# FORMAL LANGUAGES AND AUTOMATA THEORY

## UNIT-I: Finite Automata

### Objective:

To familiarize how to employ deterministic and non-deterministic finite automata.
To familiarize how to employ non-deterministic finite automata with ε transitions and finite automata with outputs.

### Syllabus:

Strings, alphabet, language, operations, finite state machine, finite automaton model, transition diagrams, acceptance of strings and languages, deterministic finite automaton and non-deterministic finite automaton, NFA to DFA conversion, NFA with epsilon transitions - significance, equivalence between NFA with and without E transitions, minimization of FSM, equivalence between two FSM's, finite automata with output- Moore and Mealy machines, applications of FA.

### Learning Outcomes:

Students will be able to:

- Understand the basic definitions like alphabet, string, language and their operations.
- Understand the model of FA.
- Design DFA and NFA for the given regular language.
- Test the designed DFA and NFA for the set of strings that belongs to L and for the set of strings that doesn't belongs to L.
- Convert NFA to DFA and NFA with epsilon transitions to NFA without Epsilon transitions.
- Minimize the given DFA.
- Test whether the two DFA's are equivalent or not.
- Design Moore and Mealy Machines

## 1.Learning Material

### 1.1 Alphabet:

An alphabet is a finite, nonempty set of symbols. It is denoted by ∑.

***Example:***

∑= {0, 1} is binary alphabet consisting of the symbols 0 and 1.
∑= {a, b, c ...z} is lowercase English alphabet.

#### 1.1.1Powers of an Alphabet

If Σ is an alphabet, we can express the set of all strings of a certain length from that alphabet by using the exponential notation. It is denoted by $\Sigma^k$ - the set of strings of length k.

***Example:***

$\Sigma^0 = \{\varepsilon\}$, regardless of what alphabet Σ is. ε is the only string of length 0.
If $\Sigma = \{0, 1\}$ then,
$\Sigma^1 = \{0, 1\}$
$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
The set of all strings over an alphabet $\Sigma$ is denoted by $\Sigma^*$. $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
For example, $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, .....\}$
The symbol $*$ is called ***Kleene star*** and is named after the mathematician and logician Stephen Cole Kleene.

The symbol + is called ***Positive closure*** i.e. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$

$$\boxed{\Sigma^* = \Sigma^+ \cup \{ \varepsilon \}}$$

## 1.2 String:

A string (or word) is a finite sequence of symbols chosen from some alphabet. The letters u, v, w, x, y and z are used to denote string.

***Example:***
If $\Sigma = \{a, b, c\}$ then abcb is a string formed from that alphabet.

- The ***length*** of a string w, denoted $|\mathbf{w}|$, is the number of symbols composing the string.

***Example:***
The string abcb has length 4.

- The ***empty string*** denoted by $\boldsymbol{\varepsilon}$, is the string consisting of zero symbols.
  Thus $|\varepsilon| = 0$.

### 1.2.1 Operations on strings:

- ***Concatenation of strings***

  The concatenation of two strings is the string formed by writing the first, followed by the second, with no intervening space. Concatenation of strings is denoted by $\circ$.

  That is, if w and x are strings, then wx is the concatenation of these two strings.

  ***Example:***
  The concatenation of dog and house is doghouse.
  Let x=0100101 and y= 1111 then $x \circ y$=01001011111

- ***String Reversal***

  Reversing a string means writing the string backwards.
  It is denoted by $w^R$

  ***Example:***
  Reverse of the string abcd is dcba.

  ***If w= $w^R$ , then that string is called palindrome.***

- ***Substring***

  A substring is a part of a string.

  ***Example:***
  If abcd is string then possible substrings are $\boldsymbol{\varepsilon}$,a,b,c,d,ab,bc,cd,abc,bcd are proper substrings for the given string

  A ***prefix*** of a string is any number of leading symbols of that string.
  A ***suffix*** of a string is any number of trailing symbols.

  ***Example:***
  String abc has prefixes $\boldsymbol{\varepsilon}$, a, ab, and abc; its suffixes are $\boldsymbol{\varepsilon}$, c, bc, and abc.

A prefix or suffix of a string, other than the string itself, is called a ***proper prefix or suffix***.

## 1.3 Language:

A (formal) language is a set of strings of symbols from some one alphabet.It is denoted by L. We denote this language by ∑*.

- The empty set, Ø, and the set consisting of the empty string {**ε**} are languages.

***Example:***

If ∑= {a}, then ∑* = {**ε**, a, aa, aaa, ...}.

If ∑ = {0, 1}, then ∑* = {**ε**, 0, 1, 00, 01, 10, 11, 000,...}.

## 1.4Operations on languages:

- ***Union***

  If L1 and L2 are two languages over an alphabet ∑.Then the union of L1 and L2 is denoted by L1 U L2.

  ***Example:***

  L1={0,01,011} and L2={001},then L1 U L2={0,01,011,001}

- ***Intersection***

  If L1 and L2 are two languages over an alphabet ∑.Then the intersection of L1 and L2 is denoted by L1 ∩ L2.

  ***Example:***

  L1= {0, 01,011} and L2= {01}, then L1 ∩ L2= {01}

- ***Complementation***

  ***L is a language*** over an alphabet ∑, then the complement of L denoted by L¯ ,is the language consisting of those strings that are not in L over the alphabet.

  ***Example:***

  If ∑={a,b} and L={a,b,aa} then L¯= ∑* - L={ **ε,a,b,aa,bb,ab.........}-{a,b,aa}={ε,bb,ab,ba.........}**

- ***Concatenation***

  Concatenation of two languages L1 and L2 is the language L1 o L2 ,each element of which is a string formed by combining one string of L1 with another string of L2.

  ***Example:***

  ***L1***={bc,bcc,cc}andL2={cc,ccc},thenL1oL2={bccc,bcccc,bcccccc,cccc,ccccc}

- ***Reversal***

  If L is language, then $L^R$ is obtained by reversing the corresponding string in L. This operation is similar to the reversal of a string.

$$\boxed{L^R = \{w^R \mid w \in L\}}$$

***Example:***

If L= {0, 011, 0111}, then $L^R$ = {0, 110, 1110}

- ***Kleene Closure***

  The Kleene closure (or just closure) of L, denoted L*, is the set

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

and the positive closure of L, denoted $L^+$ , is the set

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

That is, L* denotes words constructed by concatenating any number of words from L.
L+ is the same, but the case of zero words, whose "concatenation" is defined to be **ε**, is excluded. Note that L+ contains **ε** if and only if L does.

***Example:***

Let L1 = {10, 1}

Then **L *** = L0 U L1 U L2.................. = {ε, 1, 10, 11, 111, 1111 ,..........}

**L** $^{+}$ = L1 U L2 U L3..................= {1, 10, 11, 111, 1111...........}

## 1.5 Finite Automaton:

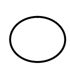
- A finite automaton (FA) consists of a finite set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet ∑.
- For each input symbol there is exactly one transition out of each state (possibly back to the state itself).
- One state, usually denoted $q_0$ is the initial state, in which the automaton starts. Some states are designated as final or accepting states.

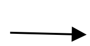
Formally, a finite automaton is denoted by a 5-tuple **(Q, ∑, δ, q₀, F),** where

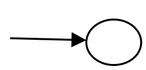
- Q is a finite set of states.
- ∑ is a finite input alphabet.
- **δ** is the transition function mapping Q x ∑ to Q i.e., **δ** (q,a) is a state for each state q and input symbol a.
- qo ∈ Q is the initial state.
- F ⊆ Q is the set of final states. It is assumed here that there may be
- more than one final state.

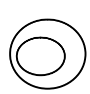
***Transition Diagram***

- A transition diagram is a directed graph associated with an FA in which the vertices of the graph correspond to the states of the FA.
- If there is a transition from state q to state p on input a, then there is an arc labelled a from state q to state p in the transition diagram.

State is denoted by 

Transition is denoted by 

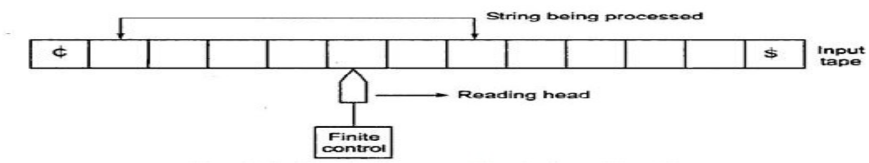
Initial state is denoted by 

Final state is denoted by 

## Transition Table

A tabular representation in which rows correspond to states, columns correspond to inputs and entries correspond to next states.

