# Part I : Basic Concepts of Finite Automata

## 1.1 Basic Concepts

SPPU : May-09, Marks 4

- Automata theory deals with the definitions and properties of mathematical models of computation. These models play a role in several applied areas of computer science.
- There are two popularly used models in automata theory - **finite automaton** and **context free grammars**.
- The **finite automaton** is used in text processing, compilers and so on.
- The **context free grammars** are used in programming languages, XML and artificial intelligence.
- The automata theory helps us to begin the study of theory of computation.

**Definition of automaton :** An automaton is defined as a system where information is transformed and transmitted and is used for performing some functions without involvement of man.

**Examples :** Automatic printing machines, automatic washing machines.

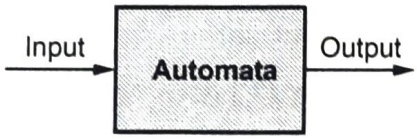



**Fig. 1.1.1 Basic model of automata**

The **characteristics** are -

1. **Input :** There can be finite number of inputs to the automata. These values are from the input set denoted by $\Sigma$.
2. **Output :** There can be finite number of outputs from the automata.
3. **States :** At any instance of time the automata can be in one particular state.
4. **State function :** This function is useful in determining the next state at any instance of time with the help of present state and present input.
5. **Output function :** This function is useful in determining the output at any instance of time with the help of present input and present state.

**Example 1.1.1** *What is the basic machine ? Enlist the important features of basic machine.*

SPPU : May-09, Marks 4

**Solution :**

**Basic machine :** The basic machine is a system which models the given problem and it halts on obtaining solution to the given problem.

**Important features of basic machine**

1. There should be definite number of inputs to the basic machine.
2. The machine must produce proper output.
3. The performance of machine should be optimum.
4. The machine must be simple to design and implement.
5. The machine must halt after sufficient number of steps. That means it should not loop for infinitely long period.

## Part II : Finite Automata

## 1.2 Formal Definition and Notations for FSM

A finite automata is a collection of 5-tuple $(Q, \Sigma, \delta, q_0, F)$

where,

Q is a finite **set of states**, which is non empty.

$\Sigma$ is **input** alphabet, indicates input set.

$q_0$ is an **initial state** and $q_0$ is in Q i.e. $q_0 \in Q$.

F is a set of **final states.**

$\delta$ is a **transition function** or a mapping function. Using this function the next state can be determined.

### 1.2.1 Control of FA over String

The finite automata can be represented as :

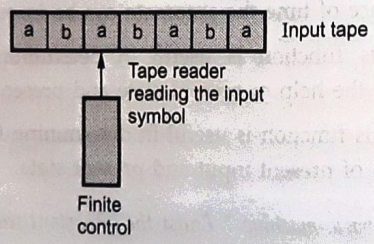



**Fig. 1.2.1 Model for finite automata**





The finite automata can be represented using :

**i) Input tape -** It is a linear tape having some number of cells. Each input symbol is placed in each cell.

**ii) Finite control -** The finite control decides the next state on receiving particular input from input tape. The tape reader reads the cells one by one from left to right and at a time only one input symbol is read.

For example, suppose current state is q, and suppose reader is reading the symbol 'a' then it is finite control which decides what will be the next state at input 'a'. The transition from current state q with input w to next state q′ producing w′ will be represented as,

$$(q, w) \vdash (q', w')$$

If w is a string and M is a finite automata, then w is accepted by the FA.

iff $(w, s) \vdash^* (q, \varepsilon)$

with q as final state.

The set of strings accepted by a FA given by M then M is accepted by language L. The acceptance of M by some language L is denoted by L(M).

A machine M accepts a language L iff,

$$L = L(M).$$

## 1.3 Concept of State Transition Diagram and Transition Table for FA

SPPU : Dec.-18, Marks 2

The strings and languages can be accepted by a finite automata, when it reaches to a final state. There are two preferred notations for describing automata :

**1. Transition diagram -**

A transition diagram or transition graph can be defined as collection of -

1) Finite set of states K.

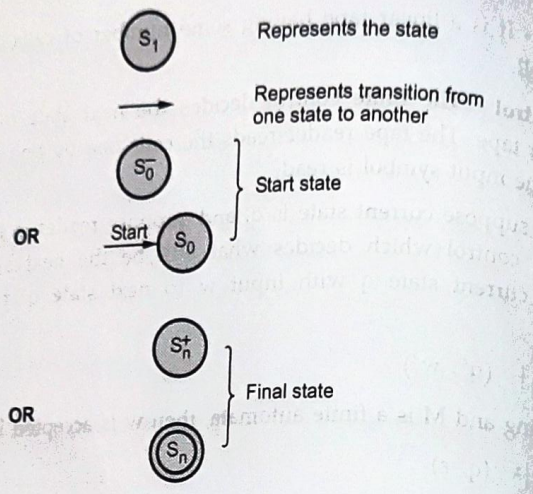
2) Finite set of symbols $\Sigma$.

3) A non empty set S of K. It is called **start state.**

4) A set $F \subseteq K$ of **final states.**

5) A transition function $K \times A \rightarrow K$ with K as state and A as input from $\Sigma^*$.

The notations used in transition diagram are -




**For example :**

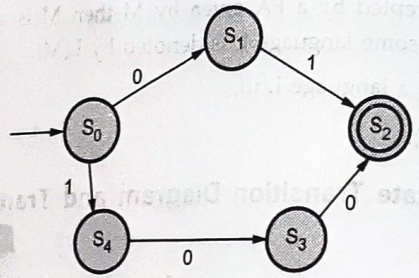
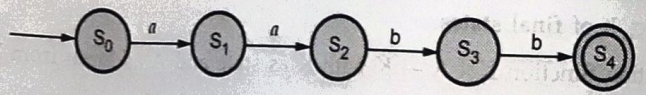



**Fig. 1.3.1 Transition diagram**

The FA can be represented using transition graph. The machine initially is in start state $S_0$ then on receiving input 0 it changes to state $S_1$. From $S_0$ on receiving input 1 the machine changes its state to $S_4$. The state $S_2$ is a final state or accept state. When we trace the input for transition diagram and reach to a final state at end of input string then it is said that the given input is **accepted by** transition diagram.

**Another example :**




We have drawn a transition diagram for the input **aabb**.





Note that the start state is $S_0$ and final state is $S_4$. The input set is $\Sigma = \{a, b\}$. The states $S_1$, $S_2$, $S_3$ are all intermediate states.

### 2. Transition table

This is a tabular representation of finite automata. For transition table the transition function is used.

**For example :**

| States \ Input | a | b |
|---|---|---|
| $q_0$ | $q_1$ | - |
| $q_1$ | - | $q_2$ |
| $q_2$ | $q_2$ | - |

The rows of the table corresponds to states and columns of the table corresponds to inputs.

**Example 1.3.1** *State properties and limitations of FSM.* **SPPU : Dec.-18, Marks 2**

**Solution : Properties :**

1) FSM consists of a) States and b) State transitions.

2) An object remains in one of the state and when certain conditions are met, object changes its state.

**Limitations :**

1) It has no capability to remember already read input.

2) The FSM with many states and transitions can be difficult to manage.

## 1.4 Types of Finite Automata

- There are two types of finite automata -

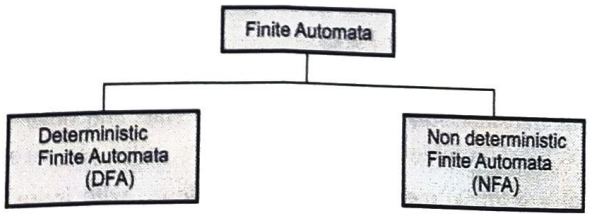



**Fig. 1.4.1 DFA and NFA**

i) Deterministic Finite Automata.

ii) Non deterministic Finite Automata.

The deterministic finite automata is deterministic in nature, in the sense, each move in this automata is uniquely determined on current input and current state. On the other hand, it is non deterministic in nature, in NFA i.e. non deterministic finite automata.

## 1.5 DFA

SPPU : Dec.-08,09,10,11,13,14,16,18, May-11,12,14,15,18,19, Aug.-17, Oct.-18, Marks 10

The finite automata is called Deterministic Finite Automata if there is only one path for a specific input from current state to next state. For example, the DFA can be shown as below.

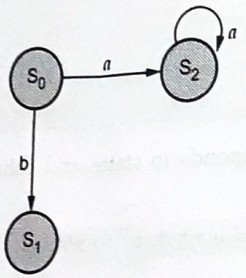



**Fig. 1.5.1 Deterministic finite automata**

From state $S_0$ for input '*a*' there is only one path, going to $S_2$. Similarly from $S_0$ there is only one path for input b going to $S_1$.

The DFA can be represented by the same 5-tuples described in the definition of FSM. All the above examples are actually the DFAs.

**Definition of DFA**

A deterministic finite automation is *a* collection of following things -

1) The finite set of states which can be denoted by Q.

2) The finite set of input symbols $\Sigma$.

3) The start state $q_0$ such that $q_0 \in Q$.

4) A set of final states F such that $F \in Q$.

5) The mapping function or transition function denoted by $\delta$. Two parameters are passed to this transition function : One is current state and other is input symbol. The transition function returns a state which can be called as next state.

For example, $q_1 = \delta (q_0, a)$ means from current state $q_0$, with input *a* the next state transition is $q_1$.





In short, the DFA is a five tuple notation denoted as :

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

The name of DFA is A which is a collection of above described five elements.

### Problems Based on DFA

**Example 1.5.1** *Design DFA which accepts the strings that start with 1 and ends in 0.*

**Solution :**

- **Logic** : As the DFA contains every string which is starting with 1, there should be a transition with input 1 from start state to next state. Similarly the edge incoming to the final stage should carry 0 along with it as the strings in this DFA end with 0.
- **Design :**

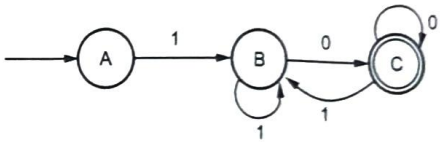



**Fig. 1.5.2**

- **Simulation :** Consider string **1011010**. We will read each character and according move to next states.

| | |
|---|---|
| A | ⊢ 1011010 |
| 1B | ⊢ 011010 |
| 10C | ⊢ 11010 |
| 101B | ⊢ 1010 |
| 1011B | ⊢ 010 |
| 10110C | ⊢ 10 |
| 101101B | ⊢ 0 |
| 1011010C | ⊢ ACCEPT |

Thus on reading the complete string we reach to the final state. Hence it is a valid string and this string belongs to the given DFA.

**Example 1.5.2** *Design FA which accepts odd number of 1's and any number of 0's.*

SPPU : Dec.-11, Marks 4

**Solution :**

- **Logic :** The odd number of 1's means the DFA can accept the 1, 111,11111,1111111,.... But there is no restriction on number of 0's i..e any number of 0's are allowed in the string.
- **Design :**

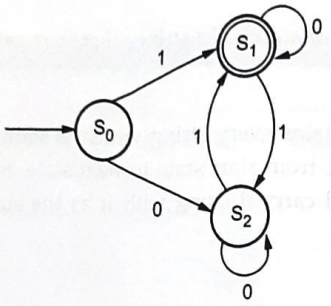



**Fig. 1.5.3**

- **Simulation :** At the start if we read input 1 then we will go to state $S_1$ which is a final state as we have read odd number of 1's. There can be any number of zeros at any state and therefore the self loop is applied to state $S_2$ as well as to state $S_1$. For example, if the input is 10101101, in this string there are any number of zeros but odd number of ones. The machine will derive this input as follows -

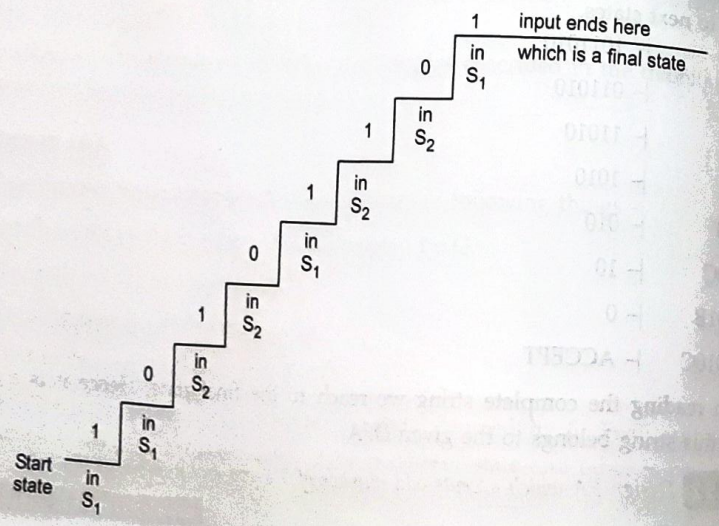



**Fig. 1.5.4 Ladder diagram of processing the input**





**Example 1.5.3** *Design FA which checks whether the given unary number is divisible by 3.*

**SPPU : Dec.-10, Marks 6**

**Solution :**

- **Logic :** The unary number is made up of ones. The number 3 can be written in unary form as 111, number 5 can be written as 11111 and so on. The unary number which is divisible by 3 can be 111 or 111111 or 111111111 and so on.
- **Design :** The transition table is as follows -

| State \ Input | 1 |
|---|---|
| → $q_0$ | $q_1$ |
| $q_1$ | $q_2$ |
| $q_2$ | $q_3$ |
| ($q_3$) | $q_1$ |

- **Simulation :**

Consider a number **111111** which is equal to 6 i.e. divisible by 3. So after complete scan of this number we reach to final state $q_3$.

Start 111111

State $q_0$

$1q_1 11111$

$11q_2 1111$

$111q_3 111$

$1111q_1 11$

$11111q_2 1$

$111111q_3$ → Now we are in final state.

**Example 1.5.4** *Design FA to check whether given decimal number is divisible by three.*

**SPPU : May-12, Marks 8, Dec.-13, Marks 6**

**Solution : Logic :**

- To determine whether the given decimal number is divisible by three, we need to take the input number digit by digit.
- Also, while considering its divisibility by three, we have to consider that the possible remainders could be 1, 2 or 0. The remainder 0 means, it is divisible by 3.
- Hence from input set {0, 1, 2, ...., 9} (since decimal number is a input), we will get either remainder 0 or 1 or 2 while testing its divisibility by 3.

- So we need to group these digits according to their remainders. The groups are given below -

  **remainder 0 group : $S_0$ : (0, 3, 6, 9)**

  **remainder 1 group : $S_1$ : (1, 4, 7)**

  **remainder 2 group : $S_2$ : (2, 5, 8)**

- **Design :** We have named out these states as $S_0$, $S_1$ and $S_2$. The state $S_0$ will be the final state as it is remainder 0 state.

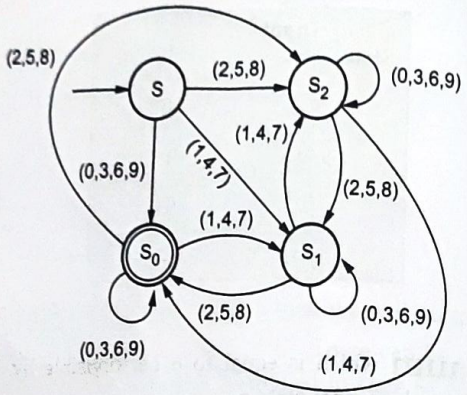



**Fig. 1.5.5**

- **Simulation :** Let us test the above FA, if the number is 36 then it will proceed by reading the number digit by digit.

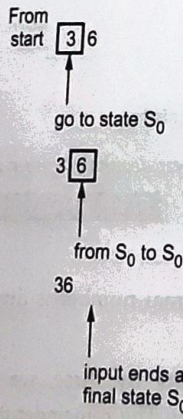



**Fig. 1.5.6**





Hence the number is divisible 3, isn't it ? Similarly if number is 121 which is not divisible by 3, it will be processed as

S 121

$1S_1$ 21

$12S_0$ 1

$121S_1$ which is remainder 1 state.

**Example 1.5.5** *Design FA which checks whether a given binary number is divisible by three.*

**SPPU : Dec.-10, Oct.-18 (In Sem), Marks 6; May-18 (End Sem), Marks 4**

**Solution :**

- **Logic :** The input number is a binary number. Hence the input set will be $\Sigma = \{0, 1\}$. The start state is denoted by S, the remainder 0 is by $S_0$, remainder 1 by $S_1$ and remainder 2 by $S_2$.

- **Design :**

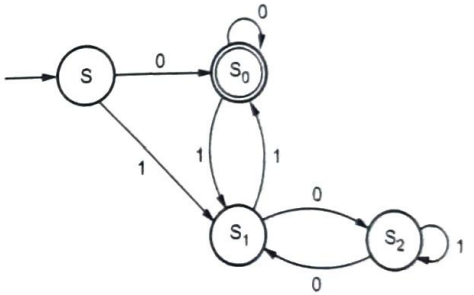



**Fig. 1.5.7**

- **Simulation :** Let us take some number 010 which is equivalent to 2. We will scan this number from MSB to LSB.

$$\begin{array}{ccc} 0 & 1 & 0 \\ \uparrow & \uparrow & \uparrow \\ S_0 & S_1 & S_2 \end{array}$$

Then we will reach to state $S_2$ which is remainder 2 state. Similarly for input 1001 which is equivalent to 9 we will be in final state after scanning the complete input.

$$\begin{array}{cccc} 1 & 0 & 0 & 1 \\ \uparrow & \uparrow & \uparrow & \uparrow \\ S_1 & S_2 & S_1 & \boxed{S_0} \end{array}$$

Thus the number is really divisible by 3.

**Example 1.5.6** *Design FA which accepts even number of 0's and even number of 1's.*

**Solution :**

- **Logic :** This FA will consider four different stages for input 0 and 1. The stages could be

  even number of 0 and even number of 1,

  even number of 0 and odd number of 1,

  odd number of 0 and even number of 1,

  odd number of 0 and odd number of 1.

- **Design :** Let us try to design the machine

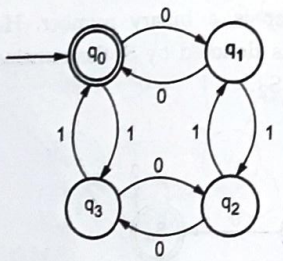



**Fig. 1.5.8**

Here $q_0$ is a start state as well as final state. Note carefully that a symmetry of 0's and 1's is maintained. We can associate meanings to each state as :

$q_0$ : State of even number of 0's and even number of 1's.

$q_1$ : State of odd number of 0's and even number of 1's.

$q_2$ : State of odd number of 0's and odd number of 1's.

$q_3$ : State of even number of 0's and odd number of 1's.

The transition table can be as follows -

| State \ Input | 0 | 1 |
|---|---|---|
| → (q0) | q1 | q3 |
| q1 | q0 | q2 |
| q2 | q3 | q1 |
| q3 | q2 | q0 |

**Example 1.5.7** *Design FA to accept L, where L = {Strings in which a always appears trippled} over the set Σ = {a, b}.*





**Solution :**

- **Logic :** In this language the a always appear in a clump of three. This clump may or may not be surrounded by b.

- **Design :**

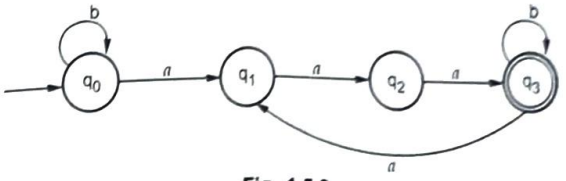



**Fig. 1.5.9**

- **Simulation :**

  baaabaaa

  q0 baaabaaa

  bq0 aaabaaa

  baq1 aabaaa

  baaq2 abaaa

  baaaq3 baaa

  baaabq3 aaa

  baaabaq1 aa

  baaabaaq2 a

  baaabaaaq3

**Example 1.5.8** *Design FA to accept L where all the strings in L are such that total number of a's in them are divisible by 3.*

**Solution :**

- **Logic :** As we have seen earlier, while testing divisibility by 3, we group the input as remainder 0, remainder 1 and remainder 2.

Hence,

**$S_0$ : State of remainder 0.**

**$S_1$ : State of remainder 1.**

**$S_2$ : State of remainder 2.**

- **Design :**

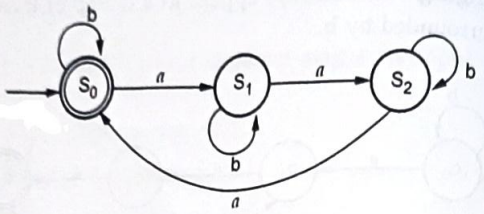



**Fig. 1.5.10**

**Example 1.5.9** *Design a FA that reads strings made up of letters in the word. CHARIOT and recognize those strings that contain the word 'CAT' as a substring.*

**SPPU : May-12, Marks 8; May-19, Marks 5**

**Solution :**

- **Logic :** To design this FA we will first consider the input set which will be Σ = {C, H, A, R, I, O, T}. From this input set we have to recognize the word 'CAT'. We can do that as follows -
- **Design :**

**Step 1 :**

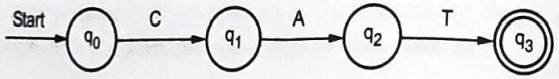



**Fig. 1.5.11**

**Step 2 :**

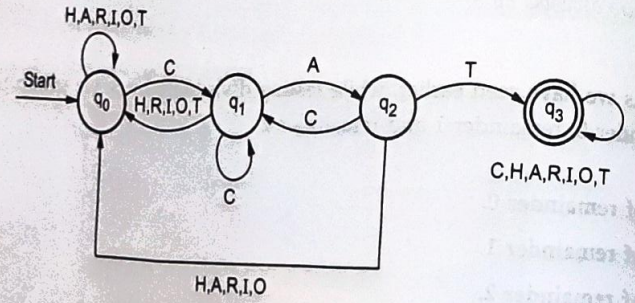



**Fig. 1.5.12 Required finite automata**

From Fig. 1.5.11 it is clear that on recognizing C initially from Σ = {C, H, A, R, I, O, T} we are moving to state $q_1$, other than C all the characters are remaining in state $q_0$ only.





From $q_1$ the character A will be identified and from $q_2$ character T will be identifier. The transition table can be drawn as follows :

| State \ Input | C | H | A | R | I | O | T |
|---|---|---|---|---|---|---|---|
| → $q_0$ | $q_1$ | $q_0$ | $q_0$ | $q_0$ | $q_0$ | $q_0$ | $q_0$ |
| $q_1$ | $q_1$ | $q_0$ | $q_2$ | $q_0$ | $q_0$ | $q_0$ | $q_0$ |
| $q_2$ | $q_1$ | $q_0$ | $q_0$ | $q_0$ | $q_0$ | $q_0$ | $q_3$ |
| ($q_3$) | $q_3$ | $q_3$ | $q_3$ | $q_3$ | $q_3$ | $q_3$ | $q_3$ |

The substring may appear anywhere in the input string for any number of times. For example, 'HACCATHA'. From this 'CAT' can be recognized and we should reach to $q_3$ state finally.

