ones are between elements of Q2. It follows that x=x1^x2=x1x2, where x1 is accepted by M1 and x2 is accepted by M2; in other words, $x \in L1L2$.

**Construction Of $M_k$**

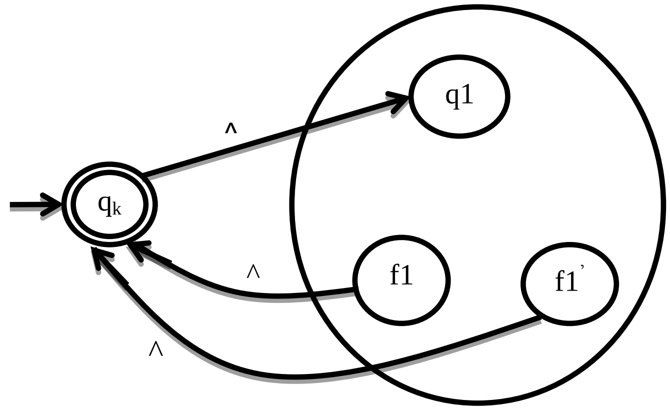


**Fig. 2.34 Construction Of $M_k$**

- Construction of $M_k = (Q_k, \Sigma, q_k, A_k, \delta_k)$. Let $q_k$ be a new state not in Q1 and let $Q_k = Q1 \cup \{q_k\}$. Once again all the transitions of M1 will be allowed in $M_k$, but in addition there is a ^-transition from $q_k$ to $q_1$ and there is a ^-transition from each elements of A1 to $q_k$. More precisely,

  $\delta_k(q_k,\wedge) = \{q1\}$ and $\delta_k(q_k,a) = \emptyset$ for $a \in \varepsilon$.
  for $q \in A1$, $\delta_k(q_k,\wedge) = \delta_k(q_k,\wedge) \cup \{q_k\}$.

- Suppose $x \in L1^*$. if x=^ then clearly x is accepted by $M_k$. Otherwise, for some $m \geq 1$, x=x1x2….xm, where xi ∈ L1 for each i, $M_k$ can move from qk to q1 by a ^-transition; for each i, $M_k$ moves from q1 to an element $f_i$ of A1 by a sequence of transition corresponding to $x_i$; and for each i, $M_k$ then moves from $f_i$ back to qk by a ^-transition.

- It follows that (^x1^) (^x2^)…… (^xm^)=x is accepted by $M_k$. On the other hand, if x is accepted by $M_k$, there is a sequence of transition corresponding to x that begins and ends at $q_k$. Since the only transition from qk is a ^-transition to q1, and the only transition to qk are ^-transition from elements of A1, x can be decomposed in the form
  x= (^x1^) (^x2^)…… (^xm^)
- Where, for each i, there is a sequence of transition corresponding to $x_i$ from q1 to an element of A1. Therefore, x∈ L1*.
- Since we have constructed an NFA-^ recognizing L in each of the three cases, the proof is complete.