## 1.6 Finite Automata Model:




**Block diagram of a finite automaton**

The various components are explained as follows:

(i) ***Input tape:***

- The input tape is divided into squares, each square containing a single symbol from the input alphabet ∑.
- The end squares of the tape contain the endmarker ¢ at the left end and the endmarker $ at the right end.
- The absence of endmarkers indicates that the tape is of infinite length. The left-to-right sequence of symbols between the two endmarkers is the input string to be processed.

(ii) ***Reading head:***

- The head examines only one square at a time and can move one square either to the left or to the right.
- For further analysis, we restrict the movement of the R-head only to the right side.

(iii) ***Finite control:*** The input to the finite control will usually be the symbol under the R-head, say a, and the present state of the machine, say q, to give the following outputs:

(a) A motion of R-head along the tape to the next square (in some a null move, i.e. the R-head remaining to the same square is permitted)

(b) The next state of the finite state machine given by $\delta(q, a)$.

## 1.7Acceptance of String by a Finite Automaton:

The FA accepts a string x if the sequence of transitions corresponding to the symbols of x leads from the start state to an accepting state and the entire string has to be consumed, i.e., a string x is accepted by a finite automaton $M = \mathbf{(Q, \Sigma, \delta, q_0, F)}$

$$\boxed{\text{if } \boldsymbol{\delta}(q_0, x) = q \text{ for some } q \in F.}$$

This is basically the acceptability of a string by the final state.

***Note: A final state is also called an accepting state.***

Transition function δ and for any two input strings x and y,

$$\boxed{\boldsymbol{\delta(q, xy) = \delta(\delta(q, x), y)}}$$

***Example:***

Consider the finite state machine whose transition function δ is given in the form of a transition table. Here $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma=\{0,1\}$, $F=\{q_0\}$.Give the entire sequence of states for the input string 110101.

**Transition Table**

| State | Input | |
|---|---|---|
| | 0 | 1 |
| → $q_0$ | $q_2$ | $q_1$ |
| $q_1$ | $q_3$ | $q_0$ |
| $q_2$ | $q_0$ | $q_3$ |
| $q_3$ | $q_1$ | $q_2$ |

**$\delta(q_0, 110101) = \delta(q_1,10101)$**

**$= \delta(q_0,0101)$**

**$= \delta(q_2,101)$**

**$= \delta(q_3,01)$**

**$= \delta(q_2,1)$**

**$=q0$**

**$q_0$ is final state, therefore given string is accepted by finite automata.**

## 1.8 Deterministic finite automaton:

Formally, a deterministic finite automaton can be represented by a 5-tuple

M= **$(Q, \sum, \delta, q_0, F)$.**

where

- Q is a finite set of states.
- $\sum$ is a finite input alphabet.
- **$\delta$** is the transition function mapping Q x $\sum$ to Q i.e., **$\delta$** (q,a) is a state for each state q and input symbol a.
- $q_o \in Q$ is the initial state.
- $F \subseteq Q$ is the set of final states. It is assumed here that there may be more than one final state.

***Steps to design a DFA***

1. Understand the language for which the DFA has to be designed and write the language for the set of strings starting with minimum string that are accepted by FA.
2. Draw transition diagram for the minimum length string.
3. Obtain the possible transitions to be made for each state on each input symbol.
4. Draw the transition table.

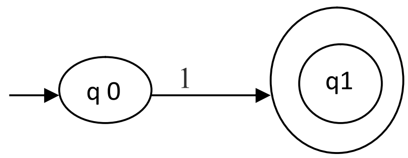
5. Test DFA with few strings that are accepted and few strings that are rejected by the given language.
6. Represent DFA with tuples.

***Examples***

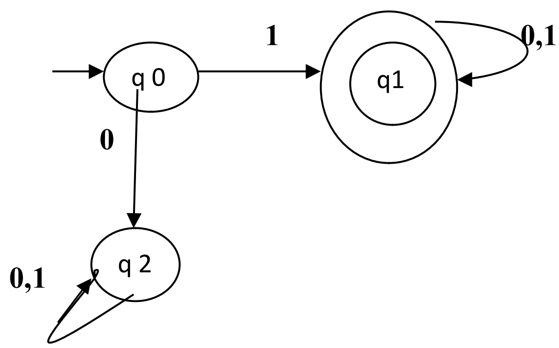
1. **Design DFA that accepts all strings which starts with '1' over the alphabet {0,1}**

   **Step 1:** Understand the language for which the DFA has to be designed and write the language for the set of strings starting with minimum string that is accepted by FA.
   L = {1, 10, 11, 100, 110, 101, 111, .........................................}

   **Step 2:** Draw transition diagram for the minimum length string.




   **Step 3:** Obtain the possible transitions to be made for each state on each input symbol.




   **Step 4:** Draw the transition table.

| State | Input | |
|---|---|---|
| | **0** | **1** |
| $q_0$ | $q_2$ | $q_1$ |
| $q_1$ | $q_1$ | $q_1$ |
| $q_2$ | $q_2$ | $q_2$ |

   **Step 5:** Test DFA with few strings that are accepted and few strings that are rejected by the given language.
   ***Case i)*** Let w=1001 ∈ L
   $\delta(q_0,1001) = \delta(q1,010)$
   $= \delta(q_1,10) = \delta(q_1,0) = q_1$
   $q_1$ is final state and the entire string has been consumed i.e., given string is accepted by DFA.

***Case ii)*** Let w=0001 $\notin$ L

$\delta(q0,0001) = \delta(q2,001)$
$= \delta(q2,10)$
$= \delta(q2,0)$
$= q_2$

$q_2$ is not final state and the entire string has been consumed i.e., given string is rejected by DFA.

**Step 6:** Represent DFA with tuples.

DFA, M= **(Q, $\sum$, δ, qo, F)**

where Q = $\{q_0, q_1, q_2\}$

$\sum = \{ 0,1 \}$

**δ:** $\delta(q_0,0)=q_2$

$\delta(q_0,1)=q_1$

$\delta(q_1,0)=q_1$

$\delta(q_1,1)=q_1$

$\delta(q_2,0)=q_2$

$\delta(q_2,1)=q_2$

$q_0$ – initial state

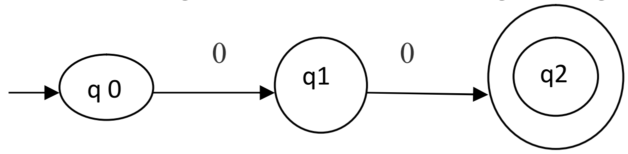
F – final state = $\{ q_1\}$

**2. Design DFA that accepts all strings which contains '00' as substring over the alphabet {0,1}**

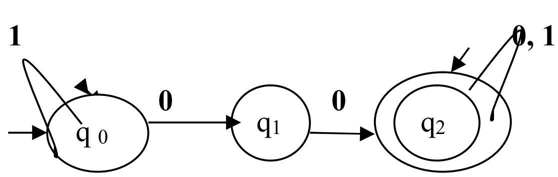
**Step 1:** Understand the language for which the DFA has to be designed and write the language for the set of strings starting with minimum string that is accepted by FA.

L={00,100,000,001,1100,1000,0100,1001,0001,11000,11100,................}

**Step 2:** Draw transition diagram for the minimum length string.




**Step 3 :** Obtain the possible transitions to be made for each state on each input symbol.



1

**Step 4:** Draw the transition table.

| State | Input | |
|---|---|---|
| | **0** | **1** |
| $\rightarrow q_0$ | $q_1$ | $q_0$ |
| q1 | $q_2$ | $q_0$ |
| $q_2$ | $q_2$ | $q_2$ |

**Step 5:** Test DFA with few strings that are accepted and few strings that are rejected by the given language.

***Case i)*** Let w = 1001 $\in$ L

$$\begin{aligned}\delta(q_0,1001) &= \delta(q_0,001)\\ &= \delta(q_1,01)\\ &= \delta(q_2,1)\\ &= q_2\end{aligned}$$

It is final state and the entire string has been consumed i.e., given string is accepted by DFA.

***Case ii)*** Let w=1011 $\notin$ L

$$\begin{aligned}\delta(q_0,1011) &= \delta(q_0,011)\\ &= \delta(q_1,11)\\ &= \delta(q_0,1)\\ &= q_0\end{aligned}$$

It is not final state and the entire string has been consumed i.e., given string is rejected by DFA.

