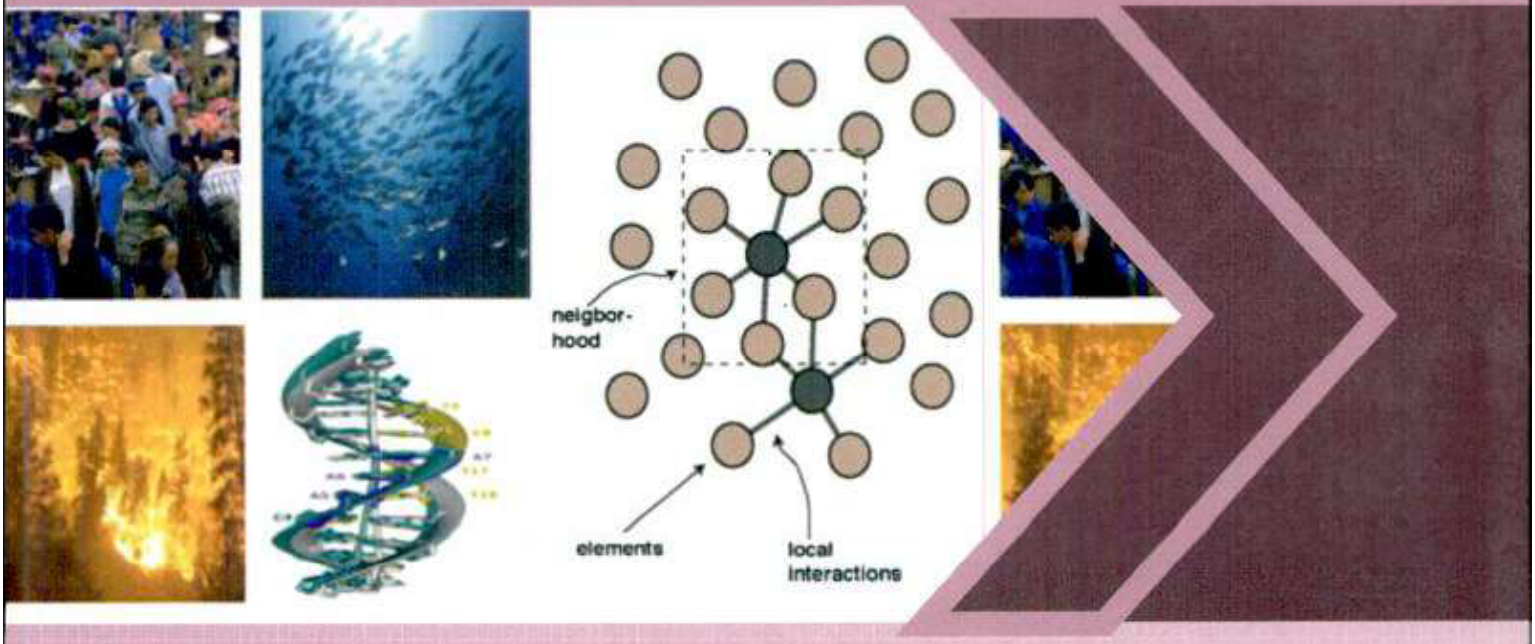
First Edition : 2008



Formal Languages & Automata Theory

A. A. Puntambekar

TECHNICAL PUBLICATIONS PUNE

Technical Publications Pune®

Strictly According to the Revised Syllabus of

JNTU - 2005 Course & JNTU - 2007 Course

Formal Languages and Automata Theory

2005 Course { *[CS 05263] Third Year, B.Tech., Semester - I (CSE)*

2007 Course { *Third Year, B.Tech., Semester - I (CSE)*

Mrs. Anuradha A. Puntambekar

M.E.(Computer)

Price Rs. 280/-

Visit us at : **www.vtubooks.com**

Technical Publications Pune®

UNIT VII (Chapter-7)

Turing Machine : Turing Machine, Definition, Model, Design of TM, Computable functions, Recursively enumerable languages. Church's hypothesis, Counter machine, Types of turing machines (Proofs not required).

Unit VIII (Chapter-8)

Computability Theory : Chomsky hierarchy of languages, Linear bounded automata and context sensitive language, LR(0) grammar, Decidability of problems, Universal turing machine, Undecidability of posts. Correspondence problem, Turing reducibility, Definition of P and NP problems, NP complete and NP hard problems.

Table of Contents :

Features of Book

- **Stepwise presentation of computational problems for easier understanding.**
- **Neat diagrams for easier understanding of concepts.**
- **More detailed and modular representation of chapters**
- **More than 250 solved examples.**
- **Book provides detailed insight into the subject.**
- **Examination oriented approach.**

Formal Languages and Automata Theory

Mrs. Anuradha A. Puntambekar

M.E.(Computer)

Visit us at : **www.vtubooks.com**

Technical Publications Pune®

Formal Languages and Automata Theory

ISBN 978 - 81 - 8431- 302 - 4

Published by :

Technical Publications Pune®

#1, Amit Residency, 412, Shaniwar Peth, Pune - 411 030, India.

Printer :

Alert DTPrinters
Sr.no. 10/3,Sinhagad Road,
Pune - 411 041

Preface

The importance of **Formal Languages and Automata Theory** is well known in Computer engineering field. Overwhelming response to my books on various subjects inspired me to write this book. The book is structured to cover the key aspects of the subject Formal Languages and Automata Theory.

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All the chapters in the book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of the subject.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

I wish to express my profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by my whole family. I wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Author

A. A. Puntambekar

Dedicated to God

Table of Contents

1 Fundamentals

1.1 Introduction

Formal languages and automata theory is based on mathematical computations. These computations are used to represent various mathematical models. In this subject we will study many interesting models such as finite automata, push down automata and turing machines. We will also discuss regular languages, non regular languages, context free languages. This subject is a fundamental subject, and it is very close to the subjects like compilers design, operating system, system software and pattern recognition system.

The automata theory is a base of this subject. Automata theory is a theory of models. Working of every process can be represented by means of models. The model can be a theorotical or mathematical model. The model helps in representing the concept of every activity. In this chapter we will discuss all the fundamentals of automata theory and those are strings, languages, operations on the languages. In the latter part of the chapter we will understand the concept of finite state machine, transition diagrams; and language recognizers.

1.2 Basic Concepts

The automata theory has a basic fundamental unit called set. The set is used to represent the mathematical model. Hence let us discuss basics of set theory.

1.2.1 Set

Set is defined as collection of objects. These objects are called elements of the set. All the elements are enclosed within curly brackets '{' and '}' and every element is separated by commas. If 'a' is an element of set A then we say that $a \in A$ and if 'a' is not an element of A then we say that $a \notin A$.

Typically set is denoted by a capital letter. The set can be represented using three methods.

1) Listing method

The elements are listed in the set.

For example : A set of element which are less than 5. Then,

$$A = \{0,1, 2, 3, 4\}$$

2) Describing properties

The properties of elements of a set define the set.

For example : A set of vowels. Hence here vowel is a property of the set which defines the set as :

$$A = \{a, e, i, o, u\}$$

3) Recursion method

The recursion occurs to define the elements of the set.

For example : $A = \{ x | x \text{ is square of n}\}$ where $n \leq 10$.

This defines the set as :

$$A = \{0,1, 4, 9, 16,..... 100\}$$

Subset : The subset A is called subset of set B if every element of set A is present in set B but reverse is not true. It is denoted by $A \subseteq B$. For example A = {1, 2, 3} and B = { 1, 2, 3, 4, 5 } then $A \subseteq B$.

Empty set : The set having no element in it is called empty set. It is denoted by A = { } and it can be written as ϕ(phi).

Null string : The null element is denoted by ε or $\wedge$ character. Null element means no value character. But ε does mean ϕ.

Power set : The power set is a set of all the subsets of its elements.

For example : $A = \{1, 2, 3\}$

Then power set : $Q = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{3,2\}, \{2,3\}, \{1,2,3\}\}$

The number of elements are always equal to 2^n where n is number of elements in original set. As in set A there are 3 elements. Therefore in power set Q there are $2^3 = 8$ elements.

Equal set : The two sets are said to be equal (A = B) if $A \subseteq B$ and $B \subseteq A$ i.e. every element of set A is an element of B and every element of B is an element of A.

For example :

A = {1, 2, 3} and B = {1, 2, 3} then A = B.

|A| denotes the length of set A. i.e. number of elements in set A.

For example : if

$$A = \{1, 2, 3, 4, 5\} \text{ then}$$

$$|A| = 5$$

1.2.2 Operations on Set

Various operations that can be carried out on set are -

i) Union

ii) Intersection

iii) Difference

iv) Complement

i) $A \cup B$ is union operation - If

$$A = \{1, 2, 3\} \quad B = \{1, 2, 4\} \quad \text{then}$$

$$A \cup B = \{1, 2, 3, 4\} \text{ i.e. combination of both the sets.}$$

ii) $A \cap B$ is intersection operation - If

$$A = \{1, 2, 3\} \quad \text{and} \quad B = \{2, 3, 4\} \quad \text{then}$$

$$A \cap B = \{2, 3\} \text{ i.e. collection of common elements from both the sets.}$$

iii) $A - B$ is the difference operation - If

$$A = \{1, 2, 3\} \quad \text{and} \quad B = \{2, 3, 4\} \quad \text{then}$$

$$A - B = \{1\} \text{ i.e. elements which are there in set A but not in set B.}$$

iv) $\overline{A}$ is a complement operation - If

$$\overline{A} = U - A \text{ where U is a universal set.}$$

For example :

if $U = \{10, 20, 30, 40, 50\}$

$A = \{10, 20\}$

then $\overline{A} = U - A$

$= \{30, 40, 50\}$

1.2.3 Cartesian Product of Two Sets

The cartesian product of two sets A and B is a set of all possible ordered pairs whose first component is member of A and whose second component is member of B. The cartesian product is denoted by $A \times B$.

$$\therefore \quad A \times B = \{\{a, b\} \mid a \in A \text{ and } b \in B\}$$

For example : Let A = {a, b} and B = {0, 1, 2}

then the cartesian product of A and B is,

$$A \times B = \{(a,0), (a,1), (a,2), (b,0), (b,1), (b,2)\}$$

1.2.4 Cardinality of Sets

The cardinality of the set is nothing but the number of members in the set.

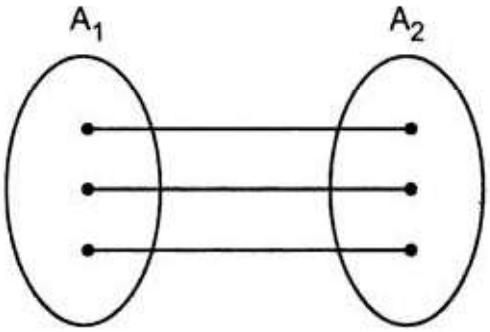


Fig. 1.1 Cardinality of two sets

These sets A_1 and A_2 have the same cardinality as there is one to one mapping of the elements of A_1 and A_2.

1.2.5 Relations

Relationship is a major aspect between two objects, even this is true in our real life. One object can be related with the other object by a 'mother of' relation. Then those two objects form a pair based on this certain relationship.

Definition : The relation R is a collection for the set S which represents the pair of elements.

For example : (a, b) is in R. We can represent their relation as a R b. The first component of each pair is chosen from a set called domain and second component of each pair is chosen from a set called range.

Properties of Relations

A relation R on set S is

1. Reflexive if iRi for all i in S.
2. Irreflexive if iRi is false for all i in S.
3. Transitive if iRj and jRk imply iRk.
4. Symmetric if iRj implies jRi.
5. Asymmetric if iRj implies that jRi is false.

Every asymmetric relation must be irreflexive.

For example : if A = { *a*, b } then

reflexive relation R can be = { (*a*, *a*), (b, b) }

irreflexive relation R can be = { (*a*, b) }

transitive relation R can be = { (*a*, b), (b, *a*), (*a*, *a*) }

symmetric relation R can be = { (*a*, b), (b, *a*) }

asymmetric relation R can be = { (*a*, b) }

Equivalent Relation

A relation is said to be equivalent relation if it is **reflexive, symmetric** and **transitive,** over some set S.

Suppose R is a set of relations and S is the set of elements.

For example : S is the set of lines in a plane and R is the relation of lines intersecting to each other.

1.3 Introduction to Formal Proofs

The formal proof can be using deductive proof and inductive proof. The deductive proof consists of sequence of statements given with logical reasoning in order to prove the first or initial statement. The initial statement is called **hypothesis.**

The inductive proof is a recursive kind of proof which consists of sequence of parameterized statements that use the statement itself with lower values of its parameter.

In short, formal proofs are the proofs in which we try to prove that statement B is true because statement A is true. The statement A is called hypothesis and B is called conclusion statement. In other words, "if A then B" we say that B is deduced from A.

Let us see some additional forms of proofs.

1.4 Additional Forms of Proof

We will discuss additional forms of proofs with the help of some examples. We will discuss

1. Proofs about sets.
2. Proofs by contradiction.
3. Proofs by counter example.

1.4.1 A Proof about Sets

The set is a collection of elements or items. By giving proofs about the sets we try to prove certain properties of the sets.

For example, if there are two expressions A and B and we want to prove that both the expressions A and B are equivalent then we need to show that the set represented by expression A is same as the set represented by expression B. Let $P \cup Q = Q \cup R$ if we map expression A with $P \cup Q$ and expression B with $Q \cup R$ then to prove A = B we need to prove $P \cup Q = Q \cup P$.

This proof is of the kind "if and only if" that means an element x is in A if and only if it is in B. We will make use of sequence of statements along with logical justification in order to prove this equivalence.

Sr. No.	Statement	Justification
1.	x is in $P \cup Q$	Given
2.	x is in P or x is in Q	1) And by definition of union
3.	x is in Q or x is in P	2) And by definition of union
4.	x is in $Q \cup P$	3) (2) And by definition of union

Table 1.1

Sr. No.	Statement	Justification
1.	x is in $Q \cup P$	Given
2.	x is in Q or x is in P	1) And by definition of union
3.	x is in P or x is in Q	2) And by definition of union
4.	x is in $P \cup Q$	3) (2) by And by definition of union

Table 1.2

Hence $P \cup Q = Q \cup P$. Thus A = B is true as element x is in B if and only if x is in A.

1.4.2 Proof by Contradiction

In this type of proof, for the statement of the form if A then B we start with statement A is not true and thus by assuming false A we try to get the conclusion of statement B. When it becomes impossible to reach to statement B we contradict our self and accept that A is true.

For example :

Prove $P \cup Q = Q \cup P$.

Proof - Assume $P \cup Q \neq Q \cup P$. Now consider x is in P or x is in Q that means x is in $P \cup Q$ (by definition of union), in other words x is in Q or a is in P which can also be written as x is in $Q \cup P$ (by definition of union). This contradicts our assumption $P \cup Q \neq Q \cup P$. Hence the assumption which we made initially is false and therefore $P \cup Q = Q \cup P$ is proved.

1.4.3 Proof by Counter Example

In order to prove certain statements, we need to see all possible conditions in which that statement remains true. There are some situations in which the statement cannot be true. For example

There is no such pair of integers such that a mod b = b mod a

Proof - Here the possibilities are either a > b or a < b. For example if a = 2 and b = 3 then 2 mod 3 ≠ 3 mod 2.

Similarly, if a = 3 and b = 2 then also 3 mod 2 ≠ 2 mod 3. Thus the given statement is true for any pair of integers but if a = b then naturally, a mod b = b mod a. Thus we need to change the statement of theorem slightly and it will be a mod b = b mod a is true only when a = b.

This type of proof is called counter example. Such proofs are true only at some specific conditions.

1.5 Inductive Proofs

Inductive proofs are special proofs based on some observations. It is used to prove recursively defined objects. This type of proof is also called as proof by mathematical induction.

The proof by mathematical induction can be carried out using following steps -

1. **Basis :** In this step we assume the lowest possible value. This is an initial step in the proof of mathematical induction.

   For example, we can prove that the result is true for n = 0 or n = 1.

2. **Induction Hypothesis :** In this step we assign value of n to some other value k. That mean we will check whether the result is true for n = k or not.

3. **Inductive Step :** In this step, if n = k is true then we check whether the result is true for n = k + 1 or not. If we get the same result at n = k + 1 then we can state that given proof is true by principle of mathematical induction.

Example 1.1 : *Prove :* $1 + 2 + 3 \ldots + n = \frac{n(n+1)}{2}$

Solution : Initially,

1) Basis of induction -

Assume, $n = 1$. Then,

$$\text{L.H.S} = n$$

$$= 1$$

$$\text{R.H.S} = \frac{n(n+1)}{2} = \frac{1(1+1)}{2}$$

$$= \frac{2}{2}$$

$$= 1$$

Now,

2) Induction hypothesis -

Now we will assume n = K and will obtain the result for it. The equation then becomes,

$$1 + 2 + 3 + \ldots + K = \frac{K(K+1)}{2} \quad \ldots (1)$$

3) Inductive step -

Now we assume that equation is true for n = K. And we will then check if it is also true for n = K + 1 or not.

