FORMAL LANGUAGES & AUTOMATA THEORY

UNIT- I FINITE AUTOMATA

1

FINITE AUTOMATA

After going through this chapter, you should be able to understand :

- Alphabets, Strings and Languages
- Mathematical Induction
- Finite Automata
- Equivalence of NFA and DFA
- NFA with $\in$ - moves

1.1 ALPHABETS, STRINGS & LANGUAGES

Alphabet

An alphabet, denoted by Σ , is a finite and nonempty set of symbols.

Example:

1. If Σ is an alphabet containing all the 26 characters used in English language, then Σ is finite and nonempty set, and $\Sigma = \{a, b, c, \ldots, z\}$.
2. $X = \{0,1\}$ is an alphabet.
3. $Y = \{1, 2, 3, \ldots\}$ is not an alphabet because it is infinite.
4. $Z = \{ \}$ is not an alphabet because it is empty.

String

A string is a finite sequence of symbols from some alphabet.

Example :

" xyz " is a string over an alphabet $\Sigma = \{a, b, c, \ldots, z\}$. The empty string or null string is denoted by $\in$.

Length of a string

The length of a string is the number of symbols in that string. If w is a string then its length is denoted by $|w|$.

Example :

1. $w=abcd$, then length of w is $|w|=4$
2. $n = 010$ is a string, then $|n| = 3$
3. $\in$ is the empty string and has length zero.

The set of strings of length K (K $\geq$ 1)

Let Σ be an alphabet and $\Sigma = \{a, b\}$, then all strings of length $K\ (K \geq 1)$ is denoted by Σ^K.

$\Sigma^K = \{w : w \textit{ is a string of length } K, K \geq 1\}$

Example:

1. $\Sigma = \{a,b\}$, then

   $\Sigma^1 = \{a,b\}$,

   $\Sigma^2 = \{aa, ab, ba, bb\}$,

   $\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

   $|\Sigma^1| = 2 = 2^1$ (Number of strings of length one),

   $|\Sigma^2| = 4 = 2^2$ (Number of strings of length two), and

   $|\Sigma^3| = 8 = 2^3$ (Number of strings of length three)
2. $S = \{0,1,2\}$, then $S^2 = \{00, 01, 02, 11, 10, 12, 22, 20, 21\}$, and $|S^2| = 9 = 3^2$

Concatenation of strings

If w_1 and w_2 are two strings then concatenation of w_2 with w_1 is a string and it is denoted by w_1w_2. In other words, we can say that w_1 is followed by w_2 and $|w_1w_2| = |w_1| + |w_2|$.

Prefix of a string

A string obtained by removing zero or more trailing symbols is called prefix. For example, if a string $w = abc$, then a, ab, abc are prefixes of w.

Suffix of a string

A string obtained by removing zero or more leading symbols is called suffix. For example, if a string $w = abc$, then c, bc, abc are suffixes of w.
A string a is a proper prefix or suffix of a string w if and only if $a \neq w$.

Substrings of a string

A string obtained by removing a prefix and a suffix from string w is called substring of w. For example, if a string $w = abc$, then b is a substring of w. Every prefix and suffix of string w is a substring of w, but not every substring of w is a prefix or suffix of w. For every string w, both w and $\in$ are prefixes, suffixes, and substrings of w.
Substring of $\text{w} = \text{w} - (\text{one prefix}) - (\text{one suffix})$.

Language

A Language L over Σ, *is a subset of* Σ^*, *i. e., it is a collection of strings over the alphabet* Σ. ϕ, and $\{\in\}$ are languages. The language ϕ is undefined as similar to infinity and $\{\in\}$ is similar to an empty box i.e. a language without any string.

Example:

1. $L_1 = \{01, 0011, 000111\}$ is a language over alphabet $\{0,1\}$
2. $L_2 = \{\in, 0, 00, 000, ...\}$ is a language over alphabet $\{0\}$
3. $L_3 = \{0^n 1^n 2^n : n \geq 1\}$ is a language.

Kleene Closure of a Language

Let L be a language over some alphabet Σ. Then Kleene closure of L is denoted by L^* and it is also known as reflexive transitive closure, and defined as follows :

$L^* = \{Set\ of\ all\ words\ over\ \Sigma\}$

$= \{\text{word of length zero, words of length one, words of length two, ...}\}$

$$= \bigcup_{K=0}^{\infty} (\Sigma^K) = L^0 \cup L^1 \cup L^2 \cup \ldots$$

Example:

1. $\Sigma = \{a, b\}$ and a language L over Σ. Then

   $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$

   $L^0 = \{\in\}$

   $L^1 = \{a, b\}$,

   $L^2 = \{aa, ab, ba, bb\}$ and so on.

   So, $L^* = \{\in, a, b, aa, ab, ba, bb \ldots\}$
2. $S = \{0\}$, then $S^* = \{\in, 0, 00, 000, 0000, 00000, \ldots\}$

Positive Closure

If Σ is an alphabet then positive closure of Σ is denoted by Σ^+ and defined as follows :

$\Sigma^+ = \Sigma^* - \{\in\} = \{Set\ of\ all\ words\ over\ \Sigma\ excluding\ empty\ string\ \in\}$

Example :

if $\Sigma = \{0\}$, then $\Sigma^+ = \{0, 00, 000, 0000, 00000, \ldots\}$

1. 2 MATHEMATICAL INDUCTION

Based on general observations specific truths can be identified by reasoning. This principle is called mathematical induction. The proof by mathematical induction involves four steps.

Basis : This is the starting point for an induction. Here, prove that the result is true for some n = 0 or 1.

Induction Hypothesis : Here, assume that the result is true for n = k .

Induction step : Prove that the result is true for some n = k + 1.

Proof of induction step : Actual proof.

Example : Prove the following series by principle of induction $1+2+3+......+n=\frac{n(n+1)}{2}$

Solution :

Basis :

Let n = 1

L. H. S = 1 and R. H. S $=\frac{1(1+1)}{2}=1$

So the result is true for n = 1

Induction hypothesis :

By induction hypothesis we assume this result is true for n = k

i. e. $1+2+3+..........k=\frac{k(k+1)}{2}$

Inductive step :

We have to prove that the result is true for $n=k+1$

i. e. $1+2+3+........+k+k+1=\frac{(k+1)(k+1+1)}{2}$

Proof of induction step :

$$\begin{aligned}
\text{L. H. S} &= 1+2+3+........+k+k+1 \\
&= \frac{k\,(k+1)}{2}+k+1 \\
&= (k+1)\left(\frac{k}{2}+1\right) \\
&= \frac{(k+1)\,(k+2)}{2} \\
&= \frac{(k+1)\,(k+1+1)}{2} = R.H.S
\end{aligned}$$

Hence the proof.

1.3 FINITE AUTOMATA (FA)

A finite automata consists of a finite memory called input tape, a finite - nonempty set of states, an input alphabet, a read - only head , a transition function which defines the change of configuration, an initial state, and a finite - non empty set of final states.

A model of finite automata is shown in figure 1.1.

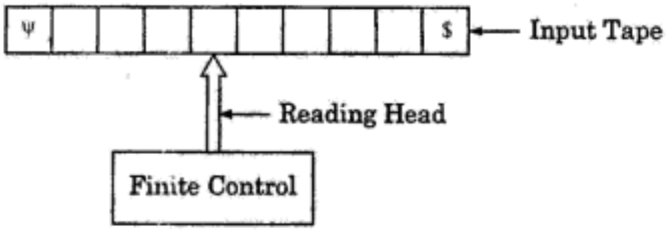


FIGURE 1.1 : Model of Finite Automata

The input tape is divided into cells and each cell contains one symbol from the input alphabet. The symbol 'ψ' is used at the leftmost cell and the symbol '$' is used at the rightmost cell to indicate the beginning and end of the input tape. The head reads one symbol on the input tape and finite control controls the next configuration. The head can read either from left - to - right or right - to -left one cell at a time. The head can't write and can't move backward. So , FA can't remember its previous read symbols. This is the major limitation of FA.

Deterministic Finite Automata (DFA)

A deterministic finite automata M can be described by 5 - tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is finite, nonempty set of states,
2. Σ is an input alphabet,
3. δ is transition function which maps $Q \times \Sigma \to Q$ i. e. the head reads a symbol in its present state and moves into next state.
4. $q_0 \in Q$, known as initial state
5. $F \subseteq Q$, known as set of final states.

Non - deterministic Finite Automata (NFA)

A non - deterministic finite automata M can be described by 5 - tuple $(Q, \Sigma, \delta, q_0, F)$, where

1 Q is finite, nonempty set of states,
2. Σ is an input alphabet,
3. δ is transition function which maps $Q \times \Sigma \rightarrow 2^Q$ i. e., the head reads a symbol in its present state and moves into the set of next state (s). 2^Q is power set of Q,
4. $q_0 \in Q$, known as initial state, and
5. $F \subseteq Q$, known as set of final states.

The difference between a DFA and a NFA is only in transition function. In DFA, transition function maps on at most one state and in NFA transition function maps on at least one state for a valid input symbol.

States of the FA

FA has following states :

1. **Initial state :** Initial state is an unique state ; from this state the processing starts.
2. **Final states :** These are special states in which if execution of input string is ended then execution is known as successful otherwise unsuccessful.
3. **Non - final states :** All states except final states are known as non - final states.
4. **Hang - states :** These are the states, which are not included into Q, and after reaching these states FA sits in idle situation. These have no outgoing edge. These states are generally denoted by ϕ. For example, consider a FA shown in figure1.2.

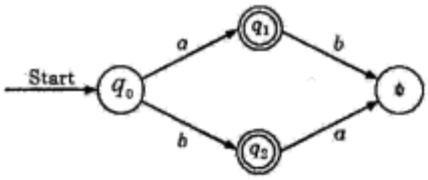


FIGURE 1.2 : Finite Automata

q_0 is the initial state, q_1, q_2 are final states, and ϕ is the hang state.

Notations used for representing FA

We represent a FA by describing all the five - terms $(Q, \Sigma, \delta, q_0, F)$. By using diagram to represent FA make things much clearer and readable. We use following notations for representing the FA :

1. The initial state is represented by a state within a circle and an arrow entering into circle as shown below :

   $\rightarrow q_0$ (Initial state q_0)

2. Final state is represented by final state within double circles :

   q_f (Final state q_f)

3. The hang state is represented by the symbol 'ϕ' within a circle as follows :

   ϕ

4. Other states are represented by the state name within a circle.
5. A directed edge with label shows the transition (or move). Suppose p is the present state and q is the next state on input - symbol 'a', then this is represented by

   $p \xrightarrow{a} q$

6. A directed edge with more than one label shows the transitions (or moves). Suppose p is the present state and q is the next state on input - symbols 'a_1' or 'a_2' or ... or 'a_n' then this is represented by $p \xrightarrow{a_1, a_2, \ldots, a_n} q$

Transition Functions

We have two types of transition functions depending on the number of arguments.

Transition Function

Direct (Represented by δ)

Indirect (Represented by δ')

Direct transition Function (δ)

When the input is a symbol, transition function is known as direct transition function.

Example : $\delta(p,a) = q$ (Where p is present state and q is the next state).
It is also known as one step transition.

Indirect transition function (δ')

When the input is a string, then transition function is known as indirect transition function.

Example : $\delta'(p,w) = q$, where p is the present state and q is the next state after | w | transitions. It is also known as one step or more than one step transition.

Properties of Transition Functions

1. If $\delta(p,a) = q$, then δ (p, ax) = δ(q, x) and if δ' (p, x) = q, then δ' (p, xa) = δ'(q, a)
2. For two strings x and y ; $\delta(p,xy) = \delta(\delta(p,x),y)$, and $\delta'(p,xy) = \delta'(\delta'(p,x),y)$

Example :1. A DFA $M = (\{q_0, q_1, q_2, q_f\}, \{0,1\}, \delta, q_0, \{q_f\})$ is shown in figure1.3 .

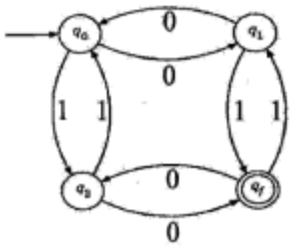


FIGURE 1.3 : Deterministic finite automata

Where δ is defined as follows :

	0	1
→ q_0	q_1	q_2
q_1	q_0	q_f
q_2	q_f	q_0
q_f	q_2	q_1

2. A NFA $M_1 = (\{q_0, q_1, q_2, q_f\}, \{0,1\}, \delta, q_0, \{q_f\})$ is shown in figure1.4.

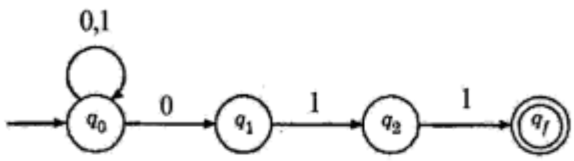


FIGURE 1.4 : Non - deterministic finite automata

Transition function δ is defined as follows :

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	-	$\{q_2\}$
q_2	-	$\{q_f\}$
q_f	-	$\{q_f\}$

Note : In first row of transition table, when present state is q_0 and input is '0', then there are two next states q_0 , and q_1 .

Acceptability of a string by DFA : Let a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and an input string $w \in \Sigma^*$. The string w is accepted by M if and only if $\delta(q_0, w) = q_f$, where $q_f \in F$.

When w is accepted by M, then the execution of string w ends in a final state and this execution is known as successful otherwise unsuccessful .

Example : Consider the DFA shown in figure1.5.

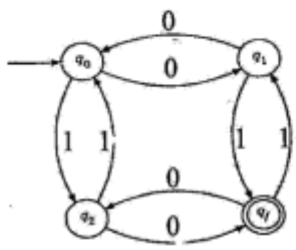


FIGURE 1.5 : Deterministic finite automata

Input strings are :

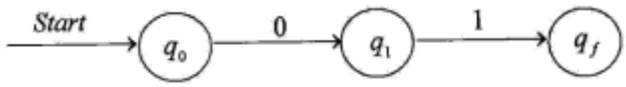
i) 01,

ii) 011

Check the acceptability of each string.

Solution :

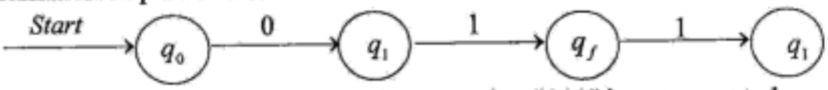
1. Let the input string $w_1 = 01$, the transition sequence is as follows :



Execution ends in final state q_f, hence string "01" is accepted.

2. Let input string $w_2 = 011$

The transition sequence is as follows :



Execution ends in non - final state q_1, hence string "011" is not accepted.

Acceptability of a string by NFA

Let a NFA $M = (Q, \Sigma, \delta, q_0, F)$ and an input string $w \in \Sigma^*$. The string w is accepted by M if and only if $\delta(q_0, w) = \{ q_i : q_i \in F$, for some i = 0, 1, ... , n $\}$.

When w is accepted by M, then the execution of string w ends in some final state and the execution is known as successful otherwise unsuccessful.

Example : Consider the NFA shown in figure1.6.

Check the acceptability of following strings : i) 011 ii) 010 iii) 011011

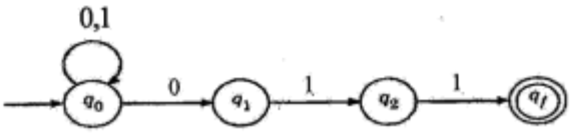
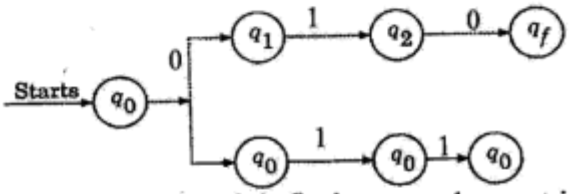


FIGURE 1.6 : Non - deterministic finite automata

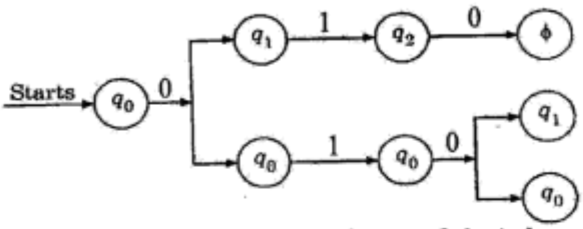
Solution :

1. Transition sequence for the string "011" is as follows :



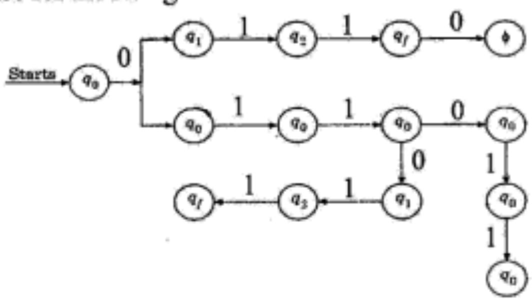
One execution sequence ends in final state q_f, hence string "011" is accepted.

2. Transition sequence for the string "010" is as follows :



The execution ends in non - final states q_0, q_1 and one ends in ϕ, hence string "010" is not accepted.

3. Transition sequence for the string "011011" is as follows :



One execution ends in hang state ϕ, second ends in non - final state q_0, and third ends in final state q_f hence string "011011" is accepted by third execution.