**Step 6:** Represent DFA with tuples.

DFA, M= **(Q, ∑, δ, qo, F)**

where Q = $\{q_0, q_1, q_2\}$

$\sum = \{ 0,1 \}$

**δ:** $\delta(q_0,0)=q_1$

$\delta(q_0,1)=q_0$

$\delta(q_1,0)=q_2$

$\delta(q_1,1)=q_0$

$\delta(q_2,0)=q_2$

$\delta(q_2,1)=q_2$

$q_0$ – initial state

F – final state = { $q_2$ }

## <u>1.9 Nondeterministic finite automaton (NDFA/NFA):</u>

A nondeterministic finite automaton is a 5-tuple (Q, ∑, δ, qo, F), where

- Q is a finite nonempty set of states;
- ∑ is a finite nonempty set of inputs;
- δ is the transition function mapping from Q x ∑ into $2^Q$ which is the power set of Q, the set of all subsets of Q;
- qo ∈ Q is the initial state; and
- F ⊆ Q is the set of final states

***Steps to design a NFA***

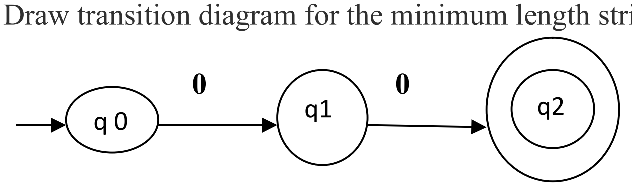
1. Understand the language for which the NFA has to be designed and write the language for the set of strings starting with minimum string that is accepted by FA.
2. Draw transition diagram for the minimum length string.
3. Obtain the possible transitions to be made for each state on each input symbol.
4. Draw the transition table.
5. Test NFA with few strings that are accepted and few strings that are rejected by the given language.
6. Represent NFA with tuples.

***Examples:***

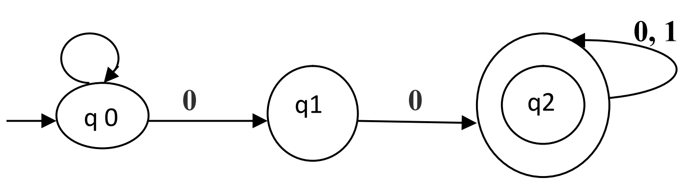
1. **Design NFA that accepts all strings which contains ‘00’ as substring over the alphabet {0,1}**

   **Step 1:** Understand the language for which the NFA has to be designed and write the language for the set of strings starting with minimum string that is accepted by FA
   L={00,100,000,001,0100,1100,1000,1001,0001,11000,11100,............}

   **Step 2:** Draw transition diagram for the minimum length string.




   **Step 3:** Obtain the possible transitions to be made for each state on each input symbol.
   **0, 1**




   **Step 4:** Draw the transition table.

| **State** | **Input** |
|---|---|

| | 0 | 1 |
|---|---|---|
| → $q_0$ | $\{q_0,q_1\}$ | $q_0$ |
| $q1$ | $q_2$ | - |
| $q_2$ | $q_2$ | $q_2$ |

**Step 5:** Test NFA with few strings that are accepted and few strings that are rejected by the given language.

***Case i)*** Let w=0100 ∈ L

$\delta(q_0,0100) = \delta(\{q_0,q_1\},100)$
$= \delta(q_0, 00)$
$= \delta(\{q_0, q1\},0)$
$= \{q_0, q1, q2\}$

$q_2$ is final state and the entire string has been consumed i.e., given string is accepted by NFA.

***Case ii)*** Let w=1011 ∉ L

$\delta(q_0,1011) = \delta(q_0,011)$
$= \delta(\{q_0,q_1\},11)$
$= \delta(q_0,1)$
$= q_0$

It is not final state and the entire string has been consumed i.e., given string is rejected by NFA.

**Step 6:** Represent NFA with tuples.

NFA, M= **(Q, ∑, δ, qo, F)**

where Q = $\{q_0, q_1, q_2\}$

∑ = { 0,1 }

**δ:** $\delta(q_0,0)=\{q_0,q_1\}$

$\delta(q_0,1) = q_0$

$\delta(q_1,0) = q_2$

$\delta(q_1,1) = \varnothing$

$\delta(q_2,0) = q_2$

$\delta(q_2,1) = q_2$

$q_0$ – initial state

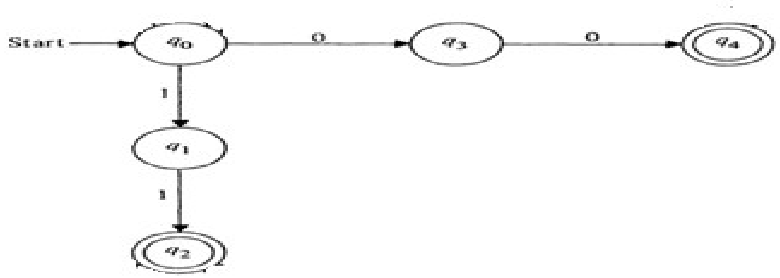
F – final state = { $q_2$ }

2. **Design NFA that accepts strings which contains either two consecutive 0's or two consecutive 1's.**

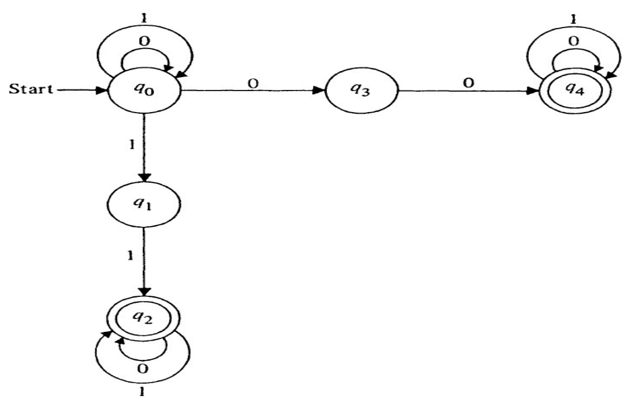
**Step 1:** Understand the language for which the NFA has to be designed and write the language for the set of strings starting with minimum string that is accepted by FA.

L={00,11,100,001,110,011,111,000,0100,1011,.............}

**Step 2:** Draw transition diagram for the minimum length string.




**Step 3:** Obtain the possible transitions to be made for each state on each input symbol.



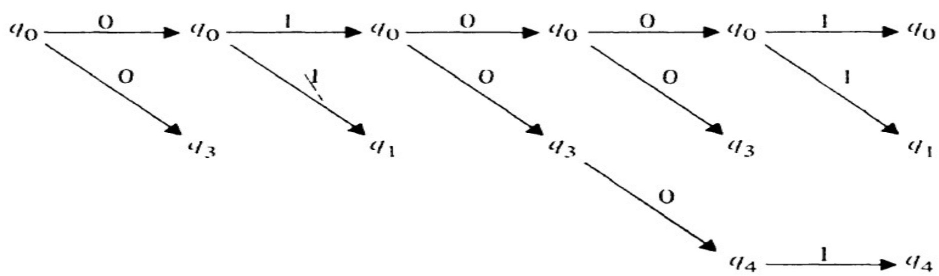

**Step 4:** Draw the transition table.

| State | Input | |
|---|---|---|
| | **0** | **1** |
| → $q_0$ | $\{q_0,q_3\}$ | $\{q_0,q_1\}$ |
| $q_1$ | - | $q_2$ |
| $q_2$ | $q_2$ | $q_2$ |
| $q_3$ | $q_4$ | - |
| $q_4$ | $q_4$ | $q_4$ |

**Step 5:** Test NFA with few strings that are accepted and few strings that are rejected by the given language.
***Case i)*** Let the input, w = 01001 ∈ L

$\delta(q_0,0) = \{q_0,q_3\}$
$\delta(q_0,01) = \delta(\delta(q_0,0),1) = \delta(\{q_0,q_3\},1) = \delta(q_0,1) \cup \delta(q_3,1) = \{q_0,q_1\}$

Similarly, we compute
$\delta(q_0,010) = \{q_0,q_3\}$, $\delta(q_0,0100) = \{q_0,q_3,q_4\}$
and
$\delta(q_0,01001) = \{q_0,q_1,q_4\}$
final state
After the entire string is consumed, the FA is in state $q_4$.As $q_4$ is the final state, the string is a accepted by FA



***Case ii)*** Let w = 010 ∉ L
$\delta(q_0,010) = \delta(\{q_0,q_3\},10)$
$= \delta(\{q_0,q_1\},0)$
$= \{q_0,q_3\}$
There is no path to the final state after the entire string is consumed. So the string is rejected by FA.
**Step 6:** Represent NFA with tuples.