**Example 1.5.10** *Design DFA to accept odd and even numbers represented using binary notation.*

**Solution :**

- **Logic :** The binary number that ends with 0 is an even number and binary number that ends with 1 is an odd number. Hence the DFA is -
- **Design :**

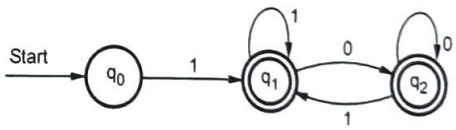



**Fig. 1.5.13**

**Example 1.5.11** *Write a DFA to accept the language L = {L : |W| mod 5 ≠ 0}.*

**Solution :**

- **Design :** The string which we obtain should not be divisible by 5. Hence the, DFA will be,

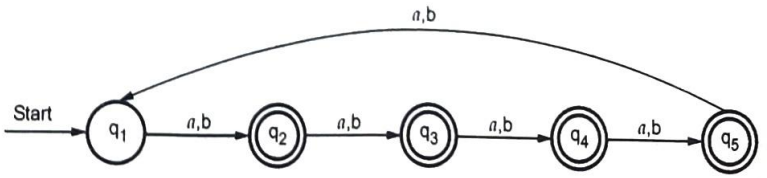



**Fig. 1.5.14**

The transition table can be drawn as follows -

| State \ Input | a | b |
|---|---|---|
| $q_1$ | $q_2$ | $q_2$ |
| $q_2$ | $q_3$ | $q_3$ |
| $q_3$ | $q_4$ | $q_4$ |
| $q_4$ | $q_5$ | $q_5$ |
| $q_5$ | $q_1$ | $q_1$ |

- **Simulation :** We can consider the string "*a*bbb". This string is a valid string as it is not divisible by 5 i.e *a*bbb mod 5 ≠ 0. That also means that we should reach to final state on acceptance of this string. The simulation of this string for given DFA is

$$\begin{aligned}\delta(q_1, abbb) &\vdash \delta(q_2, bbb)\\ &\vdash \delta(q_3, bb)\\ &\vdash \delta(q_4, b)\\ &\vdash \delta(q_5, \varepsilon)\end{aligned}$$

where $q_5$ is a final state.

Now consider string "*ababa*" which is invalid string as it is divisible by 5. That means ababa mod 5 = 0. That means we should reach to non final state on acceptance of this string. The simulation for this string is as given below.

As $q_1$ is a start we will start from state $q_1$.

$$\begin{aligned}\delta(q_1, ababa) &\vdash \delta(q_2, baba)\\ &\vdash \delta(q_3, aba)\\ &\vdash \delta(q_4, ba)\\ &\vdash \delta(q_5, a)\\ &\vdash \delta(q_1, \varepsilon)\end{aligned}$$

Here $q_1$ is a non-final state. That means "*ababa*" is invalid string for the given DFA.

**Example 1.5.12** *Design a DFA* $L(M) = \{W \mid W \in \{0,1\}^*\}$ *and W is a string that does not contain consecutive 1's.*





**Solution :**

- **Design :** When three consecutive 1's occur the DFA will be

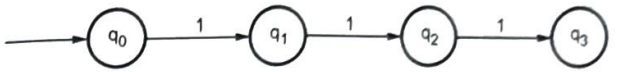



**Fig. 1.5.15**

Here two consecutive 1's or single 1 is acceptable, hence

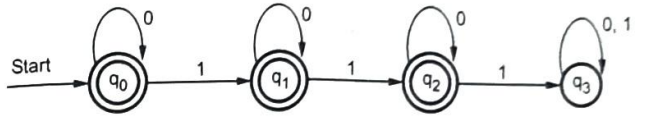



**Fig. 1.5.16**

The states $q_0$, $q_1$, $q_2$ are final states. Hence DFA M can be represented as,

$$M = \{Q, \Sigma, q_0, F\}$$

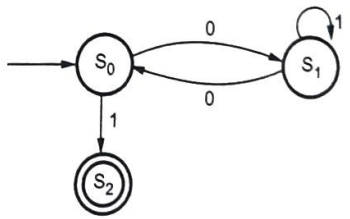
where $Q = \{q_0, q_1, q_2, q_3\}$

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

$F = \{q_0, q_1, q_2\}$

**Example 1.5.13** *Design a DFA which accepts strings with even number of 0's followed by single 1 over* $\Sigma = \{0,1\}$.

**Solution :** The DFA can be shown by a transition diagram as




Here state $S_1$ will accept all the odd number of 0's and $S_0$ will accept all even number of 0's. It should then be followed by single 1. Hence $S_2$ is a final state. Consider the input

0 0 0 1 0 1
0 $S_1$ 0 0 1 0 1
0 0 $S_0$ 0 1 0 1
0 0 0 $S_1$ 1 0 1
0 0 0 1 $S_1$ 0 1
0 0 0 1 0 $S_0$ 1
0 0 0 1 0 1 $S_2$

We now will reach to final state $S_2$, and the input ends here. Hence 000101 is a valid input string. The transition table for above drawn DFA can be represented as

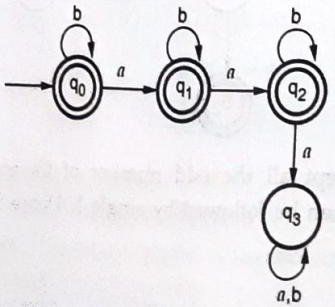
| States \ Input | 0 | 1 |
|---|---|---|
| → $S_0$ | $S_1$ | $S_2$ |
| $S_1$ | $S_0$ | $S_1$ |
| ($S_2$) | - | - |

**Transition table**

**Example 1.5.14** *Design DFA which accepts all the strings not having more than two a's over* $\Sigma = \{a, b\}$.

**Solution :**

- **Logic :** In this DFA maximum two a's are accepted. If we try to accept third a then it should not lead us to final state. Such a DFA can be as shown below.
- **Design :**








The transition table can be as shown -

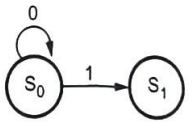
| States \ Input | a | b |
|---|---|---|
| → ($q_0$) | $q_1$ | $q_0$ |
| ($q_1$) | $q_2$ | $q_1$ |
| ($q_2$) | $q_3$ | $q_2$ |
| $q_3$ | $q_3$ | $q_3$ |

**Transition table**

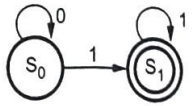
**Example 1.5.15** *Design a DFA for accepting all the strings of* $\{L = 0^m 1^n \mid m \geq 0 \text{ and } n \geq 1\}$.

**Solution :**

- **Logic :** This is a language in which all the 0's followed by all 1's. But number of zero's and number of 1's are different. The language explicitly mentions that there should be at least one 1. Any number of 0's followed by only one 1 is -




- **Design :** But we have $1^n$ in the language. Hence the DFA can be




The transition table can be

| State \ Input | 0 | 1 |
|---|---|---|
| ($S_0$) | $S_0$ | $S_1$ |
| $S_1$ | - | $S_1$ |

- **Simulation :** Consider a string 00111, which is a valid string and is accepted by above drawn DFA.

$\delta(S_0, 00111) \vdash (S_0, 0111)$
$\vdash (S_0, 111)$
$\vdash (S_1, 11)$
$\vdash (S_1, 1)$
$\vdash (S_1, \varepsilon)$

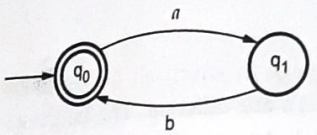
Thus we reach to final state $S_1$ on acceptance of 00111.

**Example 1.5.16** *Design DFA over $\sum = \{a, b\}$ for*
i) $(ab)^n$ with $n \geq 0$ ii) $(ab)^n$ with $n \geq 1$.

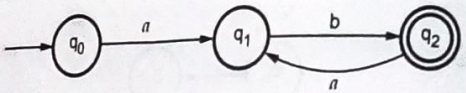
**Solution :**

- This is a language of ab in a pair in
i) Condition $\varepsilon$ is accepted and in
ii) Condition $\varepsilon$ is not accepted.

The DFA for i) can be




ii) The DFA will be -




The string accepted by

i) $\delta(q_0, ab) \vdash (q_1, b)$
$\vdash (q_0, \varepsilon)$

Here $q_0$ is a final state.

ii) $\delta(q_0, ab) \vdash (q_1, b)$
$\vdash (q_2, \varepsilon)$

Here $q_2$ is final state.

**Example 1.5.17** *Design DFA for accepting the set of integers.*

**Solution :**

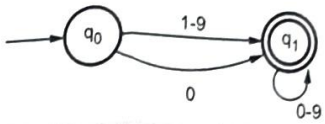
- **Logic :** For representing any integer the set {0, 1, ... 9} is used. We can represent any integer value by taking appropriate combinations of symbols from {0, ... 9}.





Hence the DFA will be

- **Design :**




**Example 1.5.18** *Construct a DFA for accepting L over {0, 1} such that every substring of length 4 contains at least three 1's.*

**SPPU : Dec.-08, Marks 8, Dec.-16, End Sem, Marks 4**

**Solution :**

- **Logic :** The r.e. for designing the required DFA is

Regular expression = $(0111 + 1011 + 1101 + 1110)^*$.

Hence we can design the required DFA as follows :

- **Design :**

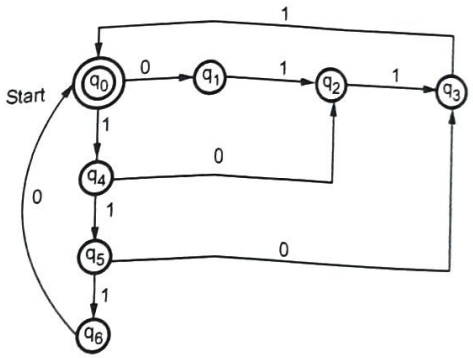



**Fig. 1.5.17**

**Example 1.5.19** *Design a FSM for divisibility of 5 tester of a given decimal number.*

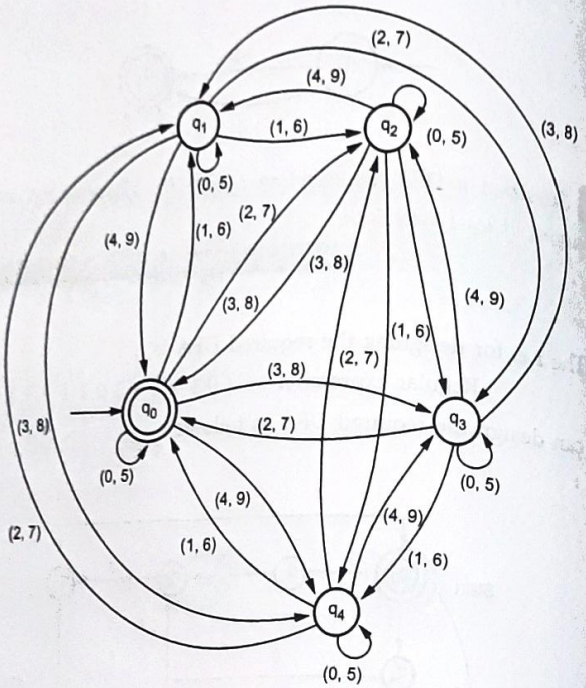
**SPPU : Dec.-11, Marks 8**

**Solution :**

- **Logic :** The logic for designing such DFA is simple. First of all we will group the decimal numbers 0 to 9 as follows :

(0, 5) → remainder 0 → $q_0$ state
(1, 6) → remainder 1 → $q_1$ state
(2, 7) → remainder 2 → $q_2$ state
(3, 8) → remainder 3 → $q_3$ state
(4, 9) → remainder 4 → $q_4$ state

Then the required DFA can be drawn as follows -




Now assume input number 365. The state transition will be

$\delta(q_0, 3) = q_3$

$\delta(q_3, 6) = q_4$

$\delta(q_4, 5) = q_4$

That means on accepting input 365 we will reach state $q_4$ which is actually a **remainder 4 state**. Similarly if we sum up the digits of number 365 then we get $3 + 6 + 5 = 14$ and $14/5$ = **remainder 4.**

If the sum of digits is 5, 10, 15 and so on then we must reach in $q_0$ state. The $q_0$ state denotes that sum of digits of a given number is divisible by 3.

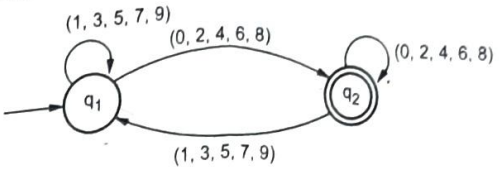
**Example 1.5.20** *Design FSM to check for divisibility of a decimal number by 2.*

**SPPU : Dec.-09, Marks 6**





**Solution :** If any decimal number contains 0, 2, 4, 6 or 8 at its unit place then that number is always divisible by 2. Hence we will make two groups.



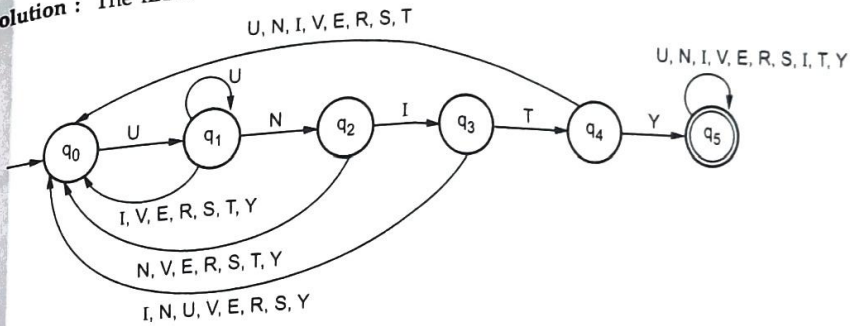

$(0, 2, 4, 6, 8) \rightarrow q_2$ state

$(1, 3, 5, 7, 9) \rightarrow q_1$ state

**Example 1.5.21** *Design a finite automaton that reads strings made of letter in the word "UNIVERSITY" and recognize these strings that contain the word UNITY as substring.*

**SPPU : Dec.-09, Marks 6**

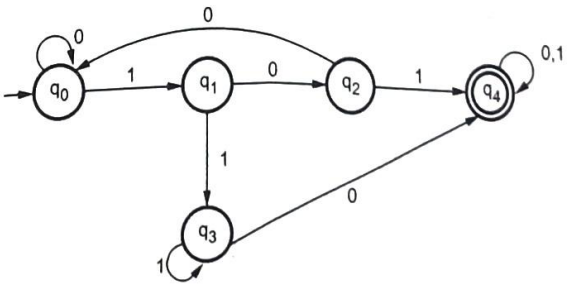
**Solution :** The finite automaton can be drawn as follows -




**Example 1.5.22** *Design a FSM to concept those strings having 101 or 110 as substring.*

**SPPU : May-11, Marks 4**

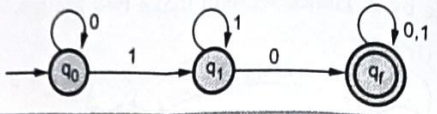
**Solution :** The FSM will be




**Example 1.5.23** *Design a DFA for a language of strings of 0's and 1's such that - i) Substring is 10 ii) Strings ending with 101.*

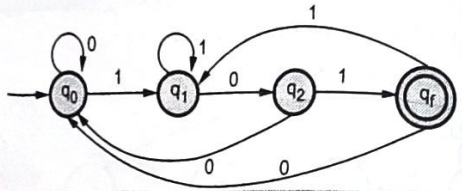
**SPPU : Dec.-12, Marks 10**

**Solution : i)**




| Q \ Σ | 0 | 1 |
|---|---|---|
| → $q_0$ | $q_0$ | $q_1$ |
| $q_1$ | $q_f$ | $q_1$ |
| * $q_f$ | $q_f$ | $q_f$ |

ii) $M = (\{q_0, q_1, q_f\}, \{0, 1\}, \delta, q_0, \{q_f\})$



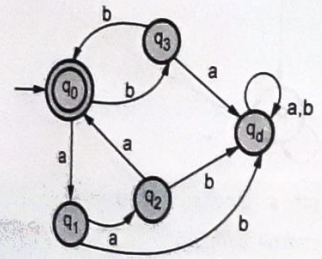

| Q \ Σ | 0 | 1 |
|---|---|---|
| → $q_0$ | $q_0$ | $q_1$ |
| $q_1$ | $q_2$ | $q_1$ |
| $q_2$ | $q_0$ | $q_f$ |
| * $q_f$ | $q_0$ | $q_1$ |

$M = (\{q_0, q_1, q_2, q_f\}, \{0, 1\}, \delta, q_0, \{q_f\})$

**Example 1.5.24** *Obtain a DFA to accept strings of a's and b's such that -*
*$L = \{W / W \in (a+b)^*$ such that $N_a(W)$ mod 3 = 0 and $N_b(W)$ mod 2 = 0$\}$*

**SPPU : Dec.-12, Marks 10**

**Solution :**








| Q \ Σ | a | b |
|---|---|---|
| → * $q_0$ | $q_1$ | $q_3$ |
| $q_1$ | $q_2$ | $q_d$ |
| $q_2$ | $q_0$ | $q_d$ |
| $q_3$ | $q_d$ | $q_0$ |
| $q_d$ | $q_d$ | $q_d$ |

$M = (\{q_0, q_1, q_2, q_3, q_f\}, \{a, b\}, \delta, q_0, \{q_0\})$

**Example 1.5.25** *Design a DFA for following language*
*$L = \{W | W$ is Binary word of length 4i (where i > = 1) such that each consecutive block of 4 bits contains atleast 2 0s$\}$ = {0000, 0110, 01101100, ...}*

**SPPU : Dec.-13, Marks 10; Aug.-17, Marks 8**

**Solution :**

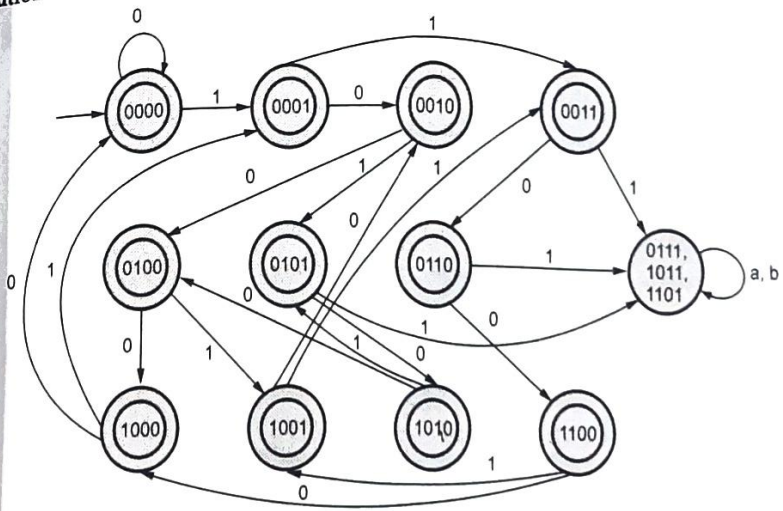



**Fig. 1.5.18**

**Example 1.5.26** *Design a FSM to check given decimal number is divisible by 4 or not.*

**SPPU : May-14, Marks 8**

**Solution :** For checking divisibility of a decimal number by 4, there are following states

| | |
|---|---|
| $q_0$ | Remainder 0 State. It is a final state |
| $q_1$ | Remainder 1 State |
| $q_2$ | Remainder 2 State |
| $q_3$ | Remainder 3 State |

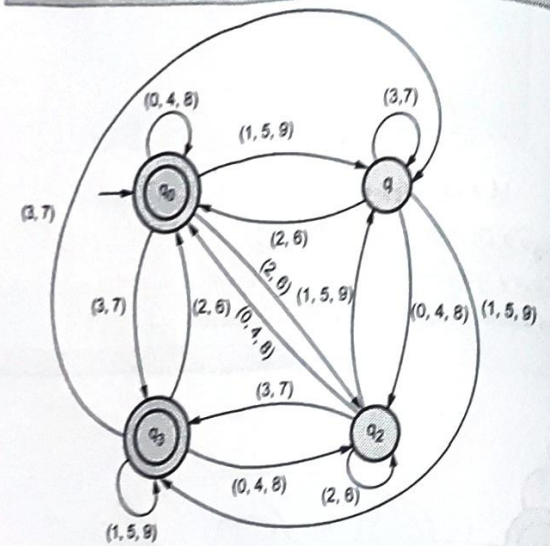



**Fig. 1.5.19**

**Example 1.5.27** *Design finite automation for the following*

*i) FA which reads strings made up of {0, 1} and accepts only those strings which end in either '00' or '11'.*

*ii) FA which accepts only those strings with 'a' at every even position. $\Sigma = \{a, b\}$.*

**SPPU : Dec.-14, Marks 6**

**Solution :**

i)

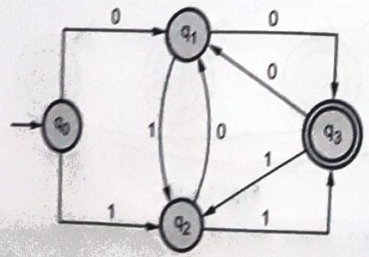



**Fig. 1.5.20**





ii)


**Fig. 1.5.21**

**Example 1.5.28** *Define Finite Automata and justify why palindrome strings cannot be checked for by FSM.*

**SPPU : May-15, Marks 4**

**Solution : Finite Automata :** Refer section 1.2.

The palindrome is a string which appears to be the same if read from left to right or from right to left. That means to recognize the palindrome, the symbols read from left to right must be remembered in order to match the symbols that can be read from right to left. To remember the symbols memory must be required in the machine. But as finite automata does not contain any memory, it is not possible to recognize palindrome using FSM.