Difference between DFA and NFA

Strictly speaking the difference between DFA and NFA lies only in the definition of δ. Using this difference some more points can be derived and can be written as shown :

DFA	NFA
1. The DFA is 5 - tuple or quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where Q is set of finite states Σ is set of input alphabets $\delta : Q \times \Sigma$ *to* Q q_0 is the initial state $F \subseteq Q$ is set of final states	The NFA is same as DFA except in the definition of δ. Here, δ is defined as follows : $\delta : Q \times (\Sigma \cup \in)$ to subset of 2^Q
2. There can be zero or one transition from a state on an input symbol	There can be zero, one or more transitions from a state on an input symbol
3. No $\in -$ transitions exist i.e., there should not be any transition or a transition if exist it should be on an input symbol	$\in -$ transitions can exist i. e., without any input there can be transition from one state to another state.
4. Difficult to construct	Easy to construct

Example 1 : Consider the FA shown in below figure. Check the acceptability of following strings:

(a) 0101 (b) 0111 (c) 001

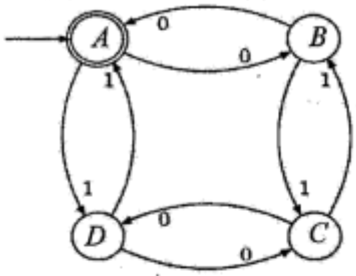
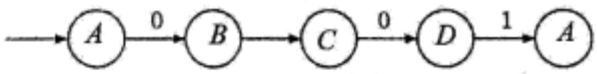


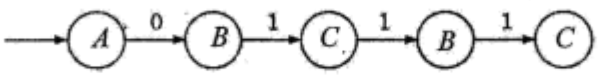
FIGURE : Finite automata

Solution : (a) The transition sequence for input string 0111 is following :



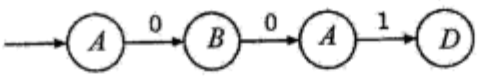
Execution ends in final state A, hence string 0101 is accepted.

(b) The transition sequence for input string 0111 is as follows :



Execution ends in non-final state C, hence string 0111 is not accepted.

(c) The transition sequence for input string 001 is as follows :



Execution ends in non-final state D, hence string 001 is not accepted

Example 2 : Let a DFA $M = (Q, \Sigma, \delta, q_0, F)$ is shown in below figure.

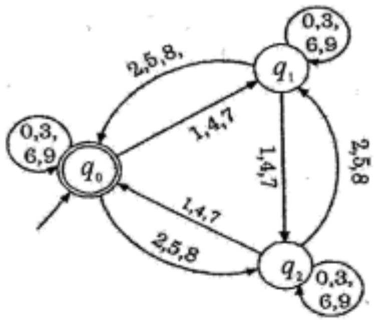


FIGURE : D F A

Check that string 33150 is recognized by above DFA or not ?

Solution :

For string 33150 the transition sequence is as follows :

$$\longrightarrow q_0 \xrightarrow{3} q_0 \xrightarrow{3} q_0 \xrightarrow{1} q_1 \xrightarrow{5} q_0 \xrightarrow{0} q_0$$

Since, transition ends in final state, q_0, so string 33150 is recognized.

Example 3 : Consider below transition diagram and verify whether the following strings will be accepted or not ? Explain.

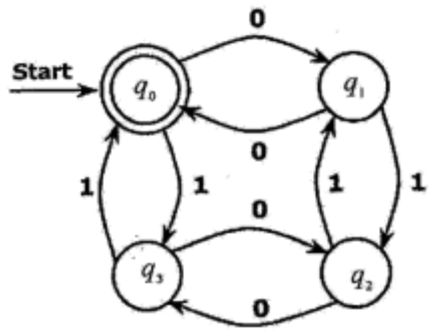


FIGURE : Given Transition Diagram

i) 0011   ii) 010101   iii) 111100   iv) 1011101 .

Solution : Transition table for the given diagram is ,

	0	1
$\rightarrow$ q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

TABLE : Transition Table for the given Transition Diagram

i) 0011

$\delta(q_0, 0011) \vdash \delta(q_1, 011)$

$\vdash \delta(q_0, 11)$

$\vdash \delta(q_3, 1)$

$\vdash q_0$

$\therefore$ 0011 is accepted.

ii) 010101

$\delta(q_0, 010101) \vdash \delta(q_1, 10101)$

$\vdash \delta(q_2, 0101)$

$\vdash \delta(q_3, 101)$

$\vdash \delta(q_0, 01)$

$\vdash \delta(q_1, 1)$

$\vdash q_2$

$\therefore$ 010101 is not accepted.

iii) 111100

$\delta(q_0, 111100) \vdash \delta(q_3, 11100)$

$\vdash (q_0, 1100)$

$\vdash \delta(q_3, 100)$

$\vdash \delta(q_0, 00)$

$\vdash \delta(q_1, 0)$

$\vdash q_0$

$\therefore$ 111100 is accepted.

iv) 1011101

$\delta(q_0, 1011101) \vdash \delta(q_3, 011101)$

$\vdash \delta(q_2, 11101)$

$\vdash \delta(q_1, 1101)$

$\vdash \delta(q_2, 101)$

$\vdash \delta(q_1, 01)$

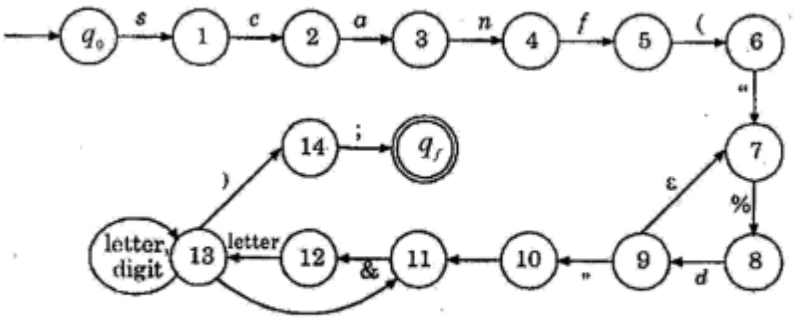
$\vdash \delta(q_0, 1)$

$\vdash q_3$

$\therefore$ 1011101 is not accepted.

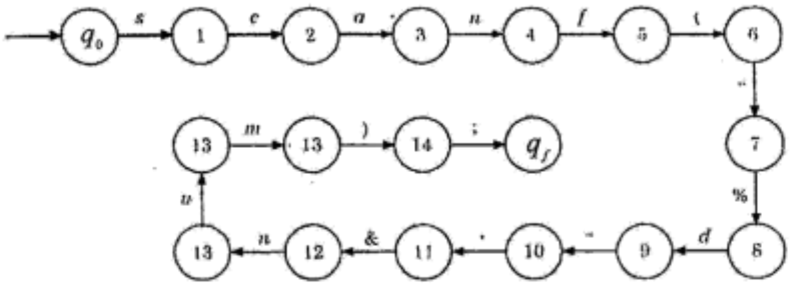
Example 4 : Consider the NFA shown in below figure. Check the acceptability of following string

scanf ("%d", & num) ;



Note : Letter stands for any symbol from { a, b, ... , z } and digit stands for any digit from { 0, 1, 2, ... 9 } .

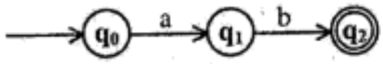
Solution : The transition sequence for given string : **scanf("%d", & num) ;**



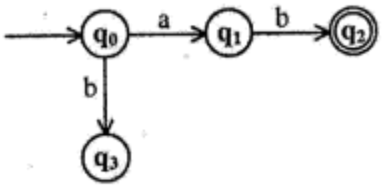
Since, execution of given string ends in final state q_f, so the string is recognized.

Example 5 : Obtain a DFA to accept strings of a's and b's starting with the string ab .

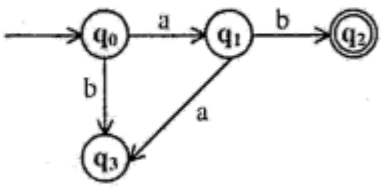
Solution :
From the problem it is clear that the string should start with ab and so, the minimum string that can be accepted by the machine is ab. To accept the string ab, we need three states and the machine can be written as



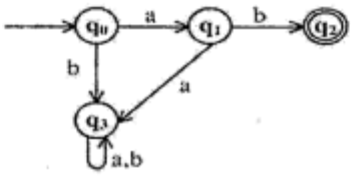
where q_2 is the final or accepting state. In state q_0, if the input symbol is b, the machine should reject b (note the string should start with a) . So, in state q_0, on input b, we enter into the rejecting state q_3. The machine for this can be of the form



The machine will be in state q_1, if the first input symbol is a. If this a is followed by another a, the string aa should be rejected by the machine . So, in state q_1, if the input symbol is a , we reject it and enter into q_3 which is the rejecting state. The machine for this can be of the form



Whenever the string is not starting with ab, the machine will be in state q_3 which is the rejecting state. So, in state q_3, if the input string consists of a's and b's of any length, the entire string can be rejected and can stay in state q_3 only. The resulting machine can be of the form



The machine will be in state q_2, if the input string starts with ab. After the string ab, the string containing any combination of a's and b's, can be accepted and so remain in state q_2 only. The complete machine to accept the strings of a's and b's starting with the string ab is shown in below figure. The state q_3 is called dead state or trap state or rejecting state.

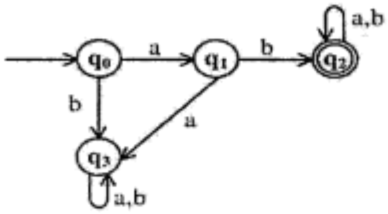


FIGURE : Transition diagram to accept string ab (a + b)*

So, the DFA which accepts strings of a's and b's starting with the string ab is given by $M = (Q, \Sigma, \delta, q_0, F)$

where $Q = \{q_0, q_1, q_2\}$; $\Sigma = \{a, b\}$;

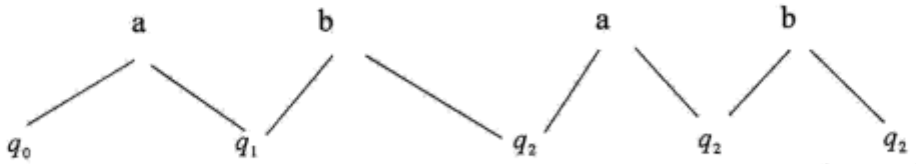
q_0 is the start state ; $F = \{q_2\}$

δ is shown the transition table.

States	δ	a	b
	$\rightarrow q_0$	q_1	q_3
	q_1	q_3	q_2
	q_2	q_2	q_2
	q_3	q_3	q_3

TABLE : Transition table for DFA shown in above figure

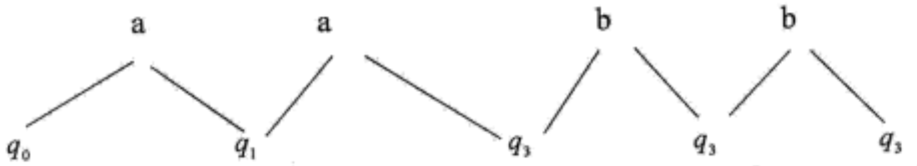
To accept the string abab : This string is accepted by the machine and is evident from the below figure.



[accepting state]

FIGURE : To accept the string abab

Here, $\delta^*(q_0, abab) = q_2$ which is the final state. So, the string abab is accepted by the machine.
To reject the string aabb : The string is rejected by the machine and is evident from the below figure .



[non - accepting state]

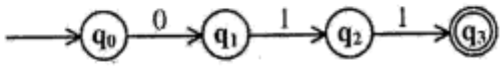
FIGURE : To reject the string aabb

Here, $\delta^*(q_0, aabb) = q_3$ which is not an accepting state. So, the string aabb is rejected by the machine.

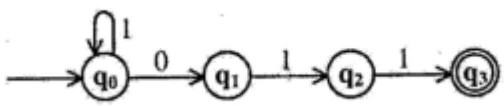
Example 6 : Draw a DFA to accept string of 0's and 1's ending with the string 011.

Solution :

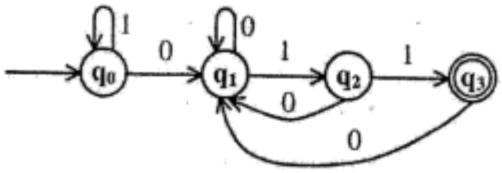
The minimum string that can be accepted by the machine is 011. It requires four states with q_0 as the start state and q_3 as the final state as shown below.



In state q_0, suppose we input the string 1111 ..... 011. Since the string ends with 011, the entire string has to be accepted by the machine. To accept the string 011 finally, the machine should be in state q_0. So, on any number of 1's the machine stays only in state q_0 and if the string ends with 011, the machine enters into the final state. The machine can be of the form



If the machine is in any of the states q_1, q_2 and q_3 and if the current input symbol is 0 and if the next input string is 11, the entire string should be accepted. This is because the string ends with 011. So, from all these states on the input symbol 0, there should be a transition to state q_1 so that if we enter the string 11 we can reach the final state. Now the machine can take the form as shown below.



In state q_3, if the input symbol is 1, enter into state q_0 so that if the next input string is 011, we can enter into the final state q_3. So, the final machine which accepts a string of 0's and 1's ending with the string 011 can take the following form.

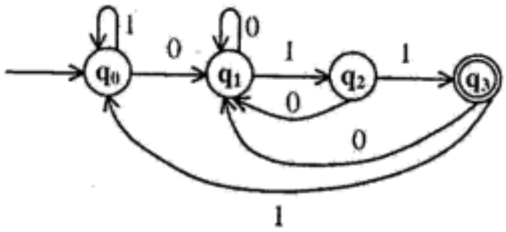


FIGURE : transition diagram to accept (0+1) *011

So, the DFA which accepts strings of 0's and 1's ending with the string 011 is given by $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3\}$; $\Sigma = \{0, 1\}$;

q_0 is the start state; $F = \{q_3\}$;

δ is shown using the transition table.

States	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3 (final)	q_1	q_0

TABLE : Transition table for the machine shown in above figure

To accept the string 0011 : This string is accepted by the machine and is evident from the below figure . Here, $\delta^*(q_0,0011)=q_3$ which is the final state. So, the string 0011 is accepted by the machine.

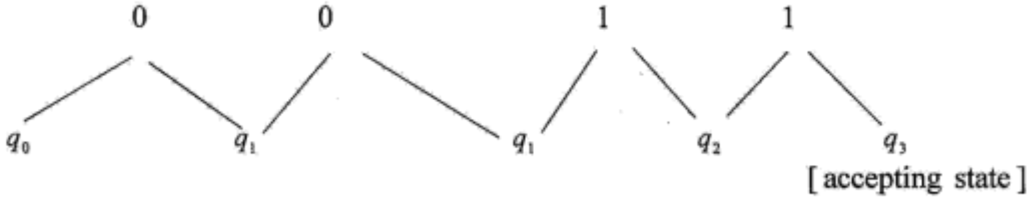


FIGURE : To accept the string 0011

To reject the string 0101 : The string is rejected by the machine and is evident from the below figure .

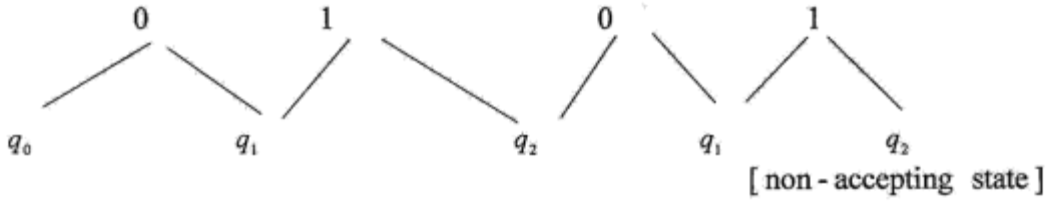


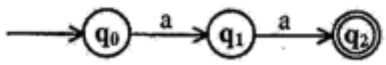
FIGURE : To reject the string 0101

Here, $\delta^*(q_0,0101)=q_2$ which is not an accepting state. So, the string 0101 is rejected by the machine.

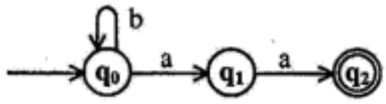
Example 7 : Obtain a DFA to accept strings of a's and b's having a substring aa .

Solution :

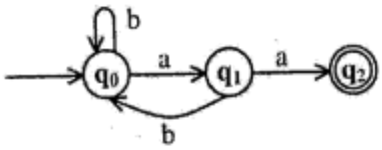
The minimum string that can be accepted by the machine is aa. To accept exactly two symbols, the DFA requires 3 states and the machine to accept the string aa can take the form



where q_0 is the start state and q_2 is the accepting state. In state q_0, if the input symbol is b, stay in q_0 so that when any number of b's ends with aa, the entire string is accepted. The machine for this can be of the form



There is a transition to state q_1 on input symbol a. In state q_1, if the input symbol is b, there will be a transition to state q_0 so that if this b is followed by aa, the machine enters into state q_2 so that the entire string is accepted by the machine. The transition diagram for this can be of the form



The machine enters into state q_2 when the string has a sub string aa. So, in this state even if we input any number of a's and b's the entire string has to be accepted. So, the machine should stay in q_2. The final machine which accepts strings of a's and b's having a sub string aa is shown in below figure

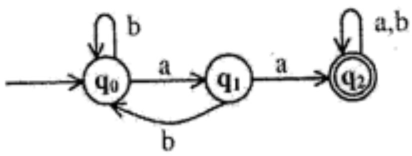


FIGURE : transition diagram to accept (a+b)* aa(a+b)*

The machine $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2\}$; $\Sigma = \{a, b\}$

q_0 is the start state; $F = \{q_2\}$

δ is shown using the transition table.