Consider the equation assuming n = K + 1

$$\text{L.H.S.} = \underbrace{1 + 2 + 3 \ldots + K}_{\frac{K(K+1)}{2}} + K + 1 \qquad \because \text{equation (1)}$$

$$= \frac{K(K+1) + 2(K+1)}{2}$$

$$= \frac{(K+1)(K+2)}{2} \qquad \because \text{taking common factor}$$

$$\text{i.e.} = \frac{(K+1)(K+1+1)}{2}$$

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

Example 1.2 : *Prove :* $n! > = 2^{n-1}$

Solution : Consider,

1. **Basis of induction** - Let n = 1 then L.H.S. = 1 R.H.S. = $2^{1-1} = 2^0 = 1$ hence $n! > = 2^{n-1}$ is proved.
2. **Induction hypothesis** - Let n = n + 1 then

$$k! = 2^{k-1} \text{ where } k > = 1$$

then
$$(k+1)! = (k+1)k! \text{ by definition of n!}$$
$$= (k+1)\,2^{k-1}$$
$$= 2 \times 2^{k-1}$$
$$= 2^k$$

Example 1.3 : *Prove that* $1^2 + 2^2 + 3^2 + ... + n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

using mathematical induction.

Solution : We will first prove it by basis of induction and then will consider induction hypothesis.

1. Basis of induction -

$$\text{Let n} = 1$$
$$\text{L.H.S.} = 1^2 = 1$$
$$\text{R.H.S.} = \frac{1(1+1)(2.1+1)}{6}$$
$$= \frac{6}{6}$$
$$\text{R.H.S.} = 1$$

As L.H.S. = R.H.S it is proved for n = 1.

2. Induction hypothesis - Let, n = K.

Then

$$1^2 + 2^2 + ... + K^2 = \sum_{i=1}^{K} i^2 = \frac{K(K+1)(2K+1)}{6} \text{ is true} \quad \ldots (1)$$

Now, we will try to prove it for n = K + 1. Hence we will substitute n by K + 1.

$$\therefore \quad 1^2+2^2+3^2+.....+(K+1)^2 = \sum_{i=1}^{K+1} i^2 = \frac{(K+1)(K+1+1)(2(K+1)+1)}{6}$$

$$\text{L.H.S.} = 1^2+2^2+3^2+.....+K^2+(K+1)^2$$

We will substitute equation (1) and we will get,

$$= \frac{K(K+1)(2K+1)}{6}+(K+1)^2$$

$$= (K+1)\left[\frac{K(2K+1)}{6}+(K+1)\right]$$

$$= (K+1)\left[\frac{K(2K+1)+6K+6}{6}\right]$$

$$= (K+1)\left[\frac{2K^2+K+6K+6}{6}\right]$$

$$= (K+1)\left[\frac{2K^2+7K+6}{6}\right]$$

$$= \frac{(K+1)\,(2K^2+4K+3K+6)}{6}$$

$$= \frac{(K+1)\,(2K(K+2)+3(K+2))}{6}$$

$$= \frac{(K+1)\,((2K+3)(K+2))}{6}$$

$$= \frac{(K+1)\,((K+2)(2K+3))}{6}$$

$$= \frac{(K+1)\,(K+1+1)(2(K+1)+1)}{6}$$

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

As L.H.S. = R.H.S., it is true that using n = K + 1 the given expression can be proved.

➠ **Example 1.4 :** *Prove* : $1+4+7+.....+(3n-2) = \frac{n(3n-1)}{2}$ *for* $n > 0$

Solution : Consider,

1. Basis of induction -

Let, n = 1,

$$\text{L.H.S.} = (3.1-2)$$

$$= 1$$

$$\text{R.H.S.} = \frac{1(3.1-1)}{2} = \frac{2}{2} = 1$$

As L.H.S. = R.H.S. given expression is true for n = 1.

2. Induction hypothesis - We assume that the given expression is true for n = K.

i.e. $$1+4+7+.....+(3K-2) = \frac{K(3K-1)}{2} \text{ is true.} \quad \ldots (1)$$

Now we will prove it for n = K + 1.

$$\text{L.H.S.} = 1+4+7+..... + (3K-2)+(3(K+1)-2)$$

We will substitute R.H.S. of equation (1).

$$= \frac{K(3K-1)}{2} + (3(K+1)-2)$$

$$= \frac{K(3K-1)+2[3(K+1)-2]}{2}$$

$$= \frac{3K^2 - K + 6K + 2}{2}$$

$$= \frac{3K^2 + 5K + 2}{2}$$

$$= \frac{3K^2 + 2K + 3K + 2}{2}$$

$$= \frac{(2K+2)+(3K^2+3K)}{2}$$

$$= \frac{2(K+1)+3K(K+1)}{2}$$

$$= \frac{(K+1)(3K+2)}{2}$$

$$= \frac{(K+1)(3(K+1)-1)}{2}$$

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

As L.H.S. = R.H.S. the given expression is true for n = K + 1.

Example 1.5 : *Prove* : $2^n > n^3$ *where* $n \geq 10$.

Solution : 1) Basis of induction -

Initially we assume,

$$n = 10 \quad \text{then}$$

$$2^{10} > (10)^3 \quad \text{is true}$$

2) Induction hypothesis -

Now assume, $n = K$ then

$$2^K > K^3$$

3) Inductive step -

Now we will assume $K = K + 1$ then the equation becomes

$$\text{L.H.S.} = 2^{(K+1)}$$

$$= 2 \cdot 2^K$$

$$\geq \left(1 + \frac{1}{10}\right)^3 \cdot 2^K$$

$$\geq \left(1 + \frac{1}{K}\right)^3 \cdot 2^K$$

But as $2^K > K^3$ we will replace 2^K by K^3 then

$$\geq \left(1 + \frac{1}{K}\right)^3 K^3$$

$$\geq \left(\frac{K+1}{K}\right)^3 K^3$$

$$\geq \frac{(K+1)^3}{K^3} \cdot K^3$$

$$\geq (K+1)^3$$

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

Hence $2^{K+1} \geq (K+1)^3$ is also true.

This proves $2^n > n^3$ for $n \geq 10$.

Example 1.6 : *Prove : $2^n > n$ using principle of mathematical induction.*

Solution : 1) Basis of induction - Assume n = 1 then,

$$2^n > n \text{ becomes}$$

$$2^1 > 1$$

$$2 > 1$$

This also implies $2^n - n > 0$

2) Induction hypothesis : Assume n = K is true

i.e. $2^K > K$

i.e. $2^K - K > 0$

Thus $2^K - K$ implies a positive number. i.e.

$$2^K - K = t \quad \ldots (1)$$

3) Inductive step : Assume K = K + 1 then

$$2^{K+1} - (K + 1)$$

$$2^K \cdot 2 - (K + 1)$$

But from equation (1) $2^K = K + t$ then

$$(K + t) \cdot 2 - K - 1$$

i.e. $2K + 2t - K - 1$

$$K + 2t - 1$$

> 0 i.e. as positive number.

This implies $2^{K+1} > K + 1$. Hence $2^n > n$ true.

1.6 Introduction to Defining Language

Alphabet : An alphabet is a finite collection of symbols. We can not define symbol formally.

The example of alphabet is,

$$S = \{a, b, c \ldots\ldots z\}$$

The elements a, b,.....z are the alphabets.

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

Here 0 and 1 are alphabets, denoted by W.

String : String is finite collection of symbols from alphabet.

For example, if $\sum$ = {a, b} then various strings that can be formed from Σ are {ab, ab, aa, aaa, bb, bbb, ba, aba ...}. An infinite number of strings can be generated from this set.

- The empty string can be denoted by ε.
- The prefix of any string is any number of leading symbols of that string and suffix is any number of trailing.

Symbols

For example, in string "0011" the prefixes can be 0, 00, 001. Similarly suffixes can be 1, 11, 011.

In string "Mango" the prefix can be 'Man' and the suffix can be 'go'.

Language : The language is a collection of appropriate strings.

The language is defined using an input set. The input set is typically denoted by letter $\sum$.

For example : $\sum = \{\varepsilon, 0, 00, 000, ...\}$

Here the language L is defined as 'any number of Zeros'.

Note that language is formed by appropriate strings and strings are formed by alphabets.

Operations on string

Various operations that can be carried out on strings are

1. Concatenation :

In this operation two strings are combined together to form a single string.

For example, x = {white} y = {house}

Then the concatenated string will be xy = {white house}.

2. Transpose :

The transpose of operation is also called reverse operation.

For example, $(xa)^T = a(x)^T$.

if $(ababbb)^T$ = (bbbaba).

3. Palindrome :

Palindrome of string is a property of a string in which string can be read same from left to right as well as from right to left.

For example, the string "MadaM" is palindrome because it is same if we read it from left to right or from right to left.

Operations on language

Language is collection of strings. Hence operations that can be carried out on strings are the operations that can be carried out on language.

If L_1 and L_2 are two languages then,

i) Union of two languages is denoted as $L_1 \cup L_2$.

ii) Concatenation of two languages is denoted by $L_1 L_2$

iii) Intersection of two languages is denoted by $L_1 \cap L_2$.

iv) Difference of two language is denoted by $L_1 - L_2$.

Example 1.7 : *Consider the string X = 110 and y = 0110 then find.*

(i) xy

(ii) yx

(iii) x^2

(iv) $\in y$

Solution : Let x = 110 and y = 0110 then

(i) xy is the concatenation operation. Hence we will concatenate the two strings from x an y respectively.

xy = 1100110.

(ii) yx suggest the concatenation of string from set y with string from x hence yx = 0110110.

(iii) x^2 = x.x. Hence concatenate the string from x with the string from set x itself.

Hence x^2 = 110110

(iv) $\in$ y means the string belonging to set y which is 0110.

Example 1.8 : *Describe the following language over the input set A = {a, b}.*

i) $L_1 = \{a, ab, ab^2\}$

ii) $L_2 = \{ a^n b^n \mid n \geq 1 \}$

iii) $L_3 = \{ a^m b^n \mid n > 0 \}$

Solution : The language is defined over the set A = {a, b}.

i) L_1 consists of all the strings with letter a followed by any number of b's.

ii) L_2 consists of all the strings having equal number of a's and equal number of b's such that all the a's are followed by all the b's.

iii) L_3 consists of all the strings with any numbers of a's followed by atleast one b.

1.7 Kleen Closures

The kleen closure is also called star closure. This is set of strings of any length (including null string ε). Each string is obtained from input set Σ the kleene star of the empty language ϕ is the empty string ε.

Example 1.9 : *Let* $\Sigma = \{a,b\}$ *obtain* $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 .....$

Solution : $\Sigma^* = \{\varepsilon, a, b, aa, bb, ab, ba, aba, aaa, bbb, baba, abaab, ...\}$ is a set of strings of any length.

The positive closure Σ^+ can be defined as $\Sigma^1 \cup \Sigma^2 \cup \Sigma^3$ that means it consists of all the strings of any length except a null string.

Example 1.10 : *Let* $\Sigma = \{a,b\}$ *obtain* $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 .....$

Solution : Σ^* = {a, b, aa, bb, ab, ba, aba, aaa, bbb, baba, abaab, ...} is a set of strings of any length except null string (epsilon).

Note that $\Sigma^* = \Sigma^+ + \varepsilon$.

1.8 Arithmetic Expressions

The arithmetic expressions are those expressions in which arithmetic operations are performed on operands. The arithmetic operators can be denoted as S ={+,–,*,/,^}. The arithmetic expressions can be formed as :

$R_1 = a + b$

$R_2 = a - b$

$R_3 = a * b$

$R_4 = a / b$

$R_5 = a \wedge b$

1.9 Graphs

A graph, denoted G = (V, E) consists of finite set of vertices (or nodes) V and set of edges, the edges are nothing but pair of vertices.

For example :

Here the E_1 is a edge connecting the vertices V_1 and V_2.

The set $V = \{V_1, V_2, V_3, V_4\}$ and

$E = \{E_1, E_2, E_3, E_4\}$

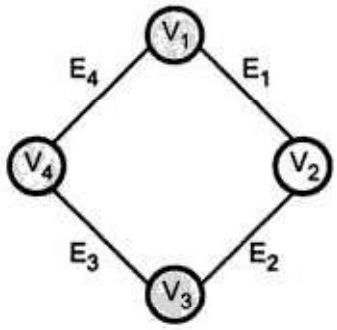


Fig. 1.2

1.9.1 Directed Graph

The directed graph is also a collection of vertices and edges where the directions are mentioned along the edges.

For example :

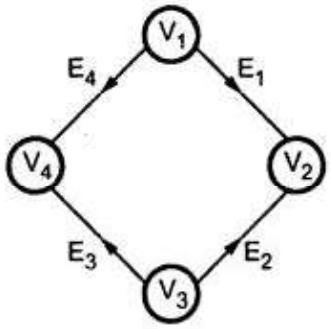


Fig. 1.3

The edge E_1 shows the direction to V_2 vertex from vertex V_1.

We will see the directed graphs in further chapters and we will call those graphs as transition graphs.

1.10 Trees

Trees is a collection of vertices and edges with following properties.

1. There is one vertex, called root which has no predecessors.
2. From this root node all the successors are ordered.

For example :

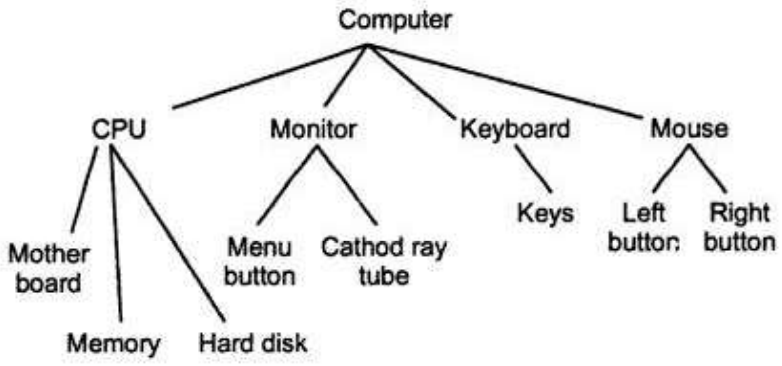


Fig. 1.4

In the above example, at the leaf node all the components of computer are given, when motherboard, memory and harddisk these components are configured they are meant for CPU always. Keys are the prominent component for keyboard. When CPU, monitor, keyboard and mouse are taken as one set, then it constitutes a device called computer.

Computer is a root node in the above figure. CPU, monitor, keyboard, mouse are the interior nodes. Motherboard, memory harddisk are the leaf nodes for the CPU. Monitor is a parent node or father of Menu button and cathode ray tube whereas menu button is a left child of monitor and cathode ray tube is a right child of monitor.

The only difference between graphs and trees is that graphs do not have special node called root node.

1.10.1 Binary Trees

A binary tree is a data structure in which each node has at the most two children which are called left child and right child.

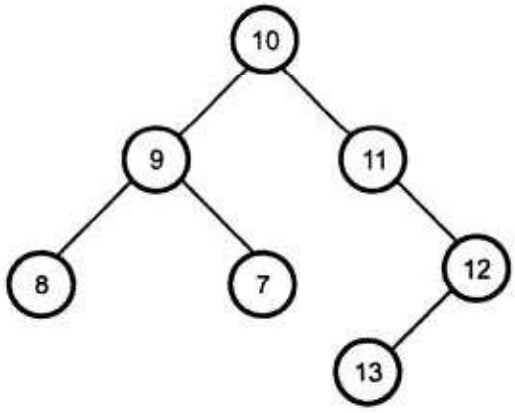


Fig. 1.5 Binary tree

For node 9 the left child is 8 and right child is 7.

- **Leaf** is a node having no children. In above given tree nodes 8, 7, 13 are leaf nodes.
- The depth of node n is length of path from the root to the node. The set of all nodes at given depth is called **level**. The node 8 has depth 2. At level 2 nodes 9 and 11 are lying.
- The **height** of tree the maximum depth of the node. The above given tree is having height 3.
- A **complete binary tree** is a binary tree in which there are 2^K nodes at every depth K where $K < n$.

1.11 Finite State Machine

The finite state system represents **a mathematical model** of a system with certain input. The model finally gives certain output. The input when is given to the machine it is processed by various states, these states are called as **intermediate states.**

The very good example of finite state system is a control mechanism of elevator. This mechanism only remembers the current floor number pressed, it does not remember all the previously pressed numbers.

The finite state system is a very good design tool for the programs such as text editors and lexical analyzers (which is used in compilers). The lexical analyzer is a program which scans your program character by character and recognizes those words as tokens.

1.11.1 Definition

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

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

Σ is **input** alphabet, indicates input set.

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

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

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

1.11.2 Finite Automata Model

The finite automata can be represented as :

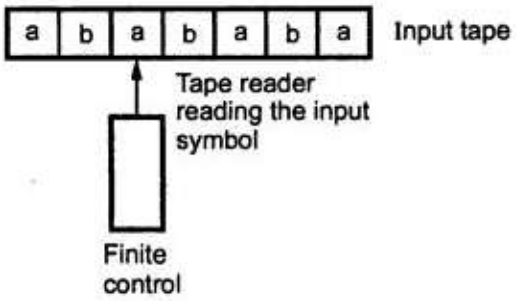


Fig. 1.6 Model for finite automata

The finite automata can be represented using

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

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

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

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

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

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

with q as final state.

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

A machine M accepts a language L iff,

$$L = L(M).$$

1.11.3 Types of Automata

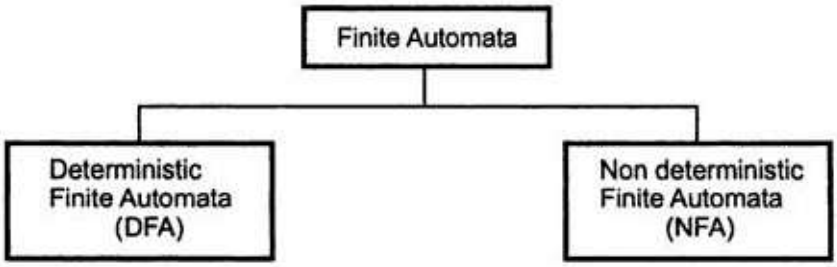


Fig. 1.7 DFA and NFA

There are two types of finite automata -

i) Deterministic Finite Automata.

ii) Non deterministic Finite Automata.