## 27. Draw NFA-^ for following regular expression.

**1. (00+1)*(10)***

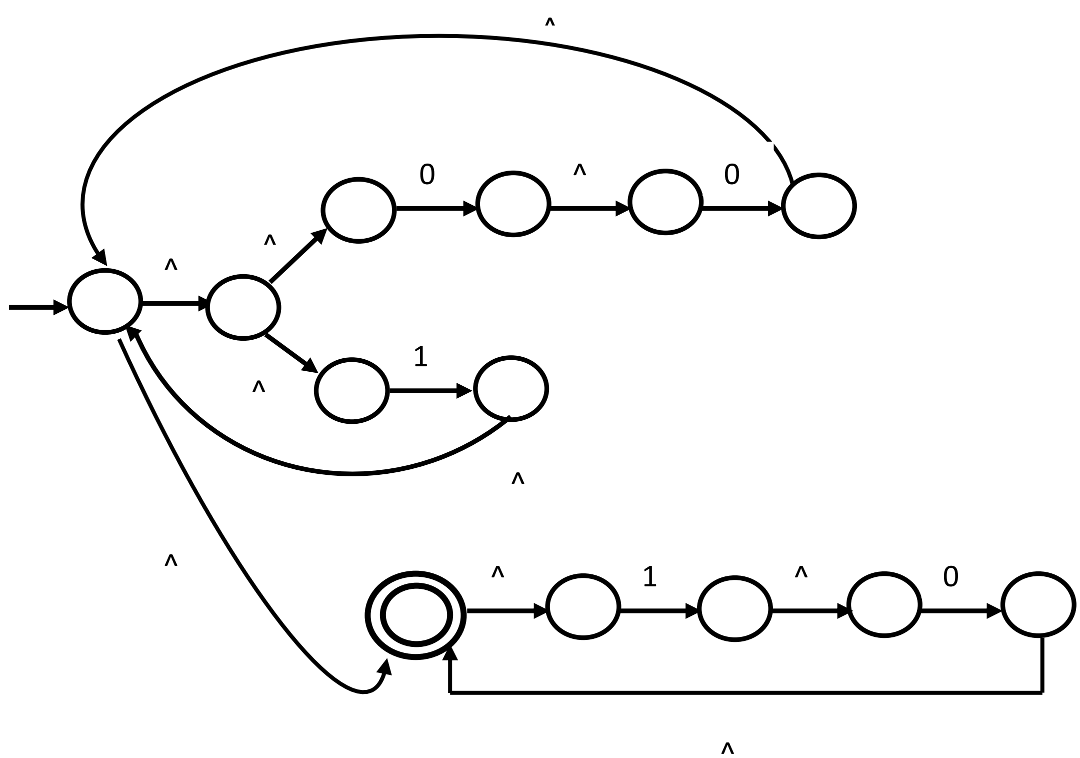


**Fig. 2.35 NFA-^ for (00+1)*(10)***

**2. (0 + 1)* (10+01)* 11**

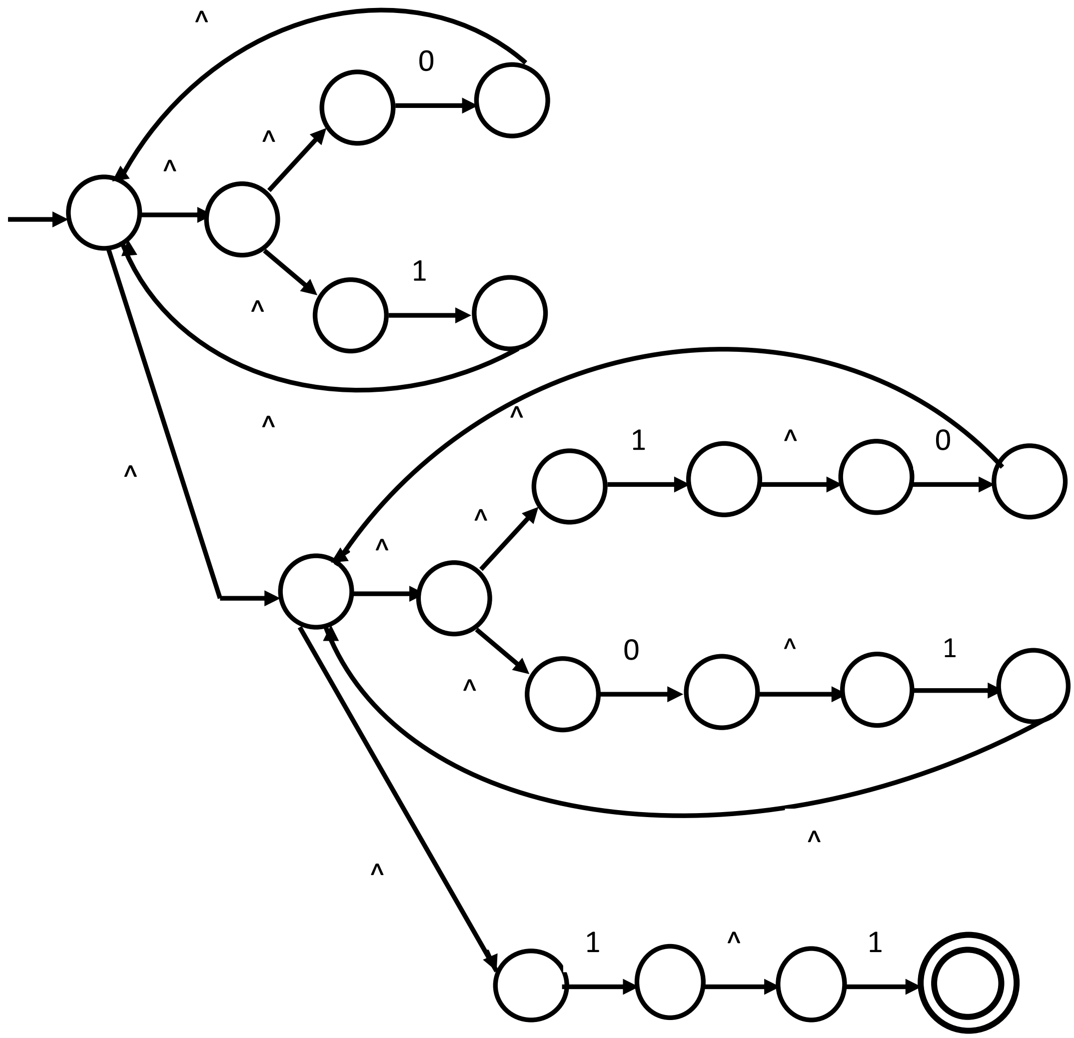


**Fig. 2.36 NFA-^ for(0+1)*(10+01)*11**

**3. (0 + 1)* (10+110)* 1**

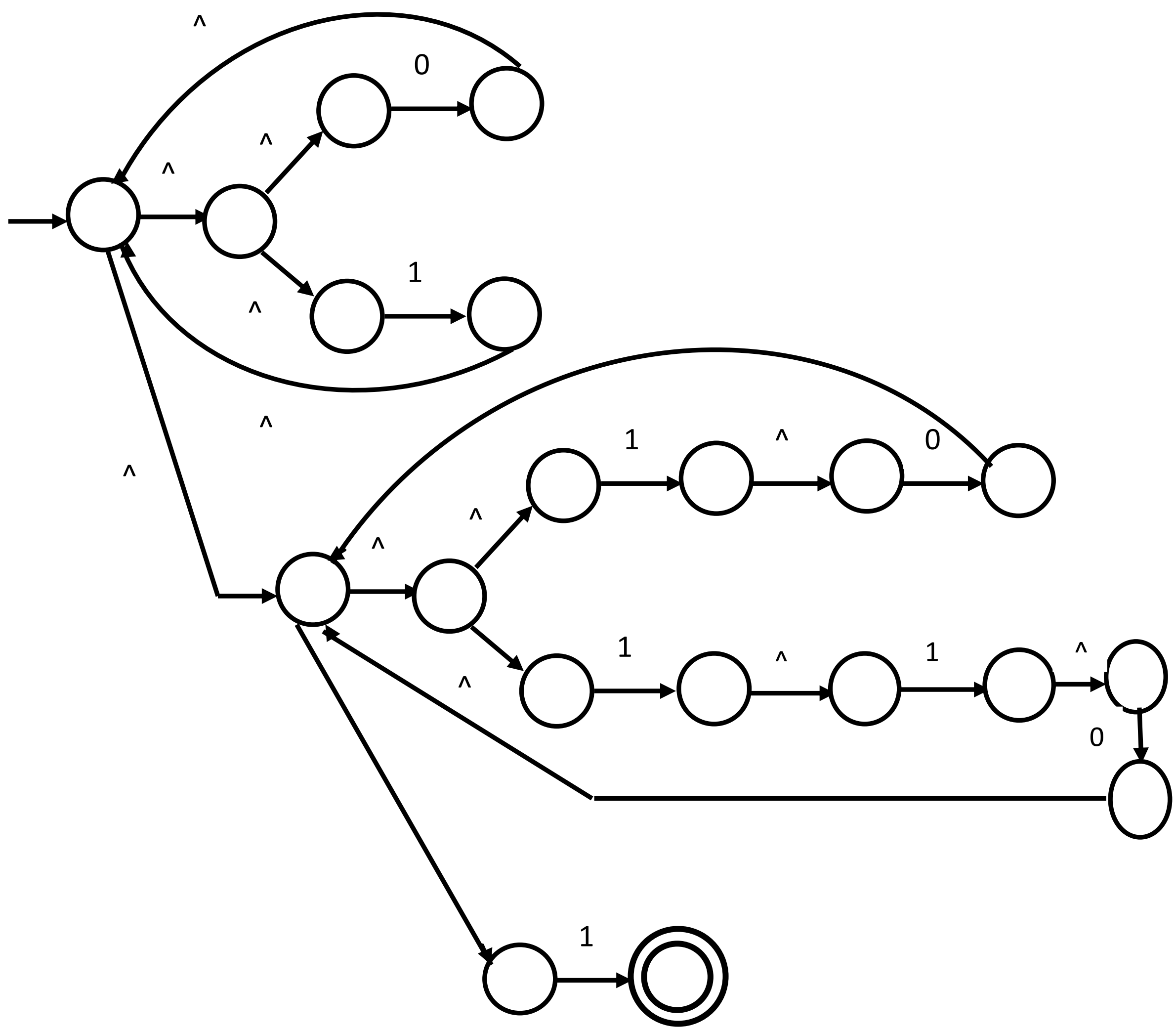


**Fig. 2.37 NFA-^ for(0+1)*(10+110)*1**

# 28. Finite Automata with Output.

- Finite automata has limited capability of either accepting a string or rejecting a string.
- Accepance of string was based on the reachability of a machine from starting state to final state. Finite automata can also be used as an output device.