	δ	a	b
↑	$\rightarrow q_0$	q_1	q_0
States	q_1	q_2	q_0
↓	q_2	q_2	q_2

TABLE : Transition table for the machine shown in above figure

To accept the string baab : This string is accepted by the machine and is evident from the below figure.

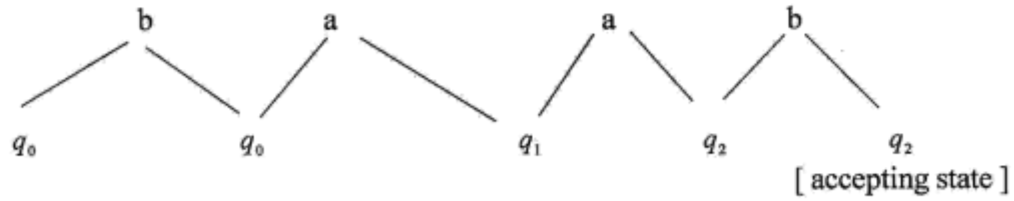


FIGURE : To accept the string baab

Here, $\delta^*(q_0, baab) = q_2$ which is the final state. So, the string baab is accepted by the machine. The string baba is rejected by the machine and is evident from the below figure.

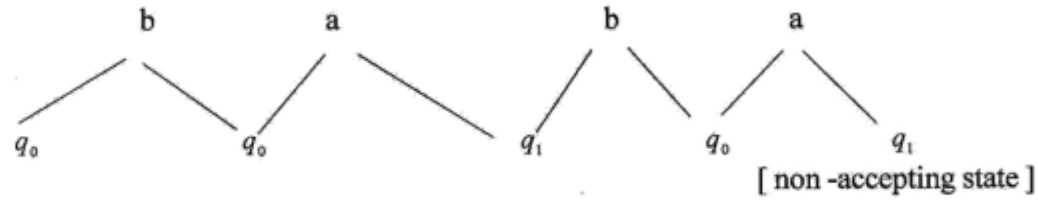


FIGURE : To reject the string baba

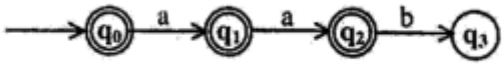
Here, $\delta^*(q_0, baba) = q_1$ which is not an accepting state. So, the string baba is rejected by the machine.

Example 8 : Obtain a DFA to accept strings of a's and b's except those containing the substring aab.

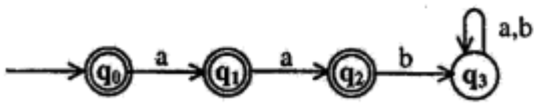
Solution :

Note : This can be solved in two ways. The first method is similar to the previous problem i. e., draw a DFA to accept strings of a's and b's having a substring aab. Then change the final states to non - final states and non final states to final states. The resulting machine will accept the strings of a's and b's except those containing the sub - string aab.

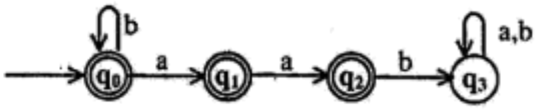
Here, the second method is explained. The minimum string that can be rejected by the machine is aab. To reject this string we need four states q_0, q_1, q_2 and q_3. Since the string aab has to be rejected, q_3 can not be the final state and the rest of the states will be the final states as shown below.



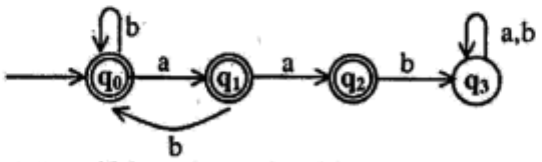
The machine enters into q_3 if the string has a sub string aab. In this state if we input any number of a's or / and b's, the entire string has to be rejected. So, stay in the state q_3 only. The machine for this is shown below.



In state q_0, if the input symbol is b, stay in q_0 so that if this b is followed by aab, the machine enters into state q_3 so that the string is rejected. The machine for this is shown below.



In state q_1, if the input symbol is b, enter into state q_0, so that if this b ends with the string aab, the entire string is rejected. The machine for this is shown below.



The machine will be in state q_2 if the string ends with aa. At this stage, if the input symbol is a, again the string ends with aa and so stay in state q_2 only. The complete machine to accept strings of a's and b's except those containing the sub string aab is shown below.

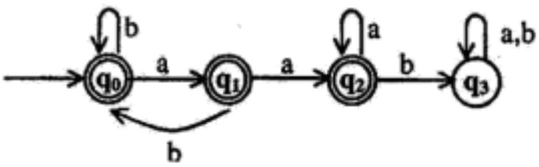


FIGURE : DFA to accept the string except the sub string aab.

So, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3\}$; $\Sigma = \{a, b\}$

q_0 is the start state ; $F = \{q_0, q_1, q_2\}$

δ is shown using the transition table

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_3
q_3	q_3	q_3

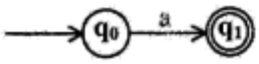
(← States →; ← Σ →)

TABLE : Transition table

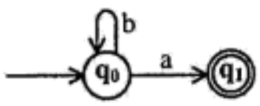
Example 9 : Obtain a DFA to accept strings of a's and b's having exactly one a, atleast one a, not more than three a's.

Solution :

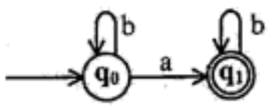
To accept exactly one a : To accept exactly one a, we need two states q_0 and q_1 and make q_1 as the final state. The machine to accept one a is shown below.



In q_0, on input symbol b, remain q_0 only so that any number of b's can end with one a. The machine for this can be of the form



In state q_1, on input symbol b remain in q_1 and the machine can take the form



But, in state q_1, if the input symbol is a, the string has to be rejected as the machine can have any number of b's but exactly one a. So, the string has to be rejected and we enter into a trap state q_2. Once the machine enters into trap state, there is no way to come out of the state and the string is rejected by the machine. The complete machine is shown in below figure.

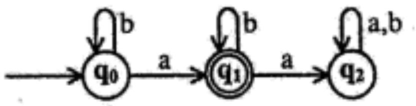


FIGURE : DFA to accept exactly one a

The machine $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{ q_0, q_1, q_2 \}$; $\Sigma = \{ a, b \}$

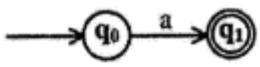
q_0 is the start state; $F = \{q_1\}$

δ is shown below using the transition table.

		← Σ →	
	δ	a	b
← States →	→ q_0	q_1	q_0
	q_1	q_2	q_1
	q_2	q_2	q_2

TABLE : Transition table

The machine to accept at least one a : The minimum string that can be accepted by the machine is a. For this, we need two states q_0 and q_1 where q_1 is the final state. The machine for this is shown below.



In state q_0, if the input symbol is b, remain in q_0. Once the final state q_1 is reached, whether the input symbol is a or b, the entire string has to be accepted. The machine to accept at least one a is shown in below figure.

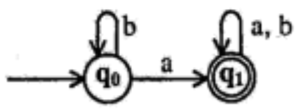


FIGURE : DFA to atleat one a

The machine $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1\}$; $\Sigma = \{a, b\}$
q_0 is the start state ; $F = \{q_1\}$
δ is shown using the transition table .

δ	$\leftarrow \Sigma \rightarrow$ a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_1

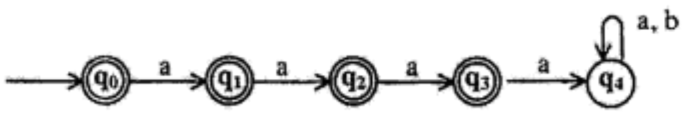
TABLE : Transition table

The machine to accept not more than three a's : The machine should accept not more than three a's means

- It can accept zero a's i. e., no a's
- It can accept one a
- It can accept two a's
- It can accept 3 a's
- But, it can not accept more than three a's.

In this machine maximum of three a's can be accepted i. e., the machine can accept zero a's, one a, two a's or three a's. So, we need maximum four states q_0, q_1, q_2 and q_3 where all these states are final states and q_0 is the start state. The machine can take the form

In state q_3, if the input symbol is a, the string has to be rejected and we enter into a trap state q_4. Once this trap state is reached, whether the input symbol is a or b, the entire string has to be rejected and remain in state q_4. Now, the machine can take the form as shown below.



In state q_0, q_1, q_2 and q_3, if the input symbol is b, stay in their respective states and the final transition diagram is shown in below figure.

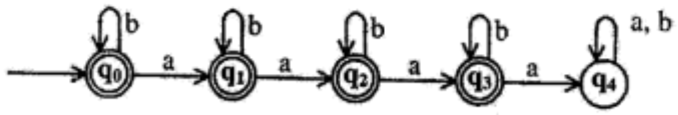


FIGURE : DFA to accept not more than 3 a's

The DFA $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3, q_4\}$; $\Sigma = \{a, b\}$

q_0 is the start state; $F = \{q_0, q_1, q_2, q_3\}$

δ is shown using the transition table.

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_3
q_4	q_4	q_4

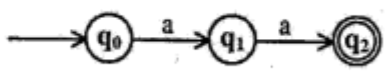
TABLE : Transition table for DFA shown in above figure

Example 10 : Obtain a DFA to accept the language $L = \{ awa \mid w \in (a+b)^* \}$.

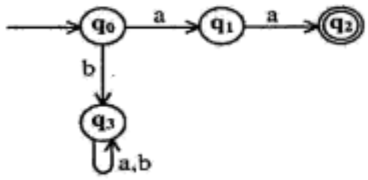
Solution :

Here, $w \in (a+b)^*$ indicates the string consisting of a's and b's of any length including the null string. So, the language accepted by DFA is a string which starts with a, followed by a string of a's and b's (possibly including $\in$) of any length and followed by one a.

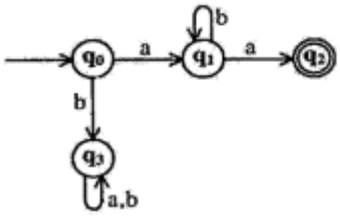
If w is $\in$ (null string), the minimum string that can be accepted by the machine is aa and so, we need three states q_0, q_1 and q_2 to accept the string. The machine can be of the form



where q_0 is the start state and q_2 is the final state. In state q_0, if the input symbol is b, the string has to be rejected and so, we enter into a trap state q_3. Once the machine enters into trap state, whether the input is either a or b, the string has to be rejected and the machine for this is shown below.



In state q_1, if the input symbol is b, remain in q_1 and the machine takes the form



In state q_2, if the input symbol is a, the string ends with a and so remain in q_2. In state q_2, if the input symbol is b, enter into state q_1 so that after inputting the symbol a, the machine enters into q_2. The complete machine is shown in below figure.

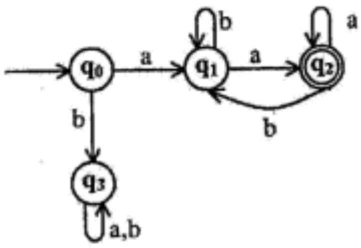


FIGURE : DFA to accept awa

So, the machine $M = (Q, \Sigma, \delta, q_0, F)$ where $Q = \{q_0, q_1, q_2, q_3\}$; $\Sigma = \{a, b\}$

q_0 is the start state ; $F = \{q_2\}$

δ is shown using the transition table.

δ	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_1
q_2	q_2	q_1
q_3	q_3	q_3

TABLE : Transition table for DFA shown in above figure

Example 11 : Obtain a DFA to accept even number of a's, odd number of a's.

Solution :

The machine to accept even number of a's is shown in figure (a) and odd number of a's is shown in figure(b).

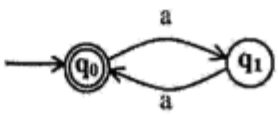


Figure : (a)

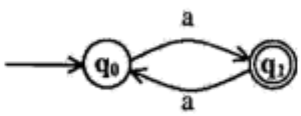


Figure : (b)

Example 12 : Obtain a DFA to accept strings of a's and b's having even number of a's and b's.

Solution :

The machine to accept even number of a's and b's is shown in figure 1.

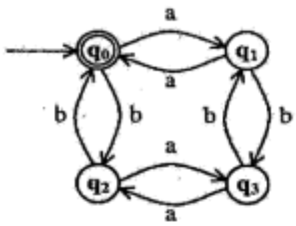


FIGURE 1 : DFA to accept even no. of a's and b's

Note : In the DFA shown in figure 1, instead of making q_0 as the final state, make q_2 as the final state. The DFA to accept even number of a's and odd number of b's is obtained and is shown in figure 2.

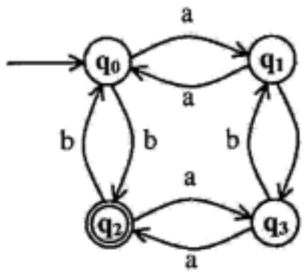


FIGURE 2 : DFA to accept even no. of a's and odd number of b's

Note : In the DFA shown in figure 1, instead of making q_0 as the final state, make q_1 as the final state. The DFA to accept odd number of a's and even number of b's is obtained and is shown in figure 3 .

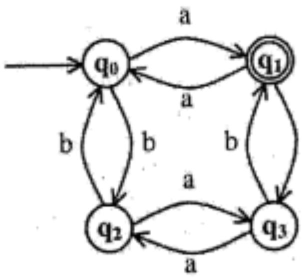


FIGURE 3 : DFA to accept odd no. of a's and even number of b's

Note : In the DFA shown in figure 1, instead of making q_0 as the final state, make q_3 as the final state. The DFA to accept odd number of a's and odd number of b's is obtained and is shown in figure 4.

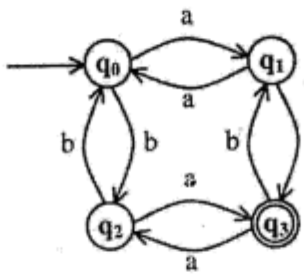


FIGURE 4 : DFA to accept odd no. of a's and odd number of b's

Example 13 : Design a DFA, M that accepts the language $L(M) = \{ w | w \in \{a, b\}^* \}$ and w does not contain 3 consecutive b's.

Solution :

We first consider a language $L_1(M) = \{ w | w \in \{a, b\}^* \}$ and w contain 3 consecutive b's.

Then DFA for L_1 is,

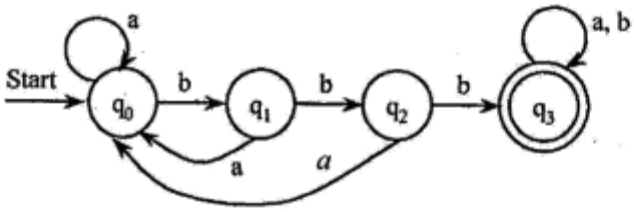


FIGURE : (A)

Now we can get language L(M) by converting non - final states to final states and final states to non - final states.

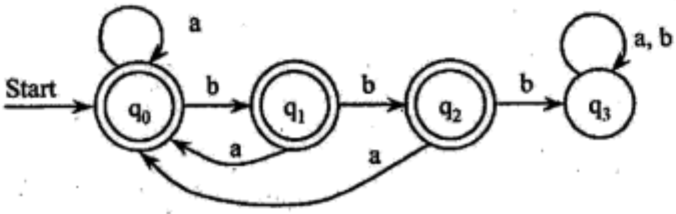


FIGURE : (B)

FIGURE : Construction of DFA from the language $L = \{ w | w \in \{a, b\}^* \}$

Example 14 : Design DFA which accepts language L = { 0, 000, 00000, ... } over { 0 } .

Solution : L = { 0, 000, 00000 , ... } over { 0 } means L accepts the strings of odd number of 0's. So the DFA for L is ,

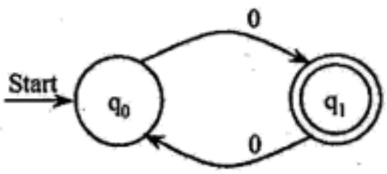
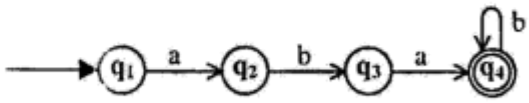


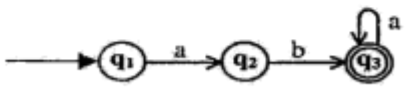
FIGURE : DFA for the given language L.

Example 15 : Obtain an NFA to accept the following language $L = \{ w | w \in abab^n$ or aba^n where $n \geq 0 \}$

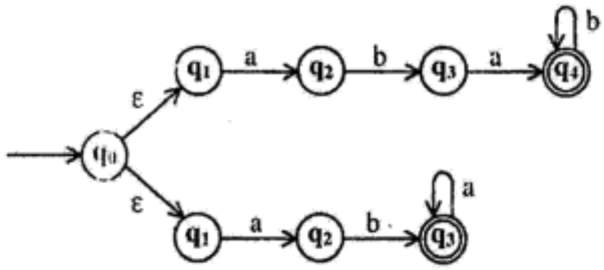
Solution : The machine to accept $abab^n$ where $n \geq 0$ is shown below :



The machine to accept aba^n where $n \geq 0$ is shown below :



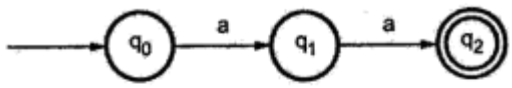
The machine to accept either $abab^n$ or aba^n where $n \geq 0$ is shown below :



Example 16 : Design NFA to accept strings with a's and b's such that the string end with 'aa'.