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

1.12 Acceptance of Strings and Languages

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

1. Transition diagram -

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

1) Finite set of states K.

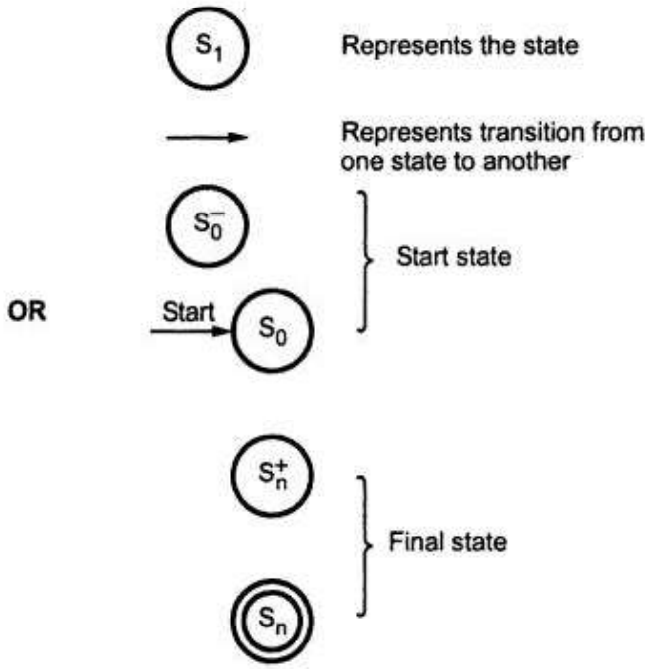
2) Finite set of symbols Σ.

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

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

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

The notations used in transition diagram are -



For example :

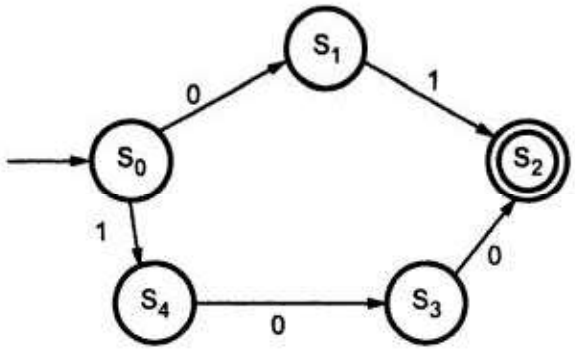
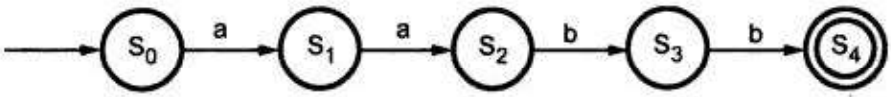


Fig. 1.8 Transition diagram

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

Another example :



We have drawn a transition diagram for the input aabb.

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

2. Transition table

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

For example :

Input / States	a	b
q_0	q_1	-
q_1	-	q_2
q_2	q_2	-

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

1.13 Deterministic Finite Automata (DFA)

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

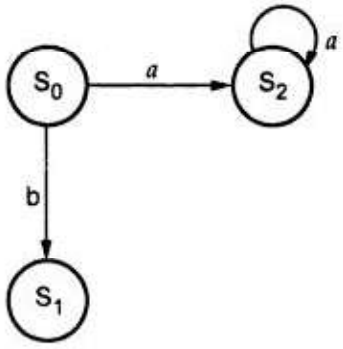


Fig. 1.9 Deterministic finite automata

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

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

Definition of DFA

A deterministic finite automation is a collection or following things -

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

2) The finite set of input symbols Σ.

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

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

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

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

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

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

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

Example 1.11 : *Design a FA which accepts the only input 101 over the input set* $Z = \{0, 1\}$.

Solution :

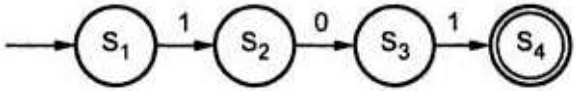


Fig. 1.10 For Ex. 1.11

Note that in the problem statement it is mentioned as only input 101 will be accepted. Hence in the solution we have simply shown the transitions, for input 101. There is no other path shown for other input.

Example 1.12 : *Design a FA which checks whether the given binary number is even.*

Solution : The binary number is made up of 0's and 1's when any binary number ends with 0 it is always even and when a binary number ends with 1 it is always odd. For example,

0100 is a even number, it is equal to 4.

0011 is a odd number, it is equal to 3.

And so while designing FA we will assume one start state, one state ending in 0 and other state for ending with 1. Since we want to check whether given binary number is even or not, we will make the state for 0, the final state.

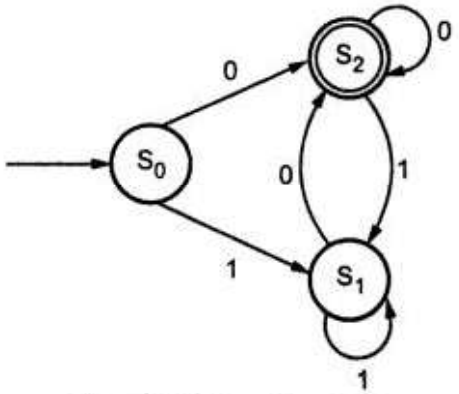


Fig. 1.11 For Ex. 1.12

The FA indicates clearly S_1 is a state which handles all the 1's and S_2 is a state which handles all the 0's. Let us take some input.

$$01000 \Rightarrow 0\underline{S_2}1000$$

$$01\underline{S_1}000$$

$$010\underline{S_2}00$$

$$0100\underline{S_2}0$$

$$01000S_2$$

Now at the end of input we are in final or in accept state so it is a even number. Similarly let us take another input.

$$1011 \Rightarrow 1\underline{S_1}011$$
$$10\underline{S_2}11$$
$$101\underline{S_1}1$$
$$1011S_1$$

Now we are at the state S_1 which is a non final state.

Another idea to represent FA with the help of transition table.

States \ Input	0	1
→ S_0	S_2	S_1
S_1	S_2	S_1
(S_2)	S_2	S_1

Transition table

Thus the table indicates in the first column all the current states. And under the column 0 and 1 the next states are shown.

The first row of transition table can be read as : when current state is S_0, on input 0 the next state will be S_2 and on input 1 the next state will be S_1. The arrow marked to S_0 indicates that it is a start state and circle marked to S_2 indicates that it is a final state.

Example 1.13 : *Design FA which accepts only those strings which start with 1 and ends with 0.*

Solution : The FA will have a start state A from which only the edge with input 1 will go to next state.

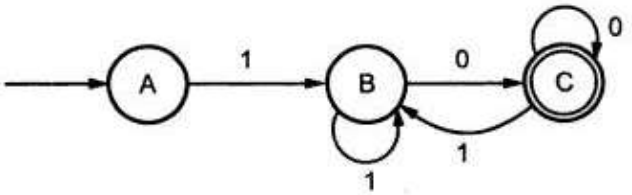


Fig. 1.12 For Ex. 1.13

In state B if we read 1 we will be in B state but if we read 0 at state B we will reach to state C which is a final state. In state C if we read either 0 or 1 we will go to state C or B respectively. Note that the special care is taken for 0, if the input ends with 0 it will be in final state.

Example 1.14 : *Design FA which accepts odd number of 1's and any number of 0's.*

Solution : In the problem statement, it is indicated that there will be a state which is meant for odd number of 1's and that will be the final state. There is no condition on number of 0's.

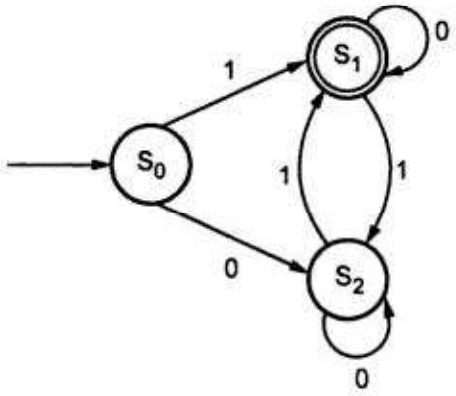


Fig. 1.13 For Ex. 1.14

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

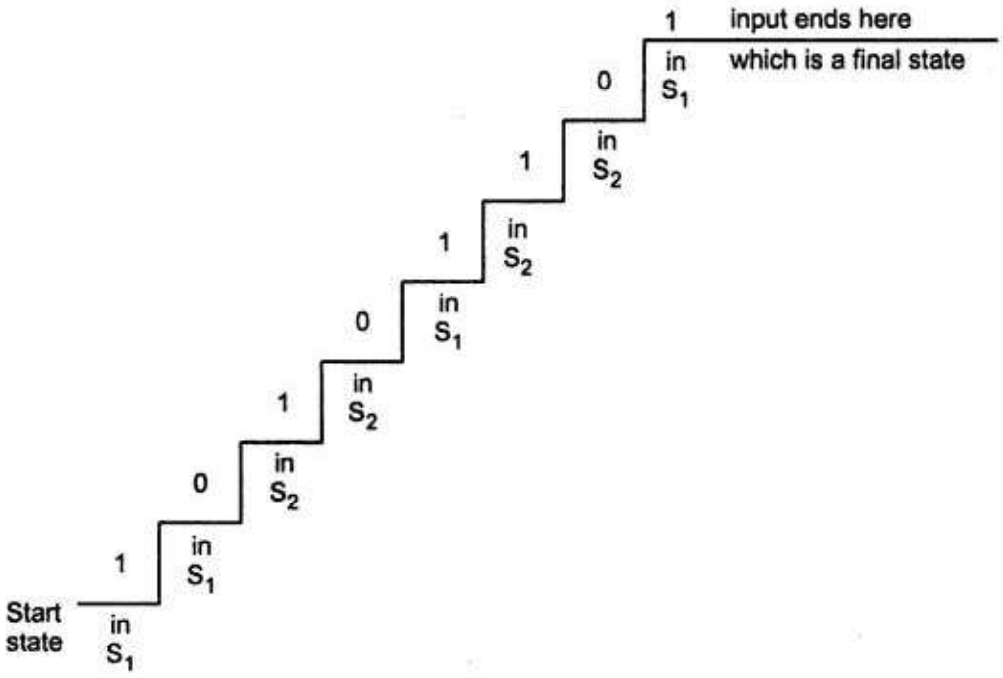


Fig. 1.14 Ladder diagram of processing the input

Example 1.15 : *Design FA which checks whether the given unary number is divisible by 3.*

Solution :

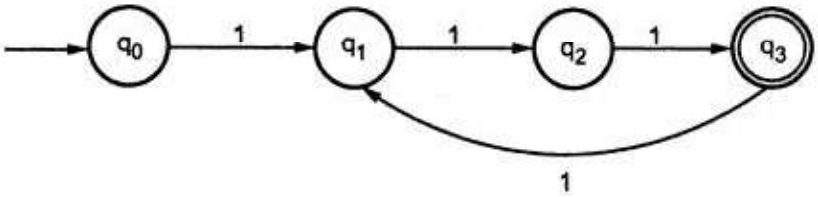


Fig. 1.15 For Ex. 1.15

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

Input	1
→ q_0	q_1
q_1	q_2
q_2	q_3
(q_3)	q_1

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

Start 111111

State q_0

$1\underline{q_1}11111$

$11\underline{q_2}1111$

$111q_3111$

$1111\underline{q_1}11$

$11111\underline{q_2}1$

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

Example 1.16 : *Design FA to check whether given decimal number is divisible by three.*

Solution : To determine whether the given decimal number is divisible by three, we need to take the input number digit by digit. Also, while considering its divisibility by three, we have to consider that the possible remainders could be 1, 2 or 0. The

remainder 0 means, it is divisible by 3. Hence from input set {0, 1, 2, ..., 9} (since decimal number is a input), we will get either remainder 0 or 1 or 2 while testing its divisibility by 3. So we need to group these digits according to their remainders. The groups are as given below -

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

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

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

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

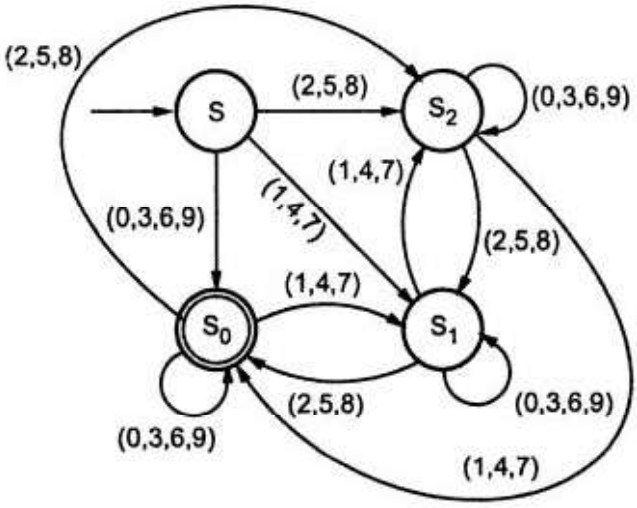


Fig. 1.16 For Ex. 1.16

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

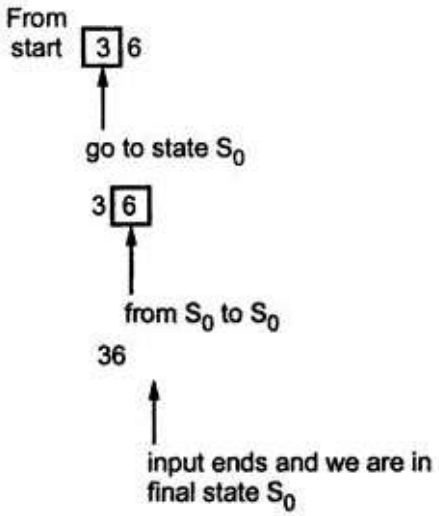


Fig. 1.17

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

S 121

$1\underline{S_1}\,21$

$12\underline{S_0}\,1$

$121\,S_1$ which is remainder 1 state.

Example 1.17 : *Design FA which checks whether a given binary number is divisible by three.*

Solution : As we have discussed in previous example, the same logic is applicable to this problem even ! The input number is a binary number. Hence the input set will be $\Sigma = \{0, 1\}$. The start state is denoted by S, the remainder 0 is by S_0, remainder 1 by S_1 and remainder 2 by S_2.

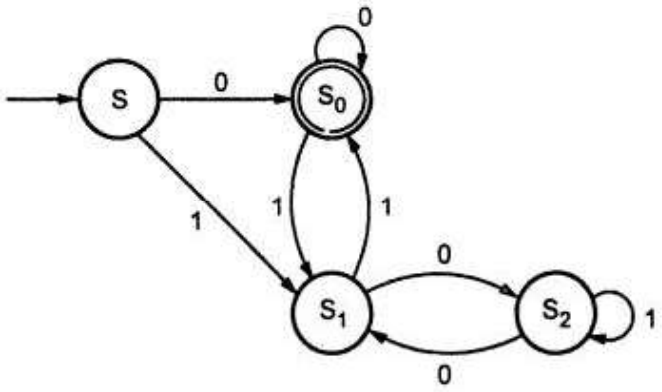


Fig. 1.18 For Ex. 1.17

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

0	1	0
↑	↑	↑
S_0	S_1	S_2

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

1	0	0	1
↑	↑	↑	↑
S_1	S_2	S_1	$\boxed{S_0}$

Thus the number is really divisible by 3.

Example 1.18 : *Design FA which accepts even number of 0's and even number of 1's.*

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

even number of 0 and even number of 1,

even number of 0 and odd number of 1,

odd number of 0 and even number of 1,

odd number of 0 and odd number of 1.

Let us try to design the machine

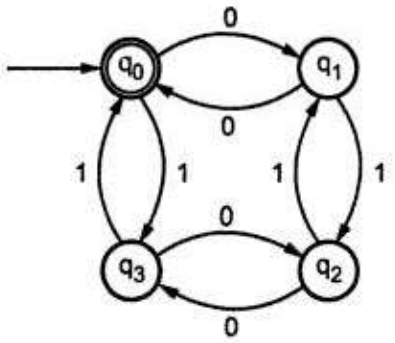


Fig. 1.19

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

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

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

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

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

The transition table can be as follows -

	0	1
→ q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

Example 1.19 : *Design FA to accept the string that always ends with 00.*

Solution :

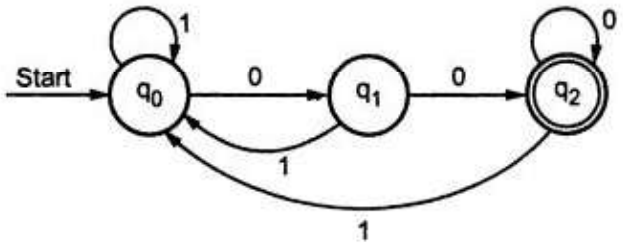


Fig. 1.20

If the input is 01001100 it will be processed as

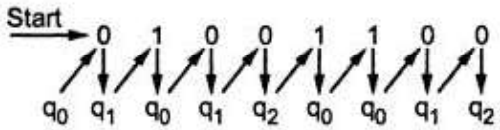


Fig. 1.21

The q_2 is a final state, hence the input is accepted.