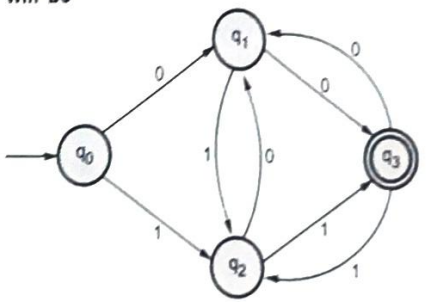
**Example 1.5.29** *Design an FA over $\sum = \{0, 1\}$ For the following :*

*i) Strings which end either "00" or "11".*

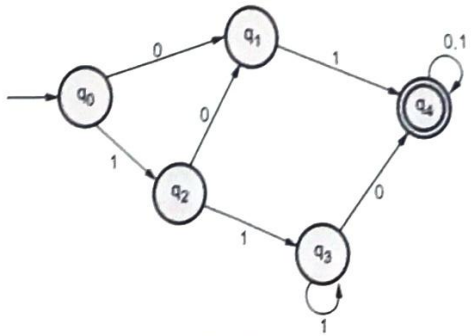
*ii) Strings which contain either "01" or "110".*

**SPPU : Oct.-16, In Sem., Marks 5**

**Solution : i) The FA will be**




ii)



**Example 1.5.30** *Design FA for accepting strings over Σ = {a,b}.*

*i) String containing at least one 'a' and at least one 'b'.*

*ii) Set of all strings that do not contain three or more consecutive a's*

**SPPU : Oct.-18 (In Sem), Marks 6**

**Solution : i)**

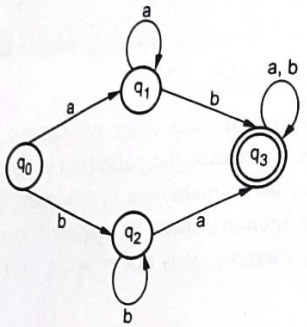



**Fig. 1.5.22**

**ii)**

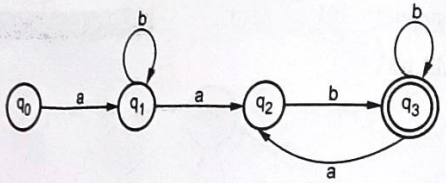



**Fig. 1.5.23**

**Example 1.5.31** *Construct transition graph for the following regular expression.*

$r = 1^*{\cdot}0{\cdot}0{\cdot}(0+1)^*$

**SPPU : Dec.-18 (End Sem), Marks 4**

**Solution :** $r = 1^*{\cdot}0{\cdot}0\ (0+1)^*$

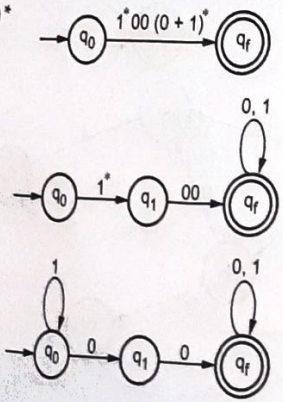



**Fig. 1.5.24**





## 1.6 NFA

### Definition of NFA

The NFA can be formally defined as a collection of 5-tuples.

1) Q is a finite set of states.

2) Σ is a finite set of inputs.

3) δ is called next state or transition function.

4) $q_0$ is initial state.

5) F is a final state where $F \subseteq Q$.

There can be multiple final states. Thus the next question might be what is the use of NFA. The NFA is basically used in theory of computations because they are more flexible and easier to use than the DFAs.

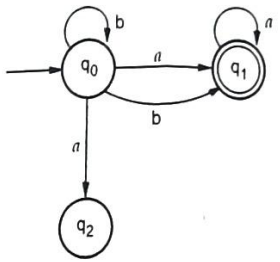



**Fig. 1.6.1 Non deterministic finite automata**

### Problems Based on NFA

**Example 1.6.1** *Design the NFA transition diagram for the transition table as given below -*

| | 0 | 1 |
|---|---|---|
| $q_0$ | $\{q_0, q_1\}$ | $\{q_0, q_2\}$ |
| $q_1$ | $\{q_3\}$ | |
| $q_2$ | $\{q_2, q_3\}$ | $\{q_3\}$ |
| $(q_3)$ | $\{q_3\}$ | $\{q_3\}$ |

Here the NFA is $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$.

**Solution :** The transition diagram can be drawn by using the mapping function as given in table.

$$\delta(q_0, 0) = \{q_0, q_1\}$$

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

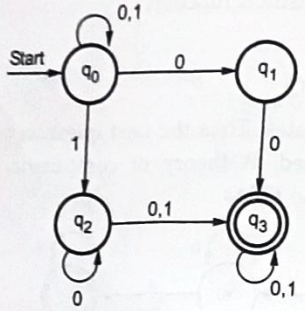
Then, $\delta(q_1, 0) = \{q_3\}$

Then, $\delta(q_2, 0) = \{q_2, q_3\}$

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

Then, $\delta(q_3, 0) = \{q_3\}$

$\delta(q_3, 1) = \{q_3\}$




**Example 1.6.2** *Construct a transition diagram for the NDFA*

$M = (\{q_1, q_2, q_3\}, \delta, q_1, \{q_3\})$ where $\delta$ is given by,

$\delta(q_1, 0) = \{q_2, q_3\}$ $\quad \delta(q_1, 1) = \{q_1\}$

$\delta(q_2, 0) = \{q_1, q_2\}$ $\quad \delta(q_2, 1) = \phi$

$\delta(q_3, 0) = \{q_2\}$ $\quad \delta(q_3, 1) = \{q_1, q_2\}$

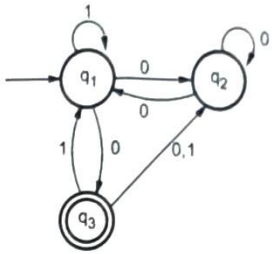
**Solution :** Firstly we will design a transition table using the given mapping function $\delta$.

| States \ Input | 0 | 1 |
|---|---|---|
| → $q_1$ | $\{q_2, q_3\}$ | $q_1$ |
| $q_2$ | $\{q_1, q_2\}$ | $\phi$ |
| (*$q_3$) | $q_2$ | $\{q_1, q_2\}$ |





The NDFA will be




## 1.7 NFA with Epsilon Moves

**SPPU : Dec.-15, Oct.-16, Marks 5**

The ε - transitions in NFA are given in order to move from one state to another without having any symbol from input set Σ.

Consider the NFA with ε as :

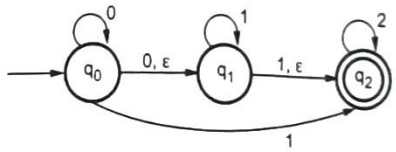



**Fig. 1.7.1 NFA with ε - transitions**

**Definition :**

The language L accepted by NFA with ε, denoted by M = (Q, Σ, δ, $q_0$, F) can be defined as :

Let, M = (Q, Σ, δ, $q_0$, F) be a NFA with ε.

where

Q is a finite set of states.

Σ is input set.

δ is a transition or a mapping function for transitions from $Q \times \{\Sigma \cup \varepsilon\}$ to $2^Q$.

$q_0$ is a start state.

F is a set of final states such that $F \in Q$.

The string w in L accepted by NFA can be represented as

$L(M) = \{w \mid w \in \Sigma^*$ and δ transition for w from $q_0$ reaches to F$\}$.

**Example 1.7.1** *Construct NFA with ε which accepts a language consisting the strings of any number of a's followed by any number of b's. Followed by any number of c's.*

**Solution :**

- **Logic :** Here any number of a's or b's or c's means zero or more in number means there can be zero or more a's followed by zero or more b's follow zero or more c's. Hence NFA with ε can be -
- **Design :**

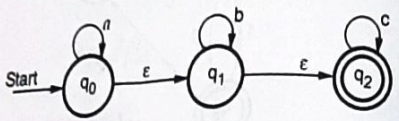



**Fig. 1.7.2**

Normally ε's are not shown in the input string. The transition table can be

| State \ Input | a | b | c | ε |
|---|---|---|---|---|
| $q_0$ | $q_0$ | ϕ | ϕ | $q_1$ |
| $q_1$ | ϕ | $q_1$ | ϕ | $q_2$ |
| $q_2$ | ϕ | ϕ | $q_2$ | ϕ |

- **Simulation :**

We can parse the string *aa*bbcc as follows -

$$\begin{aligned}
\delta(q_0, aabbcc) &\vdash \delta(q_0, abbcc)\\
&\vdash \delta(q_0, bbcc)\\
&\vdash \delta(q_0, \varepsilon bbcc)\\
&\vdash \delta(q_1, bbcc)\\
&\vdash \delta(q_1, bcc)\\
&\vdash \delta(q_1, cc)\\
&\vdash \delta(q_1, \varepsilon cc)\\
&\vdash \delta(q_2, cc)\\
&\vdash \delta(q_2, c)\\
&\vdash \delta(q_2, \varepsilon)
\end{aligned}$$

Thus we reach to accept state, after scanning the complete input string.





**Definition of ε - closure**

The ε - closure (p) is a set of all states which are reachable from state p on ε -transitions such that :

i) ε - closure (p) = p where $p \in Q$.

ii) If there exists ε - closure (p) = {q} and $\delta(q, \varepsilon) = r$ then ε - closure (p) = {q, r}.

**Example 1.7.2** *Find ε - closure for the following NFA with ε .*

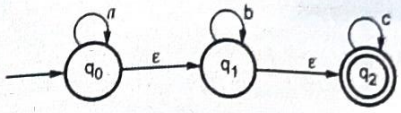



**Fig. 1.7.3**

**Solution :**

ε - closure $(q_0) = \{q_0, q_1, q_2\}$ means self state + ε - reachable states.

ε - closure $(q_1) = \{q_1, q_2\}$ means $q_1$ is a self state and $q_2$ is a state obtained from $q_1$ with ε input.

ε - closure $(q_2) = \{q_2\}$

**University Question**

1. *Write Formal definition of NFA with epsilon. Also define epsilon closure.*

**SPPU : Dec.-15, End Sem, Oct.-16 In Sem, Marks 5**

## 1.8 Conversion of NFA to DFA

**SPPU : Dec.-10, May-11,16, Marks 6**

The conversion method will follow following steps -

1) The start state of NFA M will be the start for DFA M'. Hence add $q_0$ of NFA (start state) to Q'. Then find the transitions from this start state.

2) For each state $[q_1, q_2, q_3, \ldots\ldots q_i]$ in Q' the transitions for each input symbol Σ can be obtained as,

i) $\delta'([q_1, q_2, \ldots\ldots q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \ldots\ldots \delta(q_i, a)$

$= [q_1, q_2, \ldots\ldots q_k]$ may be some state.

ii) Add the state $[q_1, q_2, \ldots\ldots q_k]$ to DFA if it is not already added in Q'.

iii) Then find the transitions for every input symbol from Σ for state $[q_1, q_2, \ldots\ldots q_k]$. If we get some state $[q_1, q_2, \ldots\ldots q_n]$ which is not in Q' of DFA then add this state to Q' .

iv) If there is no new state generating then stop the process after finding all transitions.

3) For the state $[q_1, q_2, \ldots\ldots q_n] \in Q'$ of DFA if any one state $q_i$ is a final state NFA then $[q_1, q_2, \ldots\ldots q_n]$ becomes a final state. Thus the set of all the final $\in F'$ of DFA.

## Problem based on NFA to DFA

**Example 1.8.1** *Let $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\})$. be NFA where $\delta(q_0, 0) = \{q_0, q_1\}, \delta(q_0, 1) = \{q_1\}$ $\delta(q_1, 0) = \phi, \delta(q_1, 1) = \{q_0, q_1\}$. Construct its equivalent DFA.* **SPPU : May-11, Marks 6**

**Solution :** Let the DFA $M' = (Q', \Sigma, \delta', q_0', F')$.

Now, the $\delta'$ function will be computed as follows -

As $\delta(q_0, 0) = \{q_0, q_1\}$ $\delta'([q_0], 0) = [q_0, q_1]$

As in NFA the initial state is $q_0$, the DFA will also contain the initial state $[q_0]$.

Let us draw the transition table for $\delta$ function for a given NFA.

| | | State \ Input | 0 | 1 | | |
|---|---|---|---|---|---|---|
| $\delta(q_0, 0)$ | $\Rightarrow$ | $\rightarrow q_0$ | $\{q_0, q_1\}$ | $\{q_1\}$ | $\Leftarrow$ | $\delta(q_0, 1)$ |
| $\delta(q_1, 0)$ | $\Rightarrow$ | $(q_1)$ | $\phi$ | $\{q_0, q_1\}$ | $\Leftarrow$ | $\delta(q_1, 1)$ |

$\delta$ function for NFA

From the transition table we can compute that there are $[q_0]$, $[q_1]$, $[q_0, q_1]$ states for its equivalent DFA. We need to compute the transition from state $[q_0, q_1]$.

$$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

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

Similarly,

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





$$\delta'([q_0, q_1], 1) = [q_0, q_1]$$

So,

As in the given NFA $q_1$ is a final state, then in DFA wherever $q_1$ exists that state becomes a final state. Hence in the DFA final states are $[q_1]$ and $[q_0, q_1]$. Therefore set of final states $F = \{[q_1], [q_0, q_1]\}$.

The equivalent DFA is -

| State \ Input | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $[q_0, q_1]$ | $[q_1]$ |
| $([q_1])$ | $\phi$ | $[q_0, q_1]$ |
| $([q_0, q_1])$ | $[q_0, q_1]$ | $[q_0, q_1]$ |

Transition diagram for equivalent DFA

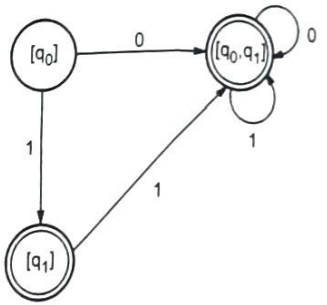



**Fig. 1.8.1**

Even we can change the names of the states of DFA.

$A = [q_0]$

$B = [q_1]$

$C = [q_0, q_1]$

With these new names the DFA will be as follows -

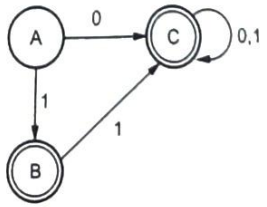



**Fig. 1.8.2 An equivalent DFA**

**Example 1.8.2** *Construct DFA equivalent to the given NFA.* **SPPU : Dec.-10, Marks**

| State \ Input | 0 | 1 |
|---|---|---|
| → p | {p, q} | p |
| q | r | r |
| r | s | – |
| s | s | s |

**Solution :** The NFA M = {{p, q, r, s}, {0, 1}, δ {p}, {s}}

The equivalent DFA will be constructed.

| State \ Input | 0 | 1 |
|---|---|---|
| → [p] | [p, q] | [p] |
| [q] | [r] | [r] |
| [r] | [s] | – |
| (s) | [s] | [s] |
| [p, q] | [p, q, r] | [p, r] |

Continuing with the generated new states.

| State \ Input | 0 | 1 |
|---|---|---|
| → [p] | [p, q] | [p] |
| [q] | [r] | [r] |
| [r] | [s] | – |
| (s) | [s] | [s] |





| | | |
|---|---|---|
| [p, q] | [p, q, r] | [p, r] |
| [p, q, r] | [p, q, r, s] | [p, r] |
| [p, r] | [p, q, s] | [p] |
| (p, q, r, s) | [p, q, r, s] | [p, r, s] |
| (p, q, s) | [p, q, r, s] | [p, r, s] |
| (p, r, s) | [p, q, s] | [p, s] |
| (p, s) | [p, q, s] | [p, s] |

The final state F′ = { [s], [p, q, r, s], [p, q, s], [p, r, s], [p, s] }

The transition graph shows two disconnected parts. But part I will be accepted as final DFA because it consists of start state and final states, in part II there is no start state.

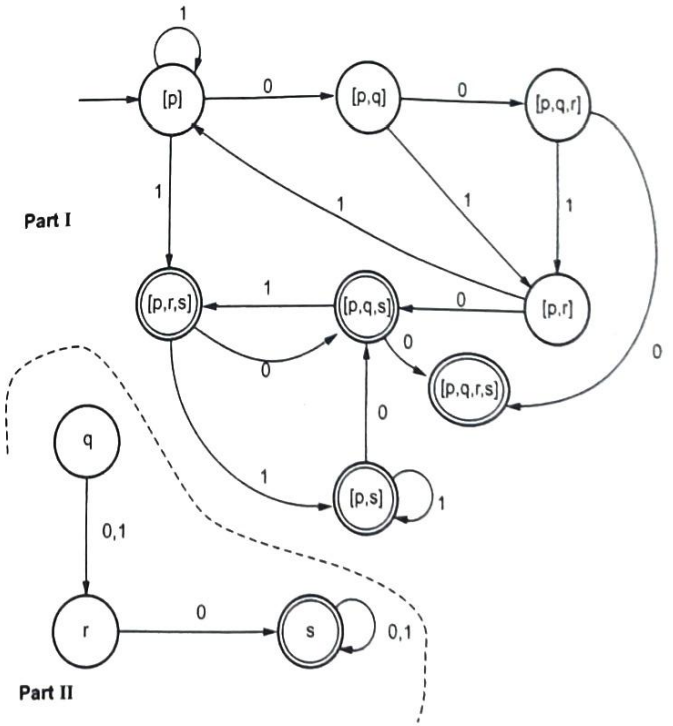



**Fig. 1.8.3**

**Example 1.8.3** *Convert the following NFA into DFA.*

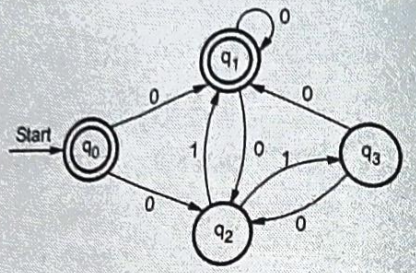



**Fig. 1.8.4**

**Solution :** We will first design a transition table from given transition diagram.

| State \ Input | 0 | 1 |
|---|---|---|
| $q_0$ | $\{q_1, q_2\}$ | $\phi$ |
| $q_1$ | $\{q_1, q_2\}$ | $\phi$ |
| $q_2$ | $\phi$ | $\{q_1, q_3\}$ |
| $q_3$ | $\{q_1, q_2\}$ | $\phi$ |

Now, $\delta(q_0, 0) = \{q_1, q_2\}$

We can write this as

$\delta'([q_0], 0) = [q_1, q_2]$

Here $[q_1, q_2]$ becomes one newly generated state when 0 input is received in state $[q_0]$.

Similarly,

$\delta'([q_0], 1) = \phi$ No state getting generated.

$\delta(q_1, 0) = \{q_1, q_2\}$ we can write it as

$\delta'([q_1], 0) = [q_1, q_2]$

$\delta'([q_1], 1) = \phi$

$\delta'([q_2], 0) = \phi$

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

i.e. $\delta'([q_2], 1) = [q_1, q_3]$ again a new state $[q_1, q_3]$ gets generated.





Similarly,

$\delta'([q_3], 0) = [q_1, q_2]$

$\delta'([q_3], 1) = \phi$

Now we will obtain $\delta'$ transitions on newly generated states $[q_1, q_2]$ and $[q_1, q_3]$.

$$\delta'([q_1, q_2], 0) = \delta([q_1], 0) \cup \delta([q_2], 0)$$
$$= \{q_1, q_2\} \cup \phi$$
$$= \{q_1, q_2\}$$

$\therefore \quad \delta'([q_1, q_2], 0) = [q_1, q_2]$

Similarly,

$$\delta'([q_1, q_2], 1) = \delta([q_1], 1) \cup \delta([q_2], 1)$$
$$= \phi \cup \{q_1, q_3\}$$
$$= \{q_1, q_3\}$$

$\therefore \quad \delta'([q_1, q_2], 1) = [q_1, q_3]$

Now, $\delta'([q_1, q_3], 0) = \delta([q_1], 0) \cup \delta([q_3], 0)$

$$= \{q_1, q_2\} \cup \{q_1, q_2\}$$
$$= \{q_1, q_2\}$$

$\therefore \quad \delta'([q_1, q_3], 0) = [q_1, q_2]$

Similarly,

$$\delta'([q_1, q_3], 1) = \delta([q_1], 1) \cup \delta([q_3], 1)$$
$$= \phi \cup \phi$$
$$= \phi$$

$\therefore \quad \delta'([q_1, q_3], 1) = \phi$

As now no new states are getting generated.

The transition table for DFA using above δ′ transitions is

| State \ Input | 0 | 1 |
|---|---|---|
| → $[q_0]$ | $[q_1, q_2]$ | $\phi$ |
| $[q_1]$ | $[q_1, q_2]$ | $\phi$ |
| $[q_2]$ | $\phi$ | $[q_1, q_3]$ |
| $[q_3]$ | $[q_1, q_2]$ | $\phi$ |
| $[q_1, q_2]$ | $[q_1, q_2]$ | $[q_1, q_3]$ |
| $[q_1, q_3]$ | $[q_1, q_2]$ | $\phi$ |

The transition diagram can be -

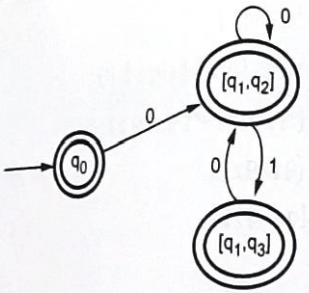
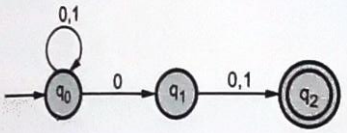



**Fig. 1.8.5**

As states $q_0$ and $q_1$ are final states original NFA states containing $q_1$ i.e. $[q_1,q$ and $[q_1, q_2]$ are also final states.

**Example 1.8.4** *Design an FA for the languages that contain strings with next-to-last symbol 0.*

**Solution :** We will construct NFA for the given problem








This NFA can be converted to DFA for that - first construct the transition table for above NFA.

| State \ i/p | 0 | 1 |
|---|---|---|
| → $q_0$ | $(q_0, q_1)$ | $q_0$ |
| $q_1$ | $q_2$ | $q_2$ |
| $q_2$ | $\phi$ | $\phi$ |

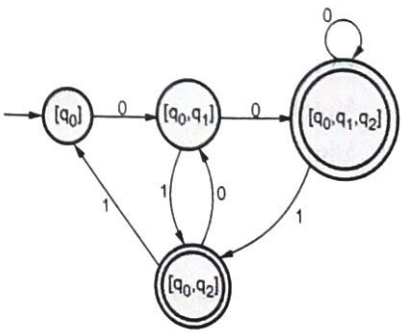
$\delta(q_0, 0) = (q_0, q_1)$ New state
$\delta(q_0, 1) = q_0$
$\delta(q_1, 0) = q_2$
$\delta(q_1, 1) = q_2$
$\delta(q_2, 0) = \phi$
$\delta(q_2, 1) = \phi$

Apply input transitions on new state

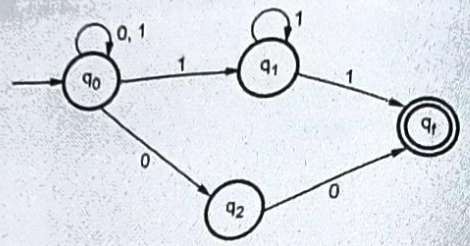
$\delta([q_0, q_1], 0) = (q_0, q_1, q_2)$ New state
$\delta([q_0, q_1], 1) = (q_0, q_2)$ New state
$\therefore \quad \delta([q_0, q_1, q_2], 0) = [q_0, q_1, q_2]$
$\delta([q_0, q_1, q_2], 1) = [q_0, q_2]$
$\delta([q_0, q_2], 0) = [q_0, q_1]$
$\delta([q_0, q_2]) = [q_0]$

As no new states are getting generated, we stop applying input transitions. The DFA can be

| State \ i/p | 0 | 1 |
|---|---|---|
| → $[q_0]$ | $[q_0, q_1]$ | $[q_0]$ |
| $[q_1]$ | $[q_2]$ | $[q_2]$ |
| $[q_2]$ | $\phi$ | $\phi$ |
| $[q_0, q_1]$ | $[q_0, q_1, q_2]$ | $[q_0, q_2]$ |
| * $[q_0, q_1, q_2]$ | $[q_0, q_1, q_2]$ | $[q_0, q_2]$ |
| * $[q_0, q_2]$ | $[q_0, q_1]$ | $[q_0]$ |



**Example 1.8.5** *Obtain DFA equivalent to NFA.* **SPPU : May-16, End Sem., Marks**




**Solution :**

**Step 1 :** We will first draw the transition table for given transition diagram.

| → State \ Input | 0 | 1 |
|---|---|---|
| $q_0$ | $\{q_0, q_2\}$ | $\{q_0, q_1\}$ |
| $q_1$ | $\phi$ | $\{q_1, q_f\}$ |
| $q_2$ | $\{q_f\}$ | $\phi$ |
| (*) $q_f$ | $\phi$ | $\phi$ |

**Step 2 :** Now we will obtain $\delta$ transitions.

For each state on each input.