- Such machines do not have a final state.
- Machine generates output on every input.
- There are two types of automata with outputs:
  1. Moore machine
  2. Mealy machine

## 29. Moore Machine

- Mathematiically moore machine is a six tuple machine and define as Mo=( Q,Σ, Δ, δ , /\’,$q_0$) where
  Q: A Nonempty finite set of state in Mo
  Σ: A Nonempty finite set of input symbols
  Δ: A Nonempty finite set of outputs
  δ: It is transition function which takes two arguments as in finite automata, one is input state and other is input symbol. The output of this function is a single state, so clearly δ is the function which is responsible for the transition of Mo.
  /\’: it is a mapping function which maps Q to Δ, giving the output associated with each state.
  $q_0$ : Is the initial state of Mo and $q_0 \in Q$.

**Examples of Moore Machine**

1. **Design a moore machine for the 1’s compliment of binary number.**

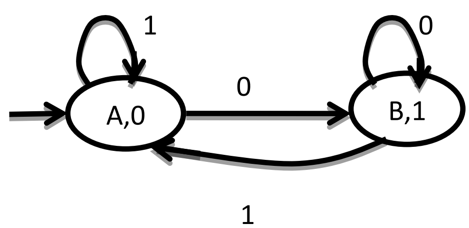


**Fig. 2.38 Moore M/c for 1’s compliment**

2. **Design a moore machine to count occurance of “ab” as substring.**

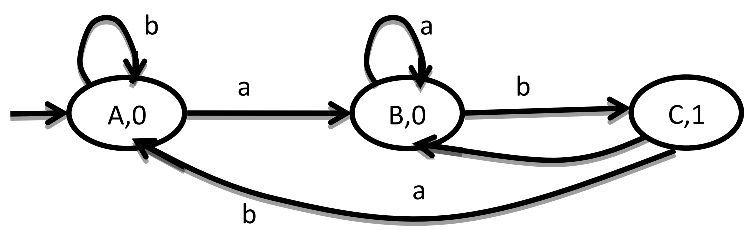


**Fig. 2.39 Moore M/c to count occurances of ab**

3. **Construct a moore machine that takes set of all strings over {0, 1} and produces ‘A’ if i/p ends witg ‘10’ or produces ‘B’ if i/p ends with ’11’ otherwise produces ‘C’.**