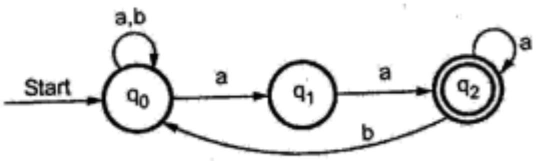
Solution :

Method - I : The simple FA which accepts a string with 'aa' is



Now there can be a situation where in

Hence we can design a required NFA as



It can be denoted by ,

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

We can test some strings for above drawn NFA.

Consider

$$\begin{aligned} \delta(q_0, aaa) &\vdash \delta(q_0, aa) \\ &\vdash \delta(q_1, a) \\ &\vdash \delta(q_2, \in) \end{aligned}$$

i. e. we reach to final state.

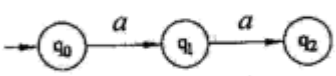
$$\begin{aligned} \delta(q_0, aaa) &\vdash \delta(q_0, aa) \\ &\vdash \delta(q_0, a) \\ &\vdash \delta(q_1, \in) \end{aligned}$$

i.e. we are not in final state.

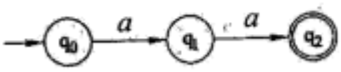
Thus there are two possibilities by which we move with string 'aaa' in above given NFA.

Method - II

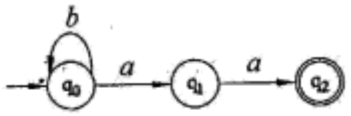
Start with two consecutive a's initially. It requires three states q_0, q_1 and q_2 respectively. Consider q_0 as the initial state



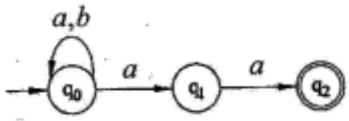
Assign q_2 as final state so that it accepts two consecutive a's



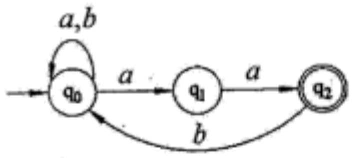
Design such a way if any number of b's preceeds first a it should be in the same state i.e., in the state q_0.



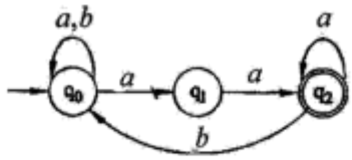
Design such a way if a's preceed by first a it should move from q_0 to q_0 only i. e., it will be in the state.



After the second a, b comes it has to move from q_2 to q_0.



Any number of a's followed by second a then it will be in the same state q_2.



The transition table is

	a	b
q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	ϕ
q_2	q_2	q_1

Test for the strings which ends with two consecutive a's.

String baa :

$\delta(q_0, baa)$ $\vdash \delta(q_0, aa)$

$\vdash \delta(q_1, a)$

$\vdash \delta(q_2, \in)$

$\vdash q_2 \in F$

String baa :

$\delta(q_0, baa)$ $\vdash \delta(q_0, aa)$

$\vdash \delta(q_0, a)$

$\vdash \delta(q_1, \in)$

$\vdash q_1 \notin F$

$\therefore$ NFA and two possibilities for the same input also shown.

String aab :

$$\begin{aligned}\delta(q_0,aab) &\mid- \delta(q_1,ab)\\ &\mid- \delta(q_2,b)\\ &\mid- \delta(q_1,\in)\\ &\mid- q_1 \notin F\end{aligned}$$

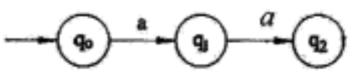
$\therefore$ If the string is not ending with two consecutive a's it will not be accepted.

String aaa :

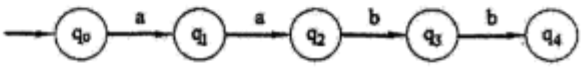
$$\begin{aligned}\delta(q_0,aaa) &\mid- \delta(q_0,aa)\\ &\mid- \delta(q_1,a)\\ &\mid- \delta(q_2,\in)\\ &\mid- q_2 \in F\end{aligned}$$

Example 17 : Design an NFA to accept a language of all strings with double 'a' followed by double 'b'.

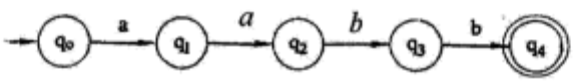
Solution : First design an NFA with three states q_0, q_1, q_2 and in which q_0 is the initial state to accept the string with two a's.



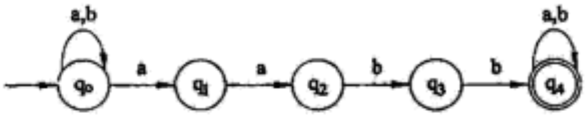
In second step we have to add another two states for the following two b's as shown below. Those states are q_3 and q_4



In the third step we assign q_4 as final state.



It can accept any number of a's or b's before first two successive a's. In the same way after the two successive b's also it can accept any number of a's or b's.



The NFA is defined as below :

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

where $Q = \{q_0, q_1, q_2, q_3, q_4\}$; $\Sigma = \{a, b\}$

$F = \{q_4\}$ and the transition table is given below :

	a	b
$\to q_0$	$\{q_0, q_1\}$	q_0
q_1	q_2	ϕ
q_2	ϕ	q_3
q_3	ϕ	q_4
q_4	q_4	q_4

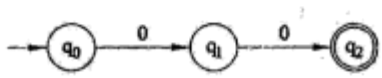
Consider the string aaa bb :

$$\begin{aligned} \delta(q_0, aaa\ bb) &\vdash \delta(q_0, aa\ bb) \\ &\vdash \delta(q_1, a\ bb) \\ &\vdash \delta(q_2, bb) \\ &\vdash \delta(q_3, b) \\ &\vdash \delta(q_4, \in) \\ &\vdash q_4 \in F \end{aligned}$$

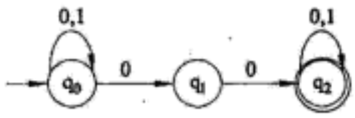
$\therefore$ $aaa\ bb \in L(M)$

Example 18 : Design an NFA to accept strings with 0's and 1's such that string contains two consecutive 0's or two consecutive 1's.

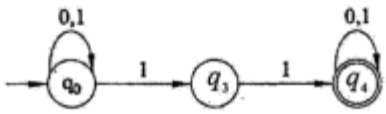
Solution : First we design NFA to accept two consecutive 0's . This



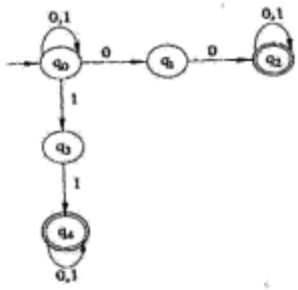
Next we can have any number of 0's and 1's before and after two consecutive zeros. i.e.,



then similarly NFA for accepting two consecutive 1's is



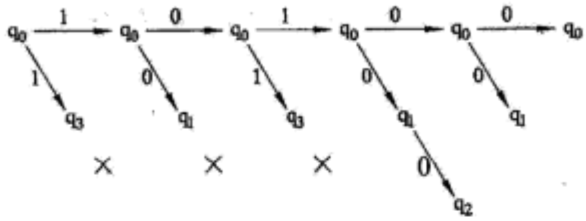
Combining above two designs



Transition table is

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_1	q_2	ϕ
q_2	q_2	q_2
q_3	ϕ	q_4
q_4	q_4	q_4

Checking 10100 string with NFA.

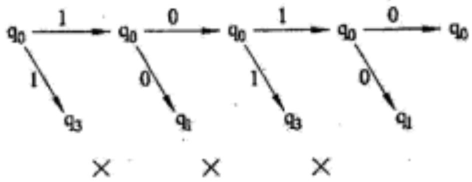


Observing above graph there are three completed paths for the string 10100. They are

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

In all these three couple paths one path is ending with final state (q_2 *or* q_4). So, the string 10100 is accepted (It contains two consecutive 0's).

Now considering another stirng 1010, then graph becomes



There are two completed paths. But no path is ending with final state (q_2 *or* q_4). So, the string 1010 is not accepted (because it does n't contain two consecutive 0's or 1's).

1.3 EQUIVALENCE OF NFA AND DFA

As we have discussed in comparison of NFA and DFA that the power of NFA and DFA is equal. It means that if a NFA M_1 accepts language L, then some DFA M_2 also accepts it and vice - versa.

In this section, we will discuss about the equivalence of NFA and DFA. It is obvious that all DFA are NFA from NFA definition. We will see this in the following theorem.

Theorem 1.3 .1 : All DFA are NFA.

Proof : While discussing the proof, we will concentrate on two things :

1. How to construct the target NFA ? And
2. The acceptability should be same for both.

Step 1 : Construction of the target NFA from given DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the given DFA and $M_1 = (Q_1, \Sigma, \delta_1, s, F_1)$ be the target NFA, then

1. $Q_1 = Q$ (States of DFA are same for NFA),
2. Σ is same for both,
3. $\delta_1 = \delta$, it means, whatever transition function given for DFA M is same for the target NFA M_1.

We also see that
For DFA M : Transition function is defined as $Q \times \Sigma \to Q$, and

For NFA M_1 : Transition function is defined as $Q_1 \times \Sigma \to 2^{Q_1}$

So, $(Q \times \Sigma \to Q) \subseteq (Q_1 \times \Sigma \to 2^{Q_1})$ or $Q \subseteq 2^{Q_1}$

4. $s = q_0$ (Same starting point or initial state)
5. $F_1 = F$ (Same terminating points or final states)

Step 2 : The acceptability of DFA and NFA : Let w be an input string and accepted by DFA

M and $w \in \Sigma^*$ if and only if $\delta'(q_0, w) = q_f, q_f \in F$ (δ' is indirect transition function)

For equivalent NFA M_1

$$\delta'_1 (s, w) = \delta'(q_0, w) = q_f, q_f \in F$$

(By construction definition $\delta_1 = \delta$, $s = q_0$, $F_1 = F$ and δ'_1 is indirect transition function for NFA).

Thus, NFA M_1 also accepts w.

It means, $L(M_1) \subseteq L(M)$ (1)

Now, let w is accepted by NFA M_1 if and only if $\delta'_1(s, w) = \delta'_1(q_0, w) = q_f, q_f \in F_1$ and by construction definition $\delta_1 = \delta$, $s = q_0$, $F_1 = F$ and δ'_1 is indirect transition function for NFA.

So, for DFA M $\delta'(q_0, w) = q_f, q_f \in F$ (δ' is indirect transition function)

Thus, DFA M also accepts w.

Hence, $M(L) \subseteq L(M_1)$ (2)

Therefore, all DFA are NFA. (From (1) and (2))

Example : Let a DFA $M = (Q, \Sigma, \delta, q_0, F)$ as shown in below figure. Find an equivalent NFA.

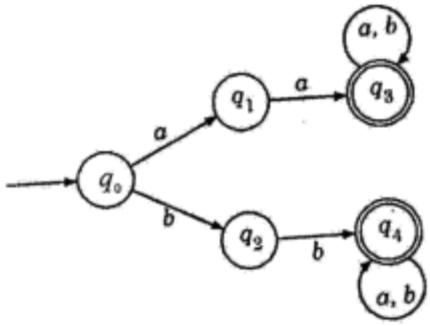


FIGURE : D F A

Solution :

Let equivalent NFA $M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$ where $Q_1 = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{a, b\}$, $F_1 = \{q_3, q_4\}$, δ_1 is defined below.

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_3	-
q_2	-	q_4
q_3	q_3	q_3
q_4	q_4	q_4

Theorem 1.3.2 : If there is a NFA M, then there exists equivalence DFA M_1 that has equal string recognizing power.

Proof : While discussing the proof, we will concentrate on two things :

1. How to construct the equivalent DFA ? And
2. The acceptability should be same for both.

Step 1 : Construction of the equivalent DFA M_1 from given NFA M

In NFA, zero, one or more next states are possible on a particular input. When we have more than one next state then we group all next states into one as $[q_1, q_2, q_3]$ and we call it one next state for equivalent DFA.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the given NFA and $M_1 = (Q_1, \Sigma, \delta_1, s, F_1)$ be the equivalent DFA, then

1. $Q_1 \subseteq 2^Q$ (2^Q is the power set of the set Q.),
2. Σ is same for both,
3. $s = [q_0]$ is initial state for M_1,
4. $F_1 \subseteq 2^Q$ such that each member of F_1 has at least one final state from F.
5. δ_1 is constructed as follows :

   Let $w = a \in \Sigma$ and

   If for given NFA $M : \delta(q_0, a) = \{q_1, q_2, \ldots\ldots\ldots\ldots q_n\}$, then

   For equivalent DFA $M_1 : \delta_1([q_0], a) = [q_1, q_2, \ldots\ldots\ldots\ldots q_n]$

And

   If for NFA $M : \delta(\{q_1, q_2, \ldots\ldots, q_n\}, a) = \{q_1, q_2, \ldots, q_m\}$, then

   For equivalent DFA $M_1 : \delta_1([q_1, q_2, \ldots\ldots, q_n], a) = [q_1, q_2, \ldots, q_m]$

Note : $[q_1, q_2, \ldots\ldots, q_i]$ denotes a single state for equivalent DFA.

Step 2 : The acceptability of DFA and NFA

We use the mathematical induction method to prove that $L(M) \subseteq L(M_1)$ and $L(M_1) \subseteq L(M)$ for all input strings $w \in \Sigma^*$.

Case 1 : Let $|w| = 0$, it means, $w = \in$ (Null string)

Let w is accepted by NFA M if and only if

$\delta(q_0, \in) = q_0$, and $q_0 \in F$ (Starting state is final state).

So, the initial state of DFA will be the final state, hence $w = \in$ is accepted by DFA also.

Case 2 : Let | w | = 1 and $w = a \in \Sigma$ is accepted by NFA M, then for NFA $M : \delta(q_0, a) = \{q_1, q_2, \ldots, q_n\}$, and $\{q_1, q_2, \ldots, q_n\}$ has at least one final state, then by constructive proof of equivalent DFA M_1 :

$\delta_1([q_0], a) = [q_1, q_2, \ldots, q_n]$ and $[q_1, q_2, \ldots, q_n]$ has at least one final state, so $[q_1, q_2, \ldots, q_n]$ is a final state for equivalent DFA M_1.

Therefore, the equivalent DFA M_1 also accepts $w = a$.

Case 3 : Suppose | w | = n and $w = a_1 a_2 \ldots a_n$ is accepted by both M and M_1 and

For NFA $M : \delta'(q_0, a_1 a_2 \ldots a_n) = \{q_1, q_2, \ldots, q_m\}$, and

For equivalent DFA M_1: $\delta'_1 = ([q_0], a_1 a_2 \ldots a_n) = [q_1, q_2, \ldots, q_m]$

Case 4 : Let | w | = n + 1 and w = yb

Where $|y| = n$, $y = a_1 a_2 \ldots a_n$ and $y, b \in \Sigma^*$ is accepted by NFA M If and only if

For NFA $M : \delta'(q_0, a_1 a_2 \ldots a_n b) = \delta(\{q_1, q_2, \ldots, q_m\}, b) = \{q_1, q_2, \ldots, q_p\}$,

($\{q_1, q_2, \ldots, q_p\}$ has at least one final state from the set F).

By constructive proof of equivalent DFA M_1

$\delta'_1([q_0], a_1 a_2 \ldots a_n b) = \delta_1([q_1, q_2, \ldots, q_m], b) = [q_1, q_2, \ldots, q_p]$

$[q_1, q_2, \ldots, q_p]$ contains one final state from F, thus it is a final state for equivalent DFA M_1.

Therefore, M_1 also accepts the string $w = yb$.

(δ', δ'$_1$ are indirect transition functions for NFA M and DFA M_1 respectively.)

It has been proved that if NFA M accepts w then DFA M_1 also accepts w for any arbitrary string w.

Thus, $L(M_1) \subseteq L(M)$. (1)

Similarly, we can prove that if equivalent DFA M_1 accepts any string $w \in \Sigma^*$, then NFA also accepts it.

Thus, $M(L) \subseteq L(M_1)$. (2)

Hence, the statement of Theorem 1.3.2 is true. (From (1) and (2))

Example 1 : Consider a NFA shown in below figure. Find equivalent DFA.