$\delta([q_0], 0) = \{q_0, q_2\} = [q_0, q_2]$ **New state**

$\delta([q_0], 1) = \{q_0, q_1\} = [q_0, q_1]$ **New state**

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

$\delta([q_1], 1) = \{q_1, q_f\} = [q_1, q_f]$ **New state**

$\delta([q_2], 0) = \{q_f\} = [q_f]$

$\delta([q_2], 1) = \phi$

$\delta([q_f], 0) = \delta([q_f], 1) = \phi$

**Step 3 :** Now we will obtain $\delta$ transitions on newly obtained states in step 2,

$\delta([q_0, q_2], 0) = \{q_0, q_2, q_f\} = [q_0, q_2, q_f]$ **New state**

$\delta([q_0, q_2], 1) = \{q_0, q_1\} = [q_0, q_1]$ New state

$\delta([q_0, q_1], 0) = \{q_0, q_2\} = [q_0, q_2]$

$\delta([q_0, q_1], 1) = \{q_0, q_1, q_f\} = [q_0, q_1, q_f]$ **New state**





$\delta([q_1, q_f], 0) = \phi$

$\delta([q_1, q_f], 1) = [q_1, q_f]$

**Step 4 :** Now we will obtain $\delta$ transitions on newly obtained states in step 3,

$\delta([q_0, q_2, q_f], 0) = [q_0, q_2, q_f]$

$\delta([q_0, q_2, q_f], 1) = [q_0, q_1]$

$\delta([q_0, q_1, q_f], 0) = [q_0, q_2]$

$\delta([q_0, q_1, q_f], 1) = [q_0, q_1, q_f]$

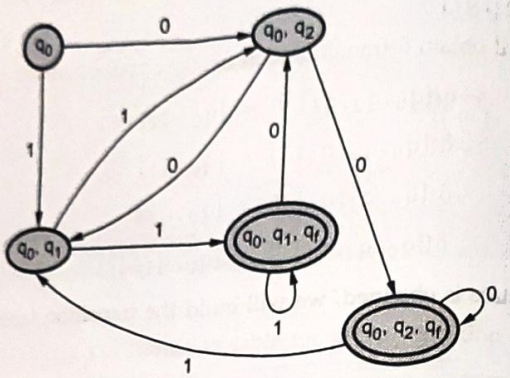
As now no new state is obtained, we will build the transition table as follows :

**Step 5 :**

| State \ Input | 0 | 1 |
|---|---|---|
| $[q_0]$ | $[q_0, q_2]$ | $[q_0, q_1]$ |
| $[q_1]$ | $\phi$ | $[q_1, q_f]$ |
| $[q_2]$ | $[q_f]$ | $\phi$ |
| $[q_f]$ | $\phi$ | $\phi$ |
| $[q_0, q_2]$ | $[q_0, q_2, q_f]$ | $[q_0, q_1]$ |
| $[q_0, q_1]$ | $[q_0, q_2]$ | $[q_0, q_1, q_f]$ |
| $[q_1, q_f]$ | $\phi$ | $[q_1, q_f]$ |
| $[q_0, q_2, q_f]$ | $[q_0, q_2, q_f]$ | $[q_0, q_1]$ |
| $[q_0, q_1, q_f]$ | $[q_0, q_2]$ | $[q_0, q_1, q_f]$ |

Clearly from above transition table, $q_1$, $q_2$, $q_1$, $q_f$ and $q_f$ are non-reachable states from start state. So we will eliminate them.

**Step 6 :** The DFA will then be as follows :




## 1.9 Conversion of NFA with Epsilon Moves to NFA without Epsilon

SPPU : May-12, Aug.-15, Oct.-16, Marks 8

In this method we try to remove all the ε transitions from given NFA. The meth will be

1. Find out all the ε transitions from each state from Q. That will be called ε - closure $\{q_i\}$ where $q_i \in Q$.
2. Then δ′ transitions can be obtained. The δ′ transitions means an ε-closure on moves.
3. Step - 2 is repeated for each input symbol and for each state of given NFA.
4. Using the resultant states the transition table for equivalent NFA without ε can b built.

**Rule for conversion**

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

where $\hat{\delta}(q,\varepsilon) = \varepsilon$ - closure$(q)$

Problems based on NFA with Epsilon to NFA without Epsilon

**Example 1.9.1** *Convert the given NFA with ε to NFA without ε.* SPPU : May-12, Marks 8

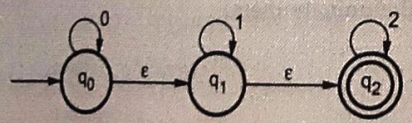



**Fig. 1.9.1**





**Solution :** We will first obtain ε - closure of each state i.e. we will find out ε - reachable states from current state.

*Hence*

$$\varepsilon \text{ - closure } (q_0) = \{q_0, q_1, q_2\}$$
$$\varepsilon \text{ - closure } (q_1) = \{q_1, q_2\}$$
$$\varepsilon \text{ - closure } (q_2) = \{q_2\}$$

As ε - closure $(q_0)$ means with null input (no input symbol) we can reach to $q_0$, $q_1$ or $q_2$. In a similar manner for $q_1$ and $q_2$ ε - closures are obtained. Now we will obtain δ′ transitions for each state on each input symbol.

$$\delta'(q_0, 0) = \varepsilon \text{ - closure } \left(\delta\left(\hat{\delta}(q_0, \varepsilon), 0\right)\right)$$
$$= \varepsilon \text{ - closure } \left(\delta(\varepsilon \text{ - closure}(q_0), 0)\right)$$
$$= \varepsilon \text{ - closure } (\delta(q_0, q_1, q_2), 0)$$
$$= \varepsilon \text{ - closure } (\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$$
$$= \varepsilon \text{ - closure } (q_0 \cup \phi \cup \phi)$$
$$= \varepsilon \text{ - closure } (q_0) = \{q_0, q_1, q_2\}$$

$$\delta'(q_0, 1) = \varepsilon \text{ - closure } \left(\delta\left(\hat{\delta}(q_0, \varepsilon), 1\right)\right)$$
$$= \varepsilon \text{ - closure } (\delta(q_0, q_1, q_2), 1)$$
$$= \varepsilon \text{ - closure } (\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$$
$$= \varepsilon \text{ - closure } (\phi \cup q_1 \cup \phi)$$
$$= \varepsilon \text{ - closure } (q_1)$$

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

$$\delta'(q_1, 0) = \varepsilon \text{ - closure } \left(\delta\left(\hat{\delta}(q_1, \varepsilon), 0\right)\right)$$
$$= \varepsilon \text{ - closure } \left(\delta(\varepsilon \text{ - closure}(q_1), 0)\right)$$
$$= \varepsilon \text{ - closure } (\delta(q_1, q_2), 0)$$
$$= \varepsilon \text{ - closure } (\delta(q_1, 0) \cup \delta(q_2, 0))$$
$$= \varepsilon \text{ - closure } (\phi \cup \phi)$$
$$= \varepsilon \text{ - closure } (\phi)$$
$$= \phi$$

$$\delta'(q_1, 1) = \varepsilon \text{ - closure } \left(\delta\left(\hat{\delta}(q_1, \varepsilon), 1\right)\right)$$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure$(q_1), 1))$

$= \varepsilon$ - closure $(\delta(q_1, q_2), 1)$

$= \varepsilon$ - closure $(\delta(q_1, 1) \cup \delta(q_2, 1))$

$= \varepsilon$ - closure $(q_1 \cup \phi)$

$= \varepsilon$ - closure $(q_1)$

$= \{q_1, q_2\}$

$\delta'(q_2, 0) = \varepsilon$ - closure $\left(\delta\left(\hat{\delta}(q_2, \varepsilon), 0\right)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(q_2), 0))$

$= \varepsilon$ - closure $(\delta(q_2, 0))$

$= \varepsilon$ - closure $(\phi)$

$= \phi$

$\delta'(q_2, 1) = \varepsilon$ - closure $\left(\delta\left(\hat{\delta}(q_2, \varepsilon), 1\right)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure$(q_2), 1))$

$= \varepsilon$ - closure $(\delta(q_2, 1))$

$= \varepsilon$ - closure $(\phi)$

$\delta'(q_2, 1) = \phi$

$\delta'(q_0, 2) = \varepsilon$ - closure $\left(\delta\left(\hat{\delta}(q_0, \varepsilon), 2\right)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure$(q_0), 2))$

$= \varepsilon$ - closure $(\delta(q_0, q_1, q_2), 2)$

$= \varepsilon$ - closure $(\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2))$

$= \varepsilon$ - closure $(\phi \cup \phi \cup q_2)$

$= \varepsilon$ - closure $(q_2)$

$= \{q_2\}$

$\delta'(q_1, 2) = \varepsilon$ - closure $\left(\delta\left(\hat{\delta}(q_1, \varepsilon), 2\right)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(q_1), 2))$

$= \varepsilon$ - closure $(\delta(q_1, q_2), 2)$

$= \varepsilon$ - closure $(\delta(q_1, 2) \cup \delta(q_2, 2))$

$= \varepsilon$ - closure $(\phi \cup q_2)$





$= \{q_2\}$

$\delta'(q_2, 2) = \varepsilon$ - closure $\left(\delta\left(\hat{\delta}(q_2, \varepsilon), 2\right)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure$(q_2), 2))$

$= \varepsilon$ - closure $(\delta(q_2, 2))$

$= \varepsilon$ - closure $(q_2)$

$= \{q_2\}$

Now we will summarize all the computed $\delta'$ transitions -

$\delta'(q_0, 0) = \{q_0, q_1, q_2\}$, $\delta'(q_0, 1) = \{q_1, q_2\}$, $\delta'(q_0, 2) = \{q_2\}$

$\delta'(q_1, 0) = \phi$, $\delta'(q_1, 1) = \{q_1, q_2\}$, $\delta'(q_1, 2) = \{q_2\}$

$\delta'(q_2, 0) = \phi$, $\delta'(q_2, 1) = \phi$, $\delta'(q_2, 2) = \{q_2\}$

From this we can write the transition table as -

| State \ Input | 0 | 1 | 2 |
|---|---|---|---|
| $q_0$ | $\{q_0, q_1, q_2\}$ | $\{q_1, q_2\}$ | $\{q_2\}$ |
| $q_1$ | $\phi$ | $\{q_1, q_2\}$ | $\{q_2\}$ |
| $q_2$ | $\phi$ | $\phi$ | $\{q_2\}$ |

The NFA will be -

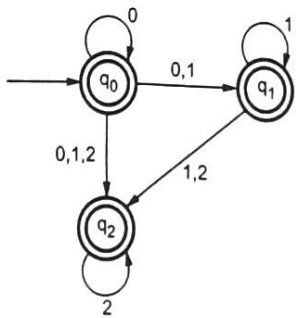



**Fig. 1.9.2**

Here $q_0$, $q_1$ and $q_2$ is a final state because $\varepsilon$ - closure $(q_0)$, $\varepsilon$ - closure $(q_1)$ and $\varepsilon$ - closure $(q_2)$ contains final state $q_2$.

**Example 1.9.2** *Convert the following NFA with ε to NFA without ε.*

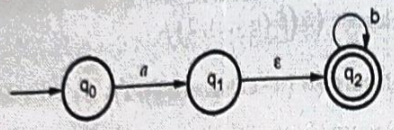



**Fig. 1.9.3**

**Solution :** We will first obtain ε - closures of $q_0$, $q_1$ and $q_2$ as follows.

ε - closure $(q_0) = \{q_0\}$

ε - closure $(q_1) = \{q_1, q_2\}$

ε - closure $(q_2) = \{q_2\}$

Now the δ′ transitions on each input symbol is obtained as

$$\delta'(q_0,a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_0,\varepsilon),a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_0),a\right)\right)$$

$$= \varepsilon\text{-closure}(\delta(q_0,a))$$

$$= \varepsilon\text{-closure}(q_1)$$

$$= \{q_1,q_2\}$$

$$\delta'(q_0,b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_0,\varepsilon),b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_0),b\right)\right)$$

$$= \varepsilon\text{-closure}(\delta(q_0,b))$$

$$= \phi$$

$$\delta'(q_1,a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_1,\varepsilon),a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_1),a\right)\right)$$

$$= \varepsilon\text{-closure}(\delta(q_1,q_2),a)$$

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

$$= \varepsilon\text{-closure}(\phi\cup\phi)$$

$$= \phi$$

$$\delta'(q_1,b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_1,\varepsilon),b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_1),b\right)\right)$$

$$= \varepsilon\text{-closure}(\delta(q_1,q_2),b)$$





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

$$= \varepsilon\text{-closure}(\phi\cup q_2)$$

$$= \varepsilon\text{-closure}(q_2)$$

$$= \{q_2\}$$

$$\delta'(q_2,a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_2,\varepsilon),a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_2),a\right)\right)$$

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

$$= \varepsilon\text{-closure}(\phi)$$

$$= \phi$$

$$\delta'(q_2,b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_2,\varepsilon),b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_2),b\right)\right)$$

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

$$= \varepsilon\text{-closure}(q_2)$$

$$= \{q_2\}$$

The transition table can be -

| State \ Input | a | b |
|---|---|---|
| → $q_0$ | $\{q_1, q_2\}$ | $\phi$ |
| $q_1$ | $\phi$ | $\{q_2\}$ |
| $q_2$ | $\phi$ | $\{q_2\}$ |

States $q_1$ and $q_2$ becomes the final as ε - closures of $q_1$ and $q_2$ contains the final state $q_2$. The NFA can be shown by following transition diagram -

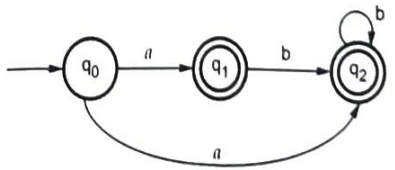
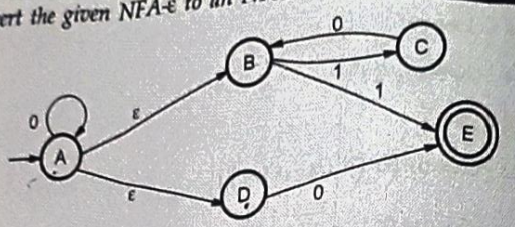



**Fig. 1.9.4**

**Example 1.9.3** *Convert the given NFA-ε to an NFA.* **SPPU : Aug.-15, Marks 8**




**Solution :**

We will obtain ε-closure of each state

ε - closure (A) = { A, B, D }

ε - closure (B) = { B }

ε - closure (C) = { C }

ε - closure (D) = { D }

ε - closure (E) = { E }

Now δ′ transitions for each state each input

$\delta'(A,0) = \varepsilon$ - closure $(\delta(\hat{\delta}(A,\varepsilon),0))$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure$(A),0))$

$= \varepsilon$ - closure $(\delta(A,B,D),0))$

$= \varepsilon$ - closure $(\delta(A,0) \cup \delta(B,0) \cup \delta(D,0))$

$= \varepsilon$ - closure $(A \cup \phi \cup E)$

$= \varepsilon$ - closure $(A) \cup \varepsilon$ - closure $(E)$

$= \{A, B, D\} \cup \{E\}$

$\delta'(A,0) = \{A, B, D, E\}$

$\delta'(A,1) = \varepsilon$ - closure $(\delta(\hat{\delta}(A,\varepsilon),1))$

$\delta'(A,1) = \varepsilon$ - closure $(\delta(\varepsilon -$ closure $(A),1))$

$= \varepsilon$ - closure $(\delta(A,B,D),1))$

$= \varepsilon$ - closure $(\delta(A,1) \cup \delta(B,1) \cup \delta(D,1))$

$= \varepsilon$ - closure $(\phi \cup (C,E) \cup \phi)$

$= \varepsilon$ - closure $(C \cup E)$

$= \varepsilon$ - closure $(C) \cup \varepsilon$ - closure $(E)$

$= \{C, E\}$

$\delta'(B,0) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(B),0))$

$= \varepsilon$ - closure $(\delta(B,0))$





$= \varepsilon$ - closure $(\phi)$

$= \phi$

$\delta'(B,1) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(B),1))$

$= \varepsilon$ - closure $(\delta(B,1))$

$= \varepsilon$ - closure $(C,E)$

$= \varepsilon$ - closure $(C) \cup \varepsilon$ - closure $(E)$

$\delta'(B,1) = \{C, E\}$

$\delta'(C,0) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(C),0))$

$= \varepsilon$ - closure $(\delta(C,0))$

$= \varepsilon$ - closure $(B)$

$\delta'(C,0) = B$

$\delta'(C,1) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(C),1))$

$= \varepsilon$ - closure $(\delta(C,1))$

$\delta'(C,1) = \phi$

$\delta'(D,0) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(D),0))$

$= \varepsilon$ - closure $(\delta(D,0))$

$= \varepsilon$ - closure $(E)$

$\delta'(D,0) = E$

$\delta'(D,1) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(D),1))$

$= \varepsilon$ - closure $(\delta(D,1))$

$= \varepsilon$ - closure $(\phi)$

$\delta'(D,1) = \phi$

$\delta'(E,0) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(E),0))$

$= \varepsilon$ - closure $(\delta(E,0))$

$= \varepsilon$ - closure $(\phi)$

$\delta'(E,0) = \phi$

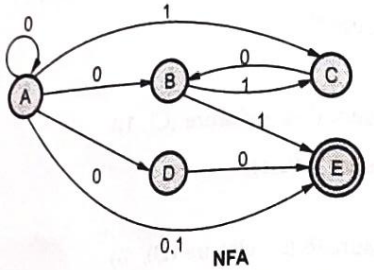
$\delta'(E,1) = \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(E),1))$

$= \varepsilon$ - closure $(\delta(E,1))$

$= \varepsilon$ - closure $(\phi)$

$\delta'(E,1) = \phi$

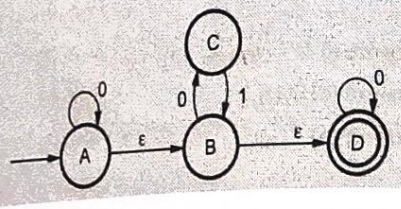
| State | Input 0 | 1 |
|---|---|---|
| A | (A, B, D, E) | (C, E) |
| B | ϕ | (C, E) |
| C | B | ϕ |
| D | E | ϕ |
| E | ϕ | ϕ |




NFA

**Example 1.9.4** *Convert the following NFA-ε to equivalent NFA.*

**SPPU : Oct.-16, In Sem. Marks**




**Solution :**

**Step 1 :** We will first obtain ε - closure of each state.

ε - closure(A) = {A, B, D}

ε - closure(B) = {B, D}

ε - closure(D) = {D}

ε - closure(C) = {C}

**Step 2 :** Now obtain the δ′ transitions

$\delta'(A, 0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(A, \varepsilon), 0))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(A), 0))$





$= \varepsilon\text{-closure}(\delta(A, B, D), 0))$

$= \varepsilon\text{-closure}(\delta(A, 0) \cup \delta(B, 0) \cup \delta(D, 0))$

$= \varepsilon\text{-closure}(A \cup C \cup D)$

$= \varepsilon\text{-closure}(A) \cup \varepsilon\text{-closure}(C) \cup \varepsilon\text{-closure}(D)$

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

$\delta'(A, 0) = \{A, B, C, D\}$

$\delta'(A, 1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(A, \varepsilon), 1))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(A), 1))$

$= \varepsilon\text{-closure}(\delta(A, B, D), 1))$

$= \varepsilon\text{-closure}(\delta(A, 1) \cup \delta(B, 1) \cup \delta(D, 1))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi)$

$\delta'(A, 1) = \phi$

$\delta'(B, 0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(B, \varepsilon), 0))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(B), 0))$

$= \varepsilon\text{-closure}(\delta(B, D), 0))$

$= \varepsilon\text{-closure}(\delta(B, 0) \cup \delta(D, 0))$

$= \varepsilon\text{-closure}(C \cup D)$

$= \varepsilon\text{-closure}(C) \cup \varepsilon\text{-closure}(D)$

$\delta'(B, 0) = \{C, D\}$

$\delta'(B, 1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(B, \varepsilon), 1))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(B), 1))$

$= \varepsilon\text{-closure}(\delta((B, D), 1))$

$= \varepsilon\text{-closure}(\delta(B, 1) \cup \delta(D, 1))$

$= \varepsilon\text{-closure}(\phi \cup \phi)$

$\delta'(B, 1) = \phi$

$\delta'(C, 0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(C, \varepsilon), 0))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(C), 0))$

$= \varepsilon\text{-closure}(\delta(C, 0))$

$= \varepsilon\text{-closure}(\phi)$

$\delta'(C, 0) = \phi$

$\delta'(C, 1) = \varepsilon$ - closure $(\delta(\hat{\delta}(C, \varepsilon), 1))$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(C), 1))$

$= \varepsilon$ - closure $(\delta(C, 1))$

$= \varepsilon$ - closure $(B)$

$\delta'(C, 1) = \{B, D\}$

$\delta'(D, 0) = \varepsilon$ - closure $(\delta(\hat{\delta}(D, \varepsilon), 0))$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(D), 0))$

$= \varepsilon$ - closure $(\delta(D, 0))$

$= \varepsilon$ - closure $(D)$

$\delta'(D, 0) = \{D\}$

$\delta'(D, 1) = \varepsilon$ - closure $(\delta(\hat{\delta}(D, \varepsilon), 1))$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(D), 1))$

$= \varepsilon$ - closure $(\delta(D, 1))$

$= \varepsilon$ - closure $(\phi)$

$\delta'(D, 1) = \{\phi\}$

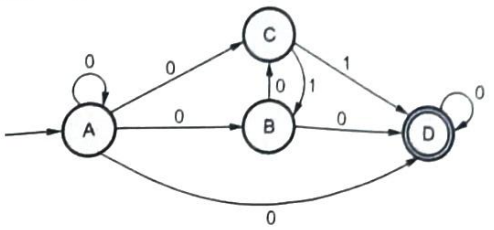
**Step 3 :** From above computations, we can draw the transition table for all transitions.

| State \ Input | 0 | 1 |
|---|---|---|
| → A | {A, B, C, D} | $\phi$ |
| B | {C, D} | $\phi$ |
| C | $\phi$ | {B, D} |
| (D) | {D} | $\phi$ |





**Step 4 :** The transition diagram for NFA will be




## 1.10 Conversion of NFA with Epsilon Moves to DFA

SPPU : Oct.-16, Dec.-18, May-09,12, Marks 8

**Method for converting NFA with $\varepsilon$ to DFA**

**Step 1 :** Consider $M = (Q, \Sigma, \delta, q_0, F)$ is a NFA with $\varepsilon$. We have to convert this NFA with $\varepsilon$ to equivalent DFA denoted by

$$M_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$$

Then obtain,

$\varepsilon$ - closure$(q_0) = \{p_1, p_2, p_3, \ldots p_n\}$ then $[p_1, p_2, p_3, \ldots p_n]$ becomes a start state of DFA.