NFA, M= $(\mathbf{Q, \sum, \delta, qo, F})$

where $Q = \{q_0, q_1, q_2, q_3,q_4\}$

$\sum = \{ 0,1 \}$

$\boldsymbol{\delta}$: $\delta(q_0,0)=\{q_0,q_3\}$

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

$\delta(q_1,0)= \emptyset$

$\delta(q_1,1)=q_2$

$\delta(q_2,0)=q_2$

$\delta(q_2,1)=q_2$

$\delta(q_3,0)=q_4$

$\delta(q_3,1)= \emptyset$

$\delta(q_4,0)=q_4$

$\delta(q_4,1)=q_4$

$q_0$ – initial state

F – final state = { $q_2$,$q_4$ }

**1.10 Language recognizers:**

A language recognizer is a device that accepts valid strings produced in a given language. Finite state automata are formalized types of language recognizers.

The language accepted by Finite Automata M designated L(M) is the set $\{x \mid \delta(q0,x) \text{ is in } F\}$.

**1.11 Applications of FA:**

- Used in Lexical analysis phase of a compiler to recognize tokens.
- Used in text editors for string matching.
- Software for designing and checking the behavior of digital circuits.

**1.12 Limitations of FA:**

- FA's will have finite amount of memory.
- The class of languages recognized by FA s is strictly the regular set. There are certain languages which are non regular i.e. cannot be recognized by any FA.

**1.13 Differences between NFA and DFA:**

| S.No | NFA | DFA |
|---|---|---|
| 1 | A nondeterministic finite automaton is a 5-tuple **M= (Q, ∑,** δ, **$q_o$, F)**, where **δ**: Q x ∑ into $2^Q$. | A deterministic finite automaton can be represented by a 5-tuple **M= (Q, ∑, δ, $q_o$, F),** where **δ**: Q x ∑ to Q. |
| 2 | NFA is the one in which there exists many paths for a specific input from current state to next state. | DFA is a FA in which there is only one path for a specific input from current state to next state. |
| 3 | NFA is easier to construct. | DFA is more difficult to construct. |
| 4 | NFA requires less space. | DFA requires more space. |
| 5 | Time required for executing an input string is more. | Time required for executing an input string is less. |

**1.14 NFA with ε transitions:**

An ε -NFA is a tuple (Q, Σ, δ, qo, F)

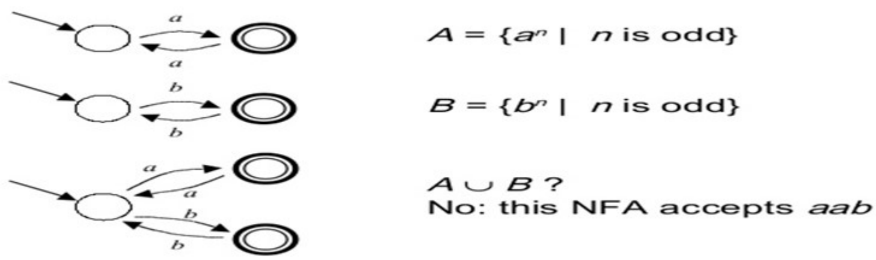
where

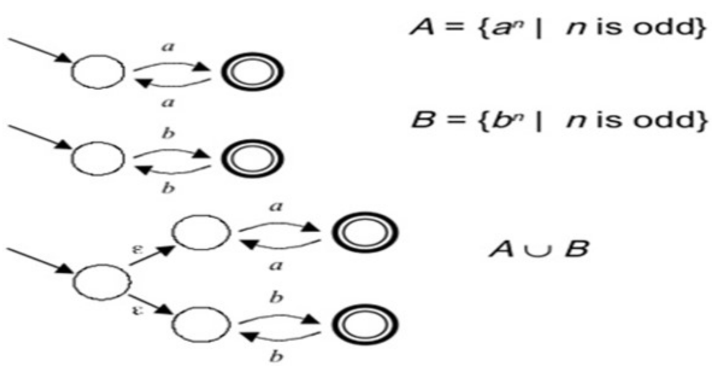
- Q is a set of states,
- Σ is the alphabet,
- δ is the transition function that maps each pair consisting of a state and a symbol in Σ ∪ { ε } to a subset of Q,
- $q_0$ is the initial state,
- F ⊂ Q is the set of final (or accepting) states.

**1.15 Significance of ε-NFA:**

It becomes very difficult or many times it seems to be impossible to draw directly NFA or DFA.

***Example:***




## 1.16 String acceptance by ε –NFA

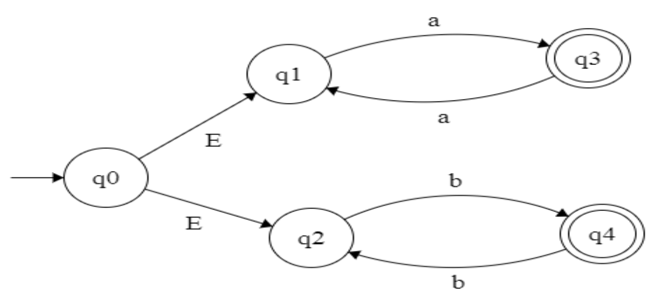



**Fig:1**

## Transition Table:

| Q/∑ | a | b | ε |
|---|---|---|---|
| → $q_0$ | - | - | $\{q_1, q_2\}$ |
| $q_1$ | $q_3$ | - | - |
| $q_2$ | - | $q_4$ | - |
| $q_3$ | $q_1$ | - | - |
| $q_4$ | - | $q_2$ | - |

**<u>Example:</u>**

**Check whether the string 'bbb' is accepted or not for the above automaton.**

$$q_0 \xrightarrow{\varepsilon} q_2 \xrightarrow{b} q_4 \xrightarrow{b} q_2 \xrightarrow{b} q_4$$
$$q_0 \xrightarrow{\varepsilon} q_1$$

As q4 is the final state, the given string is accepted by the given ε –NFA.

## 1.17 ε –NFA to NFA Conversion:

***Step 1***: Find the ε-closure for all states in the given ε-NFA.

$$\hat{\delta}(q, \epsilon) = \epsilon\text{-CLOSURE}(q)$$

ε-closure (q) denotes the set of all states p such that there is a path from q to p labelled ε.

***Step 2***: Find the extended transition function for all states on all input symbols for the given ε-NFA.

$$\boxed{\delta'(q,a) = \varepsilon\text{-closure}(\delta(\delta'(q, \varepsilon),a))}$$

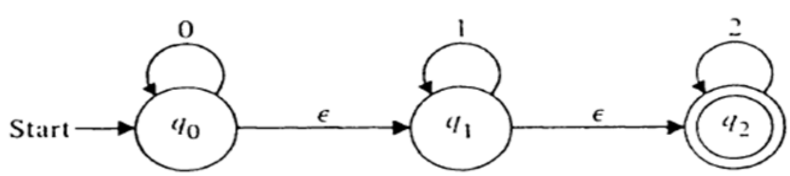
***Step 3***: Draw the transition table or diagram from the extended transition function (NFA)

***Step 4***: F is the set of final states of NFA, whose ε -closure contains the final state of ε -NFA.

***Step 5:*** To check the equivalence of ε -NFA and NFA, the string accepted by **ε** -NFA should be accepted by NFA.

***Example:***

**1. Convert NFA with ε-moves into an equivalent NFA without ε-moves.**




Finite automaton with ε-moves.