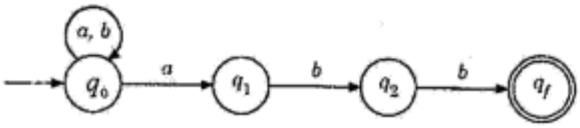


FIGURE : Non - deterministic finite Automata

Solution : Let given NFA $M = (Q, \Sigma, \delta, q_0, F)$ and equivalent DFA $M_1 = (Q_1, \Sigma, \delta_1, [q_0], F_1)$, where $Q = \{q_0, q_1, q_2, q_f\}, \Sigma = \{a, b\}$, s is starting state, $F = \{q_f\}$, and δ is defined as follows :

	q_0	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	-	$\{q_2\}$
q_2	-	$\{q_f\}$
q_f	-	-

δ_1 is defined as follows :

1. Keep the first row of NFA as it is with square bracket as follows :

	a	b
$[q_0]$	$[q_0,q_1]$	$[q_0]$

2. Now, we have two states : $[q_0],[q_0,q_1]$. We select the one next state that is not a present state till now and define the transition for it. We have only one next state $[q_0,q_1]$, which is not a present state .

	a	b
$\rightarrow[q_0]$	$[q_0,q_1]$	$[q_0]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_2]$

Since, $\delta_1([q_0,q_1],a)=[\delta(q_0,a)\cup\delta(q_1,a)]=[\{q_0,q_1\}\cup\phi]=[q_0,q_1]$, and $\delta_1([q_0,q_1],b)=[\delta(q_0,b)\cup\delta(q_1,b)]=[\{q_0\}\cup\{q_2\}]=[q_0,q_2]$)

3. Now $[q_0,q_2]$ is the next selected state, because $[q_0,q_1]$ is defined already

	a	b
$\rightarrow[q_0]$	$[q_0,q_1]$	$[q_0]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_2]$
$[q_0,q_2]$	$[q_0,q_1]$	$[q_0,q_f]$

4. Now, the state $[q_0,q_f]$ is the next selected state.

	a	b
$\rightarrow[q_0]$	$[q_0,q_1]$	$[q_0]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_2]$
$[q_0,q_2]$	$[q_0,q_1]$	$[q_0,q_f]$
$[q_0,q_f]$	$[q_0,q_1]$	$[q_0]$

5. Now, we have no new choice of the next state to be considered as present state. This is the completion of transition table. We have

$Q_1=\{[q_0],[q_0,q_1],[q_0,q_2],[q_0,q_f],\}$ (All selected states in transition), and $F_1 = \{[q_0, q_f]\}$ (Only one final state)

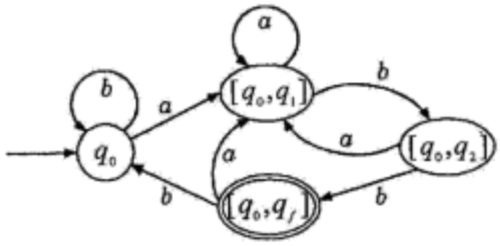


FIGURE : Transition Diagram of equivalent DFA

We see one thing here that not all states of 2^Q are selected for transition. We have selected those states, which are reachable from the initial state only and other remaining states of 2^Q are neglected.

So, finally we conclude that only those states of 2^Q are considered in transitions, which are reachable from the initial state.

Example 2 : Construct equivalent DFA for NFA M = ({ p, q, r, s }, { 0, 1 }, δ, p,{ q, s}), where δ is given below .

	0	1
p	{ q, s }	{ q }
q	{ r }	{ q, r }
r	{ s }	{ q, r }
s	-	{ p }

Solution : Let equivalent DFA is M_1 and $M_1 = (Q, \Sigma, \delta, [p], F)$

Construction of Transition table for equivalent DFA

	0	1
$\rightarrow [p]$	$[q, s]$	$[q]$
$[q]$	$[r]$	$[q, r]$
$[q, s]$	$[r]$	$[p, q, r]$
$[r]$	$[s]$	$[q, r]$
$[q, r]$	$[r, s]$	$[q, r]$
$[p, q, r]$	$[q, r, s]$	$[q, r]$

[s]	ϕ	[p]
$[r, s]$	[s]	$[p, q, r]$
$[q, r, s]$	$[r, s]$	$[p, q, r]$

Q = { [p], [q], [r], [s], [q,r], [r, s], [q, s], [p, q, r], [q, r, s] },
Σ = { 0, 1 }, [p] is the starting state,
and F = { [q], [s], [q, r] , [r, s] , [q, s], [p, q, r], [q, r, s].

Example 3 : Find a DFA equivalent to NFA $M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$, where δ is defined as follows .

PS	NS 0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_1\}$
q_2	-	$\{q_0, q_1\}$

Solution : Let $M' = (Q, \Sigma, \delta, [q_0], F)$ be the equivalent DFA, where $\Sigma = \{a, b\}$, and $[q_0]$ is the initial state.

Transition table :

PS	NS 0	1
$\rightarrow [q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_2]$	ϕ	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_0]$	$[q_0, q_1]$

$Q = \{[q_0], [q_2], [q_0, q_1], [q_1, q_2]\}$, and $F = \{[q_2], [q_1, q_2]\}$

Transition diagram :

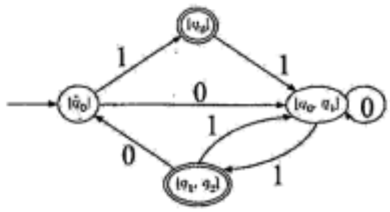


FIGURE : Equivalent DFA

Example 4 : A NFA which accepts set of strings over { 0, 1 } such that some two zero's are separated by a string over { 0, 1 } whose length is $4n$ $(n \geq 0)$ is shown in below figure . Construct equivalent DFA.

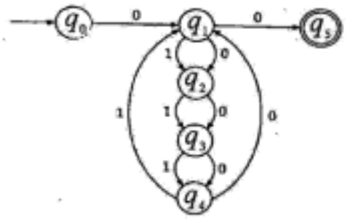


FIGURE : N F A

Solution : Let equivalent DFA $M = (Q, \Sigma, \delta, [q_0], F)$. constructing transition table for given NFA :

	(NS)	
(PS)	0	1
$\rightarrow q_0$	$\{ q_1 \}$	-
q_1	$\{ q_2, q_5 \}$	$\{ q_2 \}$
q_2	$\{ q_3 \}$	$\{ q_3 \}$
q_3	$\{ q_4 \}$	$\{ q_4 \}$
q_4	$\{ q_1 \}$	$\{ q_1 \}$
q_5	-	-

Constructing transition table for equivalent DFA :

	(NS)	
(PS)	0	1
$\rightarrow [q_0]$	$[q_1]$	ϕ
$[q_1]$	$[q_2, q_5]$	$[q_2]$
$[q_2]$	$[q_3]$	$[q_3]$
$[q_3]$	$[q_4]$	$[q_4]$
$[q_4]$	$[q_1]$	$[q_1]$
$[q_2, q_5]$	$[q_3]$	$[q_3]$

Where, $Q=\{[q_0],[q_1],[q_2],[q_3],[q_4],[q_2,q_5]\}, \Sigma=\{0,1\}, [q_0]$ is starting state, $F=\{[q_2,q_5]\}$, and transition function is defined above.

Transition diagram :

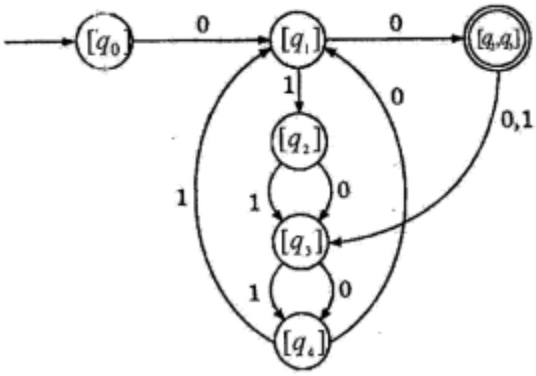


FIGURE : Equivalent DFA

1.5 NFA WITH $\in$ - MOVES

1.5.1 Finite automata With $\in$ - Transitions

This is same as NFA except we are using a special input symbol called epsilon $(\in)$. Using this symbol path we can jump to one state to other state without reading any input symbol.

This also analytically indicated as 5 - tuple notation.

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

$Q \rightarrow$ set of states in design

$\Sigma \rightarrow$ input alphabet

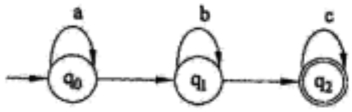
$q_0 \rightarrow$ initial state

$F \rightarrow$ final states $(\subseteq Q)$

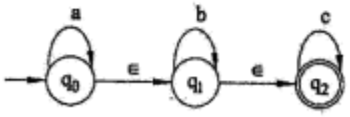
$\delta \rightarrow$ mapping function indicates $Q \times (\Sigma \cup \{\in\}) \rightarrow 2^Q$

Example : Draw a transition diagram of NFA which include transitions on the empty input $\in$ and accepts a language consisting of any number a's followed by any number of b's and which in turn followed by any number of c's.

Solution : It requires three states q_0, q_1 and q_2 and they accept any number of a's, b's and c's respectively. Assign q_2 as final state.



To reach from q_0 to q_1 and q_1 *to* q_2 no input will be given i. e., they treat $\in$ as their input and do the transition.



Normally these $\in$'s do not appear explicitly in the string.
The transition function for the NFA is shown below :

	a	b	c	$\in$
$\rightarrow q_0$	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$	ϕ

For example consider the string $\omega = ab\,c$

String $\omega = ab\,c$ (i. e., string in actual form is $a \in b \in c$ i. e., included along with epsilons).

$$
\begin{aligned}
\delta(q_0, abc) \quad & \vdash \delta(q_0, bc) \\
& \vdash \delta(q_0, \in bc) \\
& \vdash \delta(q_1, bc) \\
& \vdash \delta(q_1, \in c) \\
& \vdash \delta(q_2, c) \quad \vdash q_2 \in F
\end{aligned}
$$

The path is shown below :

$$q_0 \xrightarrow{a} q_0 \xrightarrow{\in} q_1 \xrightarrow{b} q_1 \xrightarrow{\in} q_2 \xrightarrow{c} q_2$$

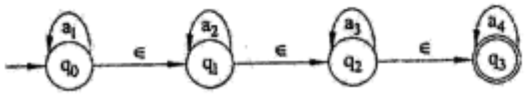
with arcs labeled a, $\in$, b, $\in$, c

Extension of Transition Function From δ to $\hat{\delta}$

The extended transition function $\hat{\delta}$ maps $Q \times \Sigma^*$ to 2^Q. It is important to compute the set of states reachable from a given state q_0 using $\in$ transitions only for constructing $\hat{\delta}$.

The $\in$ – closure (q_0) is used to denote the set of all vertices q_n such that there is a path from q_0 *to* q_n labeled $\in$.

Consider the problem



Here $\in$ - closure $(q_0) = \{q_0, q_1, q_2, q_3\}$

$\therefore$ $\hat{\delta}(q_0, \in) = \in\!-closure = \{q_0, q_1, q_2, q_3\}$

$\in$ – closure (q) is used to denote the set of all states s such that there is a path from q to s for string ω, includes edges labeled $\in$.

Note : The transition on $\in$ doesnot allow the NFA to accept Non - regular sets.

Definition : The extended transition function $\hat{\delta}$ is defined as follows :

(i) $\hat{\delta}(q, \in) = \in -$ closure (q)

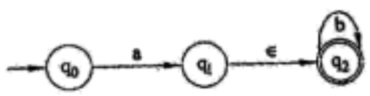
(ii) For ω in Σ^* and x in $\Sigma, \hat{\delta}(q, \omega x) = \in$ - closure (s), where s = { s | for some r in $\hat{\delta}(q, \omega), s \in \delta(r, x)\}$ δ can be extended $\hat{\delta}$ by extension to set of states.

(iii) $\delta(R, x) = \bigcup_{q \in R} \delta(q, x)$ and

(iv) $\hat{\delta}(R, \omega) = \bigcup_{q \in R} \hat{\delta}(q, \omega)$.

Note : $\hat{\delta}(q, a)$ is not necessarily equal to $\delta(q, a)$.

Example : The following NFA with $\in$ transitions accepts input strings with (a's and b's) single a or a followed by any number of b's.



The NFA accepts strings a, ab, abbb etc. by using $\in$ path between q_1 and q_2 we can move from q_1 state to q_2 without reading any input symbol. To accept ab first we are moving from q_0 to q_1 reading a and we can jump to q_2 state without reading any symbol there we accept b and we are ending with final state so it is accepted.

Equivalence of NFA with $\in-$ Transitions and NFA without $\in-$ Transitions

Theorem : If the language L is accepted by an NFA with $\in-$ transitions, then the language L_1 is accepted by an NFA without $\in-$ transitions.

Proof : Consider an NFA 'N' with $\in-$ transitions where $N = (Q, \Sigma, \delta, q_0, F)$

Construct an NFA N_1 without $\in-$ transitions $N_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$

where $Q_1 = Q$ and

$$F_1 = \begin{cases} F \cup \{q_0\} & \text{if } \in-closure(q_0) \text{ contains a state of } F \\ F & \text{otherwise} \end{cases}$$

and $\delta_1\ (q,a)$ is $\hat{\delta}\ (q,a)$ for q in Q and a in Σ.

Consider a non - empty string ω. To show by induction $|\omega|$ that $\delta_1(q_0, \omega) = \hat{\delta}\ (q_0,\omega)$

For $\omega = \in$, the above statement is not true. Because

$$\delta_1(q_0, \in) = \{q_0\},$$

while $$\hat{\delta}(q_0, \in) = \in -closure\ (q_0)$$

Basis :

Start induction with string length one.

i. e., $|\omega| = 1$

Then w is a symbol a, and $\delta_1(q_0,a) = \hat{\delta}(q_0,a)$ by definition of δ_1.

Induction : $|\omega| > 1$

Let $\omega = xy$ for symbol a in Σ.

Then $\delta_1(q_0,xy) = \delta_1(\delta_1(q_0,x), y)$

By inductive hypothesis

$$\delta_1 (q_0, x) = \hat{\delta} (q_0, x)$$

Let $\hat{\delta}(q_0, x) = s$

We have to show that $\delta_1(s, y) = \hat{\delta}(q_0, xy)$

But $\delta_1(s, y) = \bigcup_{q \in s} \delta_1(q, y) = \bigcup_{q \in s} \hat{\delta}(q, y)$

then $s = \hat{\delta}(q_0, x)$

$\therefore$ $\bigcup_{q \in s} \hat{\delta}(q, y) = \hat{\delta}(q_0, x)$

By rule (Rule : For $\omega \in \Sigma^*$ and $x \in \Sigma$, $\hat{\delta}(q, \omega x) = \in-$ closure (s),

where $s = \{ s \mid$ for some r in $\hat{\delta}(q, \omega)$, $s \in \delta(r, x)\}$ in the definition of $\hat{\delta}$).

Thus $\delta_1 (q_0, xy) = \hat{\delta} (q_0, xy)$.

To complete the proof we shall show that $\delta' (q_0, w)$ contain a state of F' if and only if $\hat{\delta} (q_0, x)$ contain a state of F. For this two cases arises.

Case I : If $\omega = \in$, this statement is true from the definition of F_1.

i. e., $\delta_1(q_0, \in) = \{ q_0 \}$

$\Rightarrow$ $q_0 \in F'_1$

Whenever $\hat{\delta} (q_0, \in)$ is $\in-$ closure (q_0), contains a state in F (possibly is q_0).

Case II : If $\omega \neq \in$ then W = xy for some symbol y.

If $\hat{\delta} (q_0, \omega)$ contains a state of F, $\Rightarrow \delta_1 (q_0, \omega)$ contains some state in F'

Conversely, if $\delta_1 (q_0, \omega) \in F_1$ other than $q_0, \Rightarrow \hat{\delta}(q_0, \omega) \in F$.

If $\delta_1(q_0, \omega) \in q_0$ and $q_0 \notin F$, then

$\hat{\delta}(q_0, \omega) = \in-$ closure $(\delta_1(\hat{\delta}(q_0, \omega), y))$,

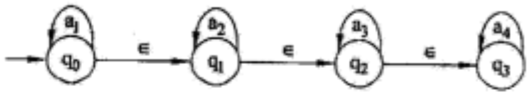
The state in $\in$ - closure (q_0) and in F must be in $\hat{\delta}(q_0, \omega)$.

Calculation of $\in$ - closure :

$\in$ - closure of state ($\in$-closure (q)) defined as it is a set of all vertices p such that there is a path from q to p labelled $\in$ (including itself).

Example :

Consider the NFA with $\in$ - moves



$\in$ – closure $(q_0) = \{ q_0, q_1, q_2, q_3 \}$

$\in$ – closure $(q_1) = \{ q_1, q_2, q_3 \}$

$\in$ – closure $(q_2) = \{ q_2, q_3 \}$

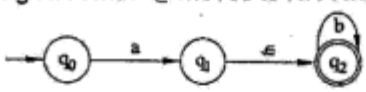
$\in$ – closure $(q_3) = \{ q_3 \}$

Procedure to convert NFA with $\in$ moves to NFA without $\in$ moves

Let $N = (Q, \Sigma, \delta, q_0, F)$ is a NFA with $\in$ moves then there exists $N' = (Q, \in, \hat{\delta}, q_0, F')$ without $\in$ moves

1. First find $\in$ – closure of all states in the design.
2. Calculate extended transition function using following conversion formulae.
   (i) $\hat{\delta}(q, x) = \in -$ closure $(\delta(\hat{\delta}(q, \in), x))$
   (ii) $\hat{\delta}(q, \in) = \in -$ closure (q)
3. F' is a set of all states whose $\in$ closure contains a final state in F.

Example 1 : Convert following NFA with $\in$ moves to NFA without $\in$ moves.