Now $[p_1, p_2, p_3, \ldots p_n] \in Q_D$

**Step 2 :** We will obtain $\delta$ transitions on $[p_1, p_2, p_3, \ldots p_n]$ for each input.

$$\delta_D([p_1, p_2, \ldots p_n], a) = \varepsilon \text{ - closure}(\delta(p_1, a) \cup \delta(p_2, a) \cup \ldots \delta(p_n, a))$$

$$= \bigcup_{i=1}^{n} \varepsilon \text{ - closure} \delta(p_i, a)$$

where a is input $\in \Sigma$.

**Step 3 :** The states obtained $[p_1, p_2, p_3, \ldots p_n] \in Q_D$. The states containing final state in $p_i$ is a final state in DFA.

### Problems based on NFA with Epsilon to DFA

**Example 1.10.1** *Convert the given NFA into its equivalent DFA - or construct DFA that accepts the language represented by 0*1*2*, Make use of NFA.* SPPU : Oct.-16, In Sem.; Dec.-18, End Sem, Marks 8

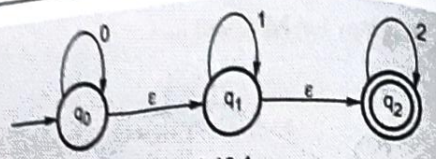



**Fig. 1.10.1**

**Solution :** Let us obtain ε - closure of each state.

ε - closure $(q_0) = \{q_0, q_1, q_2\}$

ε - closure $(q_1) = \{q_1, q_2\}$

ε - closure $(q_2) = \{q_2\}$

Now we will obtain δ' transition. Let ε - closure $(q_0) = \{q_0, q_1, q_2\}$ call it as state A.

$$\delta'(A,0) = \varepsilon\text{ - closure } \{\delta((q_0, q_1, q_2), 0)\}$$
$$= \varepsilon\text{ - closure } \{\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0)\}$$
$$= \varepsilon\text{ - closure } \{q_0\}$$
$$= \{q_0, q_1, q_2\} \quad \text{i.e. state A}$$

$$\delta'(A,1) = \varepsilon\text{ - closure } \{\delta((q_0, q_1, q_2), 1)\}$$
$$= \varepsilon\text{ - closure } \{\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1)\}$$
$$= \varepsilon\text{ - closure } \{q_1\}$$
$$= \{q_1, q_2\} \quad \text{Call it as state B}$$

$$\delta'(A,2) = \varepsilon\text{ - closure } \{\delta((q_0, q_1, q_2), 2)\}$$
$$= \varepsilon\text{ - closure } \{\delta(q_0,2) \cup \delta(q_1,2) \cup \delta(q_2,2)\}$$
$$= \varepsilon\text{ - closure } \{q_2\}$$
$$= \{q_2\} \quad \text{Call it as state C.}$$

Thus we have obtained

$\delta'(A,0) = A$

$\delta'(A,1) = B$

$\delta'(A,2) = C$





e.

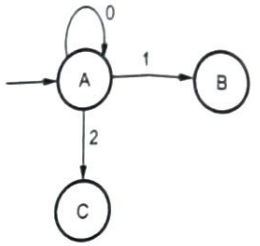



**Fig. 1.10.2**

Now we will find transitions on states B and C for each input.

Hence

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

$$\delta'(B,1) = \varepsilon\text{ - closure } \{\delta(q_1, q_2), 1\}$$
$$= \varepsilon\text{ - closure } \{\delta(q_1,1) \cup \delta(q_2,1)\}$$
$$= \varepsilon\text{ - closure } \{q_1\}$$
$$= \{q_1, q_2\} \text{ i.e state B itself.}$$

$$\delta'(B,2) = \varepsilon\text{ - closure } \{\delta(q_1, q_2), 2\}$$
$$= \varepsilon\text{ - closure } \{\delta(q_1,2) \cup \delta(q_2,2)\}$$
$$= \varepsilon\text{ - closure } \{q_2\}$$
$$= \{q_2\} \text{ i.e state C.}$$

Hence

$\delta'(B,0) = \phi$

$\delta'(B,1) = B$

$\delta'(B,2) = C$

The partial transition diagram will be

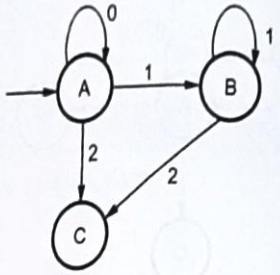



Fig. 1.10.3

Now we will obtain transitions for C :

$$\delta'(C,0) = \varepsilon\text{-closure}\{\delta(q_2,0)\}$$
$$= \varepsilon\text{-closure}\{\phi\}$$
$$= \phi$$
$$\delta'(C,1) = \varepsilon\text{-closure}\{\delta(q_2,1)\}$$
$$= \varepsilon\text{-closure}\{\phi\}$$
$$= \phi$$
$$\delta'(C,2) = \varepsilon\text{-closure}\{\delta(q_2,2)\}$$
$$= q_2$$

Hence the DFA is

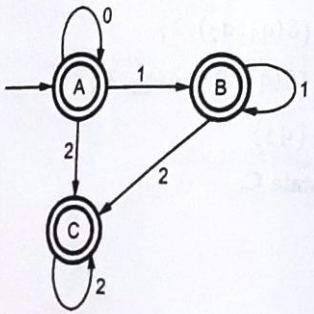



Fig. 1.10.4

As $A = \{q_0,q_1,q_2\}$ in which final state $q_2$ lies hence A is final state in $B = \{q_1$ the state $q_2$ lies hence B is also final state in $C = \{q_2\}$, the state $q_2$ lies hence C is a final state.





**Example 1.10.2** *Convert the given NFA with ε to its equivalent DFA.*

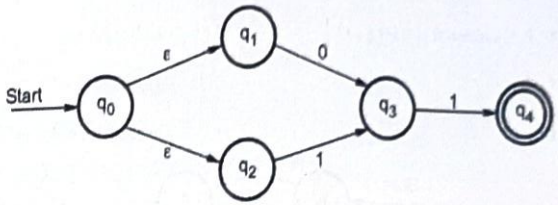



Fig. 1.10.5

**Solution :**

$$\varepsilon\text{-closure}\{q_0\} = \{q_0,q_1,q_2\}$$
$$\varepsilon\text{-closure}\{q_1\} = \{q_1\}$$
$$\varepsilon\text{-closure}\{q_2\} = \{q_2\}$$
$$\varepsilon\text{-closure}\{q_3\} = \{q_3\}$$
$$\varepsilon\text{-closure}\{q_4\} = \{q_4\}$$

Now, Let $\varepsilon\text{-closure}\{q_0\} = \{q_0,q_1,q_2\}$ be state A.

Hence

$$\delta'(A,0) = \varepsilon\text{-closure}\{\delta((q_0,q_1,q_2),0)\}$$
$$= \varepsilon\text{-closure}\{\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0)\}$$
$$= \varepsilon\text{-closure}\{q_3\}$$
$$= \{q_3\} \text{ call it as state B.}$$
$$\delta'(A,1) = \varepsilon\text{-closure}\{\delta((q_0,q_1,q_2),1)\}$$
$$= \varepsilon\text{-closure}\{\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1)\}$$
$$= \varepsilon\text{-closure}\{q_3\} = q_3 = B.$$

The partial DFA will be

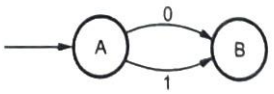



Fig. 1.10.6

Now,

$$\delta'(B,0) = \varepsilon\text{-closure}\{\delta(q_3,0)\}$$
$$= \phi$$
$$\delta'(B,1) = \varepsilon\text{-closure}\{\delta(q_3,1)\}$$
$$= \varepsilon\text{-closure}\{q_4\}$$
$$= \{q_4\} \text{ i.e. state C}$$
$$\delta'(C,0) = \varepsilon\text{-closure}\{\delta(q_4,0)\}$$

$\therefore \quad \delta(B, b) = D.$

$\delta(B, c) = \varepsilon$ - closure $(\delta(B, c))$

$= \varepsilon$ - closure$(\delta(p, q), c)$

$= \varepsilon$ - closure $(\delta(p, c) \cup \delta(q, c))$

$= \varepsilon$ - closure $(r \cup \phi)$

$= \varepsilon$ - closure $(r)$

$= \{q, r\}$

$\delta(B, c) = C$

**State C = {q, r}**

$\delta(C, a) = \varepsilon$ - closure $(\delta(C, a))$

$= \varepsilon$ - closure$(\delta(q, r), a)$

$= \varepsilon$ - closure $(\delta(q, a) \cup \delta(r, a))$

$= \varepsilon$ - closure $(q \cup r)$

$= \varepsilon$ - closure $(q) \cup \varepsilon$ - closure$(r)$

$= \{p, q\} \cup \{q, r\}$

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(C, a) = D$

$\delta(C, b) = \varepsilon$ - closure $(\delta(C, b))$

$= \varepsilon$ - closure$(\delta(q, r), b)$

$= \varepsilon$ - closure $(\delta(q, b) \cup \delta(r, b))$

$= \varepsilon$ - closure $(r \cup \phi)$

$= \varepsilon$ - closure $(r)$

$= \{q, r\}$ i.e. state C.

$\therefore \quad \delta(C, b) = C$

$\delta(C, c) = \varepsilon$ - closure $(\delta(C, c))$

$= \varepsilon$ - closure$(\delta(q, r), c)$

$= \varepsilon$ - closure $(\delta(q, c) \cup \delta(r, c))$

$= \varepsilon$ - closure $(\phi \cup p)$

$= \varepsilon$ - closure $(p)$





$= \{p\}$ i.e. state A.

$\therefore \quad \delta(C, c) = A$

**State D = {p, q, r}**

$\delta(D, a) = \varepsilon$ - closure $(\delta(D, a))$

$= \varepsilon$ - closure$(\delta(p, q, r), a)$

$= \varepsilon$ - closure $(\delta(p, a) \cup \delta(q, a) \cup \delta(r, a))$

$= \varepsilon$ - closure $(p \cup q \cup r)$

$= \varepsilon$ - closure $(p) \cup \varepsilon$ - closure$(q) \cup \varepsilon$ - closure$(r)$

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(D, a) = D$

$\delta(D, b) = \varepsilon$ - closure $(\delta(D, b))$

$= \varepsilon$ - closure$(\delta(p, q, r), b)$

$= \varepsilon$ - closure $(\delta(p, b) \cup \delta(q, b) \cup \delta(r, b))$

$= \varepsilon$ - closure $(q \cup r \cup \phi)$

$= \varepsilon$ - closure$(q, r)$

$= \varepsilon$ - closure $(q) \cup \varepsilon$ - closure$(r)$

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(D, b) = D$

$\delta(D, c) = \varepsilon$ - closure $(\delta(D, c))$

$= \varepsilon$ - closure$(\delta(p, q, r), c)$

$= \varepsilon$ - closure $(\delta(p, c) \cup \delta(q, c) \cup \delta(r, c))$

$= \varepsilon$ - closure $(r \cup \phi \cup p)$

$= \varepsilon$ - closure $(r) \cup \varepsilon$ - closure$(p)$

$= \{q, r\} \cup \{p\}$

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(D, c) = D$

The transition table from above calculations can be obtained as

| State | Input a | b | c |
|---|---|---|---|
| → A | A | B | C |
| B | B | D | C |
| (C) | D | C | A |
| (D) | D | D | D |

As state A = {p} it is a start state and states C and D contain final state r, hence are final states.

The transition diagram for the DFA is

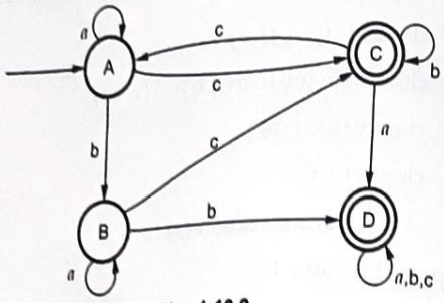
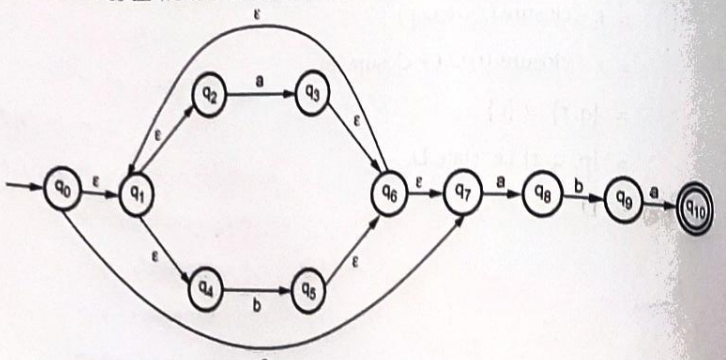



Fig. 1.10.9

**Example 1.10.4** *Construct a NFA that accepts the set of all strings over {a, b} ending in ab. Use this NFA to construct DFA accepting the same set of strings.*

**SPPU : May-09, Marks**

**Solution :** First of all we will construct NFA with ε as follows.








Now we will convert this NFA with ε to DFA.

**Let Start state**

ε - closure $(q_0) = \{q_0, q_1, q_2, q_4, q_7\}$ → **Call it as state A**

Now we will find a and b transitions on state A.

$\delta'(A, a)$ = ε - closure $(\delta(A, a))$

= ε - closure $(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_4, a) \cup \delta(q_7, a))$

= ε - closure $(\phi \cup \phi \cup q_3 \cup \phi \cup q_8)$

= ε - closure $(\{q_3, q_8\})$

= $(q_3, q_6, q_7, q_1, q_2, q_4, q_8)$ sorting it

= $(q_1, q_2, q_3, q_4, q_6, q_7, q_8)$ → **Call it as B**

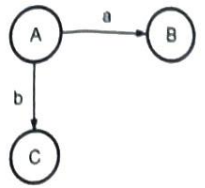
$\delta'(A, b)$ = ε - closure $(\delta(A, b))$

= ε - closure $(q_5)$

= $(q_5, q_6, q_1, q_2, q_4, q_7)$

= $(q_1, q_2, q_4, q_5, q_6, q_7)$ → **Call it as C**

The partial DFA can be




Now consider state B for input a and b transitions.

$\delta'(B, a)$ = ε - closure $(\delta(B, a))$

= ε - closure $(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a)$
$\cup \delta(q_6, a) \cup \delta(q_7, a) \cup \delta(q_8, a)$

= ε - closure $(q_3, q_8)$

= {1, 2, 3, 4, 6, 7, 8}

i.e. B

$\delta'(B, b)$ = ε - closure $(\delta(B, b))$

= ε - closure $(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_4, b)$
$\cup \delta(q_6, b) \cup \delta(q_7, b) \cup \delta(q_8, b))$

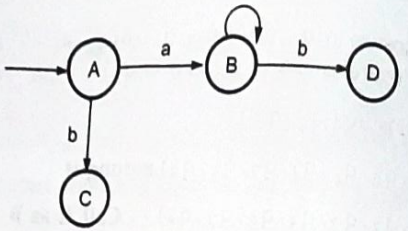
$= \varepsilon$ - closure $(q_5, q_9)$

$= (q_5, q_6, q_7, q_1, q_2, q_4, q_9)$ Sorting these states.

$= (q_1, q_2, q_4, q_5, q_6, q_7, q_9) \rightarrow$ **Call it as D state**

$= D$

The DFA partially can be drawn as




Now consider state C for input a and b transitions

$\delta'(C, a) = \varepsilon$ - closure $(\delta(C, a))$

$= \varepsilon$ - closure $(\delta(q_1, q_2, q_4, q_5, q_6, q_7), a)$

$= \varepsilon$ - closure $(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_4, a) \cup \delta(q_5, a) \cup \delta(q_6, a) \cup \delta(q_7, a))$

$= \varepsilon$ - closure $(\phi \cup q_3 \cup \phi \cup \phi \cup \phi \cup q_8)$

$= \varepsilon$ - closure $(q_3, q_8)$

$= (q_1, q_2, q_3, q_4, q_6, q_7, q_8)$

i.e. B state

$\delta'(C, b) = \varepsilon$ - closure $(\delta(c, b))$

$= \varepsilon$ - closure $(\delta((q_1, q_2, q_4, q_5, q_6, q_7), b))$

$= \varepsilon$ - closure $(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_4, b) \cup \delta(q_5, b) \cup \delta(q_6, b) \cup \delta(q_7, b))$

$= \varepsilon$ - closure $(\phi \cup \phi \cup q_5 \cup \phi \cup \phi \cup \phi)$

$= \varepsilon$ - closure $(q_5)$

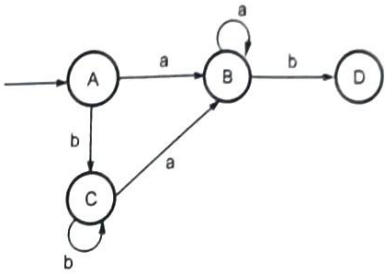
$= (q_1, q_2, q_4, q_5, q_6, q_7)$

i.e. C state





Partially DFA can be




Now consider state D for transition on input a and b

$\delta'(D, a) = \varepsilon$ - closure $(\delta((q_1, q_2, q_4, q_5, q_6, q_7, q_9), a))$

$= \varepsilon$ - closure $(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_4, a) \cup \delta(q_5, a) \cup \delta(q_6, a) \cup \delta(q_7, a) \cup \delta(q_9, a))$

$= \varepsilon$ - closure $(\phi \cup q_3 \cup \phi \cup \phi \cup \phi \cup q_8 \cup q_{10})$

$= \varepsilon$ - closure $(q_3, q_8, q_{10})$

$= (q_3, q_6, q_1, q_2, q_4, q_8, q_{10})$ Sorting it

$= (q_1, q_2, q_3, q_4, q_6, q_8, q_{10}) \rightarrow$ Call it as E.

$\delta'(D, b) = \varepsilon$ - closure $(\delta((q_1, q_2, q_4, q_5, q_6, q_7, q_9), b))$

$= \varepsilon$ - closure $(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_4, b) \cup \delta(q_5, b) \cup \delta(q_6, b) \cup \delta(q_7, b) \cup \delta(q_9, b))$

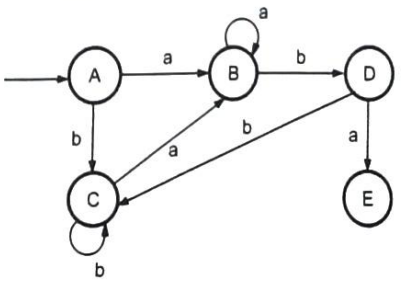
$= \varepsilon$ - closure $(\phi \cup \phi \cup q_5 \cup \phi \cup \phi \cup \phi)$

$= \varepsilon$ - closure $(q_5)$

$= (q_1, q_2, q_4, q_5, q_6, q_7)$

i.e. state C only

Partially the DFA can be constructed as



Now consider state E for transitions of input a and b.

$\delta'(E, a) = \varepsilon$ - closure $(\delta((q_1, q_2, q_3, q_4, q_6, q_8, q_{10}), a))$

$= \varepsilon$ - closure $(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a)$
$\cup \delta(q_6, a) \cup \delta(q_8, a) \cup \delta(q_{10}, a))$

$= \varepsilon$ - closure $(\phi \cup q_3 \cup \phi \cup \phi \cup \phi \cup \phi \cup \phi)$

$= \varepsilon$ - closure $(q_3)$

$= (q_1, q_2, q_3, q_4, q_6, q_7) \rightarrow$ Call it as state F.

$\delta'(E, b) = \varepsilon$ - closure $(\delta((q_1, q_2, q_3, q_4, q_6, q_8, q_{10}), b))$

$= \varepsilon$ - closure $(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b)$
$\cup \delta(q_4, b) \cup \delta(q_6, b) \cup \delta(q_8, b) \cup \delta(q_{10}, b))$

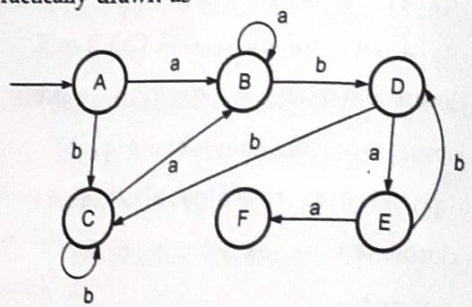
$= \varepsilon$ - closure $(\phi \cup \phi \cup \phi \cup q_5 \cup \phi \cup q_9 \cup \phi)$

$= \varepsilon$ - closure $(q_5, q_9)$

$= (q_1, q_2, q_4, q_5, q_6, q_7, q_9)$

i.e. state D.

The DFA can be practically drawn as -




Now process state F.

$\delta'(F, a) = \varepsilon$ - closure $(\delta((q_1, q_2, q_3, q_4, q_6, q_7), a))$

$= \varepsilon$ - closure $(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup$
$\delta(q_4, a) \cup \delta(q_6, a) \cup \delta(q_7, a))$

$= \varepsilon$ - closure $(\phi \cup q_3 \cup \phi \cup \phi \cup \phi \cup q_8)$

$= \varepsilon$ - closure $(q_3, q_8)$

$= (q_1, q_2, q_3, q_4, q_6, q_7, q_8)$

i.e. state B

$\delta'(F, b) = \varepsilon$ - closure $(\delta((q_1, q_2, q_3, q_4, q_6, q_7), b))$

$= \varepsilon$ - closure $(\delta(q_1, b) \cup \delta(q_2, b) \cup (q_3, b) \cup \delta(q_4, b)$
$\cup \delta(q_6, b) \cup \delta(q_7, b))$





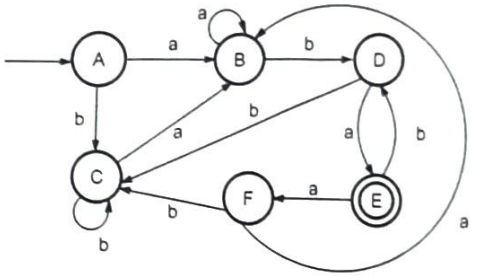
$= \varepsilon$ - closure $(\phi \cup \phi \cup \phi \cup q_5 \cup \phi \cup \phi)$

$= \varepsilon$ - closure $(q_5)$

$= (q_1, q_2, q_4, q_5, q_6, q_7)$

= State C

Finally the DFA can be




As state E contains final state $q_{10}$, in the resultant DFA, E is a final state.