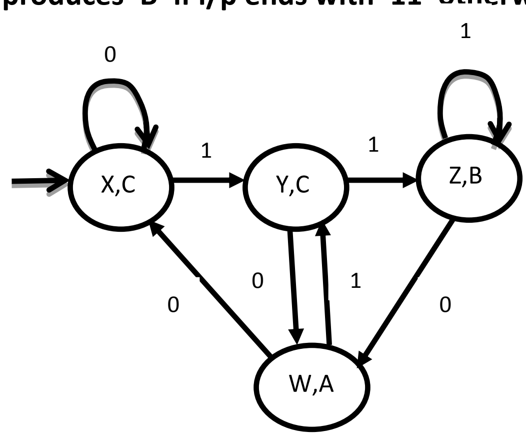

**Fig. 2.40 Moore M/c for string ending in 10 or 11**

4. **Construct a moore machine that takes binary number as an i/p and produces residue modulo '3' as an output.**

| | 0 | 1 | Δ |
|---|---|---|---|
| q0 | q0 | q1 | 0 |
| q1 | q2 | q0 | 1 |
| q2 | q1 | q2 | 2 |

**Table 2.7transition Table**

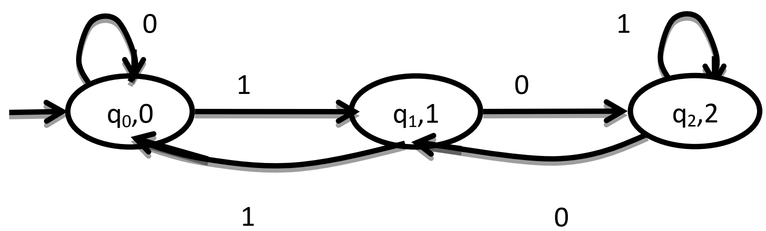


**Fig. 2.41 Moore M/c to produces residue modulo '3'**

# 30. Mealy Machine

- It is a finite automata in which output is associated with each transition.
- Mathematiically mealy machine is a six tuple machine and define as Me=( Q,Σ, Δ, δ , $\lambda'$,$q_0$) where
  Q: A Nonempty finite set of state in Me
  Σ: A Nonempty finite set of input symbols
  Δ: A Nonempty finite set of outputs
  δ: It is transition function which takes two arguments as in moore machine, one is input state and other is input symbol. The output of this function is a single state, so clearly δ is the function which is responsible for the transition of Me.
  $\lambda'$: it is a mapping function which maps Q x Σ to Δ, giving the output associated with each transition.
  $q_0$ : Is the initial state of Me and $q_0 \in Q$.

**Examples of Mealy Machine**

1. **Design a mealy machine for the 1's compliment of binary number.**

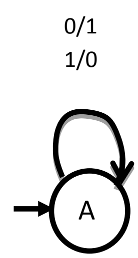


**Fig. 2.42 Mealy M/c for 1's compliment of binary**

2. **Design a mealy machine for regular expression (0+1)*(00+11).**

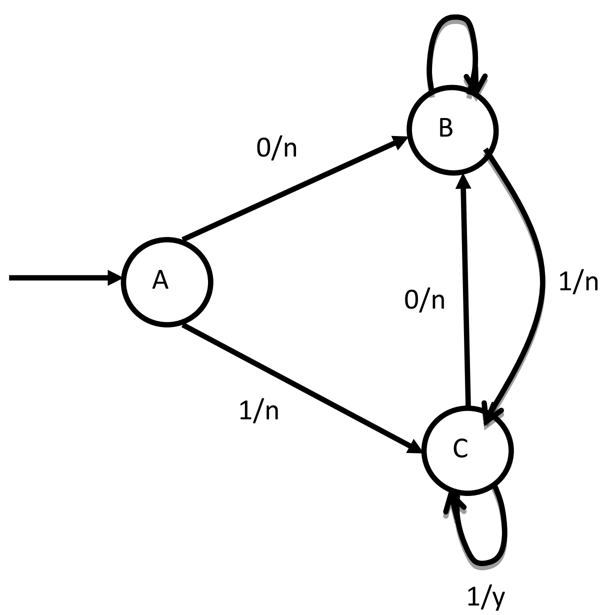


**Fig. 2.43 Mealy M/c for (0+1)*(00+11)**

3. **Design a mealy machine where Σ={0, 1, 2} print residue modulo 5 of input treated as ternary (base 3).**