Solution : Transition table for given NFA is

δ	a	b	$\in$
$\rightarrow q_0$	q_1	ϕ	ϕ
q_1	ϕ	ϕ	q_2
q_2	ϕ	q_2	ϕ

(i) Finding $\in$ closure :

$\in$– closure $(q_0) = \{q_0\}$

$\in$– closure $(q_1) = \{q_1, q_2\}$

$\in$– closure $(q_2) = \{q_2\}$

(ii) Extended Transition function :

$\hat{\delta}$	a	b
$\rightarrow q_0$	$\{q_1,q_2\}$	ϕ
q_1	ϕ	$\{q_2\}$
q_2	ϕ	$\{q_2\}$

$$
\begin{aligned}
\hat{\delta}(q_0, a) &= \in\text{-}closure\ (\delta(\hat{\delta}(q_0,\in),a)) \\
&= \in\text{-}closure\ (\delta(\in\text{-}closure\ (q_0),\ a)) \\
&= \in\text{-}closure\ (\delta(q_0, a)) \\
&= \in\text{-}closure\ (q_1) \\
&= \{q_1,q_2\}
\end{aligned}
$$

$$
\begin{aligned}
\hat{\delta}(q_0, b) &= \in\text{-}closure\ (\delta(\hat{\delta}(q_0,\in),b)) \\
&= \in\text{-}closure(\delta(\in\text{-}closure\ (q_0), b)) \\
&= \in\text{-}closure(\delta(q_0, b)) \\
&= \in\text{-}closure(\phi) \\
&= \phi
\end{aligned}
$$

$$
\begin{aligned}
\hat{\delta}(q_1, a) &= \in\text{-}closure(\delta(\hat{\delta}(q_1, \in), a)) \\
&= \in\text{-}closure(\delta(\in\text{-}closure(q_1), a)) \\
&= \in\text{-}closure(\delta((q_1, q_2), a)) \\
&= \in\text{-}closure(\delta(q_1, a) \cup \delta(q_2, a)) \\
&= \in\text{-}closure\ (\phi) \\
&= \phi
\end{aligned}
$$

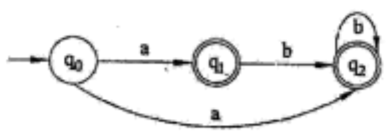
$$\begin{aligned}
\hat{\delta}(q_1, b) &= \in\text{-}closure\ (\delta(\hat{\delta}(q_1, \in), b)) \\
&= \in\text{-}closure\ (\delta(\in\text{-}closure(q_1), b)) \\
&= \in\text{-}closure\ (\delta((q_1, q_2), b)) \\
&= \in\text{-}closure\ (\delta(q_1, b) \cup \delta(q_2, b)) \\
&= \in\text{-}closure\ (q_2) \\
&= \{q_2\}
\end{aligned}$$

$$\begin{aligned}
\hat{\delta}(q_2, a) &= \in\text{-}closure\ (\delta(\hat{\delta}(q_2, \in), a)) \\
&= \in\text{-}closure\ (\delta(\in\text{-}closure(q_2), a)) \\
&= \in\text{-}closure\ (\delta(q_2, a)) \\
&= \in\text{-}closure\ (\phi) \\
&= \phi
\end{aligned}$$

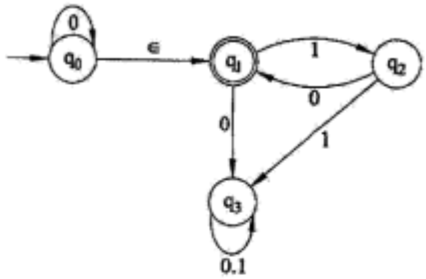
$$\begin{aligned}
\hat{\delta}(q_2, b) &= \in\text{-}closure\ (\delta(\hat{\delta}(q_2, \in), b)) \\
&= \in\text{-}closure\ (\delta(\in\text{-}closure(q_2), b)) \\
&= \in\text{-}closure\ (\delta(q_2, b)) \\
&= \in\text{-}closure\ (q_2) \\
&= \{q_2\}
\end{aligned}$$

(iii) Final states are q_1, q_2, because
$\in$- *closure* (q_1) contains final state
$\in$- *closure* (q_2) contains final state

(iv) NFA without $\in$ moves is



Example 2 : Convert the following NFA with $\in -$ moves into equivalent NFA without $\in -$ moves.



Solution : Transition table is

	0	1	$\in$
$\to q_0$	q_0	ϕ	q_1
q_1	q_3	q_2	ϕ
q_2	q_1	q_3	ϕ
q_3	q_3	q_3	ϕ

(i) Finding $\in$ - closure :

$\in -\ closure\ (q)$ is a set of states having paths on epsilon symbol from state q.

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

$$\in -\ closure\ (q_1) = \{ q_1 \}$$

$$\in -\ closure\ (q_2) = \{ q_2 \}$$

$$\in -\ closure\ (q_3) = \{ q_3 \}$$

(ii) Extended Transition function :

$$\hat{\delta}\ (q_0, 0) = \in -\ closure\ (\delta\ (\hat{\delta}\ (q_0, \in), 0))$$

$$= \in -\ closure\ (\delta\ (\in -\ closure\ (q_0), 0))$$

$$= \in -\ closure\ (\delta\ ((q_0, q_1), 0))$$

$$= \in -closure(\delta(q_0,0) \cup \delta(q_1,0))$$

$$= \in-closure\ (q_0, q_3)$$
$$= \in-closure\ (q_0) \cup \in-\text{closure}\ (q_3)$$
$$= \{q_0, q_1\} \cup \{q_3\}$$
$$= \{q_0, q_1, q_3\}$$

$$\hat{\delta}(q_0, 1) = \in-closure\ (\delta(\hat{\delta}(q_0, \in), 1))$$
$$= \in-closure\ (\delta(\in-closure\ (q_0), 1))$$
$$= \in-closure\ (\delta\ ((q_0, q_1), 1))$$
$$= \in-closure(\delta(q_0,1) \cup \delta(q_1,1)) = \in-closure(\phi \cup q_2)$$
$$= \in-closure\ (q_2)$$
$$= \{q_2\}$$

Continuing like this the table is generalised as follows.

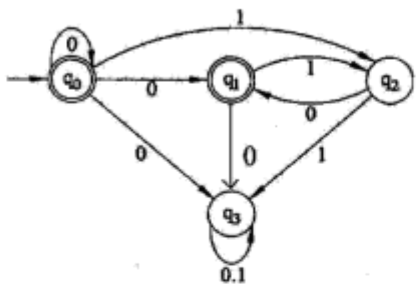
	0	1
$\rightarrow q_0$	$\{q_0, q_1, q_3\}$	q_2
q_1	q_3	q_2
q_2	q_1	q_3
q_3	q_3	q_3

(iii) Final states are q_0, q_1, because

$\in$- closure $(q_0) = \{q_0, q_1\}$ it contains final state

$\in$- closure $(q_1) = q_1$ is also final state

(iv) NFA without $\in$ moves is :



Example 3 : Find an equivalent NFA without $\in-$ transitions for NFA with $\in-$ transitions shown in below figure.

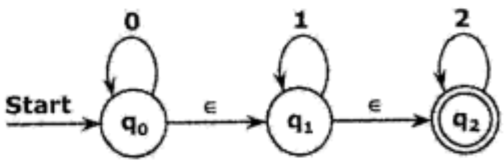


FIGURE : NFA with $\in-$ transitions

Solution : The transition table is ,

	Inputs			
States	0	1	2	$\in$
$\rightarrow q_0$	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$	ϕ

TABLE : Transition Table for the NFA in above figure.

Given NFA $M = (\{q_0, q_1, q_2\}, \{0, 1, 2, \in\}, \delta, q_0, \{q_2\})$.

Now NFA without $\in-$ moves.

$M' = (Q, \Sigma, \hat{\delta}, q_0, F')$

(i) Finding $\in-$ closure:

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

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

$\in-closure\ (q_2) = \{q_2\}$

(ii) Extended Transition function :

$\hat{\delta}\ (q_0, 0) \quad = \in-closure\ (\delta\ (\hat{\delta}(q_0, \in), 0))$

$= \in-closure\ (\delta\ \{q_0, q_1, q_2\}, 0)$

$= \in-closure(\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0))$

$$= \in\text{-}closure\ (\{q_0\} \cup \phi \cup \phi)$$
$$= \in\text{-}closure\ (q_0)$$
$$= \{q_0, q_1, q_2\}$$

$$\hat{\delta}(q_0, 1) = \in\text{-}closure\ (\delta(\hat{\delta}(q_0, \in), 1))$$
$$= \in\text{-}closure\ (\delta(\{q_0, q_1, q_2\}, 1))$$
$$= \in\text{-}closure\ [\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2, 1)]$$
$$= \in\text{-}closure\ (\phi \cup q_1 \cup \phi)$$
$$= \in\text{-}closure\ (q_1)$$
$$= \{q_1, q_2\}$$

Similarly for other transitions gives, transition table $\hat{\delta}(q, a)$

States	Inputs 0	1	2
$\rightarrow q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

TABLE : Modified Transition Table for the NFA in above figure

(iii) F' contains q_0, q_1, q_2 because $\in$– closure (q_0), $\in$– closure (q_1) and $\in$– closure (q_2) contains q_2.

(iv) $M' = (Q, \Sigma, \hat{\delta}, q_0, F')$ NFA without $\in$– transitions is,

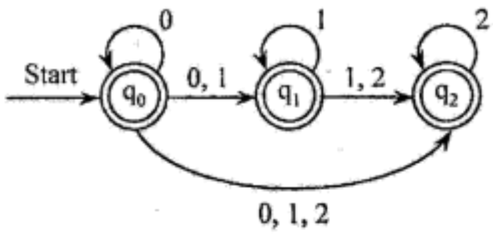


FIGURE : NFA without $\in$– transitions

Example 4 : For the following NFA with $\in-$ moves convert it into an NFA without $\in-$ moves.

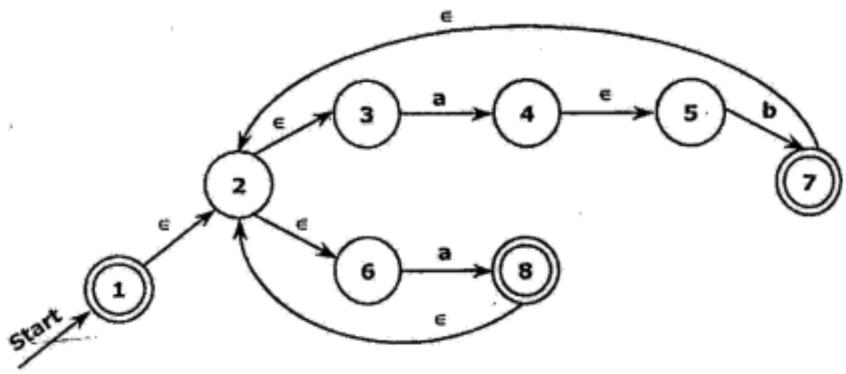


FIGURE : NFA with $\in-$ moves

Solution :

Let given NFA with $\in-$ moves be,

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

$$Q = \{1,2,3,4,5,6,7,8\} \; ; \; \Sigma = \{a, b\}$$

$$q_0 = 1; \; F = \{1, 7, 8\}$$

(i) Finding $\in-$ closure :

First we need to find $\in-$ closure of all states of M.

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

$\hat{\delta}(1,\in) = \in -closure(1) = \{1,2,3,6\}$

$\hat{\delta}(2,\in) = \in -closure(2) = \{2,3,6\}$

$\hat{\delta}(3,\in) = \in -closure(3) = \{3\}$

$\hat{\delta}(4,\in) = \in -closure(4) = \{4,5\}$

$\hat{\delta}(5,\in) = \in -closure(5) = \{5\}$

$\hat{\delta}(6,\in) = \in -closure(6) = \{6\}$

$\hat{\delta}(7,\in) = \in -closure(7) = \{2,3,6,7\}$

$\hat{\delta}(8,\in) = \in -closure(8) = \{2,3,6,8\}$

(ii) Extended Transition function:

$\hat{\delta}(1,a) = \in -closure\ (\delta(\hat{\delta}(1,\in),a))$

$= \in -\ closure\ (\delta(\{1,2,3,\ 6\},\ a))$

$= \in -\ closure\ (\{4,\ 8\})$

$= \{2,\ 4,5,\ 6,\ 8\}$

$\hat{\delta}(1,b) = \in -closure(\delta(\hat{\delta}(1,\in),b))$

$= \in -\ closure\ (\delta\ (\{1,2,\ 3,\ 6\},\ b\ ))$

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

$= \{\phi\}$

$\hat{\delta}(2,a) = \in -closure(\delta(\hat{\delta}(2,\in),a))$

$= \{2,\ 4,5,\ 6,\ 8\}$

$\hat{\delta}(2,b) = \in -closure(\delta(\hat{\delta}(2,\in),b))$

$= \{\phi\}$

$\hat{\delta}(3,a) = \in -closure(\delta(\hat{\delta}(3,\in),a))$

$=\{4,5\}$

$\hat{\delta}(3,b) = \in -closure(\delta(\hat{\delta}(3,\in),a))$

$=\{\phi\}$

$\hat{\delta}(4,a) = \in -closure\ (\delta(\hat{\delta}(4,\in),a))$

$=\{\phi\}$

$\hat{\delta}(4,b) = \in -closure\ (\delta(\hat{\delta}(4,\in),b))$

$=\{7\}$

$\hat{\delta}(5,a) = \in -closure\ (\delta(\hat{\delta}(5,\in),a))$

$=\{\phi\}$

$\hat{\delta}(5,b) = \in -closure\ (\delta(\hat{\delta}(5,\in),b))$

$=\{7\}$

$\hat{\delta}(6,a) = \in -closure\ (\delta(\hat{\delta}(6,\in),a))$

$=\{8\}$

$\hat{\delta}(6,b) = \in\text{-}closure\ (\delta(\hat{\delta}(6,\in),b))$
$=\{\phi\}$

$\hat{\delta}(7,a) = \in\text{-}closure\ (\delta(\hat{\delta}(7,\in),a))$
$=\{4,8\}$

$\hat{\delta}(7,b) = \in\text{-}closure\ (\delta(\hat{\delta}(7,\in),b))$
$=\{\phi\}$

$\hat{\delta}(8,a) = \in\text{-}closure\ (\delta(\hat{\delta}(8,\in),a))$
$=\{8\}$

$\hat{\delta}(8,b) = \in\text{-}closure\ (\delta(\hat{\delta}(8,\in),b))$
$=\{\phi\}$

Final states of M' includes all states whose $\in$– closure contains a final state of M.

$\therefore \quad F = \{1, 7, 8\}$

Transition table is,

	a	b
→ ①	{ 2, 4, 5, 6, 8 }	ϕ
2	{ 2, 4, 5, 6, 8 }	ϕ
3	{ 4, 5 }	ϕ
4	ϕ	{ 7 }
5	ϕ	{ 7 }
6	{ 8 }	ϕ
⑦	{ 4, 8 }	ϕ
⑧	{ 8 }	ϕ

FIGURE : Transition Table for the NFA in above figure.

Transition diagram of NFA without $\in -$ transitions is ,

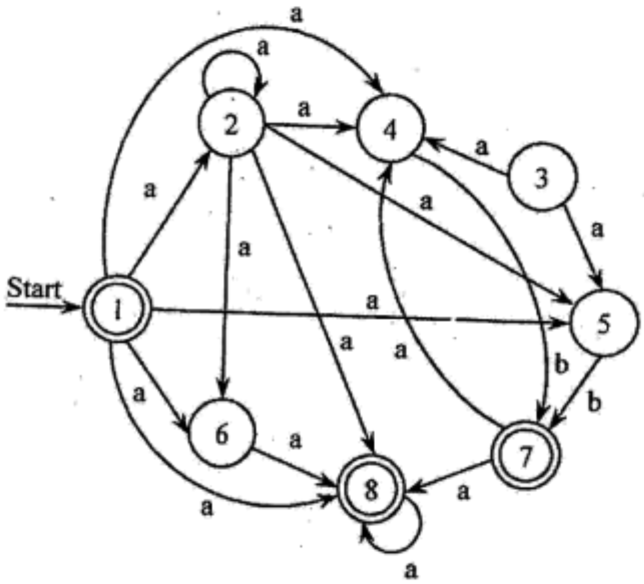


FIGURE : NFA without $\in -$ transitions

2

FINITE STATE MACHINES

After going through this chapter, you should be able to understand :

- Finite State Machines
- Moore & Mealy Machines
- Equivalence of Moore & Mealy Machines
- Equivalence of two FSMs
- Minimization of FSM

2.1 FINITE STATE MACHINES (FSMs)

A finite state machine is similar to finite automata having additional capability of outputs.

A model of finite state machine is shown in below figure.

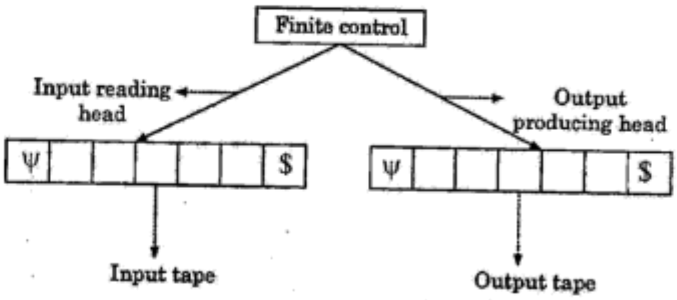


FIGURE : Model of FSM

2.1.1 Description of FSM

A finite state machine is represented by 6 - tuple $(Q,\Sigma,\Delta,\delta,\lambda,q_0)$, where

1. Q is finite and non - empty set of states,
2. Σ is input alphabet,
3. Δ is output alphabet,

4. δ is transition function which maps present state and input symbol on to the next state or $Q \times \Sigma \rightarrow Q$,
5. λ is the output function, and
6. $q_0 \in Q$, is the initial state.