## 1.11 Difference between NFA and DFA

SPPU : May-09,10, Aug.-17, Marks 2

| Sr. No. | NFA | DFA |
|---|---|---|
| 1. | NFA stands for Non deterministic Finite Automata. | DFA stands for Deterministic Finite Automata |
| 2. | For every input symbol there can be more than one state transition. | For every input symbol there can be only one state transition. |
| 3. | NFA can use empty string transition. | DFA can not use empty string transition. |
| 4. | Construction of NFA is simple | Construction of DFA is difficult |

**University Question**

1. *What is the difference between NFA and DFA ?*

SPPU : May-09, 10, Aug.-17(In Sem), Marks 2

## 1.12 Minimization of FA

SPPU : Dec.-12, Marks 8

The minimization of FSM means reducing the number of states from given FA. Thus we get the FSM with redundant states after minimizing the FSM.

While minimizing FSM we first find out which two states are equivalent represent those two states by one representative state.

**Definition of equivalent states -**

The two states $q_1$ and $q_2$ are equivalent if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final st both of them are non final states for all $x \in \Sigma^*$ ($\Sigma^*$ indicate any string of any leng can minimize the given FSM by finding equivalent states.

**Method for Construction of Minimum State Automata :**

**Step 1 :** We will create a set $\pi_0$ as $\pi_0 = \{ Q_1^0, Q_2^0 \}$ where $Q_1^0$ is set of all final and $Q_2^0 = Q - Q_1^0$ where Q is a set of all the states in DFA.

**Step 2 :** Now we will construct $\pi_{K+1}$ from $\pi_K$. Let $Q_i^K$ be any subset in $\pi_K$. and $q_2$ are in $Q_i^K$ they are (K + 1) equivalent provided $\delta(q_1, a)$ and $\delta($ are K equivalent. Find out whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are residing same equivalence class $\pi_K$. Then it is said that $q_1$ and $q_2$ are (K equivalent. Thus from $Q_i^K$ we create (K + 1) equivalence classes. R step 2 for every $Q_i^K$ in $\pi_K$ and obtain all the elements of $\pi_{K+1}$.

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

**Step 4 :** Then replace all the equivalent states in one equivalence clas representative state. This helps in minimizing the given DFA.

Let us understand this method with the help of some examples.

**Example 1.12.1** *Construct the minimum state automaton for the following transition diagr*

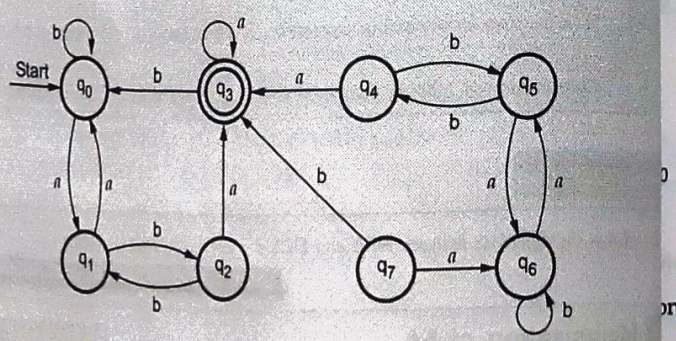



**Fig. 1.12.1** **SPPU : Dec.-12, Marks**

**Solution :** We will first construct a transition table for the given DFA.





We will start constructing equivalence classes.

| | State \ Input | a | b |
|---|---|---|---|
| → | $q_0$ | $q_1$ | $q_0$ |
| | $q_1$ | $q_0$ | $q_2$ |
| - | $q_2$ | $q_3$ | $q_1$ |
| - | $(q_3)$ | $q_3$ | $q_0$ |
| - | $q_4$ | $q_3$ | $q_5$ |
| | $q_5$ | $q_6$ | $q_4$ |
| | $q_6$ | $q_5$ | $q_6$ |
| - | $q_7$ | $q_6$ | $q_3$ |

**tep 1 :** We will first find 0 - equivalence. For that purpose we will create two sets - ne set containing all the final states and other set containing all the non-final states.

$$0 \text{ - Equivalent } = \{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}, \{q_3\}$$

Now will check equivalence among all the states of $\{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}$ by eans of $\delta$ mapping functions

$\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_0$ — Belong to same set

$\delta(q_0, b) = q_0$, $\delta(q_1, b) = q_2$ — Belong to same set

$q_0$ and $q_1$ are equivalent. In this way we will consider every pair from the set $, q_1, q_2, q_4, q_5, q_6, q_7\}$. Now consider pair $\{q_0, q_2\}$

$\delta(q_0, a) = q_1$, $\delta(q_2, a) = q_3$ ← Belong to different sets

$\therefore q_0$ and $q_2$ are not equivalent

ntinuing in this way we get -

$$1 \text{ - equivalence } = \{q_0, q_1, q_5, q_6\}\{q_2, q_4\}, \{q_3\}, \{q_7\}$$

**Step 2 :** Now will find 2 - equivalence from the sets obtained in 1 - equivalence.

Consider the set $\{q_0, q_1, q_5, q_6\}$ first. We will compare $q_0$ with $q_1, q_5, q_6$. Then we

$\delta(q_0, a) = q_1$ $\quad$ $\delta(q_0, b) = q_0$

$\delta(q_1, a) = q_0$ $\quad$ $\delta(q_1, b) = q_2$

Belong to same set $\quad$ Belong to different set

$\therefore$ $q_0$ and $q_1$ now does not belong to same set.

Now compare states $q_0, q_5$ states.

$\delta(q_0, a) = q_1$ $\quad$ $\delta(q_0, b) = q_0$ ← Belong to different sets

$\delta(q_5, a) = q_0$ $\quad$ $\delta(q_5, b) = q_4$

$\therefore$ $q_0, q_5$ are not equivalent. Now we will compare $q_1$ and $q_5$.

$\delta(q_1, a) = q_0$ $\quad$ $\delta(q_1, b) = q_2$

$\delta(q_5, a) = q_6$ $\quad$ $\delta(q_5, b) = q_4$

Belong to same set $\quad$ Belong to same set

That means - $q_1$ and $q_5$ are equivalent. Hence the 2 - equivalence set can be writ as -

2 - equivalence = $\{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_3\}, \{q_7\}$

**Step 3 :** We will compare $\{q_0, q_6\}$

$\delta(q_0, a) = q_1$ $\quad$ $\delta(q_0, b) = q_0$

$\delta(q_6, a) = q_5$ $\quad$ $\delta(q_6, b) = q_6$

Belong to same set $\quad$ Belong to same set

$\therefore$ $\{q_0, q_6\}$ are equivalent. Similarly, we will compare $\{q_1, q_5\}, \{q_2, q_4\}$ and we find them as equivalent. Hence 3 - equivalence set is as follows :

3 - equivalence = $\{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_3\}, \{q_7\}$





As 2 - equivalence and 3 - equivalence are same sets. We will stop finding further equivalences. Hence we can have minimized transition table and minimized DFA as follows.

| | State \ Input | a | b |
|---|---|---|---|
| → | $[q_0, q_6]$ | $[q_1, q_5]$ | $[q_0, q_6]$ |
| | $[q_1, q_5]$ | $[q_0, q_6]$ | $[q_2, q_4]$ |
| | $[q_2, q_4]$ | $[q_3]$ | $[q_1, q_5]$ |
| * | $[q_3]$ | $[q_3]$ | $[q_0, q_6]$ |
| | $[q_7]$ | $[q_0, q_6]$ | $[q_3]$ |

The transition diagram with minimized states is -

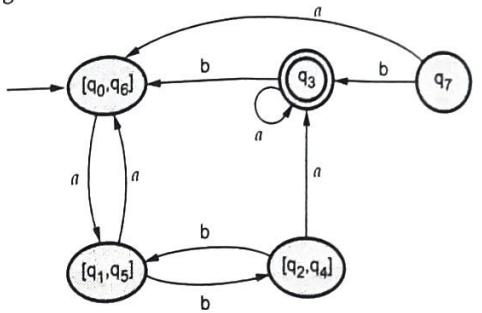



**Fig. 1.12.2**

## 1.13 Equivalence of FAs

- The two finite automata are said to be equivalent if both the automata accept the same set of strings, over an input set $\Sigma$.
- When two FAs are equivalent then there is some string x over $\Sigma$, on acceptance of that string if one FA reaches to final state other FA also reaches to final state.
- We can compare whether two FAs are equivalent or not using following method.

**Method for Comparing two FAs**

Let M and M' be two FAs and $\Sigma$ is a set of input strings.

1. We will construct a transition table have pairwise entries $(q, q')$ where $q \in M$ and $q' \in M'$ for each input symbol.
2. If we get in a pair as one final state and other nonfinal state then we terminate construction of transition table declaring that two FAs are not equivalent.

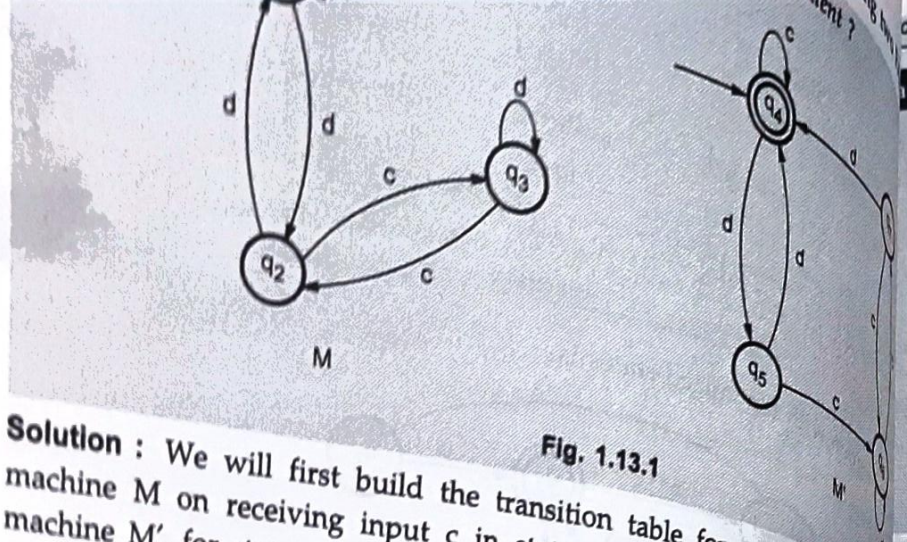



Fig. 1.13.1

**Solution :** We will first build the transition table for each input c and d machine M on receiving input c in state $q_1$ we reach to state $q_1$ only. machine M', for state $q_4$ on receiving c we reach to state $q_4$. Thus for state input c we get next state as $(q_1, q_4)$. Similarly for input d in state $(q_1, q_4)$ state as $(q_2, q_5)$.

Both $q_1$ and $q_4$ are final states obtained in pair $(q_1, q_4)$. Both $q_2$ and states obtained in pair $(q_2, q_5)$ we will obtain transition for $(q_2, q_5)$ for The complete table is as given below.

| | c | d |
|---|---|---|
| $(q_1, q_4)$ | $(q_1, q_4)$ | $(q_2, q_5)$ |
| $(q_2, q_5)$ | $(q_3, q_6)$ | $(q_1, q_4)$ |
| $(q_3, q_6)$ | $(q_2, q_7)$ | $(q_3, q_6)$ |
| $(q_2, q_7)$ | $(q_3, q_6)$ | $(q_1, q_4)$ |

From the above table note that we do not get one final state and other non in a pair. Hence we declare that two DFAs are equivalent.





**mple 1.13.2** *Following are two FAs check whether they are equivalent or not.*

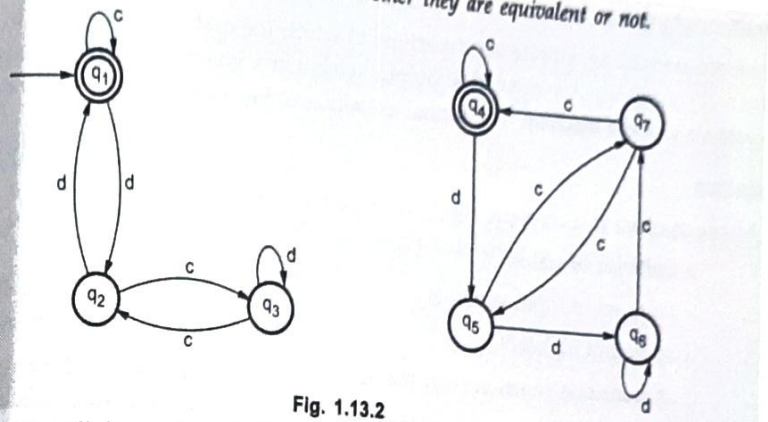



Fig. 1.13.2

**lution :** We will design the transition table for each input symbol c and d as follows.

| | c | d |
|---|---|---|
| $(q_1, q_4)$ | $(q_1, q_4)$ | $(q_2, q_5)$ |
| $(q_2, q_3)$ | $(q_3, q_7)$ | $(q_1, q_6)$ |

We will terminate the construction of transition table because we get a pair $(q_1, q_6)$ in ich $q_1$ is a final state and $q_6$ is a nonfinal state. As per equivalence rule final and nfinal state cannot form a pair. Hence the given FAs are not equivalent.

## .14 Applications of FA

Various applications of Finite Automata are -

1) In designing of lexical analysis phase of compiler, finite automata is used to recognize the tokens such as identifier, keywords, operators and so on.
2) The pattern of regular expression is recognized using finite automata.
3) It is used in text editors.
4) The finite automata is also used to design spell checkers.

# Part III : Finite State Machine with Output

## .15 Moore and Mealy Machines

SPPU : Dec.-07,11,13,16,17, May-06,10,11, Aug.-17, Oct.-16,18, Marks 6

### .15.1 Definition and Construction

There are two types of FA with output and those are :

1) Moore machine 2) Mealy machine

### Moore machine

Moore machine is a finite state machine in which the next state is decided by current state and current input symbol. The output symbol at a given time depends only on present state of the machine. The formal definition of Moore machine is,

### Definition

Moore machine is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

$Q$ is finite set of states.

$\Sigma$ is finite set of input symbols.

$\Delta$ is an output alphabet.

$\delta$ is an transition function such that $Q \times \Sigma \rightarrow Q$. This is also known as state function.

$\lambda$ is output function $Q \rightarrow \Delta$. This function is also known as machine function.

$q_0$ is the initial state of machine.

For example

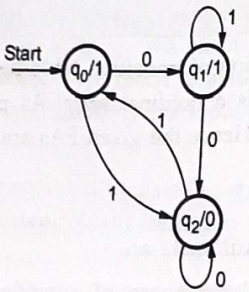



**Fig. 1.15.1**

Consider the Moore machine given below -

The transition table will be -

| Current state | Next state ($\delta$) | | Output ($\lambda$) |
|---|---|---|---|
| | 0 | 1 | |
| $q_0$ | $q_1$ | $q_2$ | 1 |
| $q_1$ | $q_2$ | $q_1$ | 1 |
| $q_2$ | $q_2$ | $q_0$ | 0 |





In Moore machine output is associated with every state. In the above given Moore machine when machine is in $q_0$ state the output will be 1. For the Moore machine if the length of input string is n then output string has length n+1.

For the string 0110 then the output will be 11110.

### Mealy machine

Mealy machine is a machine in which output symbol depends upon the present input symbol and present state of the machine. The Mealy machine can be defined as -

### Definition

Mealy machine is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

$Q$ is finite set of states.

$\Sigma$ is finite set of input symbols.

$\Delta$ is an output alphabet.

$\delta$ is state transition function such that $Q \times \Sigma \rightarrow Q$.

$\lambda$ is machine function such that $Q \times \Sigma \rightarrow \Delta$.

$q_0$ is initial state of machine.

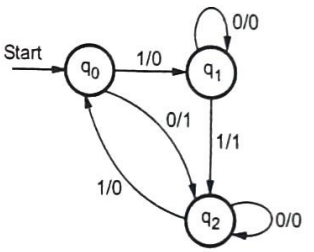



**Fig. 1.15.2**

For example,

For the input string 1001 the output will be 0001. In mealy machine the length of input string is equal to length of output string.

**Difference between Moore and Mealy Machine :**

1) Length of Moore machine is one longer than Mealy machine for given input.

2) Secondly output of the input will be along the edges in case of Mealy machine but it should be associated with the state in case of Moore machine.

## Problems on Construction of Morre and Mealy Machine

**Example 1.15.1** *Design a Moore machine to generate 1's complement of given binary number.* SPPU : May-11, Marks 4

**Solution :**

- **Logic :** 1) To generate 1's complement of given binary number the simple logic which we will apply is that if input is 0 then output will be 1 and if input is 1 then output will be 0. 2) That means there are three state one-start state, second state is for taking 0's as input and produces output as 1. Then third state is for taking 1's as input and producing output as 0.

Hence the Moore machine will be,

- **Design**

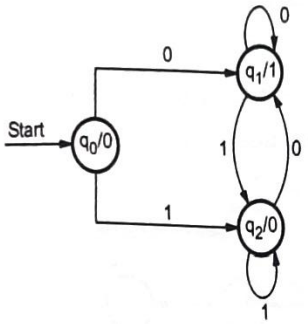



**Fig. 1.15.3**

- **Simulation**

For instance, take one binary number 1011 then

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

Thus we get 00100 as 1's complement of 1011, we can neglect the initial 0 and the output which we get is 0100 which is 1's complement of 1011. The transition table can be drawn as below -

| | Current state | Next state (δ) 0 | Next state (δ) 1 | Output (λ) |
|---|---|---|---|---|
| → | $q_0$ | $q_1$ | $q_2$ | 0 |
| | $q_1$ | $q_1$ | $q_2$ | 1 |
| | $q_2$ | $q_1$ | $q_2$ | 0 |





Thus Moore machine $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$; where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0,1\}$, $\Delta = \{0,1\}$

The transition table shows the $\delta$ and $\lambda$ functions.

**Example 1.15.2** *Design a Moore and Mealy machine for a binary input sequence such that if it has a substring 101 the machine outputs A if input has substring 110 it outputs B otherwise it outputs C.* SPPU : Aug.-17, In Sem, Marks 6

**Solution :**

- **Logic :** For designing such a machine we need to take care of two conditions and those are checking 101 and checking 110. If we get 101 the output will be A. If we recognize 110 the output will be B. For other strings the output will be C. We can make a partial design of it as follows.

- **Design**

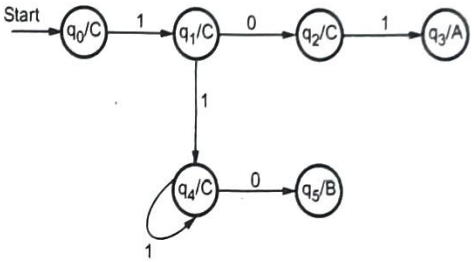



**Fig. 1.15.4**

Now we will insert the possibilities of 1's and 0's for each state. Then the Moore machine becomes.

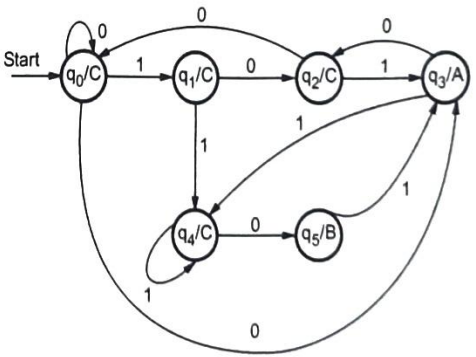



**Fig. 1.15.5**

Now the Mealy machine can be

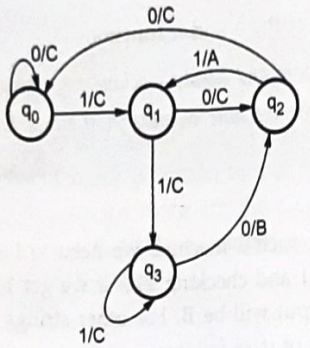



**Fig. 1.15.6**

**Example 1.15.3** *Design a Mealy machine to find 2's complement of a given binary number.* **SPPU : Oct 18, In Sem, Marks 4**

**Solution :**

- **Logic :** For designing 2's complement of a binary number we assume that input read from LSB to MSB. We will keep the binary number as it is until we read fi 1. Keep that 1 as it is then change remaining 1's by 0's and 0's by 1's.

For example,

Let the binary number be

1011

⟵

read from LSB

Keep the first 1 from LSB as it is and toggle the remaining bits we will get

0101

Thus 2's complement of 1011 is 0101. The required Mealy machine will be -

- **Design**

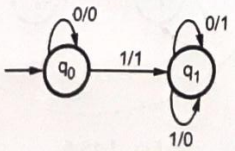



**Fig. 1.15.7**





**Example 1.15.4** *Design a Moore machine which will increment the given binary number by 1.*

**Solution :** We will read the binary number form LSB one bit at a time. We will replace each 1 by 0 until we get first 0. Once we get first 0 we will replace it by 1 and then keep remaining bits as it is.

For example -

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | ← read from LSB bit by bit |
| 1 | 0 | 1 | 0 | |
| | | | ↑ | |
| 1 | 0 | 0 | 0 | |
| | | ↑ | | |
| 1 | 1 | 0 | 0 | |
| | ↑ | | | |
| 1 | 1 | 0 | 0 | |
| ↑ | | | | |

Using this logic we can built a mealy machine as follows -

- **Design**

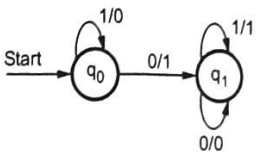



**Fig. 1.15.8**

**Example 1.15.5** *Give Moore and Mealy machine for $\Sigma = \{0, 1, 2\}$, print the residue modulo 5 of input treated as a ternary number.* **SPPU : Dec.-07, May-10, Marks 6**

**Solution :**

- **Logic :** The ternary number is made up of 0,1 and 2. The interpretation of ternary number n can be -
  - i) If we write 0 after n then number becomes 3n.
  - ii) If we write 1 after n then number becomes 3n+1.
  - iii) If we write 2 after n then number becomes 3n+2.

For example,

If n = 4 then its value is $4 \times 3^0 = 4$.

If we write 0 after 4 i.e.

$$40 = 4\times3^1+0\times3^0 = 12 \quad \text{i.e. } (3\times4)$$

If we write 1 after 4 then

$$41 = 4\times3^1+1\times3^0 = 13 \quad \text{i.e. } 3n+1$$

If we write 2 after 4 we get

$$42 = 4\times3^1+2\times3^0 = 14 \quad \text{i.e. } 3n+2$$

For residue modulo 5 we will get remainder 0, remainder 1, remainder 2, rema 3 and remainder 4 values. Then we assume various states for these remainders as,

$q_0$ - remainder 0 state

$q_1$ - remainder 1 state

$q_2$ - remainder 2 state

$q_3$ - remainder 3 state

$q_4$ - remainder 4 state

Now consider n = 4,

For 4.0 we get decimal value 12 that means 12 % 5 = 2 it gives remainder 2.

For 4.1 we get decimal value 13 that means 13 % 5 = 3 it gives remainder 3.