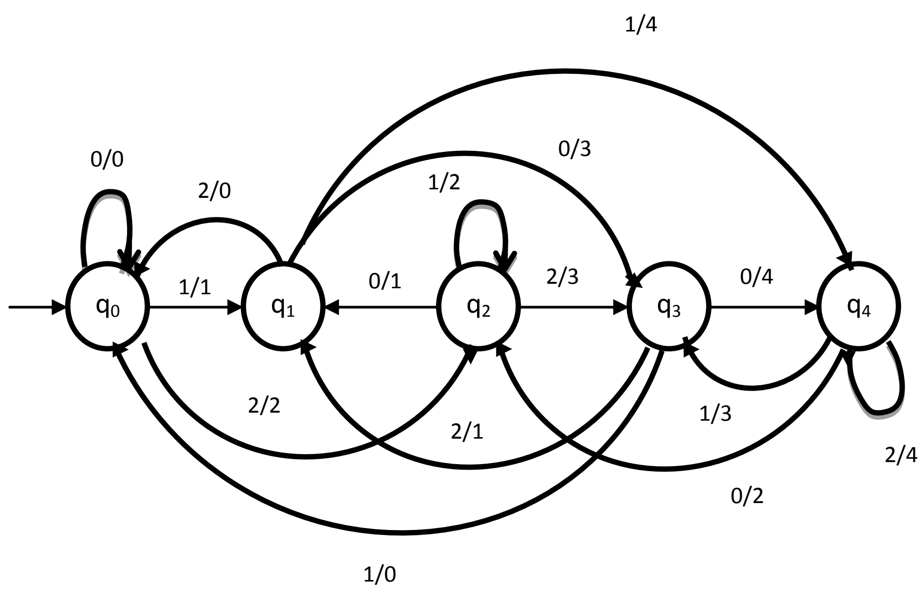


**Fig. 2.44 Mealy M/c to produces residue modulo '5'**

# 31. Explain procedure to minimize Finite Automata

**S-1**: Make final state and non-final state as distinguish.

**S-2**: Recursively interacting over the pairs of state for any transition for

δ(p,x) = r

δ(q,x) = s

and for x ∈ ε. If r and s are distinguishable make p and q as distinguish.

**S-3**: If any iteration over all possible state pairs one fails to find a new pair of states that are distinguishable terminate.