2.1.2 Representation of FSM

We represent a finite state machine in two ways; one is by transition table, and another is by transition diagram. In transition diagram, edges are labeled with Input / output.

Suppose, in transition table the entry is defined by a function F, so for input a_i and state q_i

$$F(q_i, a_i) = (\delta(q_i, a_i), \lambda(q_i, a_i))$$ (where δ is transition function, λ is output function.)

Example 1 : Consider a finite state machine, which changes 1's into 0's and 0's into 1's (1's complement) as shown in below figure.

Transition diagram :

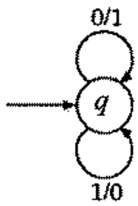


FIGURE : Finite state machine

Transition table :

	Inputs			
	0		1	
Present State(PS)	Next State (NS)	Output	Next State (NS)	Output
q	q	1	q	0

Example 2 : Consider the finite state machine shown in below figure, which outputs the 2's complement of input binary number reading from least significant bit (LSB).

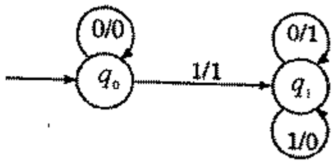
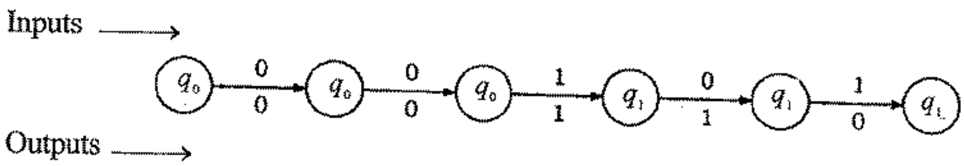


FIGURE : Finite State machine

Suppose, input is 10100. What is the output ?

Solution : The finite state machine reads the input from right side (LSB).

Transition sequence for input 10100 :



So, the output is 01100.

2.2 MOORE MACHINE

If the *output of finite state machine is dependent on present state only,* then this model of finite state machine is known as Moore machine.

A Moore machine is represented by 6-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where

1 Q is finite and non-empty set of states,

2 Σ is input alphabet,

3 Δ is output alphabet,

4 δ is transition function which maps present state and input symbol on to the next state or $Q \times \Sigma \to Q$,

5 λ is the output function which maps $Q \to \Delta$, (Present state $\to$ Output), and

6 $q_0 \in Q$, is the initial state.

If $Z(t)$, $q(t)$ are output and present state respectively at time t then

$$Z(t) = \lambda(q(t)).$$

$$\boxed{\text{For input } \in \text{ (null string), } Z(t) = \lambda \text{ (initial state)}}$$

DEPARTMENT OF CSE

Example 1 : Consider the Moore machine shown in below figure.Construct the transition table. What is the output for input 01010 ?

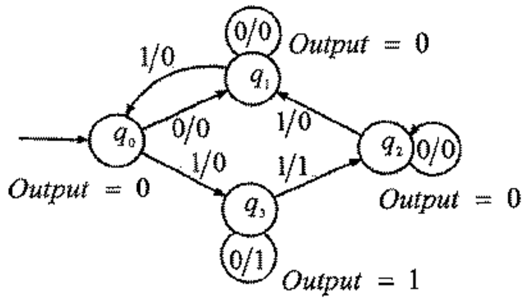
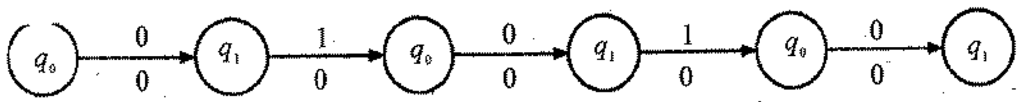


FIGURE: Moore machine

Solution : Transition table is as follows :

	Inputs 0	Inputs 1	
Present State (PS)	Next State State (NS)	Next State State (NS)	Output
q_0	q_1	q_3	0
q_1	q_1	q_0	0
q_2	q_2	q_1	0
q_3	q_3	q_2	1

Transition sequence for string 01010 :



So, the output is 00000.

Note : Since, the output of Moore machine does not depend on input. So, the first output symbol is additional from the initial state without reading the input i.e., null input and output length is one greater than the input length, but not included in the above output.

Example 2 : Design a Moore machine, which outputs residue mod 3 for each binary input string treated as a binary integer.

Solution : Let Moore machine $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where $\Sigma = \{0, 1\}$

$\Delta = \{0, 1, 2\}$ (outputs after mod 3),

Let three states $\{q_0, q_1, q_2\}$ are there and

State q_0 outputs 0,

State q_1 outputs 1, and

State q_2 outputs 2.

If input is binary string X, then

X is followed by a 0 is equivalent to twice of X

X is followed by a 1 is equivalent to twice of X plus 1.

$X0 \equiv (2 * X)_{10}$ (in decimal system), and

$X1 \equiv (2 * X)_{10} + 1$ (in decimal system)

If X mod 3 = r, for r = 0 or 1 or 2, then

X0 mod 3 = 2* r mod 3 **(For input 0)**

= 0 or 2 or 1

For transition :

$q_r \rightarrow q_{2*r \bmod 3}$ for r = 0, 1, 2

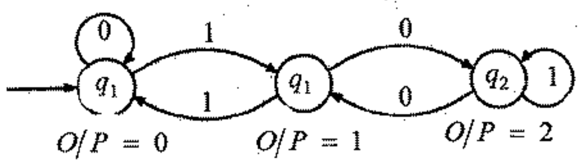
$X1 \bmod 3 = (2 * r + 1) \bmod 3$ **(For input 1)**

= 1, 0, 2

For transition :

$q_r \rightarrow q_{(2*r+1) \bmod 3}$ for r = 0, 1, 2

Transition diagram :



Example 3 : Design a Moore machine which reads input from (0+1+2)* and outputs residue mod 5 of the input. Input is considered at base 3 and it is treated as ternary integer.

Solution :

Let Moore machine $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ produces output residue mod 5 for each input string written in base 3.

$\Sigma = \{0, 1, 2\}, \Delta = \{0, 1, 2, 3, 4\}$

Let five states $\{q_0, q_1, q_2, q_3, q_4\}$ are there and

State q_0 outputs 0,

State q_1 outputs 1,

State q_2 outputs 2,

State q_3 outputs 3, and

State q_4 outputs 4.

If input is binary string w, then
w is followed by a 0 is equivalent to thrice of w,
w is followed by a 1 is equivalent to thrice of w plus 1,
w is followed by a 2 is equivalent to thrice of w plus 2.

Or

$W0 \equiv (3 * W)_{10}$ (in decimal system),

$W1 \equiv (3 * W)_{10} + 1$ (in decimal system),

$W2 \equiv (3 * W)_{10} + 2$ (in decimal system)

If $w \bmod 5 = r$, for $r = \{0,1,2,3,4\}$ (in the order of the elements), then

$W0 \bmod 5 = 3 * r \bmod 5$ **(For input 0)**

$= \{0,3,1,4,2\}$ (In the order of elements)

For transition :

$Q_r \rightarrow Q_{3*r \bmod 5}$ for $r = \{0,1,2,3,4\}$ (in the order of the elements)

W 1 mod $5 = (3 * r + 1) \bmod 5$ **(for input 1)**

$= \{1,4,2,0,3\}$ (In the order of elements)

For transition :

$Q_r \rightarrow Q_{(3*r+1) \bmod 5}$ for $r = \{0,1,2,3,4\}$ (in the order of the elements)

W 2 mod $5 = (3 * r + 2) \bmod 5$ **(for input 2)**

$= \{2,0,3,1,4\}$ (In the order of elements)

For transition :

$Q_r \rightarrow Q_{(3*r+2) \bmod 5}$ for $r = \{0,1,2,3,4\}$ (in the order of the elements)

Transition table

	Inputs 0	1	2	
PS	NS	NS	NS	Output
q_0	q_0	q_1	q_2	0
q_1	q_3	q_4	q_0	1
q_2	q_1	q_2	q_3	2
q_3	q_4	q_0	q_1	3
q_4	q_2	q_3	q_4	4

2.3 MEALY MACHINE

If the *output of finite state machine is dependent on present state and present input,* then this model of finite state machine is known as Mealy machine.

A Mealy machine is described by 6 - tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where

1. Q is finite and non-empty set of states,
2. Σ is input alphabet,
3. Δ is output alphabet,

4. δ is transition function which maps present state and input symbol on to the next state or $Q \times \Sigma \to Q$,
5. λ is the output function which maps $Q \times \Sigma \to \Delta$,((Present state, present input symbol) $\to$ Output), and
6. $q_0 \in Q$, is the initial state.

If $Z(t)$, $q(t)$, and $x(t)$ are output, present state, and present input respectively at time t,

Then, $Z(t) = \lambda(q(t), x(t))$

$$\boxed{\text{For input } \in \text{ (null string), } Z(t) = \in}$$

Example 1: Consider the Mealy machine shown in below figure. Construct the transition table and find the output for input 01010.

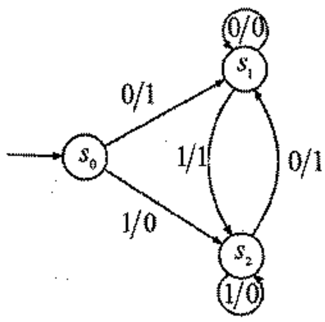
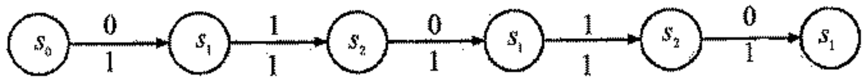


FIGURE : Mealy Machine

Solution : Transition table is constructed below.

	Inputs			
	0		1	
PS	NS	Output	NS	Output
s_0	s_1	1	s_2	0
s_1	s_1	0	s_2	1
s_2	s_1	1	s_2	0

Transition sequence for input 01010



(So, the output is 11111.)
(Note : The output length is equal to the input length).

Example 2 : Construct a Mealy machine which reads input from $\{0, 1\}$ and outputs EVEN or ODD according to total number of 1's even or odd.

Solution :

We consider two states q_0, which outputs EVEN and q_1 which outputs ODD.

Suppose, $a \in (0+1)^*$ has even number of 1's, then a11 also has even number of 1's.

Suppose, $b \in (0+1)^*$ has odd number of 1's then b1 also has odd number of 1's.

Transition diagram :

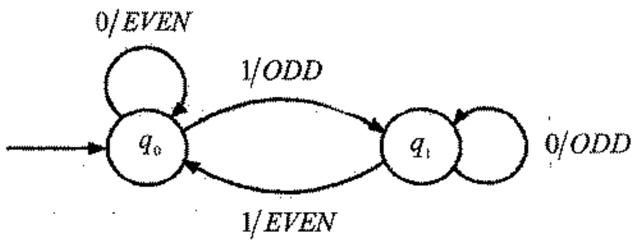


FIGURE : Mealy Machine

Example 3 : Design a Mealy machine which reads the input from (0+1)* and produces the following outputs.

(i) If input ends in 101, output is A,
(ii) If input ends 110, the output is B, and
(iii) For other inputs, output is C.

Solution : Suppose, Mealy machine $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ which reads the inputs from $(0+1)^*$, starting from the least significant bit (LSB).

Consider three LSBs of

Input	Output
...000 (X)	C
...001 (X)	C
...010 (X)	C
...011 (X)	C
...100 (X)	C
...101	A
...110	B
...111 (X)	C

Transition diagram :

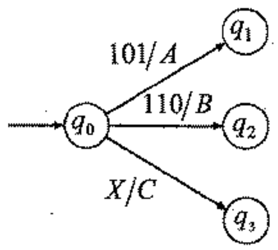


FIGURE : Moore Machine

2.4 EQUIVALENCE OF MOORE AND MEALY MACHINES

We can construct equivalent Mealy machine for a Moore machine and vice-versa. Let M_1 and M_2 be equivalent Moore and Mealy machines respectively. The two outputs $T_1(w)$ and $T_2(w)$ are produced by the machines M_1 and M_2 respectively for input string w. Then the length of $T_1(w)$ is one greater than the length of $T_2(w)$, i.e.

$$|T_1(w)| = |T_2(w)| + 1$$

The additional length is due to the output produced by initial state of Moore machine. Let output symbol x is the additional output produced by the initial state of Moore machine, then $T_1(w) = x\,T_2(w)$.

It means that if we neglect the one initial output produced by the initial state of Moore machine, then outputs produced by both machines are equivalent. *The additional output is produced by the initial state* of (for input $\in$) Moore machine without reading the input.

Conversion of Moore Machine to Mealy Machine

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

Proof : We will discuss proof in two steps.

Step 1 : Construction of equivalent Mealy machine M_2, and

Step 2 : Outputs produced by both machines are equivalent.

Step 1 (Construction of equivalent Mealy machine M_2)

Let $M_2 = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$ where all terms $Q, \Sigma, \Delta, \delta, q_0$ are same as for Moore machine and λ' is defined as following :

$$\lambda'(q, a) = \lambda(\delta(q, a)) \text{ for all } q \in Q \text{ and } a \in \Sigma$$

The first output produced by initial state of Moore machine is neglected and transition sequences remain unchanged.

Step 2 : If x is the output symbol produced by initial state of Moore machine M_1, and $T_1(w)$, $T_2(w)$ are outputs produced by Moore machine M_1 and equivalent Mealy machine M_2 respectively for input string w, then

$$T_1(w) = x\,T_2(w)$$

Or Output of Moore machine = $x| \,|$ Output of Mealy machine

(The notation | | represents concatenation).

If we delete the output symbol x from $T_1(w)$ and suppose it is $T_1'(w)$ which is equivalent to the output of Mealy machine. So we have,

$$T_1'(w) = T_2(w)$$

Hence, Moore machine M_1 and Mealy machine M_2 are equivalent.

Example 1 : Construct a Mealy machine equivalent to Moore machine M_1 given in following transition table.

	Inputs 0	Inputs 1	
Present State (PS)	Next State (NS)	Next State (NS)	Output
q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

Solution : Let equivalent Mealy machine $M_2 = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$ where

1. $Q = \{q_0, q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. $\Delta = \{0, 1\}$
4. λ' is defined as following :

For state $q_0 : \lambda'(q_0, 0) = \lambda(\delta(q_0, 0)) = \lambda(q_1) = 0$

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1)) = \lambda(q_2) = 1$

For state q_1 : $\lambda'(q_1, 0) = \lambda(\delta(q_1, 0)) = \lambda(q_3) = 1$

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

For state q_2 : $\lambda'(q_2, 0) = \lambda(\delta(q_2, 0)) = \lambda(q_2) = 1$

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

For state $q_3 : \lambda'(q_3, 0) = \lambda(\delta(q_3, 0)) = \lambda(q_0) = 1$

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

Transition table :

	Inputs 0		Inputs 1	
PS	NS	Output	NS	Output
$\rightarrow q_0$	q_1	0	q_2	1
q_1	q_3	1	q_2	1
q_2	q_2	1	q_1	0
q_3	q_0	1	q_3	1

Transition diagram :

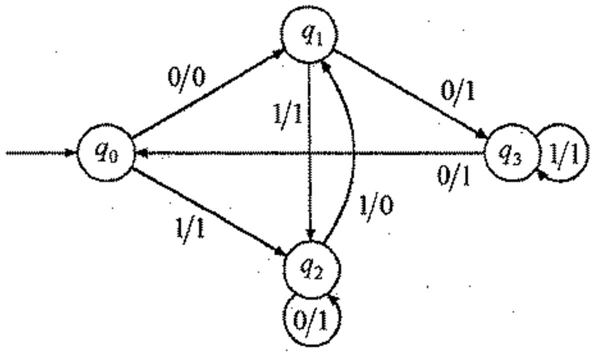


FIGURE : Mealy Machine

Example 2 : Construct a Mealy machine equivalent to Moore machine $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ described in following transition table.

	Inputs 0	Inputs 1	
Present State (PS)	Next State (NS)	Next State (NS)	Output
q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Solution : Let equivalent Mealy machine $M_2 = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$, where

1. $Q = \{q_0, q_1, q_2, q_3\}$
2. $\Sigma = \{0,1\}$
3. $\Delta = \{0, 1\}$
4. λ' is defined as following :

For state $q_0 : \lambda'(q_0, 0) = \lambda(\delta(q_0, 0)) = \lambda(q_3) = 0$

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1)) = \lambda(q_1) = 1$

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

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

For state $q_2 : \lambda'(q_2, 0) = \lambda(\delta(q_2, 0)) = \lambda(q_2) = 0$

$\lambda'(q_2, 1) = \lambda(\delta(q_2, 1)) = \lambda(q_3) = 0$

For state $q_3 : \lambda'(q_3, 0) = \lambda(\delta(q_3, 0)) = \lambda(q_3) = 0$

$\lambda'(q_3, 1) = \lambda(\delta(q_3, 1)) = \lambda(q_0) = 0$

5. Transition is same for both machines, and
6. q_0 is the initial state.

Transition table :

	Inputs 0		1	
PS	NS	Output	NS	Output
q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

Conversion of Mealy Machine to Moore Machine

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

Proof : We will discuss proof in two steps.