Example 1.20 : *Construct the transition graph for a FA which accepts a language L over* $\Sigma\{0, 1\}$ *in which every string start with 0 and ends with 1.*

Solution :

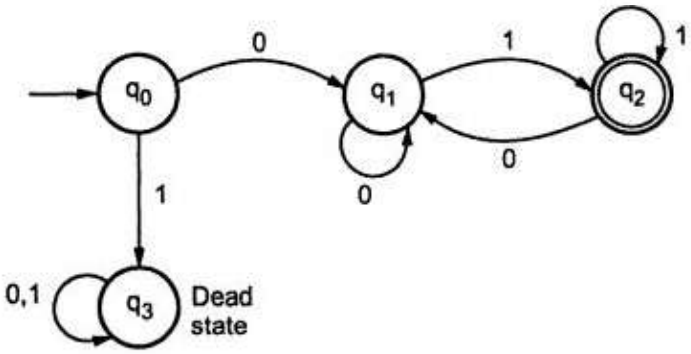


Fig. 1.22

In the above transition graph we have handled the case as if the input starts with 1 then it will be in q_3 state which is a dead state and never lead to final state. Thus this machine strictly handles the strings starting with 0 and ending with 1.

Example 1.21 : *Design FA to accept L, where L = {Strings in which a always appears trippled} over the set Σ = {a, b}.*

Solution : For this particular language the valid strings are *aaa*b, b*aaaaaa*, bb*aaa*b and so on. The *a* always appears in a clump of 3. The TG (transition graph) will look like this -

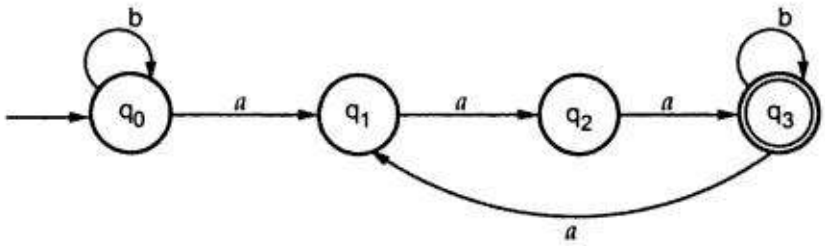


Fig. 1.23 For Ex. 1.21

Note that the sequence of tripple *a* is maintained to reach to the final state.

Example 1.22 : *Design FA to accept L where all the strings in L are such that total number of a's in them are divisible by 3.*

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

Hence,

S_0 : State of remainder 0.

S_1 : State of remainder 1.

S_2 : State of remainder 2.

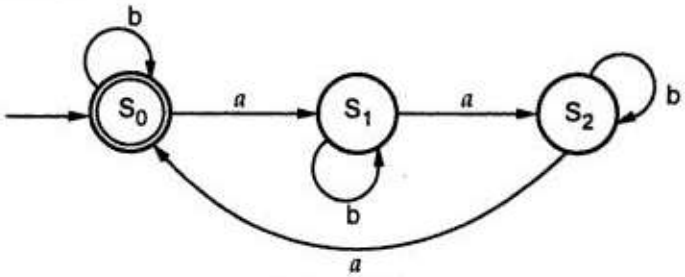


Fig. 1.24

Note the difference between previous example and this, here there is no condition as *a* should be in clump but total number of *a*'s in a string are divisible by 3. Hence b's are allowed in between.

Example 1.23 : *Design a FA that reads strings made up of letters in the word. CHARIOT and recognize those strings that contain the word 'CAT' as a substring.*

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

Step 1 :

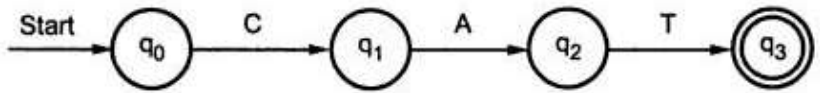


Fig. 1.25

Step 2 :

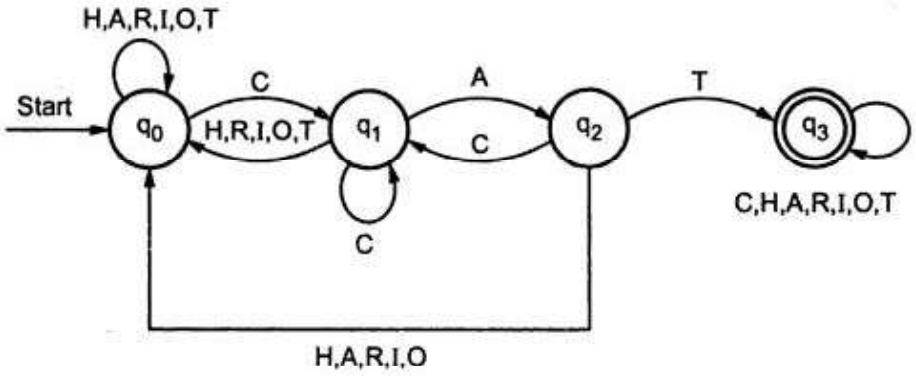


Fig. 1.26 Required finite automata

From above figure it is clear that on recognizing C initially from Σ = {C, H, A, R, I, O, T} we are moving to state q_1, other than C all the characters are remaining in state q_0 only. From q_1 the character A will be identified and from q_2 character T will be identifier. The transition table can be drawn as follows

Input / State	C	H	A	R	I	O	T
$\rightarrow q_0$	q_1	q_0	q_0	q_0	q_0	q_0	q_0
q_1	q_1	q_0	q_2	q_0	q_0	q_0	q_0
q_2	q_1	q_0	q_0	q_0	q_0	q_0	q_3
(q_3)	q_3	q_3	q_3	q_3	q_3	q_3	q_3

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

Example 1.24 : *Design a DFA to accept string of a's and b's ending with 'abb' over* $\Sigma = \{a, b\}$.

Solution : Initially we will design DFA for the string abb as.

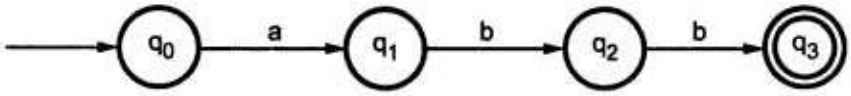


Fig. 1.27

In this transition, q_0 is a start state and q_3 is a final state. Thus only after recognizing 'abb' the DFA will enter in final state. We can design the complete DFA as

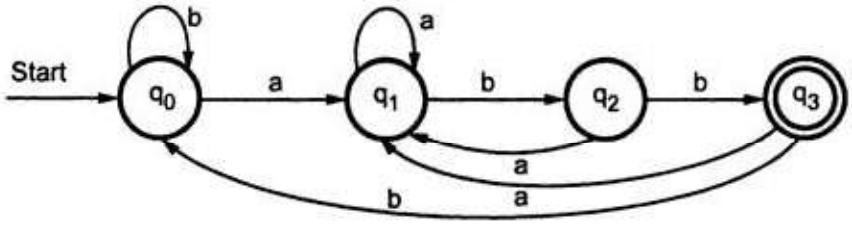


Fig. 1.28

Example 1.25 : *Design DFA to accept odd and even numbers represented using binary notation.*

Solution : The binary number that ends with 0 is an even number and binary number that ends with 1 is an odd number. Hence the DFA is

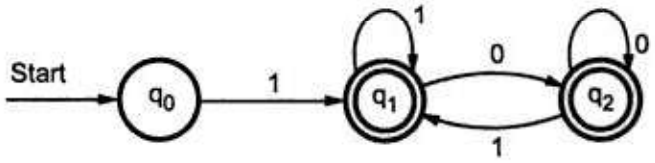


Fig. 1.29

Example 1.26 : *Write a DFA to accept the language* $L = \{L : |W| \bmod 5 \neq 0\}$.

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

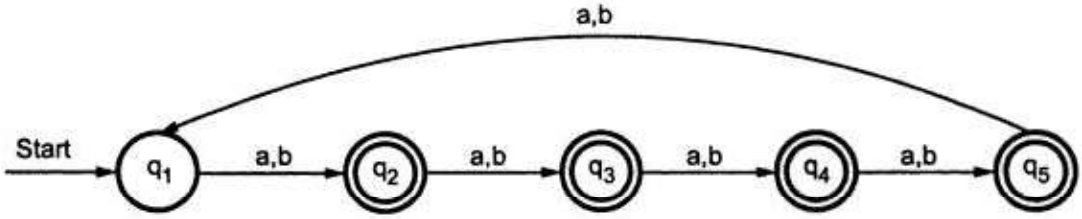


Fig. 1.30

The transition table can be drawn as follows -

	a	b
q_1	q_2	q_2
q_2	q_3	q_3
q_3	q_4	q_4
q_4	q_5	q_5
q_5	q_1	q_1

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

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

where q_5 is a final state.

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

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

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

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

Example 1.27 : *Design a DFA* $L(M) = \{W \mid W \in \{0,1\}^*\}$ *and W is a string that does not contain consequetive 1's.*

Solution : When three consequetive 1's occur the DFA will be

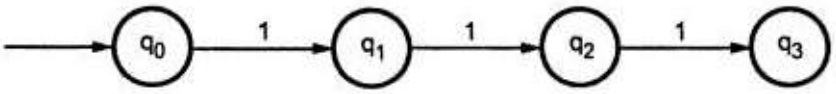


Fig. 1.31

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

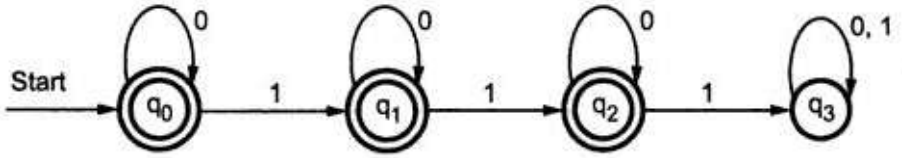


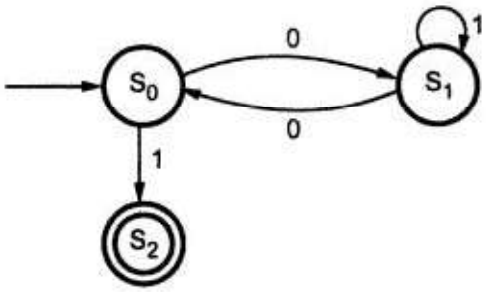
Fig. 1.32

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

$$M = \{ Q, \Sigma, q_0, F \} \quad \text{where}$$
$$Q = \{ q_0, q_1, q_2, q_3 \}$$
$$\Sigma = \{0,1\}$$
$$F = \{ q_0, q_1, q_2 \}$$

Example 1.28 : *Design a DFA which accepts strings with even number of 0's followed by single 1 over* $\Sigma = \{0,1\}$.

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



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

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

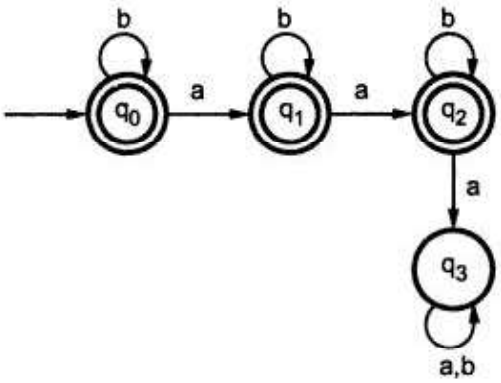
We now will reach to final state S_2, and the input ends here. Hence 000101 is a valid input string. The transition table for above drawn DFA can be represented as

States \ Input	0	1
S_0	S_1	S_2
S_1	S_0	S_1
((S_2))	-	-

Transition table

Example 1.29 : *Design DFA which accepts all the strings not having more than two a's over* $\Sigma = \{a, b\}$.

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



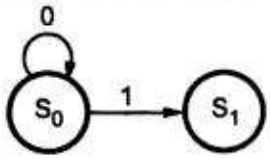
The transition table can be as shown -

States \ Input	a	b
((q_0))	q_1	q_0
((q_1))	q_2	q_1
((q_2))	q_3	q_2
q_3	q_3	q_3

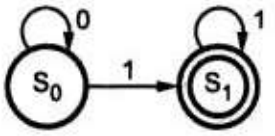
Transition table

Example 1.30 : *Design a DFA for accepting all the strings of* $\{L = 0^m 1^n \mid m \geq 0 \text{ and } n \geq 1\}$

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



But we have 1^n in the language. Hence the DFA can be



The transition table can be

State \ Input	0	1
S_0	S_0	S_1
S_1	–	S_1

Consider a string 00111, which is a valid string and is accepted by above drawn DFA.

$$\begin{aligned}\delta(S_0, 00111) &\vdash (S_0, 0111)\\ &\vdash (S_0, 111)\\ &\vdash (S_1, 11)\\ &\vdash (S_1, 1)\\ &\vdash (S_1, \varepsilon)\end{aligned}$$

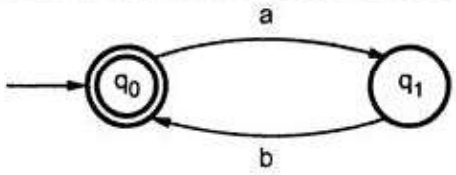
Thus we reach to final state S_1 on acceptance of 00111.

Example 1.31 : *Design DFA over* $\sum = \{a, b\}$ *for*

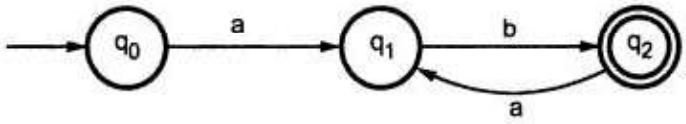
i) $(ab)^n$ *with* $n \geq 0$.

ii) $(ab)^n$ *with* $n \geq 1$.

Solution : This is a language of ab in a pair in i) condition ε is accepted and in ii) condition ε is not accepted. The DFA for i) can be



ii) The DFA will be -



The string accepted by

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

$\vdash (q_0, \varepsilon)$

Here q_0 is a final state.

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

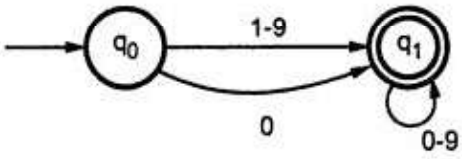
$\vdash (q_2, \varepsilon)$

Here q_2 is final state.

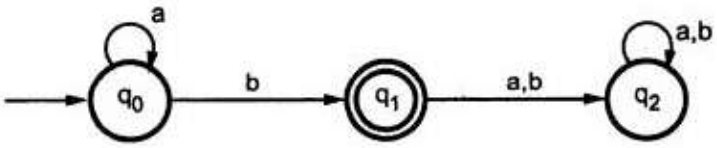
Example 1.32 : *Design DFA for accepting the set of integers.*

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

Hence the DFA will be



Example 1.33 : *Recognize the language given by following DFA.*



Solution : In the given DFA, at q_0 state a self loop for symbol a is given. That means any number of a's are allowed. Then if single 'b' comes then it leads to final state. But after this b if anything i.e. 'a' or 'b' comes then we reach to a non final state q_2 and in this state whatever may come, we will be in q_2 state only. Basically state

q_2 is called a dead state. Thus it is a language in which any number of a's should be followed by single b only. Hence this language is denoted as

$\{L = a^n b \mid n \geq 0\}$.

1.14 Non Deterministic Finite Automata (NFA)

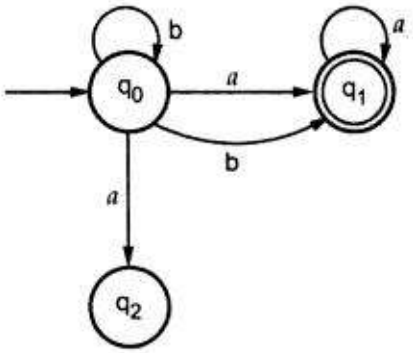


Fig. 1.27 Non deterministic finite automata

The concept of Non-deterministic Finite Automata is exactly reverse of Deterministic Finite Automata. The Finite Automata is called NFA when there exists many paths for a specific input from current state to next state. The NFA can be shown as in Fig. 1.27.

Note that the NFA shows from q_0 for input a there are two next states q_1 and q_2. Similarly, from q_0 for input b the next states are q_0 and q_1.

Thus it is not fixed or determined that with a particular input where to go next. Hence this FA is called non deterministic finite automata.

Consider the input string bb*a*. This string can be derived as

	Input	b	b	*a*
	Path	q_0	q_0	q_1
or	Input	b	b	*a*
	Path	q_0	q_0	q_2
or	Input	b	b	*a*
	Path	q_0	q_1	q_1

Thus you cannot take the decision of which path has to be followed for deriving the given string.

Definition of NFA

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

1) Q is a finite set of states.

2) Σ is a finite set of inputs.

3) δ is called next state or transition function.

4) q_0 is initial state.

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

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

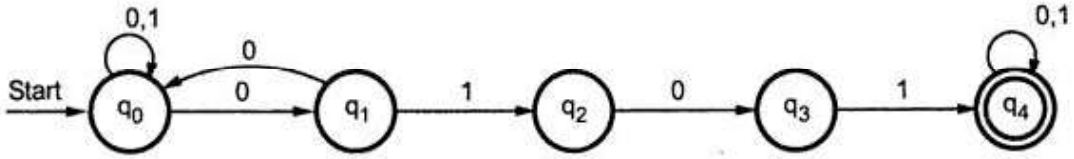
Difference between NFA and DFA

The DFA is a deterministic finite automata whereas NFA is a non-deterministic finite automata. In DFA, for a given state, on a given input we reach to a deterministic and unique state. On the other hand, in NFA we may lead to more than one states for given input. The DFA is a subset of NFA. We need to convert NFA to DFA in the design of compiler.