**S-4**: All the states that are not distinguished are equivalence.

## 32. For following FA find minimized FA accepting same language.

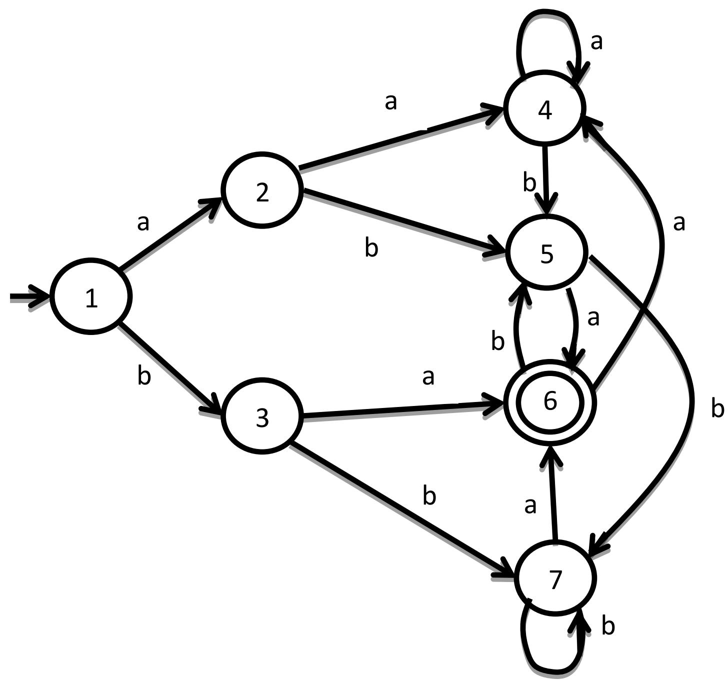


**Fig. 2.45 Finite Automata**

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | | | | | | |
| 3 | X | X | | | | |
| 4 | | | X | | | |
| 5 | X | X | | X | | |
| 6 | X | X | X | X | X | |
| 7 | X | X | | X | | X |
| | 1 | 2 | 3 | 4 | 5 | 6 |

Final state is {6}

And, Non-Final state is {1,2,3,4,5,7}

(6, 1), (6,2), (6, 3), (6, 4), (6, 5), (6, 7) are distinguish pairs.

Consider pair (1,2)

δ(1,a)=2 δ(1,b)=3

δ(2,a)=4 δ(2,b)=5

Consider pair (1,3)

δ(1,a)=2 δ(1,b)=3

δ(3,a)=6 δ(3,b)=7

pair (2,6) is distinguish, so It's a distinguished pair.

Consider pair (1,4)

δ(1,a)=2 δ(1,b)=3

δ(4,a)=4 δ(4,b)=5

Consider pair (1,5)

δ(1,a)=2 δ(1,b)=3

δ(5,a)=6 δ(5,b)=7

pair (2,6) is distinguish, so It's a distinguished pair.

Consider pair (1,7)

δ(1,a)=2 δ(1,b)=3

δ(7,a)=6 δ(7,b)=7

pair (2,6) is distinguish, so It's a distinguished pair.

Consider pair (2,3)

δ(2,a)=4 δ(2,b)=5

δ3(,a)=6 δ(3,b)=7

pair (4,6) is distinguish, so It's a distinguished pair.

Consider pair (2,4)

δ(2,a)=4 δ(2,b)=5

δ(4,a)=4 δ(4,b)=5

Consider pair (2,5)

δ(2,a)=4 δ(2,b)=5

δ(5,a)=6 δ(5,b)=7

pair (4,6) is distinguish, so It's a distinguished pair.

Consider pair (2,7)

δ(2,a)=4 δ(2,b)=5

δ(7,a)=6 δ(7,b)=7

pair (4,6) is distinguish, so It's a distinguished pair.

Consider pair (3,4)

δ(3,a)=6 δ(3,b)=7

δ(4,a)=4 δ(4,b)=5

pair (6,4) is distinguish, so (3,4) is distinguish.

Consider pair (3,5)

δ(3,a)=6 δ(3,b)=7

δ(5,a)=6 δ(5,b)=7

Consider pair (3,7)

δ(3,a)=6 δ(3,b)=7

δ(7,a)=6 δ(7,b)=7

Consider pair (4,5)

δ(4,a)=4 δ(4,b)=5

δ(5,a)=6 δ(5,b)=7

pair (6,4) is distinguish, so (4,5) is distinguish.

Consider pair (4,7)

δ(4,a)=4 δ(4,b)=5

δ(7,a)=6 δ(7,b)=7

pair (6,4) is distinguish, so (4,7) is distinguish.

Consider pair (5,7)

δ(5,a)=6 δ(5,b)=7

δ(7,a)=6 δ(7,b)=7

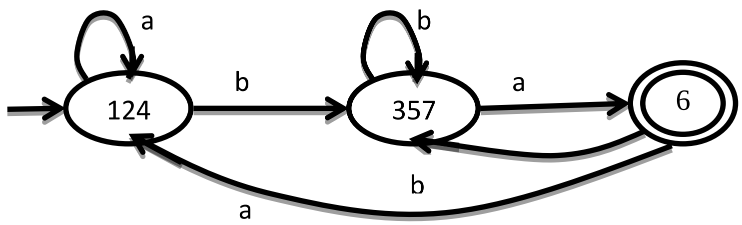


**Fig. 2.46 Minimized Finite Automata**

## 33. Define pumping lemma and its application.

Suppose L is a regular language. Then there is an integer n so that for any x∈ L with |x|>=n, there are strings u, v, and w so that