Step 1 : Construction of equivalent Moore machine M_2, and

Step 2 : Outputs produced by both machines are equivalent.

Step 1 : Construction of equivalent Moore machine M_2

We define the set of states as ordered pair over Q and Δ. There is also a change in transition function and output function.

Let equivalent Moore machine $M_2 = (Q', \Sigma, \Delta, \delta', \lambda', q_0')$,

where

1. $Q' \subseteq Q \times \Delta$ is the set of states formed with ordered pair over Q and Δ,
2. Σ remains unchanged,

3. Δ remains unchanged,
4. λ' is defined as follows :

   $\delta'([q,b],a) = [\delta(q,a), \lambda(q,a)]$, where δ and λ are transition function and output function of Mealy machine.
5. λ' is the output function of equivalent Moore machine which is dependent on present state only and defined as follows :

$$\lambda'([q,b]) = b$$

6. q_0' is the initial state and defined as $[q_0,b_0]$, where q_0 is the initial state of Mealy machine and b_0 is any arbitrary symbol selected from output alphabet Δ.

Step 2 : Outputs of Mealy and Moore Machines

Suppose, Mealy machine M_1 enters states $q_0, q_1, q_2, \ldots q_n$ on input $a_1, a_2, a_3, \ldots . a_n$ and produces outputs $b_1, b_2, b_3, \ldots b_n$, then M_2 enters the states $[q_0, b_0], [q_1, b_1], [q_2, b_2] \ldots, [q_n, b_n]$ and produces outputs $b_0, b_1, b_2, \ldots b_n$ as discussed in Step 1. Hence, outputs produced by both machines are equivalent.

Therefore, Mealy machine M_1 and Moore machine M_2 are equivalent.

Example 1 : Consider the Mealy machine shown in below figure. Construct an equivalent Moore machine.

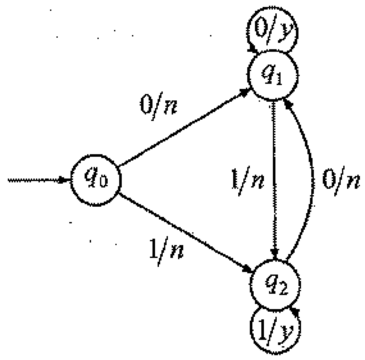


FIGURE : Mealy Machine

Solution : Let $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a given Mealy machine and $M_2 = (Q', \Sigma, \Delta, \delta', \lambda', q_0')$ be the equivalent Moore machine,

where

1. $Q' \subseteq \{[q_0,n],[q_0,y],[q_1,n],[q_1,y],[q_2,n],[q_2,y]\}$ (Since, $Q' \subseteq Q \times \Delta$)
2. $\Sigma = \{0, 1\}$

3. $\Delta = \{n, y\}$,
4. $q_0' = [q_0, y]$, where q_0 is the initial state and y is the output symbol of Mealy machine,
5. δ' is defined as following :

For initial state $[q_0, y]$:

$$\delta'([q_0, y], 0) = [\delta(q_0, 0), \lambda(q_0, 0)] = [q_1, n]$$
$$\delta'([q_0, y], 1) = [\delta(q_0, 1), \lambda(q_0, 1)] = [q_2, n]$$

For state $[q_1, n]$:

$$\delta'([q_1, n], 0) = [\delta(q_1, 0), \lambda(q_1, 0)] = [q_1, y]$$
$$\delta'([q_1, n], 1) = [\delta(q_1, 1), \lambda(q_1, 1)] = [q_2, n]$$

For state $[q_2, n]$:

$$\delta'([q_2, n], 0) = [\delta(q_2, 0), \lambda(q_2, 0)] = [q_1, n]$$
$$\delta'([q_2, n], 1) = [\delta(q_2, 1), \lambda(q_2, 1)] = [q_2, y]$$

For state $[q_1, y]$:

$$\delta'([q_1, y], 0) = [\delta(q_1, 0), \lambda(q_1, 0)] = [q_1, y]$$
$$\delta'([q_1, y], 1) = [\delta(q_1, 1), \lambda(q_1, 1)] = [q_2, n]$$

For state $[q_2, y]$:

$$\delta'([q_2, y], 0) = [\delta(q_2, 0), \lambda(q_2, 0)] = [q_1, n]$$
$$\delta'([q_2, y], 1) = [\delta(q_2, 1), \lambda(q_2, 1)] = [q_2, y]$$

(**Note :** We have considered only those states, which are reachable from initial state)

6. λ' is defined as follows :

$$\lambda'[q_0, y] = y$$
$$\lambda'[q_1, n] = n$$
$$\lambda'[q_2, n] = n$$
$$\lambda'[q_1, y] = y$$
$$\lambda'[q_2, y] = y$$

Transition diagram :

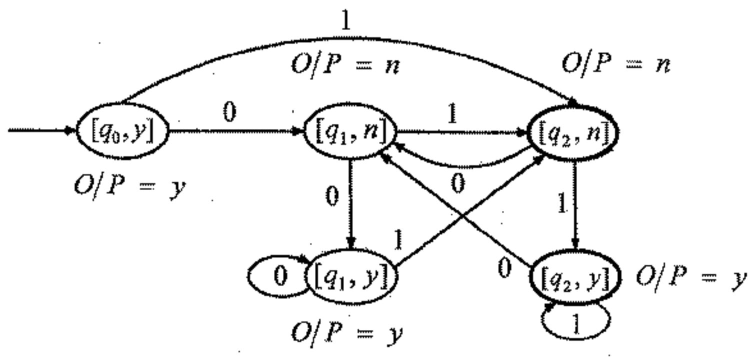


FIGURE : Moore machine

Example 2 : Construct a Moore machine equivalent to Mealy machine $M_1 = (Q,\Sigma,\Delta,\delta,\lambda,q_0)$ described in following transition table

	Inputs 0		Inputs 1	
PS	NS	Output	NS	Output
q_0	q_1	z_1	q_2	z_1
q_1	q_1	z_2	q_2	z_1
q_2	q_1	z_1	q_2	z_2

Solution :

Let $M_1 = (Q,\Sigma,\Delta,\delta,\lambda,q_0)$ is given Mealy machine and $M_2 = (Q',\Sigma,\Delta',\delta',\lambda',q_0')$ be the equivalent Moore machine, where

1. $Q' \subseteq \{[q_0,z_1],[q_0,z_2],[q_1,z_1],[q_1,z_2],[q_2,z_1],[q_2,z_2]\}$ (Since, $Q' \subseteq Q \times \Delta$)
2. $\Sigma = \{0, 1\}$
3. $\Delta' = \{z_1, z_2\}$
4. Let starting state $q_0' = [q_0, z_1]$ where q_0 is the initial state and z_1 is the output symbol of Mealy machine,

5. δ' is defined as follows :

For initial state $[q_0, z_1]$: $\delta'([q_0, z_1], 0) = [\delta(q_0, 0), \lambda(q_0, 0)] = [q_1, z_1]$

$\delta'([q_0, z_1], 1) = [\delta(q_0, 1), \lambda(q_0, 1)] = [q_2, z_1]$

(**Note :** Both states $[q_1, z_1]$ and $[q_2, z_1]$ are reachable from initial state.)

For state $[q_1, z_1]$: $\delta'([q_1, z_1], 0) = [\delta(q_1, 0), \lambda(q_1, 0)] = [q_1, z_2]$

$\delta'([q_1, z_1], 1) = [\delta(q_1, 1), \lambda(q_1, 1)] = [q_2, z_1]$

For state $[q_2, z_1]$: $\delta'([q_2, z_1], 0) = [\delta(q_2, 0), \lambda(q_2, 0)] = [q_1, z_1]$

$\delta'([q_2, z_1], 1) = [\delta(q_2, 1), \lambda(q_2, 1)] = [q_2, z_2]$

(**Note :** Both states $[q_1, z_2]$ and $[q_2, z_2]$ are reachable states.)

For state $[q_1, z_2]$: $\delta'([q_1, z_2], 0) = [\delta(q_1, 0), \lambda(q_1, 0)] = [q_1, z_2]$

$\delta'([q_1, z_2], 1) = [\delta(q_1, 1), \lambda(q_1, 1)] = [q_2, z_1]$

For state $[q_2, z_2]$: $\delta'([q_2, z_2], 0) = [\delta(q_2, 0), \lambda(q_2, 0)] = [q_1, z_1]$

$\delta'([q_2, z_2], 1) = [\delta(q_2, 1), \lambda(q_2, 1)] = [q_2, z_2]$

(**Note :** We have considered only those states, which are reachable from initial state.)

6. λ' is defined as follows :

$\lambda'[q_0, z_1] = z_1$

$\lambda'[q_1, z_1] = z_1$

$\lambda'[q_2, z_1] = z_1$

$\lambda'[q_1, z_2] = z_2$

$\lambda'[q_2, z_2] = z_2$

Transition Table

	Inputs 0	Inputs 1	
PS	NS	NS	Output
$[q_0, z_1]$	$[q_1, z_1]$	$[q_2, z_1]$	z_1
$[q_1, z_1]$	$[q_1, z_2]$	$[q_2, z_1]$	z_1
$[q_2, z_1]$	$[q_1, z_1]$	$[q_2, z_2]$	z_1
$[q_1, z_2]$	$[q_1, z_2]$	$[q_2, z_1]$	z_2
$[q_2, z_2]$	$[q_1, z_1]$	$[q_2, z_2]$	z_2

2.5 EQUIVALENCE OF FSMs

Two finite machines are said to be equivalent if and only if every input sequence yields identical output sequence.

Example :

Consider the FSM M_1 shown in figure (a) and FSM M_2 shown in figure (b).

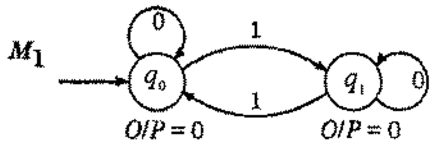


Figure (a)

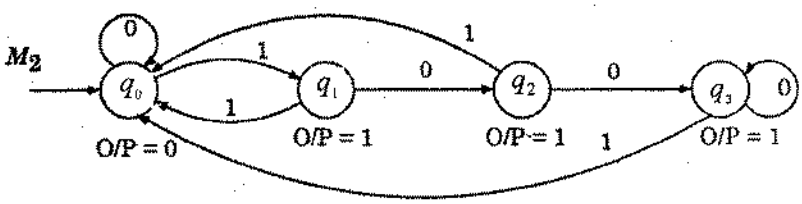


Figure (b)

Are these two FSMs equivalent ?

Solution :

We check this. Consider the input strings and corresponding outputs as given following :

Input string	Output by M_1	Output by M_2
(1) 01	00	00
(2) 010	001	001
(3) 0101	0011	0011
(4) 1000	0111	0111
(5) 10001	01111	01111

Now, we come to this conclusion that for each input sequence, outputs produced by both machines are identical. So, these machines are equivalent. In other words, both machines do the same task. But, M_1 has two states and M_2 has four states. So, some states of M_2 are doing the same

task i. e., producing identical outputs on certain input. Such states are known as equivalent states and require extra resources when implemented.
Thus, our goal is to find the simplest and equivalent FSM with minimum number of states.

2.5.1 FSM Minimization

We minimize a FSM using the following method, which finds the equivalent states, and merges these into one state and finally construct the equivalent FSM by minimizing the number of states.

Method : Initially we assume that all pairs (q_0, q_1) over states are non - equivalent states

Step 1 : Construct the transition table.

Step 2 : Repeat for each pair of non - equivalent states (q_0, q_1) :

(a) Do q_0 and q_1 produce same output ?

(b) Do q_0 and q_1 reach the same states for each input $a \in \Sigma$?

(c) If answers of (a) and (b) are YES, then q_0 and q_1 are equivalent states and merge these two states into one state $[q_0, q_1]$ and replace the all occurrences of q_0 and q_1 by $[q_0, q_1]$ and mark these equivalent states.

Step 3 : Check the all - present states, if any redundancy is found, remove that.

Step 4 : Exit.

Example 1 : Consider the following transition table for FSM. Construct minimum state FSM.

	Inputs		
	0	1	
Present State(PS)	**Next State (NS)**	**Next State (NS)**	**Output**
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_3	q_0	1
q_3	q_3	q_0	1

Solution :

Pairs formed over $\{q_0,q_1,q_2,q_3\}$ are $(q_0,q_1),(q_0,q_2),(q_0,q_3),(q_1,q_2)$, (q_1,q_3), (q_2,q_3).

Consider the pair (q_0,q_1) **:**

$$\lambda(q_0)=0$$
$$\lambda(q_1)=1$$

Hence, q_0 and q_1 are not equivalent.

Consider the pair (q_0,q_2) **:**

$$\lambda(q_0)=0$$
$$\lambda(q_2)=1$$

Hence, q_0 and q_2 are not equivalent

Consider the pair (q_0,q_3) **:**

$$\lambda(q_0)=0$$
$$\lambda(q_3)=1$$

Hence, q_0 and q_3 are not equivalent

Consider the pair (q_1,q_2) **:**

$$\lambda(q_1)=1$$
$$\lambda(q_2)=1$$

Outputs are identical.

Now, consider the transition:

$$\delta(q_1,0)=q_2,\quad \delta(q_1,1)=q_0$$
$$\delta(q_2,0)=q_3,\quad \delta(q_2,1)=q_0$$

So, transitions from q_1 and q_2 are not on the same state for 0 input.

Hence, q_1 and q_2 are not equivalent

Consider the pair (q_1,q_3) **:**

$$\lambda(q_1)=1$$
$$\lambda(q_3)=1$$

Outputs are identical.

Now, consider the transition :

$$\delta(q_1,0)= q_2,\quad \delta(q_1,1)= q_0$$
$$\delta(q_3,0)= q_3,\quad \delta(q_3,1)= q_0$$

So, transitions from q_1 and q_3 are not on the same state for 0 input.

Hence, q_1 and q_3 are not equivalent .

Consider the pair (q_2,q_3) **:**

$$\lambda(q_2)=1$$
$$\lambda(q_3)=1$$

Outputs are identical .

Now, consider the transition :

$$\delta(q_2,0)= q_3,\quad \delta(q_2,1)= q_0$$
$$\delta(q_3,0)= q_3,\quad \delta(q_3,1)= q_0$$

So, transitions from q_1 and q_2 are identical for inputs 0 and 1.

Hence, q_2 and q_3 are equivalent states.

So, merging q_2 and q_3 into $[q_2,q_3]$ to represent one state and replacing q_2 and q_3 by $[q_2,q_3]$, we have following intermediate transition table 1.

Intermediate transition table 1

Present State (PS)	Inputs 0 Next State (NS)	Next State (NS)	Output
$\rightarrow q_0$	q_0	q_1	0
q_1	$[q_2,q_3]$	q_0	1
$[q_2,q_3]$	$[q_2,q_3]$	q_0	1
$[q_2,q_3]$	$[q_2,q_3]$	q_0	1

Applying Step 2 further on intermediate transition table we see that $q_1,[q_2,q_3]$ are equivalent states.

So, replacing q_1 and $[q_2,q_3]$ by $[q_1,q_2,q_3]$, we have intermediate transition table 2.

Intermediate transition table 2

Present State (PS)	Inputs 0	Inputs 1	
	Next State (NS)	Next State (NS)	Output
$\rightarrow q_0$	q_0	$[q_1,q_2,q_3]$	0
$[q_1,q_2,q_3]$	$[q_1,q_2,q_3]$	q_0	1
$[q_1,q_2,q_3]$	$[q_1,q_2,q_3]$	q_0	1
$[q_1,q_2,q_3]$	$[q_1,q_2,q_3]$	q_0	1

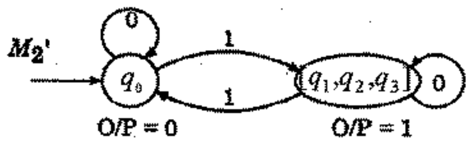
Applying Step 3 and removing redundancy, we have to delete two rows.

Now, we have the following final transition table 3 :

Transition table 3

Present State (PS)	Inputs 0	Inputs 1	
	Next State (NS)	Next State (NS)	Output
$\rightarrow q_0$	q_0	$[q_1,q_2,q_3]$	0
$[q_1,q_2,q_3]$	$[q_1,q_2,q_3]$	q_0	1

Transition diagram :



Example 2 : Consider the following transition table of a Mealy machine. Construct minimum state Mealy machine.

P S	Inputs 0		Inputs 1	
	N S	Output	N S	Output
$\rightarrow q_0$	q_0	0	q_1	0
q_1	q_0	0	q_2	1
q_2	q_0	0	q_2	1

Solution : Last two rows of transition table show that states q_1 and q_2 are equivalent states. So, replacing these states by $[q_1,q_2]$, we have the following intermediate transition table.

P S	Inputs 0		1	
	N S	Output	N S	Output
$\rightarrow q_0$	q_0	0	$[q_1,q_2]$	0
$[q_1,q_2]$	q_0	0	$[q_1,q_2]$	1
$[q_1,q_2]$	q_0	0	$[q_1,q_2]$	1

Deleting the last row, we have the following final transition table.

P S	Inputs 0		1	
	N S	Output	N S	Output
$\rightarrow q_0$	q_0	0	$[q_1,q_2]$	0
$[q_1,q_2]$	q_0	0	$[q_1,q_2]$	1