Example 1.34 : *Construct a NFA for the language*

L_1 = *{consisting a substring 0 1 0 1}*
$L_2 = \{a^n \cup b^n\}$.

Solution : We will consider L_1 first to design NFA. There can be any combination of 0 and 1 in the language but a substring 0101 must be present. We will get such a substring then it leads to a final state or accept state.



This is a NFA as for 0 input we have two different paths one going to q_0 and other is going to q_1.

The string 00010101 is acceptable by above given NFA and it is as shown below.

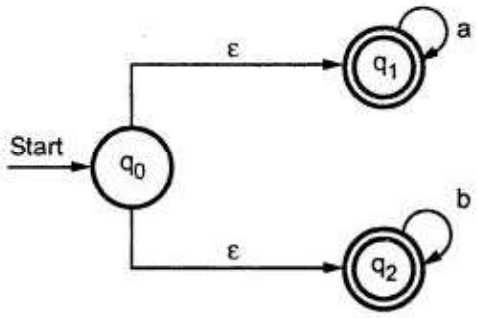
$$
\begin{aligned}
00010101 \quad & 0q_0 \vdash 010101 \\
& 00q_0 \vdash 010101 \\
& 000q_1 \vdash 10101 \\
& 0001q_2 \vdash 0101 \\
& 00010q_3 \vdash 101 \\
& 000101q_4 \vdash 01 \\
& 0001010\ q_4 \vdash 1
\end{aligned}
$$

$00010101q_4 \vdash$

00010101 is accepted as q_4 is final state.

Now we will build a NFA for L_2. The language L_2 is a language in which there be any number of a's or any number of b's. It accepts {a, b, aa, bb, aaa, bbb, ...}.

Hence the NFA will be



The NFA shows two different states q_1 and q_2 for the input ε from q_0 state.

Here ε (pronounced as epsilon) is basically a null move i.e. a move carrying no symbol from input set Σ. But a state change occurs from one state to other.

Example 1.35 : *Design the NFA transition diagram for the transition table as given below -*

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

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

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

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

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

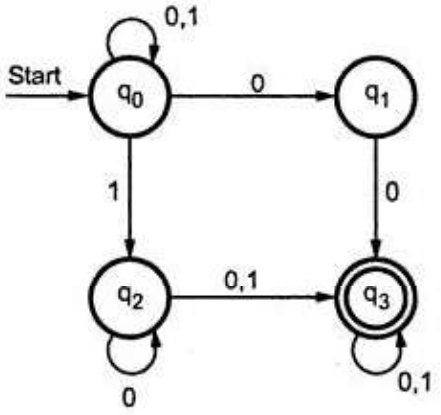
Then, $\delta(q_1, 0) = \{q_3\}$

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

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

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

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

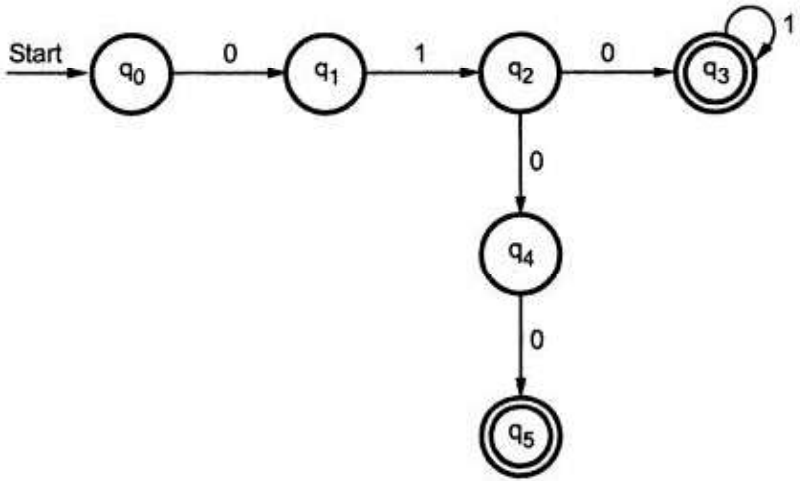


Example 1.36 : *Construct NFA for the language*

$$L = \{0101^n \cup 0100 \mid n \geq 0\}$$

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

Solution : Here in language L first three symbols are common i.e. 010. Hence the NFA can be drawn as



The states q_3 and q_5 are final states accepting 0101^n and 0100 respectively. The NFA then can be denoted by,

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \delta, q_0, \{q_3, q_5\})$$

Example 1.37 : *Construct a transition diagram for the NDFA* $M = (\{q_1, q_2, q_3\}, \delta, q_1, \{q_3\})$ *where* δ *is given by,*

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

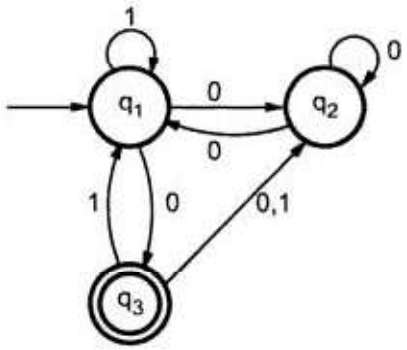
$\delta(q_2, 0) = \{q_1, q_2\}$ $\quad \delta(q_2, 1) = \phi$

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

Solution : Firstly we will design a transition table using the given mapping function δ.

States \ Input	0	1
$\rightarrow q_1$	$\{q_2, q_3\}$	q_1
q_2	$\{q_1, q_2\}$	ϕ
(q_3)	q_2	$\{q_1, q_2\}$

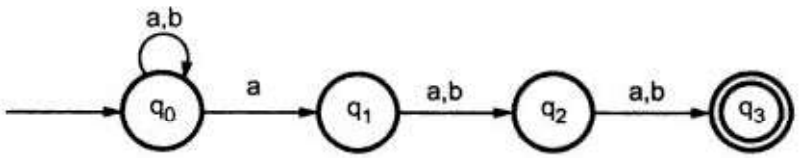
The NDFA will be



Example 1.38 : *Construct a NFA for a language L which accepts all the strings in which the third symbol from right end is always a. Over* $\Sigma = \{a, b\}$.

Solution : The strings in such a language are of the form

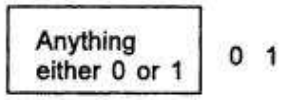
Thus we get third symbol from right end as 'a' always. The NFA then can be



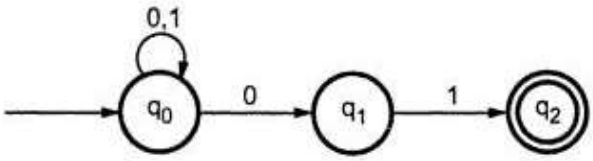
The above figure is a NFA because in state q_0 with input 'a' we can go to either state q_0 or state q_1.

Example 1.39 : *Design NFA accepting all strings ending with 01. Over* $\Sigma = \{0, 1\}$

Solution :

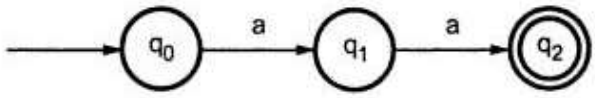


Hence, NFA would be

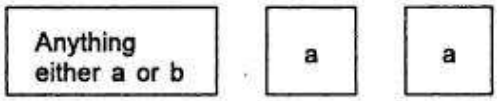


Example 1.40 : *Design NFA to accept strings with a's and b's such that the string end with 'aa'.*

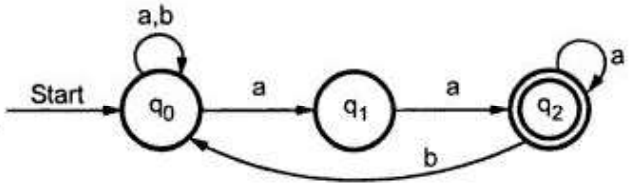
Solution : The simple FA which accepts a string with 'aa' is -



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

$$\delta(q_0, aaa) \vdash \delta(q_0, aa)$$
$$\vdash \delta(q_1, a)$$
$$\vdash \delta(q_2, \varepsilon)$$

i.e. we reach to accept state.

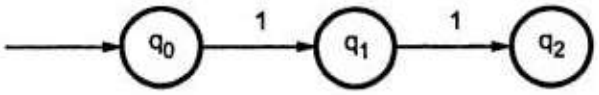
$$\delta(q_0, aaa) \vdash \delta(q_0, aa)$$
$$\vdash \delta(q_0, a)$$
$$\vdash \delta(q_1, \varepsilon)$$

i.e. we are not in final state.

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

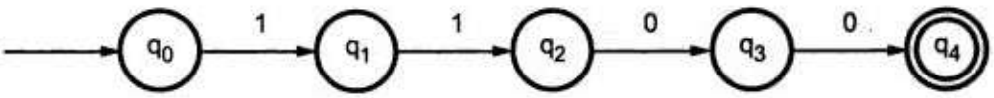
Example 1.41 : *Construct a NFA in which double '1' is followed by double '0'. Over* $\Sigma = \{0, 1\}$.

Solution : The FA with double 1 is as drawn below.



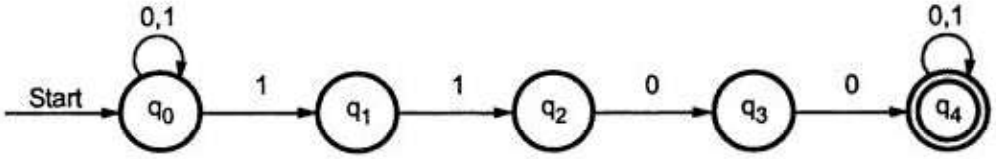
It should be immediately followed by double 0.

Then,



Now before double 1 there can be any string of 0 and 1. Similarly after double 0 there can be any string of 0 and 1.

Hence, the NFA becomes



The transition table for above transition diagram can be as given below -

States \ Input	1	0
q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	-
q_2	-	q_3
q_3	-	q_4
q_4	q_4	q_4

Consider string 11100,

$$\begin{aligned}\delta(q_0, 111000) &\vdash \delta(q_0, 1100)\\ &\vdash \delta(q_0, 100)\\ &\vdash \delta(q_1, 00)\end{aligned}$$

Got stuck! As there is no path from q_1 for input symbol 0. We can process string 11100 in another way.

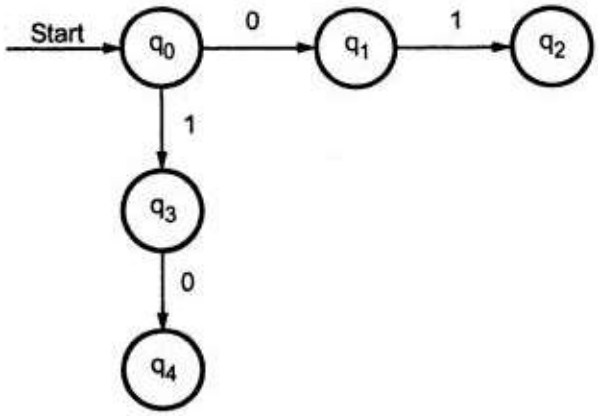
$$\begin{aligned}\delta(q_0, 11100) &\vdash \delta(q_0, 1100)\\ &\vdash \delta(q_1, 100)\\ &\vdash \delta(q_2, 00)\\ &\vdash \delta(q_3, 0)\\ &\vdash \delta(q_4, \varepsilon)\end{aligned}$$

As q_4 is accept state we get, the complete string scanned and reaching to final state.

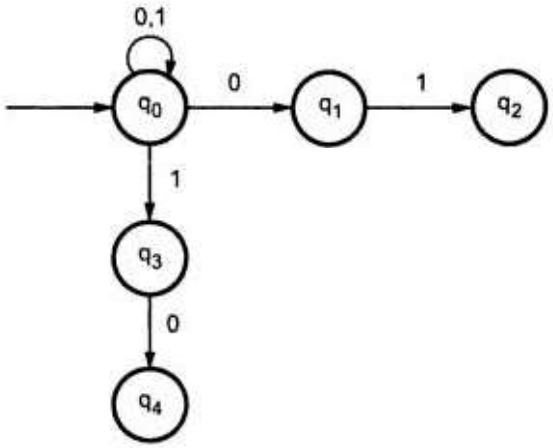
Example 1.42 : *Design NFA which accepts the string containing either '01' or '10'. Over* $\Sigma = \{0, 1\}$.

Solution : The NFA can be drawn in stages as follows.

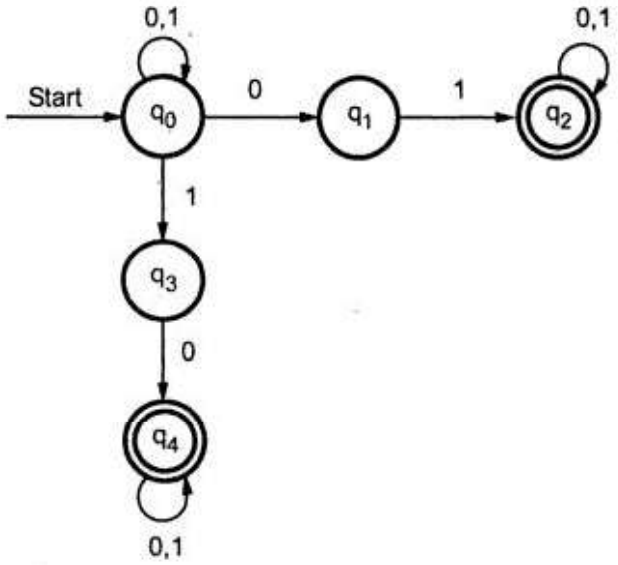
Consider a FA with '01' or '10'.



Now before '01' or '10' there can be any number or 0 and 1. Hence in next stage the FA can be



Now after '01' or '10' there can be any combination of 0 and 1. Hence the NFA in its final stage will be -



We can design a transition table as

States \ Input	0	1
q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
→ q_1	–	q_2
(q_2)	q_2	q_2
q_3	q_4	-
(q_4)	q_4	q_4

δ { } Input Σ

Thus this chapter focuses on the fundamental aspects of formal languages and automata theory. We have learnt the basic concepts of language such as alphabet, strings. We have also been introduced with the concept of finite automata, DFA and NFA. Now in the next chapter we will discuss Finite Automata in more detail.

Review Questions

1. *Explain the concept of alphabet, string and languages with suitable examples.*
2. *What is trees ?*
3. *What are the different types of proofs ?*
4. *Describe the language represented by following DFA*

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

5. *Convert the following NFA to DFA and also describe the language accepted by it*

	0	1
$\rightarrow p$	{p, q}	{p}
q	{r, s}	{t}
r	{p, r}	{t}
(s)	ϕ	ϕ
(t)	ϕ	ϕ

6. *Construct a deterministic finite automata equivalent to*

$M = (\{q_0, q_1, q_2, q_3,\}, \{0, 1\}, \delta, q_0, \{q_3\})$

Σ / State	0	1
q_0	q_0, q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_3
(q_3)	ϕ	q_2

7. *Compare NFA and DFA.*

8. *For the given transition table, obtain the transition diagram. Find out whether following strings are accepted by this machine or not.*

States \ Σ	0	1
(q_0)	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

a) 1 0 1 1 0 1.

b) 0 0 0 0 0 0.

9. *Construct a non deterministic finite automata accepting {0 1, 1 0}, and use it to find a deterministic finite automata.*

10. *Define -*

i) Empty set.

ii) Subset.

iii) Union of two sets.

iv) Intersection of two sets.

❑❑❑

2 Finite Automata

2.1 Introduction

An automaton is an abstract model of digital computer. An automaton has a mechanism to read input from input tape. Any langauge is recognized by some automaton. Hence these automaton are basically **language acceptors** or **language recognizers**. The study of theory of automata starts with finite automata. We have already discussed finite automata in earlier chapter. We also know that there are two types of finite automata : 1) Deterministic Finite Automata i.e. DFA 2) Nondeterministic Finite Automata i.e. NFA.

In this chapter we will learn DFA and NFA in more detail. Firstly we will discuss ε move in NFA, then conversion of NFA with ε to NFA without ε. We will also discuss the conversion of NFA to DFA. Minimization of FSM. In latter part of the chapter we discuss the working of Mealy and Moore machines along with illustrative examples.

2.2 Significance of Nondeterministic Finite Automaton

Computers are basically deterministic machines. That means on giving certain input we get certain output either desirable or undesirable. In similar manner deterministic finite automata are the machines in which we get some predictable state on certain input. But constructing such deterministic machines is very difficult. Hence there should some way by which some easy to construct machines can be build. In such a case we construct a nondeterministic machine which is called NFA or NDFA. This NFA then can be easily constructed to DFA. This efficient mechanism is majorly used in compilers. Thus NFA serves as a bridge between input given and DFA constructed.

2.3 NFA with ε - Transitions

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

Consider the NFA with ε as :

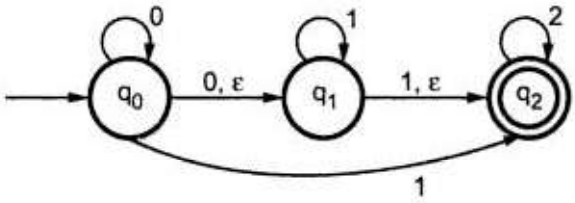


Fig. 2.1 NFA with ε - transitions

In this NFA with ε, q_0 is a start state with input 0 one can be either in state q_0 or in state q_1. If we get at the start a symbol 1 then with ε - move we can change state from q_0 to q_1 and then with input we can be in state q_1. On the other hand, from start state q_0, with input 1 we can reach to state q_2. Thus it is not definite that on input 1 whether we will be in state q_1 or q_2. Hence it is called nondeterministic finite automata (NFA) and since there are some ε moves by which we can simply change the states from one state to other. Hence it is called NFA with ε.