1. x=uvw
2. |uv|<=n
3. |v|>0
4. For any m>=0, $uv^m w$ ∈ L

**Application: (Explain the application of the Pumping Lemma to show a Language is Regular or Not)**

The pumping lemma is extremely useful in proving that certain sets are non-regular. The general methodology followed during its applications is :

- Select a string z in the language L.
- Break the string z into x, y and z in accordance with the above conditions imposed by the pumping lemma.
- Now check if there is any contradiction to the pumping lemma for any value of i.

## 34. Use the pumping lemma to show that following language is not regular: L = {ww|w ϵ {0,1}*}

Step 1: Let us assume that L is regular and L is accepted by an FA with n states.

Step 2: Let us chose the string

$$\omega = \underbrace{a^n b}_{\omega} \ \underbrace{a^n b}_{\omega}$$

$|\omega| = 2n+2 >= n$

Let us write w as xyz with

$|y|>0$
And $|xy|<=n$
Since $|xy|<=n$, x must be of the form $a^s$.
Since $|xy|<=n$, y must be of the form $a^r \mid r>0$
Now $\omega=a^n b a^n b=\underbrace{a^s}_{x}\underbrace{a^r}_{y}\underbrace{a^{n-s-r} b a^n b}_{z}$

Step 3: Let us check whether $x y^i z$ for i=2 belongs to L.
$Xy^2z= a^s a^{2r} a^{n-s-r} b a^n b= a^{n+r} b a^n b$
Since, r>0, $a^{n+r} b a^n b$ is not of the form $\omega\omega$ as the number of a's in the first half is n+r and second half is n. Therefore, $xy^2z \notin L$. Hence by contrdiction we can say language is not regular.

## 35. Prove that the language L = {$0^n$: n is a prime number} is not regular.

Step 1: Let us assume that L is regular and L is accepted by an FA with n states.
Step 2: Let us chose the string
$\omega=a^p$ ,where p is prime and p>n
$|\omega| = |a^p| = p>=n$
Let us write w as xyz with
$|y|>0$
And $|xy|<=n$
Since $|xy|<=n$, x must be of the form $a^s$.
We assume that $y= a^m$ for m>0.
Step 3: Length of $x y^i z$ can be written as given below.
$Xy^iz= |xyz|+|y^{i-1}|=p+(i-1)m$
As $|y|=|a^m|=m$
Let us check whether P(i-1) m is prime for every i.
For i=p+1, p+(i-1)m= P+ $P_m$ =P(1+m)
So $xy^{p+1}z \notin L$. Hence by contrdiction we can say language is not regular.

## 36. Use Pumping Lemma to show that following language is not regular. L = { $ww^R$ / w ε {0,1}* }

Step 1: Let us assume that L is regular and L is accepted by an FA with n states.
Step 2: Let us chose the string
$\omega=\underbrace{a^n b}_{\omega}\ \underbrace{b a^n}_{\omega^R}$
$|\omega|=2n+2>=n$
Let us write w as xyz with
$|y|>0$
And $|xy|<=n$
Since $|xy|<=n$, x must be of the form $a^s$.
Since $|xy|<=n$, y must be of the form $a^r \mid r>0$
Now $\omega=a^n bba^n=\underbrace{a^s}_{x}\underbrace{a^r}_{y}\underbrace{a^{n-s-r} bba^n}_{z}$

Step 3: Let us check whether x $y^i$ z for i=2 belongs to L.

$Xy^2z= a^s\, a^{2r}\, \underline{a^{n-s-r}}\underline{bba^n}= a^{n+r}bba^n$

Since, r>0, $a^{n+r}bba^n$ is not of the form $\omega\omega^R$ as the string starts with (n+r) a's but ends in (n) a's. Therefore, $Xy^2z \notin L$. Hence by contrdiction we can say language is not regular.