Similarly 4.2 given remainder 4.

n = 4 itself is remainder 4. Hence we can design,

- **Design**

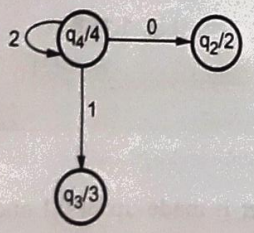



**Fig. 1.15.9**





Considering all possible cases we can design Moore machine as,

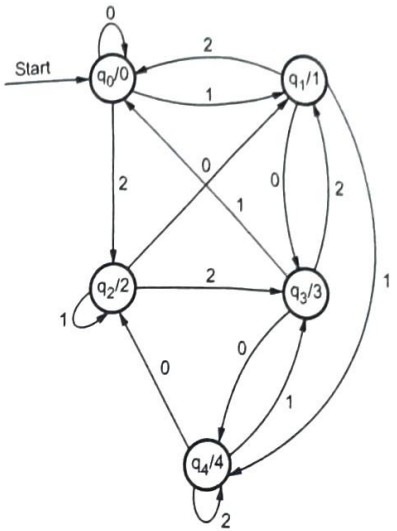



**Fig. 1.15.10**

Similarly the Mealy machine can be -

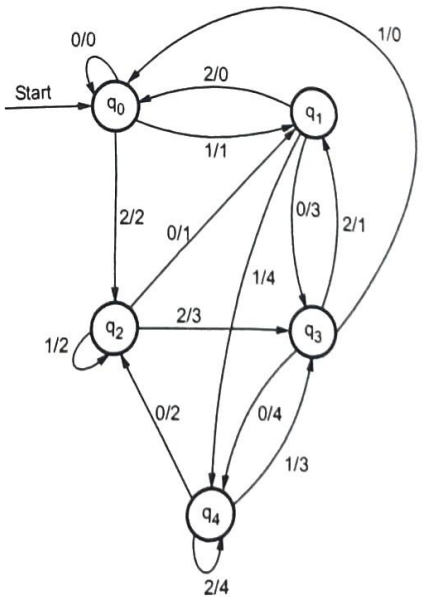



**Fig. 1.15.11**

**Example 1.15.6** *Design a Moore machine that will read sequences made up of letters A, O, U and will give as output the same sequences except that in case where an I follows an E, it will be changed to U. Design the Mealy machine for the same.* **SPPU : May-06, ...**

**Solution :**

- **Logic :** We will assume a separate state for each alphabet. The output of the ... will be corresponding letter. For instance For alphabet A the state will be ... output of $q_0$ will be A. For alphabet E the state will be $q_1$ and output of $q_1$ ... E. Continuing in this fashion we will have $q_0$, $q_1$, $q_2$, $q_3$ and $q_4$ states. The ... state will be $q_0$ we will take care of one thing and that is when I ... immediately after E it will lead to the state $q_4$ because output of state $q_4$ ... With this logic the corresponding Moore machine will be -
- **Design**

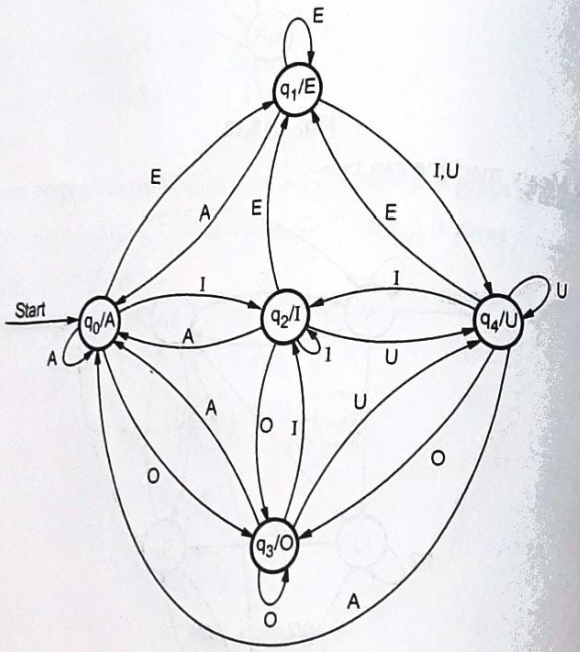



**Fig. 1.15.12**





Now we will draw the Mealy machine for the same problem.

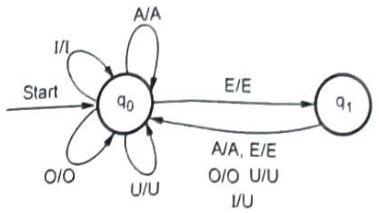



**Fig. 1.15.13**

Here the Mealy machine has the output along the edge itself we have got only two state. The transition changes from $q_0$ to $q_1$ when we get E. The I which is immediately following E will have the output U. Consider the string AIEOAEI will be give an output as AIEOAEU.

**Example 1.15.7** *Construct a Moore machine to determine residue mod 3 for binary number.* **SPPU : Dec.-17, End Sem, Marks 6**

**Solution :**

- **Logic :** This Moore machine is also called remainder 3 tester. In this machine we will get remainder 0, remainder 1 and remainder 2. To interpret the given binary number in its decimal value we consider n as a number if 0 is written after n then its value becomes 2n. If 1 is written after n then it's value becomes 2n + 1. For instance, if n = 0 then its decimal value is 0 then,

$$01 = 2n+1 = 1\times0+1 = 1$$

$$011 = 2n+1 = (1\times2)+1 = 3$$

Consider n = 1 its decimal value is 1. After 1 if 0 comes then its value will be

$$10 = 2n = 2\times1 = 2$$

If 1 comes after 10 then its value becomes,

$$101 = 2n+1 = (2\times2)+1 = 5$$

With this logic we can construct a Moore machine with 3 states. $q_0$ is the start state and is considered as remainder 0 state. $q_1$ is considered to be remainder 1 state and $q_2$ is considered as remainder 2 state.

- **Design**

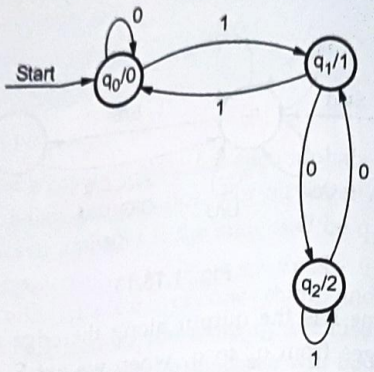
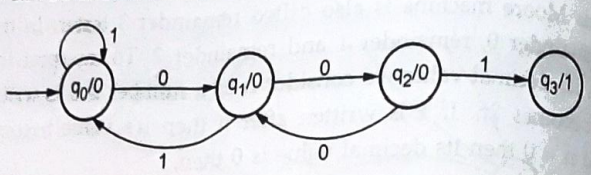



**Fig. 1.15.14**

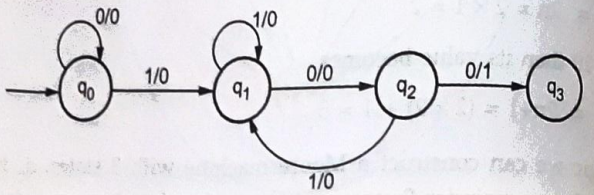
**Example 1.15.8** *Design a Moore machine for generating output 1 if input of binary seq... 1 is preceded with exactly two zeroes.*

**Solution :** In this Moore machine on receiving input 001 the output will be 1. Moore machine can be shown as below -




**Example 1.15.9** *Design a Mealy machine for a binary input sequence such that if sequence ends with 100 the output is 1 otherwise output is 0.* **SPPU : May-10, Mark...**

**Solution :** The Mealy machine will be -




**Example 1.15.10** *Design a Moore machine for checking divisibility by 3 of a given decimal number (residue of 3).* **SPPU : Dec.-11, Marks...**

**Solution :** The output at each state of Moore machine will be the remainder.





The Moore machine will be -

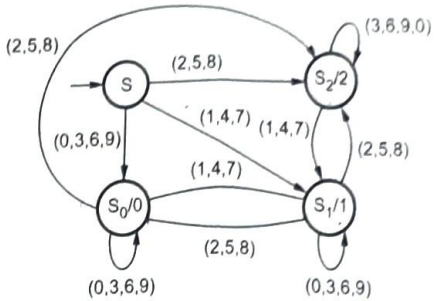



**Fig. 1.15.15**

**Example 1.15.11** *Construct FSM for binary adder.* **SPPU : Dec.-13, Marks 6**

**Solution :**

In binary addition there will be two states - no carry state and a carry state.

The transition table will be.

| x1 | x2 | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

But carry gets generated

| Present State | Next State, Output | | | |
|---|---|---|---|---|
| | $x_1x_2 = 00$ | 01 | 10 | 11 |
| No carry | No carry, 0 | carry, 1 | No carry, 1 | carry, 0 |
| carry | No carry, 1 | carry, 0 | carry, 0 | carry, 1 |

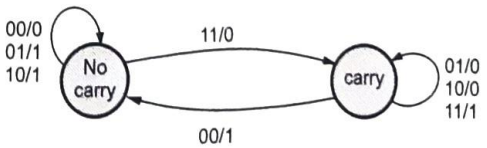



**Fig. 1.15.16**

**University Questions**

1. Define and compare Moore and Mealy Machine. SPPU : Dec.-16 In Sem.
2. Compare Moore and Mealy Machine with suitable example. SPPU : Oct.-16 In Sem.

## 1.16 Conversion of Moore to Mealy Machine

SPPU : May-11, Dec.-14,

Let $M = (Q, \Sigma, \delta, \lambda, q_0)$ be a Moore machine. The equivalent Mealy machine represented by $M' = (Q, \Sigma, \delta, \lambda', q_0)$. The output function $\lambda'$ can be obtained as -

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

**Example 1.16.1** *The Moore machine to determine residue mod 3 for binary number is given below. Convert it to Mealy equivalent machine.*

| Q \ Σ | 0 | 1 | Output (λ) |
|---|---|---|---|
| $q_0$ | $q_0$ | $q_1$ | 0 |
| $q_1$ | $q_2$ | $q_0$ | 1 |
| $q_2$ | $q_1$ | $q_2$ | 2 |

**Solution :** The transition diagram for the given problem can be drawn as -

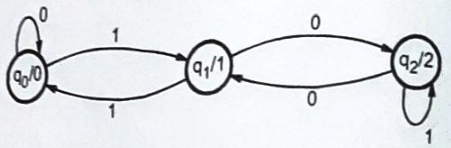



**Fig. 1.16.1**

The output function $\lambda'$ can be obtained using following rule,

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

Hence we will obtain output for every transition corresponding to input symbol

$$\lambda'(q_0, 0) = \lambda(\delta(q_0, 0))$$
$$= \lambda(q_0) \quad \text{i.e. output of } q_0$$
$$\boxed{\lambda'(q_0, 0) = 0}$$
$$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1))$$
$$= \lambda(q_1) \quad \text{i.e. output of } q_1$$
$$\boxed{\lambda'(q_0, 1) = 1}$$





$$\lambda'(q_1, 0) = \lambda(\delta(q_1, 0))$$
$$= \lambda(q_2)$$
$$\boxed{\lambda'(q_1, 1) = 2}$$
$$\lambda'(q_1, 1) = \lambda(\delta(q_1, 1))$$
$$= \lambda(q_0)$$
$$\boxed{\lambda'(q_1, 1) = 0}$$
$$\lambda'(q_2, 0) = \lambda(\delta(q_2, 0))$$
$$= \lambda(q_1)$$
$$\boxed{\lambda'(q_2, 0) = 1}$$
$$\lambda'(q_2, 1) = \lambda(\delta(q_2, 1))$$
$$= \lambda(q_2)$$
$$\boxed{\lambda'(q_2, 1) = 2}$$

Hence the transition table can be drawn as follows -

| Q \ Σ | Input 0 | | Input 1 | |
|---|---|---|---|---|
| | State | O/P | State | O/P |
| $q_0$ | $q_0$ | 0 | $q_1$ | 1 |
| $q_1$ | $q_2$ | 2 | $q_0$ | 0 |
| $q_2$ | $q_1$ | 1 | $q_2$ | 2 |

The transition diagram of Mealy machine is,

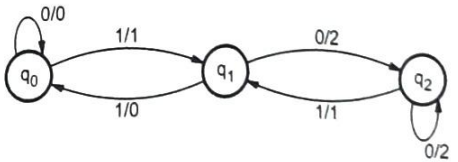



**Fig. 1.16.2**

The input string 10011 then the output for Mealy machine will be

| | | | | | |
|---|---|---|---|---|---|
| next state | $q_1 \rightarrow$ | $q_2 \rightarrow$ | $q_2 \rightarrow$ | $q_1 \rightarrow$ | $q_0$ |
| output | 1 | 2 | 2 | 1 | 0 |

The output sequence for Moore machine will be

| | | | | | |
|---|---|---|---|---|---|
| Input | | 1 | 0 | 0 | 1 | 1 |
| next state | $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2 \rightarrow q_1 \rightarrow q_0$ | | | | | |
| output | 0 | 1 | 2 | 2 | 1 | 0 |

The length of output sequence is n+1 in Moore machine and is n in Mealy m which is desired.

**Example 1.16.2** *Convert the following Moore machine into equivalent Mealy machine* $M = (\{q_0, q_1\}, \{a, b\}, \{0,1\}, \delta, \lambda, q_0)$

**Solution :**

| δ | a | b | Output (λ) |
|---|---|---|---|
| $q_0$ | $q_0$ | $q_1$ | 0 |
| $q_1$ | $q_0$ | $q_1$ | 1 |

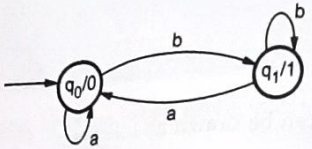



**Fig. 1.16.3**

The equivalent Mealy machine can be obtained as follows -

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

$= \lambda(q_0)$

$= 0$

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

$= \lambda(q_1)$

$= 1$

$\lambda'(q_1, a) = \lambda(\delta(q_1, a))$

$= \lambda(q_0)$

$= 0$

$\lambda'(q_1, b) = \lambda(\delta(q_1, b))$

$= \lambda(q_1)$

$= 1$





Hence the transition table can be drawn as follows -

| Q \ Σ | Input a | | Input b | |
|---|---|---|---|---|
| | State | O/P | State | O/P |
| → $q_0$ | $q_0$ | 0 | $q_1$ | 1 |
| $q_1$ | $q_0$ | 0 | $q_1$ | 1 |

The equivalent Mealy machine will be,

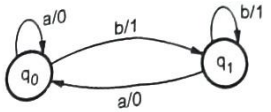



**Fig. 1.16.4**

Consider the output sequence for Moore machine for input sequence,

a b b a

Then

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_0$$

0 0 1 1 0

Similarly output sequence for Mealy machine will be

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \longrightarrow q_0$$

0 1 1 1 0

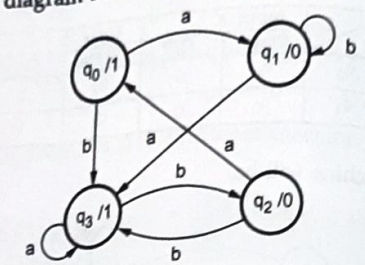
We can note that length of Moore machine is n+1 and that of Mealy machine is n.

**Example 1.16.3** *Convert the following Moore machine to Mealy machine.*

**SPPU : May-11, Marks 6**

| State | Input | | Output |
|---|---|---|---|
| | a | b | |
| $q_0$ | $q_1$ | $q_3$ | 1 |
| $q_1$ | $q_3$ | $q_1$ | 0 |
| $q_2$ | $q_0$ | $q_3$ | 0 |
| $q_3$ | $q_3$ | $q_2$ | 1 |

**Solution :** The transition diagram for the given transition table will be




The output function $\lambda'$ can be obtained using following rule,

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

$\therefore$ $\lambda'(q_0, a) = \lambda(\delta(q_0, a))$

$= \lambda(q_1)$

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

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

$= \lambda(q_3)$

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

$\lambda'(q_1, a) = \lambda(\delta(q_1, a))$

$= \lambda(q_3)$

$\lambda'(q_1, a) = 1$

$\lambda'(q_1, b) = \lambda(\delta(q_1, b))$

$= \lambda(q_1)$

$\lambda'(q_1, b) = 0$

$\lambda'(q_2, a) = \lambda(\delta(q_2, a))$

$= \lambda(q_0)$

$\lambda'(q_2, a) = 1$

$\lambda'(q_2, b) = \lambda(\delta(q_2, b))$

$= \lambda(q_3)$

$\lambda'(q_2, b) = 1$

$\lambda'(q_3, a) = \lambda(\delta(q_3, a)) = \lambda(q_3)$

$\lambda'(q_3, a) = 1$





$\lambda'(q_3, b) = \lambda(\delta(q_3, b))$

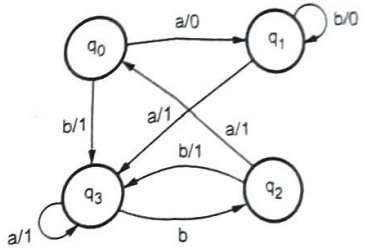
$= \lambda(q_2)$

$\lambda'(q_3, b) = 0$

The transition table can be as follows -

| Q | Σ | Input a | | Input b | |
|---|---|---|---|---|---|
| | | State | Output | State | Output |
| $q_0$ | | $q_1$ | 0 | $q_3$ | 1 |
| $q_1$ | | $q_3$ | 1 | $q_1$ | 0 |
| $q_2$ | | $q_0$ | 1 | $q_3$ | 1 |
| $q_3$ | | $q_3$ | 1 | $q_2$ | 0 |

The transition diagram for Mealy machine is




**Example 1.16.4** *Construct Mealy machine equivalent to the given Moore machine.*

**SPPU : Dec.-14 In Sem, Marks 6**

| | 0 | 1 | O/P |
|---|---|---|---|
| $q_0$ | $q_0$ | $q_1$ | N |
| $q_1$ | $q_0$ | $q_2$ | N |
| $q_2$ | $q_0$ | $q_3$ | N |
| $q_3$ | $q_0$ | $q_3$ | Y |

*Start state :* $q_0$ *; Final state :* $q_3$

**Solution :** The transition diagram for given Moore machine is shown below.

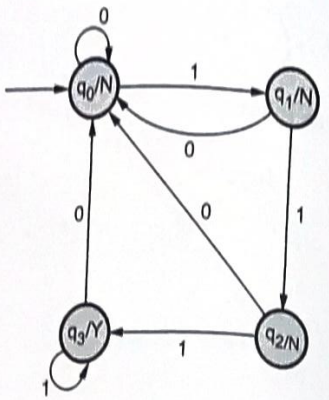



**Fig. 1.16.5**

The output function $\lambda'$ can be obtained by following rule,

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

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

$$= \lambda(q_0)$$

$$\therefore \boxed{\lambda'(q_0, 0) = N}$$

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

$$= \lambda(q_1)$$

$$\therefore \boxed{\lambda'(q_0, 1) = N}$$

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

$$= \lambda(q_0)$$

$$\therefore \boxed{\lambda'(q_1, 0) = N}$$

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

$$= \lambda(q_2)$$

$$\therefore \boxed{\lambda'(q_1, 1) = N}$$

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

$$= \lambda(q_0)$$

$$\therefore \boxed{\lambda'(q_2, 0) = N}$$





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

$$= \lambda(q_3)$$

$$\therefore \boxed{\lambda'(q_2, 1) = Y}$$

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

$$= \lambda(q_0)$$

$$\therefore \boxed{\lambda'(q_3, 0) = N}$$

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

$$= \lambda(q_3)$$

$$\therefore \boxed{\lambda'(q_3, 1) = Y}$$

Hence the transition table can be drawn as follows

| Q \ Σ | Input 0 | | Input 1 | |
|---|---|---|---|---|
| | State | O/P | State | O/P |
| $q_0$ | $q_0$ | N | $q_1$ | N |
| $q_1$ | $q_0$ | N | $q_2$ | N |
| $q_2$ | $q_0$ | N | $q_3$ | Y |
| $q_3$ | $q_0$ | N | $q_3$ | Y |

The Mealy machine will be

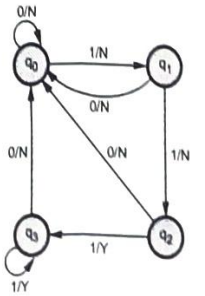



**Fig. 1.16.6**

## 1.17 Conversion of Mealy to Moore Machine

SPPU : May-13,18, Dec.-16, Mark

**Method for conversion of Mealy Machine to Moore Machine**

Let M = ( Q, Σ, Δ, δ, λ, $q_0$ ) be a Mealy machine then there exists a Moore machine M' equivalent to M.

The M' can be given as M' = (Q×Δ, Σ, Δ, δ', λ',[$q_0$, $b_0$]) where $b_0$ is an output symbol selected from Δ. We will calculate δ' and λ' as follows -

δ'([q,b], a) = [δ(q,a), λ(q,a)]

λ'([q,b]) = b

δ' defines the move made by M' on input a.

M' on input a.

λ' defines the output made for the corresponding state q.

Thus we have to calculate δ' and λ' for $q_0$, $q_1$, $q_2$, .....$q_n$ on input $a_0$, $a_1$,.....$a_n$ should emit the outputs $b_1$, $b_2$, $b_3$.....$b_n$.

**Example 1.17.1** *Convert the following Mealy machine into equivalent Moore machine.*

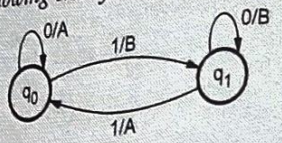



**Fig. 1.17.1**

**Solution :** The states for Moore machine will be [$q_0$, A], [$q_0$, B], [$q_1$, A], [$q_1$, B]. Now we will calculate δ' and λ' as follows -

δ' (($q_0$, A), 0) = [δ($q_0$, 0), λ ($q_0$, 0)]

i.e [next state of $q_0$ on input 0, output of $q_0$ on input 0 ]

$q_0$ A

= [$q_0$, A]

λ' ([$q_0$, A]) = A → i.e. output of state $q_0$.