2.3.1 Significance of NFA with ε

As construction of DFA is very difficult for certain input set, we construct NFA. This NFA then can be converted to DFA. ε is an empty string. The ε - transitions are used simply to change one state to other. Sometimes to reach to final state we do not get proper state from start state. In such a case we simply want to reach to certain state which leads to final state. Such a transition to that specific state should be without any input symbol. Hence we require some ε - moves by which a proper state can be obtained for reaching to final state. Thus ε - moves play an important role in NFA.

2.3.2 Acceptance of Language

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

Let, $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA with ε.

where

Q is a finite set of states.

Σ is input set.

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

q_0 is a start state.

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

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

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

Example 2.1 : *Construct NFA with ε which accepts a language consisting the strings of any number of a's followed by any number of b's. Followed by any number of c's.*

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

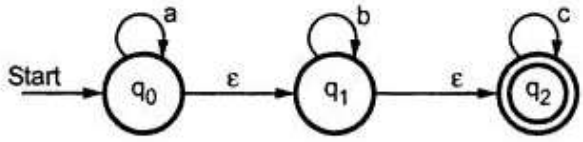


Fig. 2.2

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

State \ Input	a	b	c	ε
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

We can parse the string aabbcc as follows -

$\delta(q_0, aabbcc) \vdash \delta(q_0, abbcc)$

$\vdash \delta(q_0, bbcc)$

$\vdash \delta(q_0, \varepsilon bbcc)$

$\vdash \delta(q_1, bbcc)$

$\vdash \delta(q_1, bcc)$

$\vdash \delta(q_1, cc)$

$\vdash \delta(q_1, \varepsilon cc)$

$\vdash \delta(q_2, cc)$

$\vdash \delta(q_2, c)$

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

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

Definition of ε - closure

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

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

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

Example 2.2 : *Find ε - closure for the following NFA with ε.*

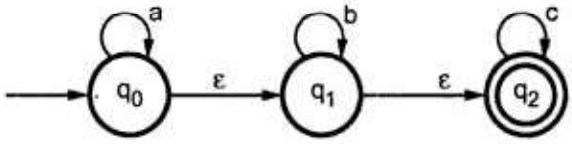


Fig. 2.3

Solution :

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

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

ε - closure $(q_2) = \{q_2\}$

2.4 Conversions and Equivalence

The NFA with ε can be converted to NFA without ε. And this NFA without ε can be converted to DFA. In this section we will discuss these conversions and will find them equivalent to each other.

Fig. 2.4 Moves to DFA

2.4.1 Conversion From NFA with ε to NFA without ε

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

1. Find out all the ε transitions from each state from Q. That will be called as ε-closure $\{q_i\}$ where $q_i \in Q$.
2. Then δ′ transitions can be obtained. The δ′ transitions means an ε-closure on δ moves.

3. Step-2 is repeated for each input symbol and for each state of given NFA.
4. Using the resultant states the transition table for equivalent NFA without ε can be built.

Theorem : If L is accepted by NFA with ε transitions, then there exist L which is accepted by NFA without ε transitions.

Proof :

Let, $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ε transitions.

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

$$F' = \begin{cases} F \cup \{q_0\} & \text{if } \varepsilon\text{ - closure contains state off} \\ F & \text{otherwise} \end{cases}$$

M' is a NFA without ε moves. The δ' function can be denoted by δ" with some input. For example, $\delta'(q, a) = \delta''(q, a)$ for some q in Q and a from Σ. We will apply the method of induction with input x. The x will not be ε because

$\delta'(q_0, \varepsilon) = \{q_0\}$.

$\delta''(q_0, \varepsilon) = \varepsilon$-closure (q_0). Therefore we will assume length of string to be 1.

Basis : $|x| = 1$. Then x is a symbol a.

$$\delta'(q_0, a) = \delta''(q_0, a)$$

Induction : $|x| > 1$ Let $x = wa$

$$\delta'(q_0, wa) = \delta'(\delta'(q_0, w), a)$$

By inductive hypothesis,

$$\delta'(q_0, w) = \delta''(q_0, w) = p$$

Now we will show that $\delta'(p, a) = \delta(q_0, wa)$

But $$\delta'(p, a) = \bigcup_{q \text{ in } p} \delta'(q, a) = \bigcup_{q \text{ in } p} \delta''(q, a)$$

As $$p = \delta''(q_0, w)$$

We have, $$\bigcup_{q \text{ in } p} \delta''(q, a) = \delta''(q_0, wa)$$

Thus by definition δ"

$$\delta'(q_0, wa) = \delta''(q_0, wa)$$

Rule for conversion

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

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

Before solving some examples based on conversion of NFA with ε to NFA without ε above rule should be remembered.

Example 2.3 : *Convert the given NFA with ε to NFA without ε.*

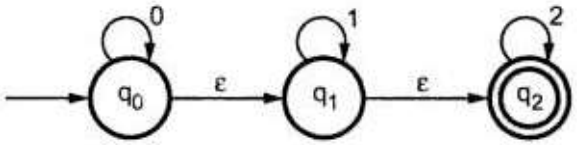


Fig. 2.5

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

Hence

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

ε - closure $(q_1) = \{q_1, q_2\}$

ε - closure $(q_2) = \{q_2\}$

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

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

$$= \varepsilon \text{ - closure } \left(\delta(\varepsilon \text{ - closure}(q_0), 0)\right)$$

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

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

$$= \varepsilon \text{ - closure } (q_0 \cup \phi \cup \phi)$$

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

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

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

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

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

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

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

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

$$= \varepsilon\text{ - closure }\left(\delta\left(\varepsilon\text{ - closure}(q_1), 0\right)\right)$$

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

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

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

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

$$= \phi$$

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

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

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

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

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

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

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

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

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

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

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

$$= \phi$$

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

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

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

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

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

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

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

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

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

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

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

$$= \{q_2\}$$

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

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

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

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

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

$$= \{q_2\}$$

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

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

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

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

$$= \{q_2\}$$

Now we will summarize all the computed δ' transitions –

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

From this we can write the transition table as -

State \ Input	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

The NFA will be -

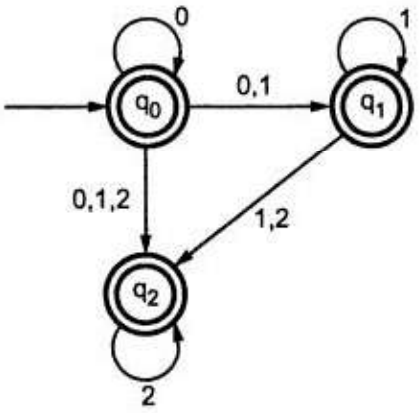


Fig. 2.6

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

Example 2.4 : *Convert the following NFA with ε to NFA without ε.*

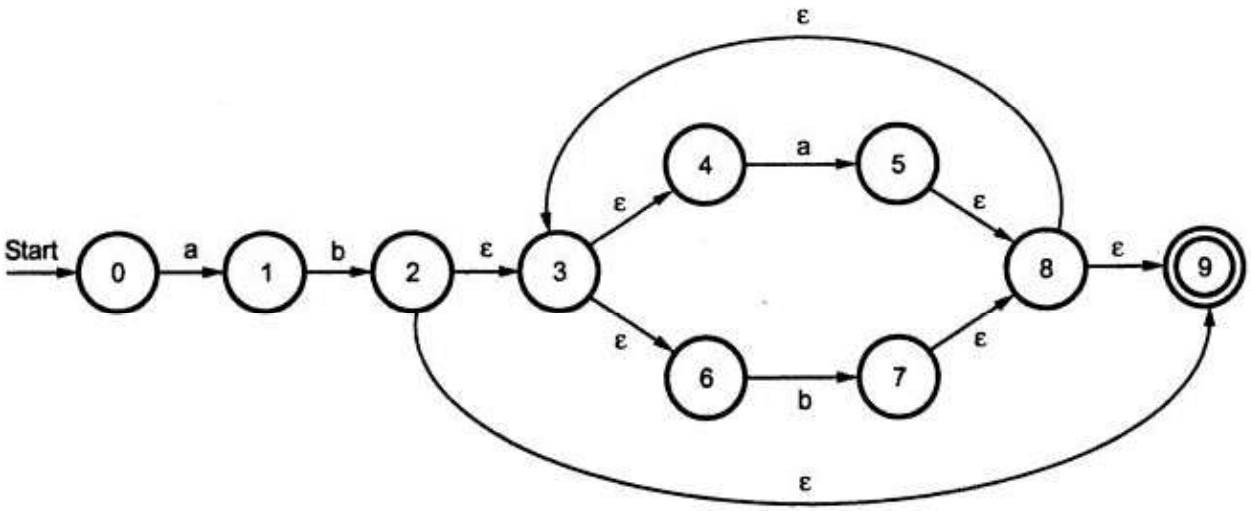


Fig. 2.7

Solution : We will first obtain ε - closure of every state. The ε - closure is basically an ε - transition from one state to other. Hence

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

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

$$\varepsilon\text{ - closure }(2) = \{2, 3, 4, 6, 9\}$$

$$\varepsilon\text{ - closure }(3) = \{3, 4, 6\}$$

$$\varepsilon\text{ - closure }(4) = \{4\}$$

$$\varepsilon\text{ - closure }(5) = \{5, 8, 3, 4, 6, 9\}$$

$$= \{3, 4, 5, 6, 8, 9\} \text{ sorted it!}$$

$$\varepsilon\text{ - closure }(6) = \{6\}$$

$$\varepsilon\text{ - closure }(7) = \{7, 8, 3, 4, 6, 9\}$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\varepsilon\text{ - closure }(8) = \{8, 3, 4, 6, 9\}$$

$$= \{3, 4, 6, 8, 9\}$$

$$\varepsilon\text{ - closure }(9) = \{9\}$$

Now we will obtain δ′ transitions for each state and for each input symbol.

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

$$= \varepsilon\text{ - closure}\left(\delta\left(\varepsilon\text{ - closure}(0), a\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(0, a)\right)$$

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

$$= \{1\}$$

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

$$= \varepsilon\text{ - closure}\left(\delta\left(\varepsilon\text{ - closure}(0), b\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(0, b)\right)$$

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

$$= \phi$$

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

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

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

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

$$= \phi$$

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

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

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

$$= \varepsilon\text{-closure}(2)$$

$$= \{2, 3, 4, 6, 9\}$$

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

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(2), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(2, 3, 4, 6, 9), a\right)$$

$$= \varepsilon\text{-closure}\left(\delta(2, a) \cup \delta(3, a) \cup \delta(4, a) \cup \delta(6, a) \cup \delta(9, a)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup 5 \cup \phi \cup \phi)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

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

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(2), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(2, 3, 4, 6, 9), b\right)$$

$$= \varepsilon\text{-closure}\left(\delta(2, b) \cup \delta(3, b) \cup \delta(4, b) \cup \delta(6, b) \cup \delta(9, b)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 7 \cup \phi)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(3, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(3,\varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(3), a\right)\right)$$

$$= \varepsilon\text{-closure}(\delta(3,4,6), a)$$

$$= \varepsilon\text{-closure}(\delta(3,a) \cup \delta(4,a) \cup \delta(6,a))$$

$$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(3,b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(3,\varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(3), b)\right)$$

$$= \varepsilon\text{-closure}(\delta(3,4,6), b)$$

$$= \varepsilon\text{-closure}\left(\delta(3,b) \cup \delta(4,b) \cup \delta(6,b)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(4,a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(4,\varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(4), a)\right)$$

$$= \varepsilon\text{-closure}(\delta(4,a))$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(4,b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(4,\varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(4), b)\right)$$

$$= \varepsilon\text{-closure}(\delta(4,b))$$

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

$$= \phi$$

$$\delta'(5,a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(5,\varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(5), a)\right)$$

$$= \varepsilon\text{-closure}(\delta(3,4,5,6,8,9), a)$$

$$= \varepsilon\text{-closure}(\delta(3,a) \cup \delta(4,a) \cup \delta(5,a) \cup \delta(6,a) \cup \delta(8,a) \cup \delta(9,a))$$

$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi \cup \phi \cup \phi \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(5)$

$= \{3, 4, 5, 6, 8, 9\}$

$\delta'(5, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(5,\varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(5), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(3, 4, 5, 6, 8, 9), b\right)$

$= \varepsilon\text{-closure}\left(\delta(3, b) \cup \delta(4, b) \cup \delta(5, b) \cup \delta(6, b) \cup \delta(8, b) \cup \delta(9, b)\right)$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 7 \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(7)$

$= \{3, 4, 6, 7, 8, 9\}$

$\delta'(6, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(6,\varepsilon), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(6), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(6, a)\right)$

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

$= \phi$

$\delta'(6, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(6,\varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(6), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(6, b)\right)$

$= \varepsilon\text{-closure}(7)$

$= \{3, 4, 6, 7, 8, 9\}$

$\delta'(7, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(7,\varepsilon), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(7), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(3, 4, 6, 7, 8, 9), a\right)$

$= \varepsilon\text{-closure}\left(\delta(3, a) \cup \delta(4, a) \cup \delta(6, a) \cup \delta(7, a) \cup \delta(8, a) \cup \delta(9, a)\right)$

$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi \cup \phi \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(5)$

$= \{3, 4, 5, 6, 8, 9\}$

$\delta'(7, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(7,\varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(7), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(3, 4, 6, 7, 8, 9), b\right)$

$= \varepsilon\text{-closure}\left(\delta(3, b) \cup \delta(4, b) \cup \delta(6, b) \cup \delta(7, b)\right.$

$\left. \cup \delta(8, b) \cup \delta(9, b)\right)$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7 \cup \phi \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(7)$

$= \{3, 4, 6, 7, 8, 9\}$

$\delta'(8, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(8,\varepsilon), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(8), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(3, 4, 6, 8, 9), a\right)$

$= \varepsilon\text{-closure}(\delta(3, a) \cup \delta(4, a) \cup \delta(6, a) \cup \delta(8, a) \cup \delta(9, a))$

$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(5)$

$= \{3, 4, 5, 6, 8, 9\}$

$\delta'(8, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(8,\varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(8), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(3, 4, 6, 8, 9), b\right)$

$= \varepsilon\text{-closure}\left(\delta(3, b) \cup \delta(4, b) \cup \delta(6, b) \cup \delta(8, b) \cup \delta(9, b)\right)$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7 \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(7)$

$= \{3, 4, 6, 7, 8, 9\}$

$\delta'(9, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(9,\varepsilon), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(9), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(9, a)\right)$

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

$$= \phi$$

$$\delta'(9, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(9,\varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(9), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(9, b)\right)$$

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

$$= \phi$$

Now we will build the transition table using above calculated δ' transitions.

Input / State	a	b
{0}	{1}	ϕ
{1}	ϕ	{2, 3, 4, 6, 9}
({2})	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
{3}	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
{4}	{3, 4, 5, 6, 8, 9}	ϕ
({5})	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
{6}	ϕ	{3, 4, 6, 7, 8, 9}
({7})	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
({8})	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
({9})	ϕ	ϕ

State {2} is a final state since ε - closure (2) contains 9 which is actually a final state in given NFA. Similarly state 5, 7, 8 and 9 are final states.

Example 2.5 : *Convert the following NFA with ε to NFA without ε.*

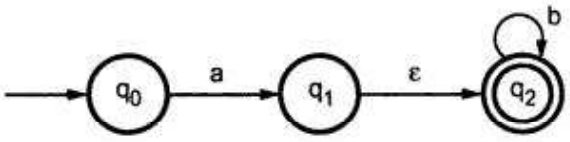


Fig. 2.8

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

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

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

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

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

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

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

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

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

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

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

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

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

$$= \phi$$

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

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

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

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

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

$$= \phi$$

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

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

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

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

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

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

$$= \{q_2\}$$

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

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

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

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

$$= \phi$$

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

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

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

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

$$= \{q_2\}$$

The transition table can be -

	State \ Input	a	b
→	q_0	$\{q_1, q_2\}$	ϕ
	(q_1)	ϕ	$\{q_2\}$
	(q_2)	ϕ	$\{q_2\}$

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

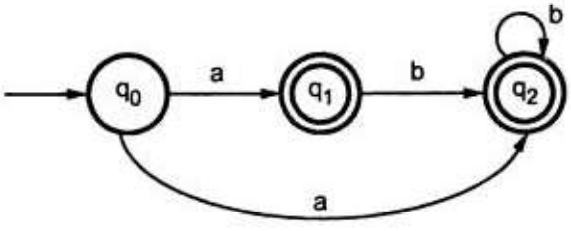


Fig. 2.9

2.5 NFA to DFA Conversion

As we have discussed, the finite automata can either be DFA or NFA. You might be thinking now, who is better NFA or DFA. Which has more power ? Here is a theorem which tell you that any NFA can be converted to its equivalent DFA. That is any language L acceptable by NFA can be acceptable by its equivalent DFA. The basic idea in this theorem is that, DFA keeps track of all the states, that NFA could be in reading the same input as the DFA has read.

Let us see the theorem.