***Step 1***: Find the ε-closure for all states in the given ε-NFA.

ε -CLOSURE ($q_0$) = {$q_0$, $q_1$, $q_2$}
ε -CLOSURE ($q_1$) = {q1, $q_2$}
ε -CLOSURE ($q_2$) = {q2}

***Step 2:*** Find the extended transition function for all states on all input symbols for the given ε-NFA.

$\delta' (q_0,0) = \varepsilon\text{-closure}(\delta (\delta'(q_0, \varepsilon),0))$
$= \varepsilon\text{-closure}(\delta \{q_0, q_1, q_2\},0)$
$= \varepsilon\text{-closure}(\delta(q_0, 0) \cup \delta (q_1,0) \cup \delta (q_2,0))$
$= \varepsilon\text{-closure}(q_0 \cup \emptyset \cup \emptyset)$
$= \{q_0, q_1, q_2\}$

$\delta' (q_0,1) = \varepsilon\text{-closure}(\delta (\delta'(q_0, \varepsilon),1))$
$= \varepsilon\text{-closure}(\delta \{q_0,q_1, q_2\},1)$
$= \varepsilon\text{-closure}(\delta (q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1))$
$= \varepsilon\text{-closure}(\emptyset \cup q_1 \cup \emptyset)$
$=\{q_1,q_2\}$

$\delta' (q_0,2) = \varepsilon\text{-closure}(\delta (\delta'(q_0, \varepsilon),2))$
$= \varepsilon\text{-closure}(\delta \{ q_0,q_1, q_2\},2)$
$= \varepsilon\text{-closure}(\delta (q_0,2) \cup \delta (q_1,2) \cup \delta (q_2,2))$
$= \varepsilon\text{-closure}(q_2 \cup \emptyset)$
$=\{q_2\}$

$\delta' (q_1,0) = \varepsilon\text{-closure}(\delta (\delta'(q_1, \varepsilon),0))$
$= \varepsilon\text{-closure}(\delta \{q_1, q_2\},0)$
$= \varepsilon\text{-closure}(\delta (q_1,0) \cup \delta (q_2,0))$
$= \varepsilon\text{-closure}(\emptyset)$
$=\{ \emptyset \}$

$\delta' (q_1,1) = \varepsilon\text{-closure}(\delta (\delta'(q_1, \varepsilon),1))$
$= \varepsilon\text{-closure}(\delta \{q_1, q_2\},1)$
$= \varepsilon\text{-closure}(\delta (q_1,1) \cup \delta (q_2,1))$
$= \varepsilon\text{-closure}(q_1)$
$=\{q_1, q_2 \}$

$\delta (q_1,2) = \varepsilon\text{-closure}(\delta (\delta'(q1, \varepsilon),2))$
$= \varepsilon\text{-closure}(\delta \{q1, q2\},2)$
$= \varepsilon\text{-closure}(\delta (q1,2) \cup \delta (q2,2))$
$= \varepsilon\text{-closure}(q2)$
$=\{q2\}$

$\delta (q_2,0) = \varepsilon\text{-closure}(\delta (\delta'(q_2, \varepsilon),0))$
$= \varepsilon\text{-closure}(\delta (q_2,2))$
$= \varepsilon\text{-closure}(\emptyset)$
$=\{ \emptyset\}$

$\delta (q_2,1) = \varepsilon\text{-closure}(\delta (\delta'(q_2, \varepsilon),1))$
$= \varepsilon\text{-closure}(\delta (q_2,1))$
$= \varepsilon\text{-closure}(\emptyset)$
$=\{ \emptyset \}$

$\delta (q_2,2) = \varepsilon\text{-closure}(\delta (\delta'(q_2, \varepsilon),2))$
$= \varepsilon\text{-closure}(\delta (q_2,2))$
$= \varepsilon\text{-closure}(q_2)$
$=\{ q_2\}$

***Step 3***: Draw the transition table or diagram from the extended transition function (NFA)

| State | Inputs | | |
|---|---|---|---|
| | **0** | **1** | **2** |
| → $q_0$ | $\{q_0, q_1, q_2\}$ | $\{q_1, q_2\}$ | $q_2$ |
| $q_1$ | Ø | $\{q_1, q_2\}$ | $q_2$ |
| $*q_2$ | Ø | Ø | $q_2$ |

***Step 4***: F is the set of final states of NFA, whose ε -closure contains the final state of ε - NFA.

| State | Inputs | | |
|---|---|---|---|
| | **0** | **1** | **2** |
| → $q_0$ | $\{q_0, q_1, q_2\}$ | $\{q_1, q_2\}$ | $q_2$ |
| q1 | Ø | $\{q_1, q_2\}$ | $q_2$ |
| q2 | Ø | Ø | $q_2$ |

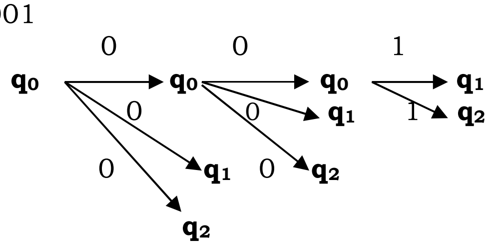
***Step 5:*** To check the equivalence of ε -NFA and NFA, the string accepted by **ε** -NFA should be accepted by NFA.

***String acceptance by ε-NFA:***
Let w=001

$$\mathbf{q_0} \xrightarrow{0} \mathbf{q_0} \xrightarrow{0} \mathbf{q_0} \xrightarrow{\varepsilon} \mathbf{q_1} \xrightarrow{1} \mathbf{q_1} \xrightarrow{\varepsilon} \mathbf{q_2}$$

As q2 is the final state, the string is accepted by the given ε-NFA.

***String acceptance by NFA:***
If w=001




As q1 and q2 are final states, the string is accepted by the NFA.

## 1.18 NFA to DFA Conversion:

***Step 1:*** First take the starting state of NFA as the starting state of DFA.

***Step 2:*** Apply the inputs on initial state and represent the corresponding states in the transition table.

***Step 3:*** For each newly generated state, apply the inputs and represent the corresponding states in the transition table.

***Step 4:*** Repeat step 3 until no more new states are generated.

***Step 5:*** The states which contain any of the final states of the NFA are the final states of the equivalent DFA.
***Step 6:*** Represent the transition diagram from the constructed table.
***Step7:*** To check the equivalence of NFA and DFA, the string accepted by NFA should be accepted by DFA.
***Step 8:*** Write the tuple representation for the obtained DFA.

**Note:** If the NFA has $n$ states, the resulting DFA may have up to $2^n$ states, an exponentially larger number, which sometimes makes the construction impractical for large NFAs.

***Example:***

**1.** Construct DFA equivalent to the NFA $M=(\{q_0,q_1\},\{0,1\}, \delta,q_0,\{q_1\})$
where $\delta(q_0,0) = \{q_0,q_1\}$ $\delta(q_0,1) = \{q_1\}$ $\delta(q_1,0) = \emptyset$ $\delta(q_1,1) = \{q_0,q_1\}$

***Step 1:*** First take the starting state of NFA as the starting state of DFA

| Q/∑ | 0 | 1 |
|---|---|---|
| → $[q_0]$ | | |

***Step 2:*** Apply the inputs on initial state and represent the corresponding states in the transition table.

| Q/∑ | 0 | 1 |
|---|---|---|
| → $[q_0]$ | $[q_0,q_1]$ | $[q_1]$ |

***Step 3:*** For each newly generated state, apply the inputs and represent the corresponding states in the transition table.

| Q/∑ | 0 | 1 |
|---|---|---|
| → $[q_0]$ | $[q_0,q_1]$ | $[q_1]$ |
| $[q_0,q_1]$ | $[q_0,q_1]$ | $[q_0,q_1]$ |
| $[q_1]$ | $\emptyset$ | $[q_0,q_1]$ |

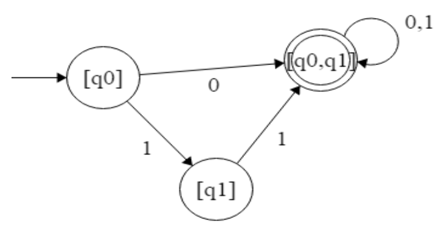
***Step 4:*** Stop the procedure as there are no more new states being generated.

***Step 5:*** The states which contain any of the final states of the NFA are the final states of the equivalent DFA.

$q_1$ is the final state in NFA. $q_1$ is included in the state $[q_0,q_1]$ and $[q_1]$. So $[q_0,q_1]$ and $[q_1]$ are the final states of the DFA.

| **Q/∑** | **0** | **1** |
|---|---|---|
| → $[q_0]$ | $[q_0,q_1]$ | $[q_1]$ |
| $[q_0,q_1]$ | $[q_0,q_1]$ | $[q_0,q_1]$ |
| $[q_1]$ | Ø | $[q_0,q_1]$ |

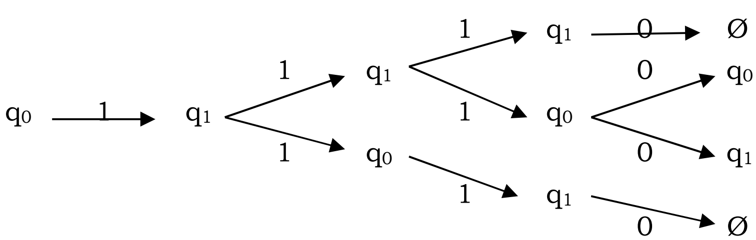
***Step 6:*** Represent the transition diagram from the constructed table.