δ' ([$q_1$, A], 0) = [δ($q_1$,0), λ ($q_1$,0)]

= [$q_1$, B]

λ' ([$q_1$, A]) = A





δ' ([$q_1$, A], 1) = [δ($q_1$,1), λ ($q_1$,1)]

= [$q_0$, A]

λ' ([$q_1$, A]) = A

Hence,

| | 0 | 1 | Output |
|---|---|---|---|
| [$q_0$, A] | [$q_0$, A] | [$q_1$, B] | A |
| [$q_1$, A] | [$q_1$, B] | [$q_0$, A] | A |

Continuing in this fashion we can calculate δ' and λ' as follows -

δ'([$q_0$, B], 0) = [δ ($q_0$,0), λ ($q_0$,0)]

= [$q_0$, A]

λ' [$q_0$, B]) = B

δ'(($q_1$, B), 1) = [δ ($q_0$,1), λ ($q_0$,1)]

= [$q_1$, B]

λ' ([$q_0$, B]) = B

δ'([$q_1$, B], 0) = [δ ($q_1$,0), λ($q_1$,0)]

= [$q_1$, B]

λ' ([$q_1$, B]) = B

δ'(($q_1$, B), 1) = [δ ($q_1$,1), λ ($q_1$,1)]

= [$q_0$, B]

λ' ([$q_1$, B]) = B

Thus we get the transition table as

| State \ I/P | 0 | 1 | Output |
|---|---|---|---|
| [$q_0$, A] | [$q_0$, A] | [$q_1$, B] | A |
| [$q_0$, B] | [$q_0$, A] | [$q_1$, B] | B |
| [$q_1$, A] | [$q_1$, B] | [$q_0$, A] | A |
| [$q_1$, B] | [$q_1$, B] | [$q_0$, A] | B |

The transition diagram will be,

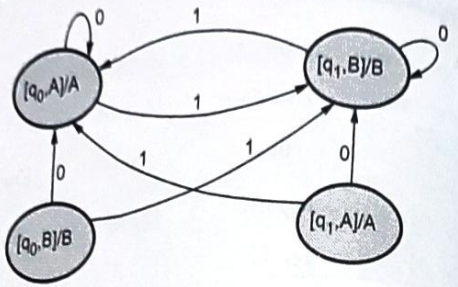



Fig. 1.17.2

**Example 1.17.2** *Convert the following Mealy machine into equivalent Moore machine.*

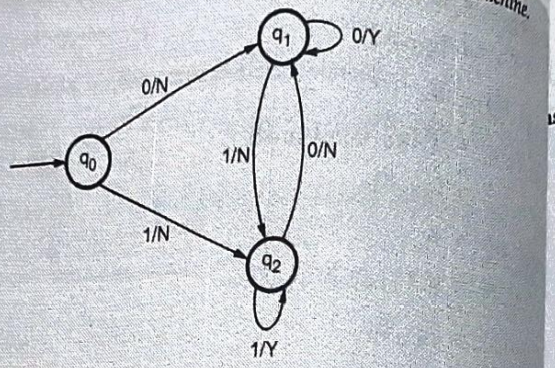



Fig. 1.17.3

**Solution :** We will write the transition table for given transition graph -

| | Input 0 | Output | Input 1 | Output |
|---|---|---|---|---|
| $q_0$ | $q_1$ | N | $q_2$ | N |
| $q_1$ | $q_1$ | Y | $q_2$ | N |
| $q_2$ | $q_1$ | N | $q_2$ | Y |

Now we will find out the states and corresponding outputs for Moore machine states for Moore machine will be -

$[q_0,N]$, $[q_0,Y]$, $[q_1,N]$, $[q_1,Y]$, $[q_2,N]$, $[q_2,Y]$

Now let us calculate $\delta'$ and $\lambda'$ for all the above given states -

$\delta'([q_0,N)], 0) = [\delta(q_0,0), \lambda(q_0,0)]$

$= [q_1, N]$





$\lambda'([q_0,N]) = N$

Similarly,

$\delta'([q_0,N), 1) = [\delta(q_0,1), \lambda(q_0,1)]$

$= [q_2, N]$

$\lambda'([q_0,N]) = N$

The partial transition table is -

| State \ I/P | 0 | 1 | Output |
|---|---|---|---|
| $[q_0, N]$ | $[q_1, N]$ | $[q_2, N]$ | N |

Now, for remaining states the corresponding transitions and outputs can be obtained as follows -

$\delta'([q_0,Y], 0) = [\delta(q_0,0), \lambda(q_0,0)]$

$= [q_1, N]$

$\lambda'([q_0,Y]) = Y$

$\delta'([q_0,Y], 1) = [\delta(q_0,1), \lambda(q_0,1)]$

$= [q_2 N]$

$\lambda'([q_0,Y]) = Y$

$\delta'([q_1N], 0) = [\delta(q_1,0), \lambda(q_1,0)]$

$= [q_1 Y]$

$\lambda'([q_1,N]) = N$

$\delta'([q_1,N], 1) = [\delta(q_1,1), \lambda\ (q_1,1)]$

$= [q_2, N]$

$\lambda'([q_1,N]) = N$

$\delta'([q_1,Y], 0) = [\delta(q_1,0), \lambda\ (q_1,0)]$

$= [q_1, Y]$

$\lambda'([q_1,Y]) = Y$

$\delta'([q_2,N], 0) = [\delta(q_2,0), \lambda\ (q_2,0)]$

$= [q_1, N]$

$\lambda'([q_2,N]) = N$

$\delta'([q_2,N], 1) = [\delta(q_2,1), \lambda\ (q_2,1)]$

$$= [q_2, Y]$$

$$\lambda'([q_2, N]) = N$$

$$\delta'([q_1, Y], 1) = [\delta(q_1, 1), \lambda(q_1, 1)]$$

$$= [q_2, N]$$

$$\lambda'([q_1, Y]) = Y$$

$$\delta'([q_2, Y], 0) = [\delta(q_2, 0), \lambda(q_2, 0)]$$

$$= [q_1, N]$$

$$\lambda'([q_2, Y]) = Y$$

$$\delta'([q_2, Y], 1) = [\delta(q_2, 1), \lambda(q_2, 1)]$$

$$= [q_2, Y]$$

$$\lambda'([q_2, Y]) = Y$$

The transition table can be drawn as -

| State \ Σ | 0 | 1 | Output |
|---|---|---|---|
| $[q_0, N]$ | $[q_1, N]$ | $[q_2, N]$ | N |
| $[q_0, Y]$ | $[q_1, N]$ | $[q_2, N]$ | Y |
| $[q_1, N]$ | $[q_1, Y]$ | $[q_2, N]$ | N |
| $[q_1, Y]$ | $[q_1, Y]$ | $[q_2, N]$ | Y |
| $[q_2, N]$ | $[q_1, N]$ | $[q_2, Y]$ | N |
| $[q_2, Y]$ | $[q_1, N]$ | $[q_2, Y]$ | Y |

**Example 1.17.3** *Convert the following Mealy machine into equivalent Moore machine.*

**SPPU : May-13, Marks**

**Solution :** We will design transition table for Mealy machine.

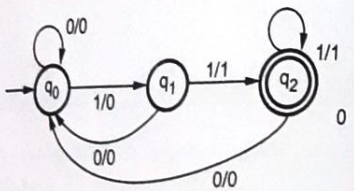



**Fig. 1.17.4**





| Current state | A = 0 | | A = 1 | |
|---|---|---|---|---|
| | Next state | Output | Next state | Output |
| $q_0$ | $q_0$ | 0 | $q_1$ | 0 |
| $q_1$ | $q_0$ | 0 | $q_2$ | 1 |
| $q_2$ | $q_0$ | 0 | $q_2$ | 1 |

$q_0$ is associated with output 0, $q_1$ is associated with output 0, $q_2$ is associated with output 1

Moore machine

| Current state | Next state | | Output |
|---|---|---|---|
| | A = 0 | A = 1 | |
| $q_0$ | $q_0$ | $q_1$ | 0 |
| $q_1$ | $q_0$ | $q_2$ | 0 |
| $q_2$ | $q_0$ | $q_2$ | 1 |

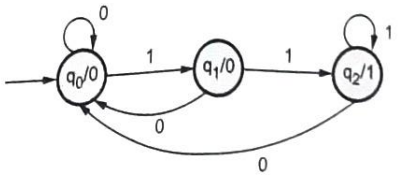
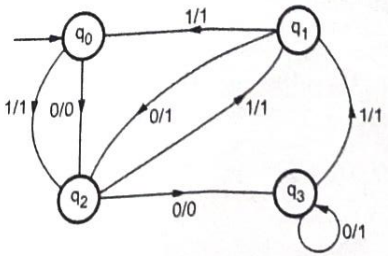



**Fig. 1.17.5**

**Example 1.17.4** *Construct Moore machine equivalent for the given Mealy machine.*



**Solution :** The states for Moore machine will be $[q_0, 0]$, $[q_0, 1]$, $[q_1, 0]$, $[q_1, 1]$, $[q_2, 1]$, $[q_3, 0]$, $[q_3, 1]$.

Then we will calculate $\delta'$ and $\lambda'$ functions.

$\delta'([q_0, 0], 0) = [\delta(q_0, 0), \lambda(q_0, 0)] = [q_2, 0]$

$\lambda'([q_0, 0]) = 0$

$\delta'([q_0, 0], 1) = [\delta(q_0, 1), \lambda(q_0, 1)] = [q_2, 1]$

$\lambda'([q_0, 0]) = 0$

The partial transition table will be

| State \ Input | 0 | 1 | Output |
|---|---|---|---|
| $[q_0, 0]$ | $[q_2, 0]$ | $[q_2, 1]$ | 0 |

Now remaining states for corresponding transitions are as given below.

$\delta'([q_0, 1], 0) = [\delta(q_0, 0), \lambda(q_0, 0)] = [q_2, 0]$

$\lambda'([q_0, 1]) = 1$

$\delta'([q_0, 1], 1) = [\delta(q_0, 1), \lambda(q_0, 1)] = [q_2, 1]$

$\lambda'([q_0, 1]) = 1$

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

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

$\delta'([q_1, 0], 1) = (\delta(q_1, 1), \lambda(q_1, 1)) = [q_0, 1]$

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

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

$\delta'([q_1, 1]) = 1$

$\delta'([q_1, 1], 1) = (\delta(q_1, 1), \lambda(q_1, 1)) = [q_0, 1]$

$\lambda'([q_1, 1]) = 1$

$\delta'([q_2, 0], 0) = (\delta(q_2, 0), \lambda(q_2, 0)) = [q_3, 0]$

$\lambda'([q_2, 0]) = 0$





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

$\lambda'([q_2, 0]) = 0$

$\delta'([q_2, 1], 0) = (\delta(q_2, 0), \lambda(q_2, 0)) = [q_3, 0]$

$\lambda'([q_2, 1]) = 1$

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

$\lambda'([q_2, 1]) = 1$

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

$\lambda'([q_3, 0]) = 0$

$\delta'([q_3, 0], 1) = (\delta(q_3, 1), \lambda(q_3, 1)) = [q_1, 1]$

$\lambda'([q_3, 0]) = 0$

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

$\lambda'([q_3, 1]) = 1$

$\delta'([q_3, 1], 1) = (\delta(q_3, 1), \lambda(q_3, 1)) = [q_1, 1]$

$\lambda'([q_3, 1]) = 1$

The transition table for Moore machine will be

| State \ Input | 0 | 1 | Output |
|---|---|---|---|
| $[q_0, 0]$ | $[q_2, 0]$ | $[q_2, 1]$ | 0 |
| $[q_0, 1]$ | $[q_2, 0]$ | $[q_2, 1]$ | 1 |
| $[q_1, 0]$ | $[q_2, 1]$ | $[q_0, 1]$ | 0 |
| $[q_1, 1]$ | $[q_2, 1]$ | $[q_0, 1]$ | 1 |
| $[q_2, 0]$ | $[q_3, 0]$ | $[q_1, 1]$ | 0 |
| $[q_2, 1]$ | $[q_3, 0]$ | $[q_1, 1]$ | 1 |
| $[q_3, 0]$ | $[q_3, 1]$ | $[q_1, 1]$ | 0 |
| $[q_3, 1]$ | $[q_3, 1]$ | $[q_1, 1]$ | 1 |

## 1.17 Conversion of Mealy to Moore Machine

SPPU : May 13, 18, Dec. 16, May

### Method for conversion of Mealy Machine to Moore Machine

Let $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ be a Mealy machine then there exists a Moore m
M' equivalent to M.

The M' can be given as $M' = (Q \times \Delta, \Sigma, \Delta, \delta', \lambda', [q_0, b_0])$ where $b_0$ is an
symbol selected from $\Delta$. We will calculate $\delta'$ and $\lambda'$ as follows -

$\delta'([q, b], a) = [\delta(q, a), \lambda(q, a)]$

$\lambda'([q, b]) = b$

$\delta'$ defines the move made by M' on input a.

M' on input a.

$\lambda'$ defines the output made for the corresponding state q.

Thus we have to calculate $\delta'$ and $\lambda'$ for $q_0, q_1, q_2, \ldots\ldots q_n$ on input $a_0, a_1, \ldots$
should emit the outputs $b_1, b_2, b_3, \ldots\ldots b_n$.

**Example 1.17.1** *Convert the following Mealy machine into equivalent Moore machine.*

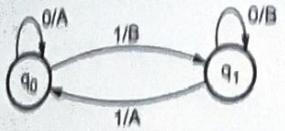



**Fig. 1.17.1**

**Solution :** The states for Moore machine will be $[q_0, A]$, $[q_0, B]$, $[q_1, A]$, $[q_1, B]$,
we will calculate $\delta'$ and $\lambda'$ as follows -

$$\delta'((q_0, A), 0) = [\delta(q_0, 0), \lambda(q_0, 0)]$$

i.e [next state of $q_0$ on input 0, output of $q_0$ on input 0 ]

$q_0$ A

$$= [q_0, A]$$

$\lambda'([q_0, A]) = A \rightarrow$ i.e. output of state $q_0$.

$$\delta'([q_1, A], 0) = [\delta(q_1, 0), \lambda(q_1, 0)]$$

$$= [q_1, B]$$

$\lambda'([q_1, A]) = A$





$\delta'([q_1, A], 1) = [\delta(q_1, a), \lambda(q_1, a)]$

$= [q_0, B]$

$\lambda'([q_1, A]) = A$

Hence,

Continuing in the same manner

$\delta'([q_0, B], 1) = [\delta(q_0, a), \lambda(q_0, a)]$

$= [q_0, B]$

$\lambda'([q_0, B]) = B$

$\delta'([q_1, B], 0) = [\delta(q_1, a), \lambda(q_1, a)]$

$= [q_1, B]$

$\lambda'([q_1, B]) = B$

$\delta'([q_1, B], 1) = [\delta(q_1, a), \lambda(q_1, a)]$

$= [q_0, B]$

$\lambda'([q_0, B]) = B$

$\delta'([q_0, B], 1) = [\delta(q_0, a), \lambda(q_0, a)]$

$= [q_0, B]$

$\lambda'([q_0, B]) = B$

Thus we get the

$= [q_3, 0]$

$\lambda'([q_0, 0]) = 0$

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

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

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

$\delta'([q_1, 1], a) = [\delta(q_1, a), \lambda(q_1, a)] = [q_0, 0]$

$\lambda([q_1, 1]) = 1$

$\delta'([q_1, 1], b) = [\delta(q_1, b), \lambda(q_1, b)] = [q_1, 1]$

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

$\lambda([q_2, 0]) = 0$

$\delta'([q_2, 0], b) = [\delta(q_2, b), \lambda(q_2, b)] = [q_2, 0]$

$\delta'([q_2, 1], a) = [\delta(q_2, a), \lambda(q_2, a)] = [q_1, 1]$

$\lambda([q_2, 1]) = 1$

$\delta'([q_3, 0], a) = [\delta(q_3, a), \lambda(q_3, a)] = [q_2, 0]$

$\lambda([q_3, 0]) = 0$

$\delta'([q_3, 0], b) = [\delta(q_3, b), \lambda(q_3, b)] = [q_0, 1]$

$\delta'([q_3, 1], a) = [\delta(q_3, a), \lambda(q_3, a)] = [q_2, 0]$

$\lambda([q_3, 1]) = 1$

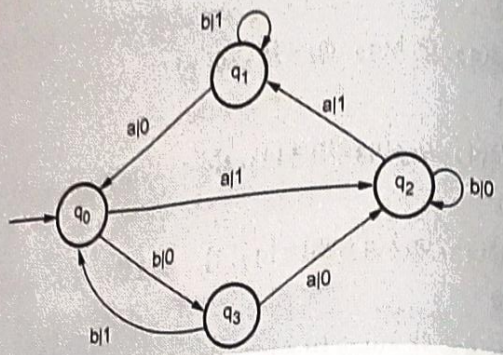
$\delta'([q_3, 1], b) = [\delta(q_3, b), \lambda(q_3, b)] = [q_0, 1]$

$\lambda([q_3, 1]) = 1$

tained next state and output. The transition table can be drawn as

| $\Sigma$ | a | b | Output |
|---|---|---|---|
| | $[q_2, 1]$ | $[q_3, 0]$ | 0 |
| | $[q_2, 1]$ | $[q_3, 0]$ | 1 |
| | $[q_0, 0]$ | $[q_1, 1]$ | 0 |
| | $[q_0, 0]$ | $[q_1, 1]$ | 1 |

**Example 1.17.5** *Construct Moore machine for given Mealy machine.* **SPPU : Dec.-16 End Sem.**




**Solution :** We will write the transition table for given transition graph

| State \ Transit | Input a | Output | Input b | Output |
|---|---|---|---|---|
| $q_0$ | $q_2$ | 1 | $q_3$ | 0 |
| $q_1$ | $q_0$ | 0 | $q_1$ | 1 |
| $q_2$ | $q_1$ | 1 | $q_2$ | 0 |
| $q_3$ | $q_2$ | 0 | $q_0$ | 1 |

Now we will find out the states and corresponding outputs for Moore machine. The states for Moore machine will be $[q_0,1]$, $[q_0,0]$, $[q_1,0]$, $[q_1,1]$, $[q_2,0]$, $[q_2,1]$, $[q_3,0]$, $[q_3,1]$

Now let us compute $\lambda'$ and $\delta'$ for the above states.

$$\delta'([q_0,1],a) = [\delta(q_0,a),\lambda(q_0,a)]$$
$$= [q_2,1]$$

Similarly
$$\lambda'([q_0,1]) = 1$$
$$\delta'([q_0,1],b) = [\delta(q_0,b),\lambda(q_0,b)]$$
$$= [q_3,0]$$
$$\delta'([q_0,0],a) = [\delta(q_0,a),\lambda(q_0,a)] = [q_2,1]$$
$$\lambda'([q_0,0]) = 0$$
$$\delta'([q_0,0],b) = [\delta(q_0,b),\lambda(q_0,b)]$$





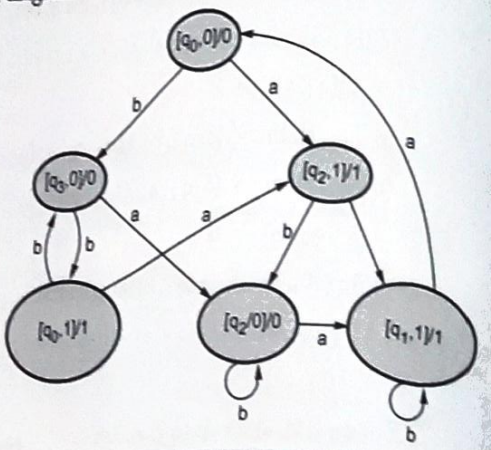
$$= [q_3,0]$$
$$\lambda'([q_0,0]) = 0$$
$$\delta'([q_1,0],a) = [\delta(q_1,a),\lambda(q_1,a)] = [q_0,0]$$
$$\delta'([q_1,0]) = 0$$
$$\delta'([q_1,0],b) = [\delta(q_1,b),\lambda(q_1,b)] = [q_1,1]$$
$$\delta'([q_1,1],a) = [\delta(q_1,a),\lambda(q_1,a)] = [q_0,0]$$
$$\lambda([q_1,1]) = 1$$
$$\delta'([q_1,1],b) = [\delta(q_1,b),\lambda(q_1,b)] = [q_1,1]$$
$$\delta'([q_2,0],a) = [\delta(q_2,a),\lambda(q_2,a)] = [q_1,1]$$
$$\lambda([q_2,0]) = 0$$
$$\delta'([q_2,0],b) = [\delta(q_2,b),\lambda(q_2,b)] = [q_2,0]$$
$$\delta'([q_2,1],a) = [\delta(q_2,a),\lambda(q_2,a)] = [q_1,1]$$
$$\lambda([q_2,1]) = 1$$
$$\delta'([q_3,0],a) = [\delta(q_3,a),\lambda(q_3,a)] = [q_2,0]$$
$$\lambda([q_3,0]) = 0$$
$$\delta'([q_3,0],b) = [\delta(q_3,b),\lambda(q_3,b)] = [q_0,1]$$
$$\delta'([q_3,1],a) = [\delta(q_3,a),\lambda(q_3,a)] = [q_2,0]$$
$$\lambda([q_3,1]) = 1$$
$$\delta'([q_3,1],b) = [\delta(q_3,b),\lambda(q_3,b)] = [q_0,1]$$
$$\lambda([q_3,1]) = 1$$

Thus we have obtained next state and output. The transition table can be drawn as follows :

| State \ Σ | a | b | Output |
|---|---|---|---|
| $[q_0,0]$ | $[q_2,1]$ | $[q_3,0]$ | 0 |
| $[q_0,1]$ | $[q_2,1]$ | $[q_3,0]$ | 1 |
| $[q_1,0]$ | $[q_0,0]$ | $[q_1,1]$ | 0 |
| $[q_1,1]$ | $[q_0,0]$ | $[q_1,1]$ | 1 |

| | | | |
|---|---|---|---|
| $[q_2, 0]$ | $[q_1, 1]$ | $[q_2, 0]$ | 0 |
| $[q_2, 1]$ | $[q_1, 1]$ | $[q_2, 0]$ | 0 |
| $[q_3, 0]$ | $[q_2, 0]$ | $[q_0, 1]$ | 1 |
| $[q_3, 1]$ | $[q_2, 0]$ | $[q_0, 1]$ | 1 |

The transition diagram for Moore machine will be




**Example 1.17.6** *Construct the Mealy machine to accept strings ending with '00' or '11'. $\Sigma = \{0, 1\}$. Convert Mealy machine into equivalent Moore machine.*

SPPU : May-18, ...

**Solution :** The Mealy machine is

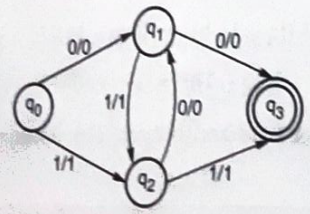



**Fig. 1.17.6**





The conversion of Mealy to Moore - We will write transition table as,

| | i/p 0 | Output | i/p 1 | Output |
|---|---|---|---|---|
| $q_0$ | $q_1$ | 0 | $q_2$ | 1 |
| $q_1$ | $q_3$ | 0 | $q_2$ | 1 |
| $q_2$ | $q_1$ | 0 | $q_3$ | 1 |

Now we will obtain states and corresponding outputs for Moore machine. The states for Moore machine will be

$[q_0, 0]$, $[q_0, 1]$, $[q_1, 0]$, $[q_1, 1]$

We will calculate $\delta'$ and $\lambda'$ for all above states

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

Similarly,

$$\delta'([q_0, 0], 1) = [\delta(q_0, 1)\lambda(q_0, 1)]$$
$$= [q_2, 1]$$
$$\lambda'[(q_0, 0)] = 0$$

The partial transistion table

| State \ i/p | 0 | 1 | Output |
|---|---|---|---|
| $[q_0, 0]$ | $[q_1, 0]$ | $[q_2, 1]$ | 0 |

Thus, if we compute $\delta'$ and $\lambda'$ in above manner we get following Moore machine.

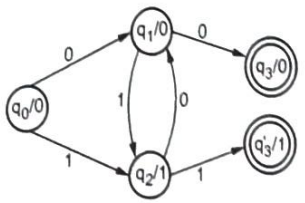



**Fig. 1.17.7**