Theorem : Let L be a set accepted by non-deterministic finite automation. Then there exists a deterministic finite automation that accepts L.

Proof : Let

$M = (Q, \Sigma, \delta, q_0, F)$ be an NFA for language L. Then define DFA M' such that

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

The states of M' are all the subset of M'. The $Q' = 2^Q$.

F' be the set of all the final states in M.

The elements in Q' will be denoted by $[q_1, q_2, q_3, \ldots q_i]$ and the elements in Q are denoted by $\{q_0, q_1, q_2, \ldots\}$ The $[q_1, q_2, \ldots q_i]$ will be assumed as one state in Q' if in the NFA q_0 is a initial state it is denoted in DFA as $q_0' = [q_0]$. We define,

$\delta'([q_1, q_2, q_3, \ldots q_i], a) = [p_1, p_2, p_3, \ldots p_j]$

if and only if,

$\delta(\{q_1, q_2, q_3, \ldots q_i\}, a) = \{p_1, p_2, p_3, \ldots p_j\}$

This means that whenever in NFA, at the current states $\{q_1, q_2, q_3, \ldots q_i\}$ if we get input a and it goes to the next states $\{p_1, p_2, \ldots p_j\}$ then while constructing DFA for it the current state is assumed to be $[q_1, q_2, q_3, \ldots q_i]$ At this state, the input is a and the next is assumed to be $[p_1, p_2, \ldots p_j]$. On applying δ function on each of the states $q_1, q_2, q_3, \ldots q_i$ the new states may be any of the states from $[p_1, p_2, \ldots p_j]$. The theorem can be proved with the induction method by assuming length of input string x

$$\delta'(q_0', x) = [q_1, q_2, \ldots q_i]$$

if and only if,

$$\delta(q_0, x) = \{q_1, q_2, q_3, \ldots q_i\}$$

Basis : If length of input string is 0

i.e. $|x| = 0$, that means x is ε then $q_0' = [q_0]$

Induction : If we assume that the hypothesis is true for the input string of length m or less than m. Then if x a is a string of length m+1. Then the function δ' could be written as

$$\delta'(q_0', xa) = \delta'(\delta'(q_0, x), a)$$

By the induction hypothesis,

$$\delta'(q_0', x) = [p_1, p_2, \cdots p_j]$$

if and only if,

$$\delta(q_0, x) = \{p_1, p_2, p_3, \cdots p_j\}$$

By definition of δ'

$$\delta'([p_1, p_2, \cdots p_j], a) = [r_1, r_2, \ldots r_k]$$

if and only if,

$$\delta(\{p_1, p_2, \cdots p_j\} a) = \{r_1, r_2, \ldots r_k\}$$

Thus

$$\delta'(q_0', xa) = [r_1, r_2, \ldots r_k]$$

if and only if

$$\delta(q_0, x\ a) = \{r_1, r_2, \ldots r_k\}$$

is shown by inductive hypothesis.

Thus $L(M) = L(M')$

With the help of this theorem, let us try to solve few examples.

Conversion from NFA to DFA

We will discuss the method of converting NFA to its equivalent DFA.

Let, $M = (Q, \Sigma, \delta, q_0, F)$ is a NFA which accepts the language L(M). There should be equivalent DFA denoted by $M' = (Q', \Sigma', \delta', q_0', F')$ such that L(M) = L(M').

The conversion method will follow following steps -

1) The start state of NFA M will be the start for DFA M'. Hence add q_0 of NFA (start state) to Q'. Then find the transitions from this start state.
2) For each state $[q_1, q_2, q_3, \ldots\ldots q_i]$ in Q' the transitions for each input symbol Σ can be obtained as,
   i) $\delta'([q_1, q_2, \ldots\ldots q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \ldots\ldots \delta(q_i, a)$
   $= [q_1, q_2, \ldots\ldots q_k]$ may be some state.
   ii) Add the state $[q_1, q_2, \ldots\ldots q_k]$ to DFA if it is not already added in Q'.

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

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

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

Example 2.6 : *Let* $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\})$

be NFA where $\delta(q_0, 0) = \{q_0, q_1\}, \delta(q_0, 1) = \{q_1\}$,

$\delta(q_1, 0) = \phi, \delta(q_1, 1) = \{q_0, q_1\}$

Construct its equivalent DFA.

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

Now, the δ' function will be computed as follows -

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

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

Let us draw the transition table for δ function for a given NFA.

		Input / State	0	1		
$\delta(q_0, 0)$	$\Rightarrow$	$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$	$\Leftarrow$	$\delta(q_0, 1)$
$\delta(q_1, 0)$	$\Rightarrow$	(q_1)	ϕ	$\{q_0, q_1\}$	$\Leftarrow$	$\delta(q_1, 1)$

δ Function for NFA

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

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

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

Similarly,

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

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

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

The equivalent DFA is

State \ Input	0	1
$\rightarrow q_0$	$[q_0, q_1]$	$[q_1]$
(q_1)	ϕ	$[q_0, q_1]$
(q_0, q_1)	$[q_0, q_1]$	$[q_0, q_1]$

Transition table for equivalent DFA

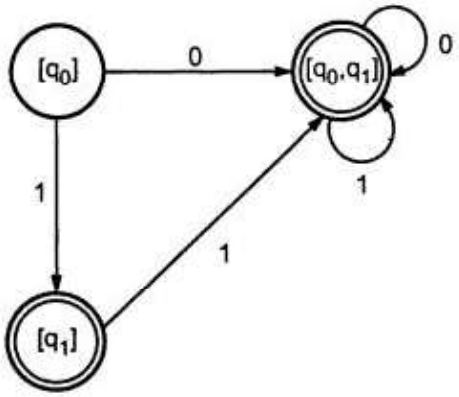


Fig. 2.10

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

$A = [q_0]$

$B = [q_1]$

$C = [q_0, q_1]$

With these new names the DFA will be as follows -

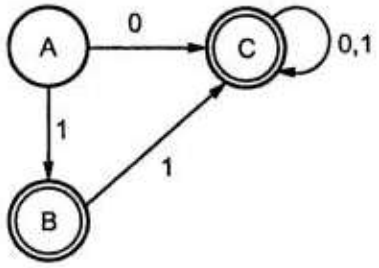


Fig. 2.11 An equivalent DFA

Example 2.7 : *Convert the given NFA to DFA.*

State \ Input	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
(q_3)	ϕ	q_2

Solution : As we know, first we will compute δ' function.

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

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

Similarly,

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

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

Thus we have got a new state $[q_0, q_1]$.

Let us check how it behaves on input 0 and 1.

So,

$$\begin{aligned}\delta'([q_0, q_1], 0) &= \delta([q_0], 0) \cup \delta([q_1], 0) \\ &= \{q_0, q_1\} \cup \{q_2\} \\ &= \{q_0, q_1, q_2\}\end{aligned}$$

Hence a new state is generated i.e. $[q_0, q_1, q_2]$

Similarly

$$\begin{aligned}\delta'([q_0, q_1], 1) &= \delta([q_0], 1) \cup \delta([q_1], 1) \\ &= \{q_0\} \cup \{q_1\} \\ &= \{q_0, q_1\}\end{aligned}$$

No new state is generated here.

Again δ' function will be computed for $[q_0, q_1, q_2]$, the new state being generated.

State	0	1
$[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$

As, you have observed in the above table for a new state $[q_0, q_1, q_2]$ the input 0 will give a new state $[q_0, q_1, q_2, q_3]$ and input 1 will give a new state $[q_0, q_1, q_3]$, because

$$\begin{aligned}\delta'([q_0, q_1, q_2], 0) &= \delta'([q_0], 0) \cup \delta'([q_1], 0) \cup \delta'([q_2], 0) \\ &= \{q_0, q_1\} \cup \{q_2\} \cup \{q_3\} \\ &= \{q_0, q_1, q_2, q_3\} \\ &= [q_0, q_1, q_2, q_3]\end{aligned}$$

Same procedure for input 1. Thus the final DFA is as given below.

State	0	1
$\rightarrow [q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_1]$	$[q_2]$	$[q_1]$
$(\![q_2]\!)$	$[q_3]$	$[q_3]$
$[q_3]$	ϕ	$[q_2]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$

$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

DFA for example 2.7

Example 2.8 : *Construct DFA equivalent to the given NFA.*

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

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

The equivalent DFA will be constructed.

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

Continuing with the generated new states.

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

[p, q]	[p, q, r]	[p, r]
[p, q, r]	[p, q, r, s]	[p, r]
[p, r]	[p, q, s]	[p]
[p, q, r, s]	[p, q, r, s]	[p, r, s]
[p, q, s]	[p, q, r, s]	[p, r, s]
[p, r, s]	[p, q, s]	[p, s]
[p, s]	[p, q, s]	[p, s]

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

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

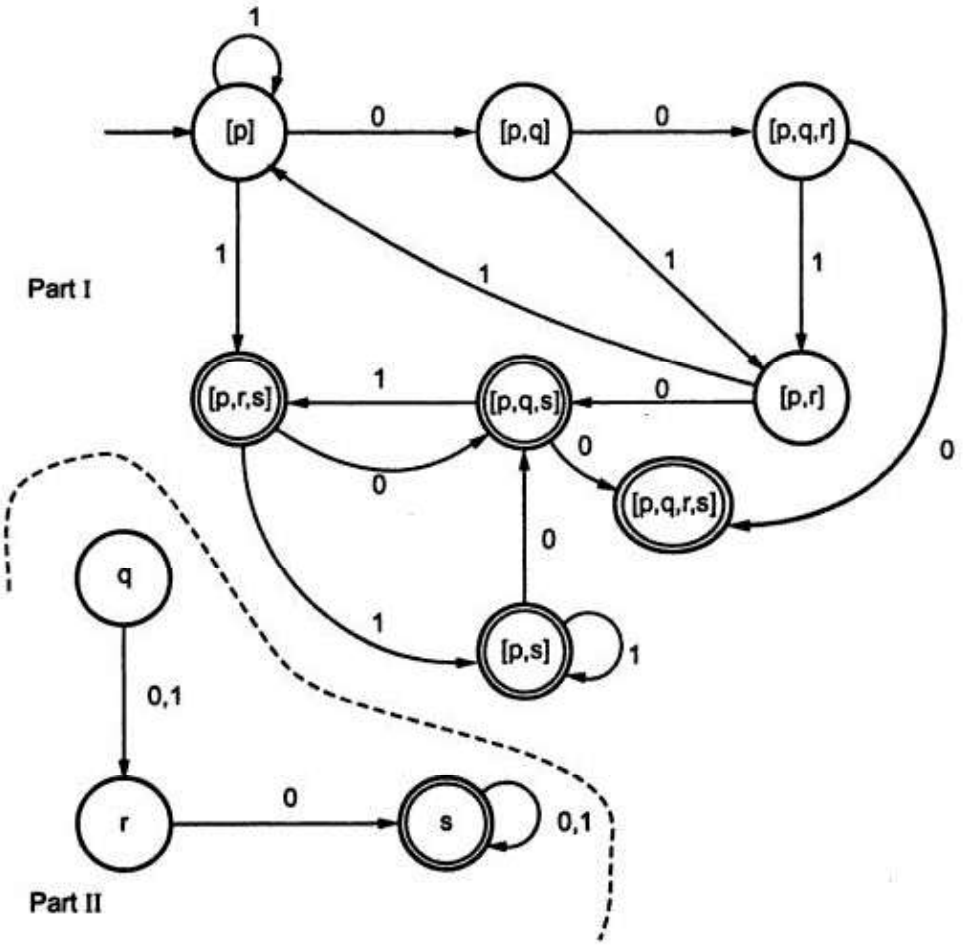


Fig. 2.12

Example 2.9 : *Convert the given NFA to its equivalent DFA.*

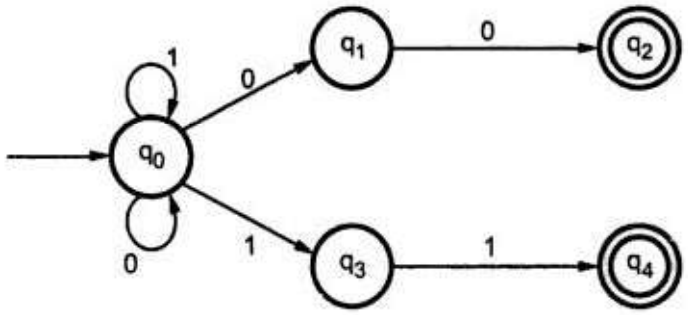


Fig. 2.13

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

State \ Input	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_1	$\{q_2\}$	ϕ
((q_2))	ϕ	ϕ
q_3	ϕ	$\{q_4\}$
((q_4))	ϕ	ϕ

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

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

$\delta'([q_0], 1) = [q_0, q_3]$

As we have got a new state $[q_0, q_1]$ we will compute δ' transitions for input 0 and 1.

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

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

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

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

$= \{q_0, q_3\} \cup \{\phi\}$

$= \{q_0, q_3\}$

$\delta'([q_0, q_1], 1) = [q_0, q_3]$

The new state $[q_0, q_3]$ is obtained on $\delta(q_0, 1)$ we will process it.

$$\delta'([q_0, q_3], 0) = \delta(q_0, 0) \cup \delta(q_3, 0)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$
$$\delta'([q_0, q_3], 0) = [q_0, q_1]$$
$$\delta'([q_0, q_3], 1) = \delta(q_0, 1) \cup \delta(q_3, 1)$$
$$= \{q_0, q_3\} \cup \{q_4\}$$
$$= \{q_0, q_3, q_4\}$$

Hence we get a new state.

$$\delta'([q_0, q_3], 1) = [q_0, q_3, q_4]$$
$$\delta'([q_1], 0) = [q_2]$$
$$\delta'([q_1], 1) = \phi$$
$$\delta'([q_2], 0) = \phi$$
$$\delta'([q_2], 1) = \phi$$
$$\delta'([q_3], 0) = \phi$$
$$\delta'([q_3], 1) = [q_4]$$
$$\delta'([q_4], 0) = \phi$$
$$\delta'([q_4], 1) = \phi$$

Now consider newly generated states.

$$\delta'([q_0, q_1, q_2], 0) = \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$$
$$= \{q_0, q_1\} \cup \{q_2\} \cup \phi$$
$$= \{q_0, q_1, q_2\}$$
$$\delta'([q_0, q_1, q_2], 0) = [q_0, q_1, q_2]$$
$$\delta'([q_0, q_1, q_2], 1) = \delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1)$$
$$= \{q_0, q_3\} \cup \phi \cup \phi$$
$$= \{q_0, q_3\}$$
$$\delta'([q_0, q_1, q_2], 1) = [q_0, q_3]$$
$$\delta'([q_0, q_3, q_4], 0) = \delta(q_0, 0) \cup \delta(q_3, 0) \cup \delta(q_4, 0)$$
$$= \{q_0, q_1\} \cup \phi \cup \phi$$

$\delta'([q_0, q_3, q_4], 0) = [q_0, q_1]$

$\delta'([q_0, q_3, q_4], 1) = \delta(q_0, 1) \cup \delta(q_3, 1) \cup \delta(q_4, 1)$

$= \{q_0, q_3\} \cup \{q_4\} \cup \phi$

$= \{q_0, q_3, q_4\}$

$\delta'([q_0, q_3, q_4], 1) = [q_0, q_3, q_4]$

The transition table can be -

	State \ Input	0	1
	$[q_0]$	$[q_0, q_1]$	$[q_0, q_3]$
	$[q_1]$	$[q_2]$	ϕ
*	$[q_2]$	ϕ	ϕ
	$[q_3]$	ϕ	$[q_4]$
*	$[q_4]$	ϕ	ϕ
	$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_3]$
	$[q_0, q_3]$	$[q_0, q_1]$	$[q_0, q_3, q_4]$
*	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_0, q_3]$
*	$[q_0, q_3, q_4]$	$[q_0, q_1]$	$[q_0, q_3, q_4]$

The states $[q_2]$, $[q_4]$, $[q_0, q_1, q_2]$ and $[q_0, q_3, q_4]$ are all final states.

The DFA can be drawn as below.