***Step 7:*** To check the equivalence of NFA and DFA, the string accepted by NFA should be accepted by DFA.

Let **w=1110** be the string accepted by NFA.

***Acceptability by NFA:***




***Acceptability by DFA:***

$\delta([q_0],1110) = \delta([q_1],110)$ $\qquad [q0] \xrightarrow{1} [q1] \xrightarrow{1} [q0,q1] \xrightarrow{1} [q0,q1] \xrightarrow{0} [q0,q1]$

$= \delta([q_0,q_1],10)$

$= \delta([q_0,q_1],0)$

$= [q_0,q_1] \in F$

***Step 8:*** Write the tuple representation from the obtained DFA.

DFA M' = $(Q,\sum, \delta,q_0,F)$

where Q = $\{[q_0], [q_0,q_1], [q_1]\}$

$\Sigma = \{0, 1\}$

$\delta$ - transition function

$[q_0]$ - initial state
$F = \{[q_0], [q_0,q_1]\}$

## 1.19 Minimization of Finite Automata:

Two states ql and q2 are equivalent (denoted by $q1 \equiv q2$) if both $\delta(q1, x)$ and $\delta(q2, x)$ are final states. or both of them are nonfinal states for all $x \in \Sigma^*$.

Two states q1 and $q2$ are k-equivalent ($k \geq 0$) if both $\delta(q1, x)$ and $\delta(q2, x)$ are final states or both nonfinal states for all strings $x$ of length $k$ or less. In particular, any two final states are 0-equivalent and any two nonfinal states are also 0-equivalent.

***Construction of Minimum Automaton:***

***Step 1:*** (Construction of $\pi_0$)· By definition of 0-equivalence, $\pi_0 = \{Q_1^0, Q_2^0\}$ where $Q_1^0$ is the set of all final states and $Q_2^0 = Q - Q_1^0$.

***Step 2:*** (Construction of $\pi_{k+1}$ from $\pi_k$).

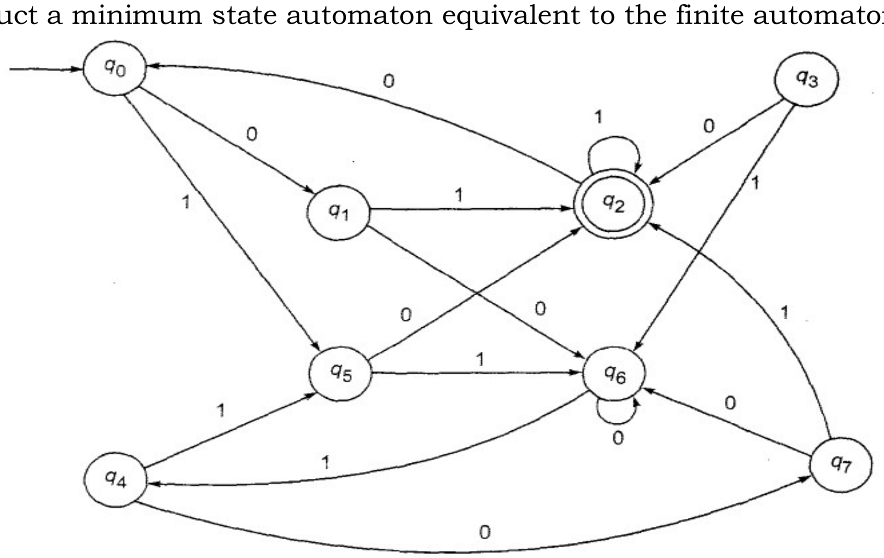
- Let $Q_i^k$ be any subset in $\pi_k$ . If $q_1$ and $q_2$ are in $Q_i^k$ , they are (k + 1)-equivalent provided $\delta$ (q1,a) and $\delta$(q2,a) are k-equivalent.
- Find out whether $\delta$ (q1, a) and $\delta$ (q2, a) are in the same equivalence class in $\pi_k$ for every $a \in \Sigma$. If so q1 and q2 are (k + 1)-equivalent.
- In this way, $Q_i^k$ is further divided into (k + 1)-equivalence classes. Repeat this for every $Q_i^k$ in $\pi_k$ to get all the elements of $\pi_{k+1}$.

***Step 3:*** Construct $\pi_n$ for n = 1, 2, .... until $\pi_n = \pi_{n+1}$.

***Step 4:*** (Construction of minimum automaton). For the required minimum state automaton, the states are the equivalence classes obtained in step 3. i.e. the elements of $\pi_n$ The state table is obtained by replacing a state q by the corresponding equivalence class [q].

***Example:***

Construct a minimum state automaton equivalent to the finite automaton.



***Solution:***
It will be easier if we construct the transition table.

| State/Σ | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_1$ | $q_5$ |
| $q_1$ | $q_6$ | $q_2$ |
| $q_2$ | $q_0$ | $q_2$ |
| $q_3$ | $q_2$ | $q_6$ |
| $q_4$ | $q_7$ | $q_5$ |
| $q_5$ | $q_2$ | $q_6$ |
| $q_6$ | $q_6$ | $q_4$ |
| $q_7$ | $q_6$ | $q_2$ |

***Step 1:*** Construction of $\pi_0$

$\pi_0 = \{Q_1^0, Q_2^0\}$

where $Q_1^0 = F = \{q2\}$ $\quad\quad Q_2^0 = Q - Q_1^0$

$\therefore \pi_0 = \{\{q2\}, \{q0,q1,q3,q4,q5,q6,q7\}\}$

***Step 2:*** The {q2} in $\pi_0$ cannot be further partitioned. So, $Q_1^1 = \{q2\}$.
Compare $q_0$ with $q_1$, $q_3$,$q_4$,$q_5$,$q_6$ and $q_7$.

Consider qo and q1 $\epsilon$ $Q_2^0$.

- The entries under the 0- column corresponding to qo and q1 are q1 and q6; they lie in $Q_2^0$.
- The entries under the 1-column are $q_5$ and $q_2$. q2 $\epsilon$ $Q_1^0$ and q5 $\epsilon$ $Q_2^0$. Therefore qo and q1 are not 1- equivalent.

| **Q/∑** | **0** | **1** |
|---|---|---|
| $q_0$ | $q_1$ | $q_5$ |
| $q_1$ | $q_6$ | $q_2$ |

Consider q0 and q3

| **Q/∑** | **0** | **1** |
|---|---|---|
| $q_0$ | $q_1$ | $q_5$ |
| $q_3$ | $q_2$ | $q_6$ |

The entries under the 0- column corresponding to qo and q3 are q1 and q2; q1 $\epsilon$ $Q_2^0$ and q2 $\epsilon$ $Q_1^0$. The entries under the 1-column are q5 and q6; they lie in $Q_2^0$. Therefore qo and q3 are not 1- equivalent

Similarly, qo is not 1-equivalent to q5 and q7.

Consider q0 and q4

| Q/∑ | 0 | 1 |
|---|---|---|
| **q0** | **q1** | **q5** |
| **q4** | **q7** | **q5** |

- The entries under the 0- column corresponding to qo and q4 are q1 and q7; they lie in $Q_2^0$.
- The entries under the 1-column are q5 and q5; they lie in $Q_2^0$. Therefore qo and q1 are 1- equivalent.

  Similarly, qo is 1-equivalent to q6.

  {qo. q4, q6} is a subset in $\pi_1$.
  So, $Q_2^1$ = {q0,q4,q6}

- Repeat the construction by considering q1 and anyone of the state's q3, q5, q7. Now, q1 is not 1-equivalent to q3 or q5 but 1-equivalent to q7.
  Hence, $Q_3^1$ = {q1,q7}.
- The elements left over in $Q_2^0$are q3 and q5. By considering the entries under the 0-column and the 1-column, we see that q3 and q5 are 1-equivalent.
  So $Q_4^1$ = {q3, q5}.
  Therefore, $\pi_1$ = {{q2}. {qo, q4, q6}. {q1, q7}, {q3, q5}}

***Step 3:*** Construct $\pi_n$ for n = 1, 2, .... until $\pi_n = \pi_{n+1}$.
Calculate 2-equivalent, $\pi_2$.

$\pi_2$ = {{q2}, {qo,q4}, {q6}, {q1,q7}, {q3,q5}}

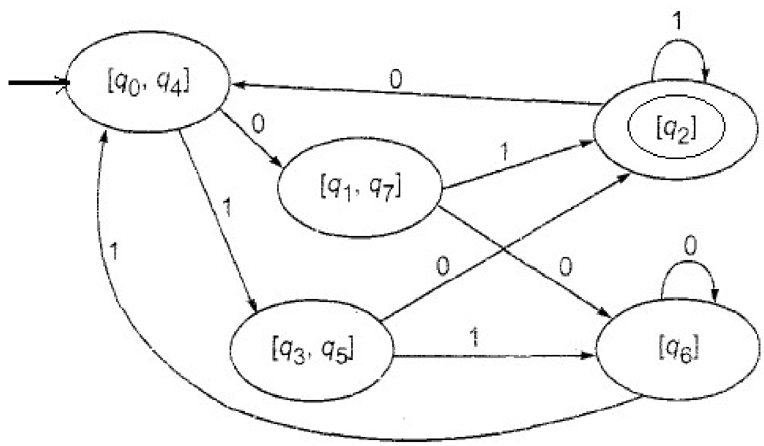
Similarly calculate 3-equivalent, $\pi_3$.

$\pi_3$ = {{q2}, {qo,q4}, {q6}, {q1,q7},{q3,q5}}

As $\pi_2 = \pi_3$, $\pi_2$ gives us the equivalence classes.

***Step 4:*** Construction of minimum automaton.
M' = (Q',{0,1},δ',$q_0$',F')
where Q' = {[$q_2$]. [$q_0$, $q_4$], [$q_6$], [$q_1$, $q_7$], [$q_3$, $q_5$]}
$q_0$' = [q0, q4]
F' = [q2]
δ' is given by

| *State/Σ* | 0 | 1 |
|---|---|---|
| [$q_0$, $q_4$] | [$q_1$, $q_7$] | [$q_3$, $q_5$] |
| [$q_1$, $q_7$] | [$q_6$] | [$q_2$] |
| [$q_2$] | [$q_0$, $q_4$] | [$q_2$] |
| [$q_3$, $q_5$] | [$q_2$] | [$q_6$] |
| [$q_6$] | [$q_6$] | [$q_0$, $q_4$] |




**1.20 Equivalence between two FSM's:**

Let M and M' be two FSM's over ∑ .We construct a comparison table consisting of n+1 columns where n is the number of input symbols.

***Step 1:*** 1st column consisting of a pair of states of form (q, q') where q belongs to M and q' belongs M'.

***Step 2:*** If (q, q') appears in the same row of 1st column then the corresponding entry in a column (a belongs to ∑**)** is (r,r') where (r,r') are pair from q and q' on a.

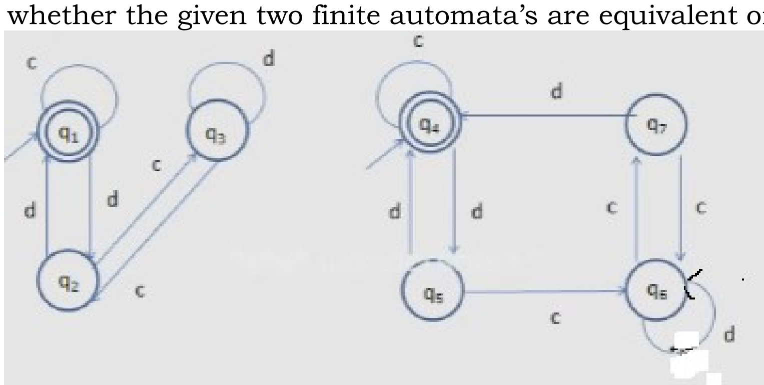
***Step 3:*** A table is constructed by starting with a pair of initial states $q_0$, $q_0$' of M and M'. We complete construction by considering the pairs in 2nd and subsequent columns which are not in the 1st column.

(i) if we reach a pair (q,q') such that q is final states of M and q' is non-final state of M' i.e. terminate contruction and conclude that M and M' are not equivalent.

(ii) if construction is terminated when no new element appears in 2nd and subsequent columns which are not in 1st column. Conclude that M and M' are equivalent.

***Example:***

Check whether the given two finite automata's are equivalent or not.




**Solution:**

$q_1$ is initial state of M1 and $q_4$ is initial state of M2 ,make them a pair and place it in 1st row of the transition table.

**Comparison table**

| **Q/∑** | **c** | **d** |
|---|---|---|
| **($q_1$,$q_4$)** | **($q_1$,$q_4$)** | **($q_2$,$q_5$)** |
| **($q_2$,$q_5$)** | **($q_3$,$q_4$)** | |

Here q3 is non-final state and q4 is final state.

Therefore, we stop constructing comparison table and conclude that the two given Finite Automata's are not equivalent.

## 1.21 Moore Machine

**A Moore machine is a six tuple (Q, ∑, Δ, δ, $q_0$, λ)**

where

- Q is a set of states,
- Σ is the alphabet,
- δ is the transition function that maps each pair consisting of a state and a symbol in Σ to Q i.e. .Q X Σ -> Q
- q0 is the initial state,
- Δ is output alphabet
- λ is a mapping from Q to Δ giving the output associated with each state

**Note:** For a Moore machine if the input string is of length n, the output string is of length n + 1. The first output is λ (qo) for all output strings.

## 1.22 Mealy Machine

**A Mealy machine is a six tuple (Q, ∑, Δ, δ, $q_0$, λ)**

where

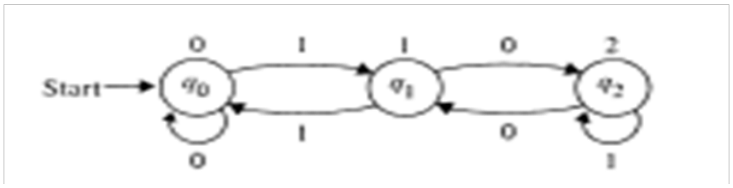
- Q is a set of states,
- Σ is the alphabet,
- δ is the transition function that maps each pair consisting of a state and a symbol in Σ to Q i.e. .Q X Σ -> Q
- Δ is output alphabet
- q0 is the initial state,
- **λ** maps Q x ∑ **to** Δ **i.e.,** λ(q,a) gives the output associated with the transition from state q on input a

**Note:** In the case of a Mealy machine if the input string is of length n , the output string is also of the same length n.

***Example:***

- The given transition diagram is moore machine because each state is associated with output.

- In the below diagram $q_0$ is representing 0 output, $q_1$ is is representing 1 output and $q_2$ is representing 2 output.
  $\lambda(q_0) = 0$   $\lambda(q_1) = 1$   $\lambda(q_2) = 2$




**w=011** the output is **0010**

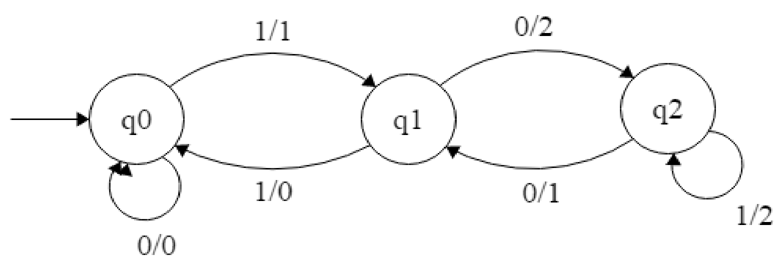
$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0$ **Transitions**

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

**0   0   1   0   Outputs**

***Example:***

- The given transition diagram is mealy machine because output depends on present state and present input.
- In the below diagram
  $\lambda(q_0,0) = 0$   $\lambda(q_1,0) = 2$   $\lambda(q_2,0) = 0$
  $\lambda(q_0,1) = 1$   $\lambda(q_1,1) = 0$   $\lambda(q_2,1) = 2$




**w=011** the output is **010**

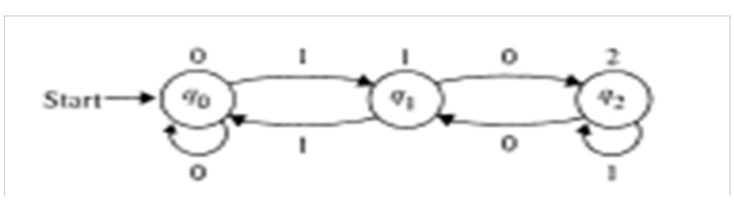
$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0$ **Transitions**

$\downarrow \quad \downarrow \quad \downarrow$

**0   1   0   Outputs**

***Example:***

**1. Design Moore machine to determine the residue mod 3 for each binary string treated as a binary integer.**



## Moore machine calculating residue mod 3

**Moore Table**

| Present State | Next State | | Output |
|---|---|---|---|
| | 0 | 1 | |
| → $q_0$ | $q_0$ | $q_1$ | 0 |
| $q_1$ | $q_2$ | $q_0$ | 1 |
| $q_2$ | $q_1$ | $q_2$ | 2 |

**Tuple Representation:**

**Q**$=\{q_0,q_1,q_2\}$　　$\Delta=\{0,1,2\}$　$\Sigma=\{0,1\}$

$\mathbf{q_0}=\{q_0\}$

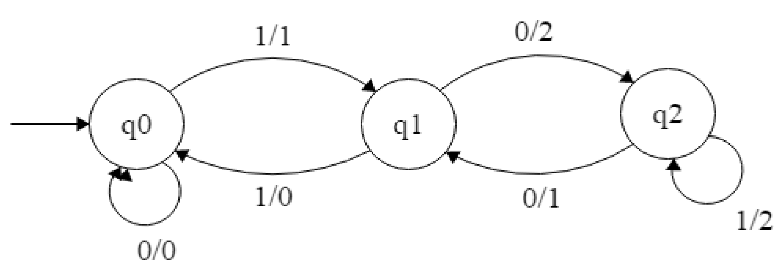
**λ :** $\lambda(q_0)=0$　　**δ:** $\delta(q_0,0) = q_0$　$\delta(q_0,1) = q_1$

$\lambda(q_1)=1$　　$\delta(q_1,0) = q_2$　$\delta(q_1,1) = q_0$

$\lambda(q_2)=2$　　$\delta(q_2,0) = q_1$　$\delta(q_2,1) = q_2$

***Example:***

**1. Design Mealy machine to determine the residue mod 3 for each binary string treated as a binary integer.**




**Mealy Table:**

| Present State | Next State | | Next State | |
|---|---|---|---|---|
| | 0 | Output | 1 | Output |
| → $q_0$ | $q_0$ | 0 | $q_1$ | 1 |
| $q_1$ | $q_2$ | 2 | $q_0$ | 0 |

| | | | | |
|---|---|---|---|---|
| $q_2$ | $q_1$ | 1 | $q_2$ | 2 |

**Tuple Representation:**

**Q**$=\{q_0,q_1,q_2\}$ $\Delta=\{0,1,2\}$ $\Sigma=\{0,1\}$

$\mathbf{q_0}=\{q_0\}$

**λ:** $\lambda(q_0,0)=0$

$\lambda(q0,1)=1$

$\lambda(q_1,0)=2$

$\lambda(q_1,1)=0$

$\lambda(q_2,0)=1$

$\lambda(q_2,1)=2$

**δ:** $\delta(q_0,0) = q_0$ $\delta(q_0,1) = q_1$

$\delta(q_1,0) = q_2$ $\delta(q_1,1) = q_0$

$\delta(q_2,0) = q_1$ $\delta(q_2,1) = q_2$

## 1.23 Moore to Mealy Conversion:

**If $M_1= (Q,\Sigma,\Delta,\delta,q_0, \lambda)$ is a Moore machine, then there is a Mealy machine $M_2$ equivalent to $M_1$.**

***Procedure:***

- Let M2 = $(Q,\Sigma,\Delta,\delta,q_0, \lambda')$ and define $\lambda'(q, a)$ to be $\lambda(\delta(q, a))$ for all states q and input symbols a.
- Then $M_1$ and $M_2$ enter the same sequence of states on the same input, and with each transition $M_2$ emits the output that $M_1$ associates with the state entered.

***Example:***

**Construct a Mealy Machine which is equivalent to the Moore machine given by table below.**

| Present State | Next State | | Output |
|---|---|---|---|
| | 0 | 1 | |
| $q_0$ | $q_3$ | $q_1$ | 0 |
| $q_1$ | $q_1$ | $q_2$ | 1 |
| $q_2$ | $q_2$ | $q_3$ | 0 |
| $q_3$ | $q_3$ | $q_0$ | 0 |

**Solution:**

$\lambda'(q, a)$ to be $\lambda(\delta(q, a))$

$\lambda'(q_0,0) =\lambda(\delta(q_0, 0))$

$=\lambda(q_3)$

$=0$

$\lambda'(q_0,1) =\lambda(\delta(q_0, 1))$

$=\lambda(q_1)$

$=1$

$\lambda' (q_1,0) = \lambda(\delta (q_1, 0))$ $\lambda' (q_1,1) = \lambda(\delta (q_1, 1))$

$= \lambda (q_1)$ $= \lambda (q_2)$

$=1$ $=0$

$\lambda' (q_2,0) = \lambda(\delta (q_2, 0))$ $\lambda' (q_1,1) = \lambda(\delta (q_2, 1))$

$= \lambda (q_2)$ $= \lambda (q_3)$

$=0$ $=0$

$\lambda' (q_3,0) = \lambda(\delta (q_3, 0))$ $\lambda' (q_3,1) = \lambda(\delta (q_3, 1))$

$= \lambda (q_3)$ $= \lambda (q_0)$

$=0$ $=0$

**Mealy Table:**

| Present State | Next State | | Next State | |
|---|---|---|---|---|
| | **0** | **output** | **1** | **Output** |
| $q_0$ | $q_3$ | **0** | $q_1$ | **1** |
| $q_1$ | $q_1$ | **1** | $q_2$ | **0** |
| $q_2$ | $q_2$ | **0** | $q_3$ | **0** |
| $q_3$ | $q_3$ | **0** | $q_0$ | **0** |

## 1.24 Mealy to Moore Conversion:

**If $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a Mealy machine, then there is a Moore machine $M_2$ equivalent to $M_1$.**

***Procedure:***

- Determine the number of different output associated with qi in the next state column.
- We split qi into different states according to different output associated with it
  For example: $q_2$ is associated with two different outputs 0 and 1, so we split $q_2$ into $q_{20}$ and $q_{21}$.

***Example:***

**Construct Moore machine for the given mealy machine.**

| Present State | Next State | | | |
|---|---|---|---|---|
| | a = 0 | | a = 1 | |
| | State | Output | State | Output |
| -> q0 | q3 | 0 | q1 | 1 |
| q1 | q0 | 1 | q3 | 0 |
| q2 | q2 | 1 | q2 | 0 |
| q3 | q1 | 0 | q0 | 1 |

**Solution:**

- We get two states (q1 and q2) that are associated with different outputs (0 and 1). so we split both states into $q_{10}$ , $q_{11}$ and $q_{20}$, $q_{21}$.
- Whole row of $q_1$ is copied to $q_{10}$, $q_{11}$ and whole row of $q_2$ is copied to $q_{20}$ and $q_{21}$ of the sample transition table of mealy machine.
- The outputs of the next state columns of $q_1$ and $q_2$ are depend on the previous output. For ex. in the first row, $q_1$ becomes $q_{11}$ because the out of $q_1$ is 1 in the fourth row, $q_2$ becomes $q_{21}$ because the output of the $q_2$ is 1 and in the subsequent column $q_2$ becomes $q_{20}$ because the output of $q_2$ in that column was 0. and so on

| Present State | Next State | | Output |
|---|---|---|---|
| | a = 0 | a = 1 | |
| -> q0 | q3 | q11 | 1 |
| q10 | q0 | q3 | 0 |
| q11 | q0 | q3 | 1 |
| q20 | q21 | q20 | 0 |
| q21 | q21 | q20 | 1 |
| q3 | q10 | q0 | 0 |

## 1.25 Applications of FA:

- Used in Lexical analysis phase of a compiler to recognize tokens.
- Used in text editors for string matching.
- Software for designing and checking the behavior of digital circuits.