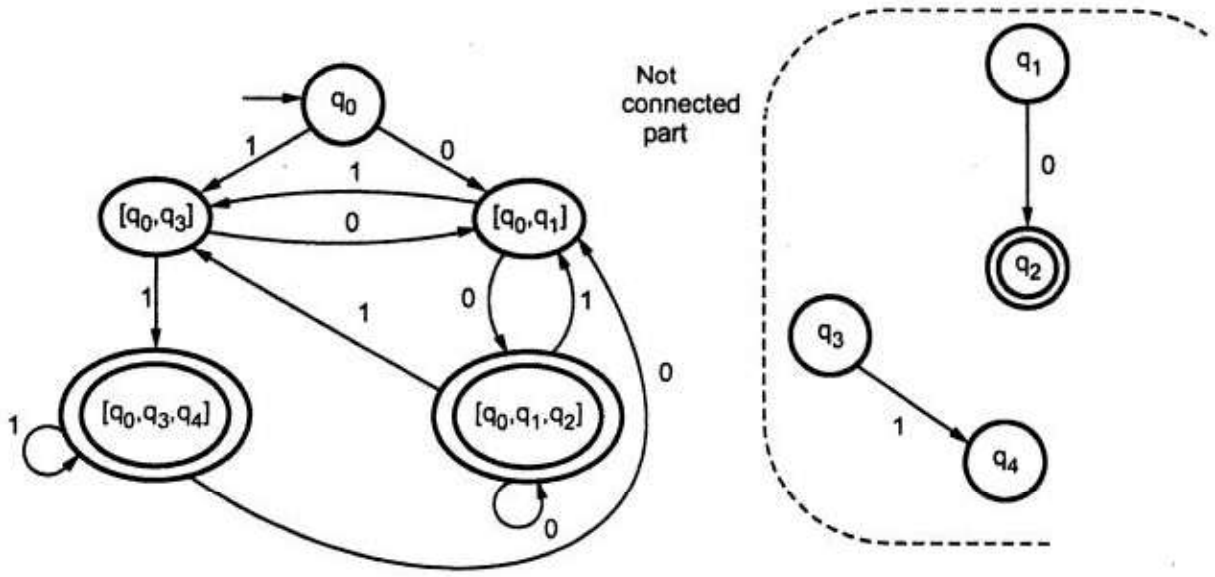


Fig. 2.14

In above transition diagram of DFA clearly q_1, q_2, q_3 and q_4 are not part of the converted DFA. Hence we can eliminate these states. Hence final DFA will

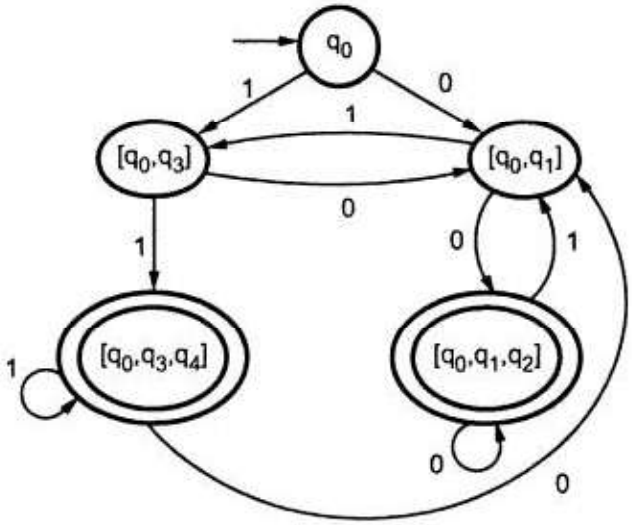


Fig. 2.15

This is a language that accepts two consecutive 0's or two consecutive 1's.

Example 2.10 : *Convert the following NFA into DFA.*

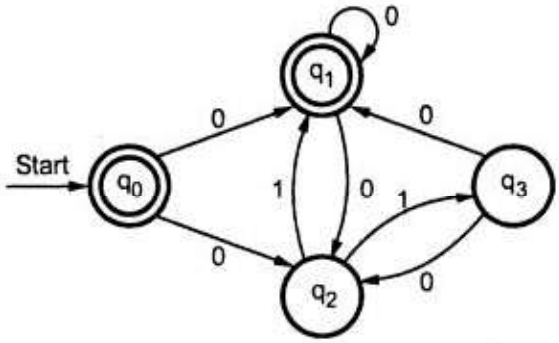


Fig. 2.16

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

State \ Input	0	1
q_0	$\{q_1, q_2\}$	ϕ
q_1	$\{q_1, q_2\}$	ϕ
q_2	ϕ	$\{q_1, q_3\}$
q_3	$\{q_1, q_2\}$	ϕ

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

We can write this as

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

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

Similarly

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

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

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

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

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

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

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

Similar,

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

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

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

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

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

Similarly,

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

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

Now, $\delta'([q_1, q_3], 0) = \delta([q_1], 0) \cup \delta([q_3], 0)$
$$= \{q_1, q_2\} \cup \{q_1, q_2\}$$

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

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

Similarly,

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

$$= \phi \cup \phi$$

$$= \phi$$

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

As now no new states are getting generated.

The transition table for DFA using above δ' transitions is

State \ Input	0	1
→ $[q_0]$	$[q_1, q_2]$	ϕ
$[q_1]$	$[q_1, q_2]$	ϕ
$[q_2]$	ϕ	$[q_1, q_3]$
$[q_3]$	$[q_1, q_2]$	ϕ
$[q_1, q_2]$	$[q_1, q_2]$	$[q_1, q_3]$
$[q_1, q_3]$	$[q_1, q_2]$	ϕ

The transition diagram can be -

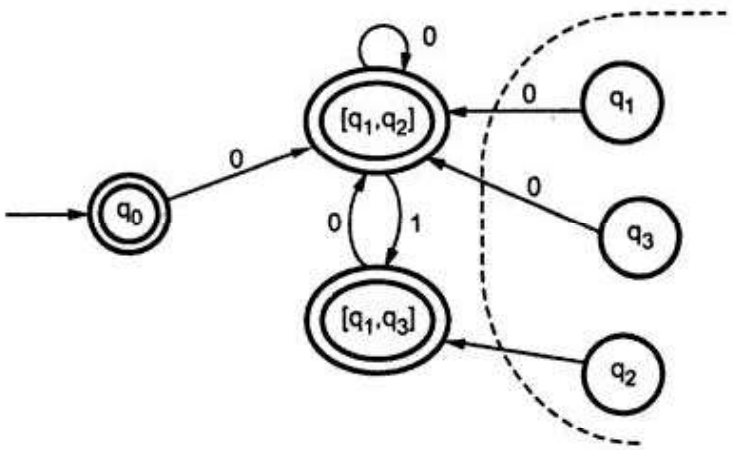


Fig. 2.17

As we can see the marked part of DFA consisting of q_1, q_2, q_3 is of no use since there is no path from start state to these states. Hence we will never reach to these states. So we can eliminate these states. Hence new DFA becomes.

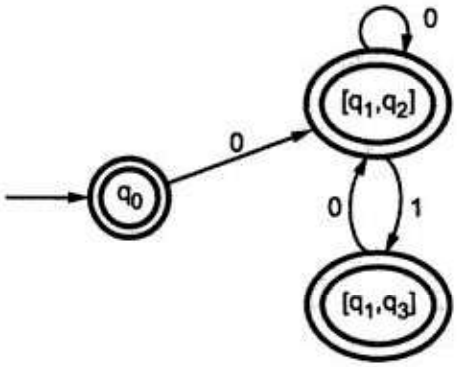


Fig. 2.18

As states q_0 and q_1 are final states original NFA states containing q_1 i.e. $[q_1, q_3]$ and $[q_1, q_2]$ are also final states.

➠ **Example 2.11 :** *Convert the following NFA into its equivalent DFA. Also recognize the language generated by this DFA.*

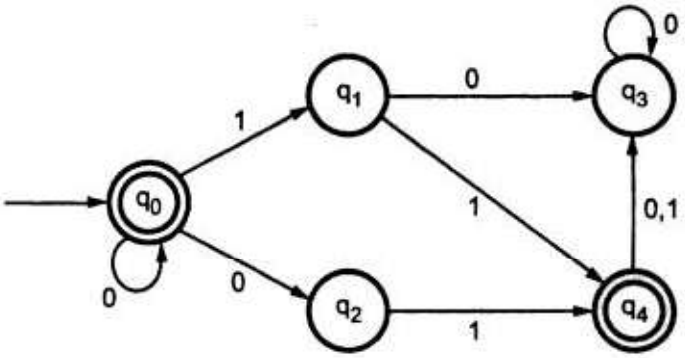


Fig. 2.19

Solution : We will first desing the transition table from given transition diagram.

State \ Input	0	1
→ (q0)	{q0, q2}	q1
q1	q3	q4
q2	ϕ	q4
q3	q3	ϕ
(q4)	q3	q3

$\delta'([q_0], 0) = [q_0, q_2]$ new state generated

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

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

$\delta'([q_1], 1) = [q_4]$

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

$\delta'([q_2], 1) = [q_4]$

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

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

$\delta'([q_4], 0) = [q_3]$

$\delta'([q_4], 1) = [q_3]$

Now we will obtain input transitions on $[q_0, q_2]$.

$$\delta'([q_0, q_2], 0) = \delta([q_0], 0) \cup \delta([q_2], 0)$$
$$= \{q_0, q_2\} \cup \phi$$
$$= \{q_0, q_2\}$$

i.e. $\delta'([q_0, q_2], 0) = [q_0, q_2]$

Similarly,

$$\delta'([q_0, q_2], 1) = \delta([q_0], 1) \cup \delta([q_2], 1)$$
$$= [q_1, q_4] \text{ new state generated.}$$

We will obtain δ' transitions for new state $[q_1, q_4]$.

$$\delta'([q_1, q_4], 0) = \delta([q_1], 0) \cup \delta([q_4], 0)$$
$$= \{q_3\} \cup \{q_3\}$$
$$= \{q_3\}$$

$\therefore \quad \delta'([q_1, q_4], 0) = [q_3]$

Now $\delta'([q_1, q_4], 1) = \delta([q_1], 1) \cup \delta([q_4], 1)$

$$= \{q_4\} \cup \{q_3\}$$
$$= \{q_4, q_3\}$$

$\therefore \quad \delta'([q_1, q_4], 1) = [q_4, q_3]$ new state generated.

We will obtain δ' transitions for new state $[q_4, q_3]$.

$$\delta'([q_4, q_3], 0) = \delta([q_4], 0) \cup \delta([q_3], 0)$$
$$= \{q_3\} \cup \{q_3\}$$
$$= \{q_3\}$$
$$\therefore \quad \delta'([q_4, q_3], 0) = [q_3]$$

Similarly,

$$\delta'([q_4, q_3], 1) = \delta([q_4], 1) \cup \delta([q_3], 1)$$
$$= \{q_3\} \cup \phi$$
$$= \{q_3\}$$
$$\delta'([q_4, q_3], 1) = [q_3]$$

Now since no new state is being generated.

I will now design the transition diagrams from the above δ' transitions.

State \ Input	0	1
→ $[q_0]$	$[q_0, q_2]$	$[q_1]$
$[q_1]$	$[q_3]$	$[q_4]$
$[q_2]$	ϕ	$[q_4]$
$[q_3]$	$[q_3]$	ϕ
$[q_4]$	$[q_3]$	$[q_3]$
$[q_0, q_2]$	$[q_0, q_2]$	$[q_1, q_4]$
$[q_1, q_4]$	$[q_3]$	$[q_4, q_3]$
$[q_4, q_3]$	$[q_3]$	$[q_3]$

The states q_0 and q_4 are basically final states. Hence newly generated states containing q_4 are final states. Therefore $[q_1, q_4]$ and $[q_4, q_3]$ are final states. Now the transition diagram for DFA are -

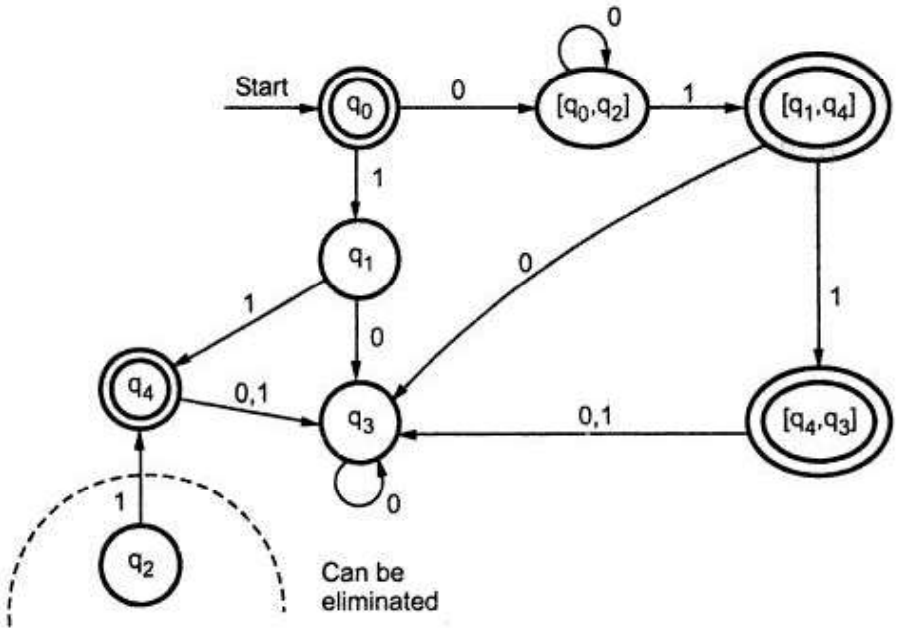


Fig. 2.20

The DFA will then be -

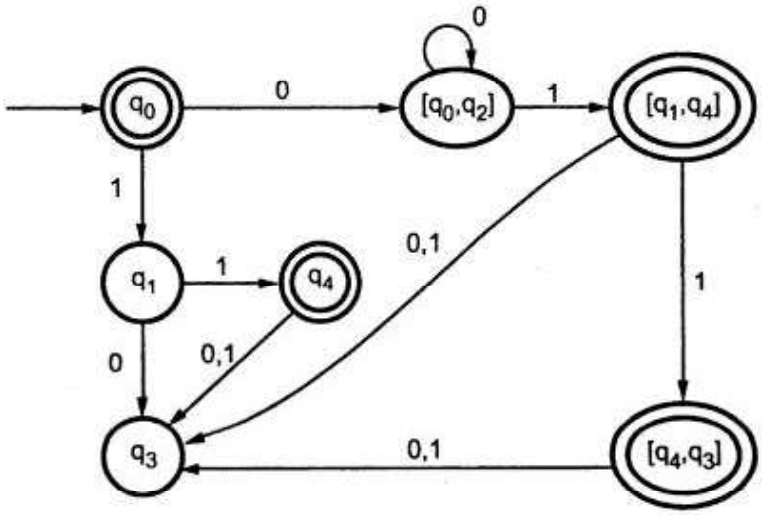


Fig. 2.21

This is a language accepting '11' or 0^n 11 where $n \geq 0$. That means it is a language of any number of zeros followed by 11 only.

2.5.1 Conversion of NFA with ε to DFA

We have already seen the method of converting NFA with ε to NFA without ε. Now we will learn how to convert NFA with ε to its equivalent DFA. In this method we first convert NFA with ε to NFA without ε. Then the NFA without ε can be converted to its equivalent DFA.

Method for converting NFA with ε to DFA

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

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

Then obtain,

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

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

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

$$\delta_D([P_1, P_2, \ldots P_n], a) = \varepsilon \text{ - closure}(\delta(P_1, a) \cup \delta(P_2, a) \cup \ldots \delta(P_n, a))$$

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

where a is input $\in \Sigma$.

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

Now let us see some examples of conversion based on this procedure.

Example 2.12 : *Convert the following NFA with ε to equivalent DFA.*

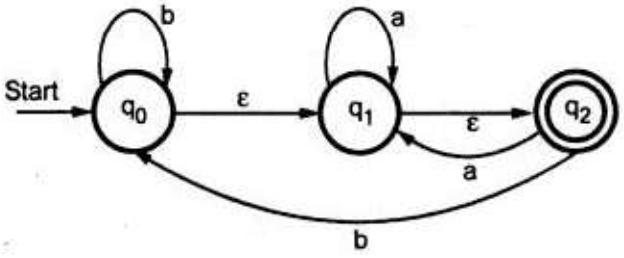


Fig. 2.22

Solution : To convert this NFA we will first find ε-closures.

ε-closures $\{q_0\} = \{q_0, q_1, q_2\}$

ε-closures $\{q_1\} = \{q_1, q_2\}$

ε-closures $\{q_2\} = \{q_2\}$

Let us start from ε-closure of start state

ε-closure $\{q_0\} = \{q_0, q_1, q_2\}$ we will call this state as A.

Now let us find transitions on A with every input symbol.

$$\begin{aligned}
\delta'(A, a) &= \varepsilon\text{-closure } (\delta(A, a)) \\
&= \varepsilon\text{-closure } (\delta(q_0, q_1, q_2), a) \\
&= \varepsilon\text{-closure } \{\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)\} \\
&= \varepsilon\text{-closure } \{q_1\} \\
&= \{q_1, q_2\}. \text{ Let us call it as state B.}
\end{aligned}$$

$$\begin{aligned}
\delta'(A, b) &= \varepsilon\text{-closure } (\delta(A, b)) \\
&= \varepsilon\text{-closure } (\delta(q_0, q_1, q_2), b) \\
&= \varepsilon\text{-closure } \{\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)\} \\
&= \varepsilon\text{-closure } \{q_0\} \\
&= \{q_0, q_1, q_2\} \text{ i.e. A.}
\end{aligned}$$

Hence we can state that

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

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

Now let us find transitions for state $B = \{q_1, q_2\}$.

$$\begin{aligned}
\delta'(B, a) &= \varepsilon\text{-closure } (\delta(q_1, q_2), a) \\
&= \varepsilon\text{-closure } \{q_1\} \\
&= \{q_1, q_2\} \text{ i.e. B}
\end{aligned}$$

$$\begin{aligned}
\delta'(B, b) &= \varepsilon\text{-closure } (\delta(q_1, q_2), b) \\
&= \varepsilon\text{-closure } \{\delta(q_1, b) \cup \delta(q_2, b)\} \\
&= \varepsilon\text{-closure } \{q_0\} \\
&= \{q_0, q_1, q_2\} \text{ i.e. A.}
\end{aligned}$$

Hence the generated DFA is

	a	b
→ (A)	B	A
(B)	B	A

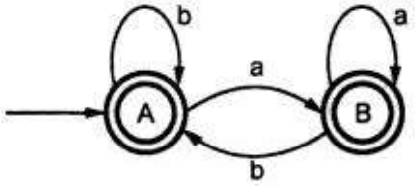


Fig. 2.23

Example 2.13 : *Convert the given NFA into its equivalent DFA -*

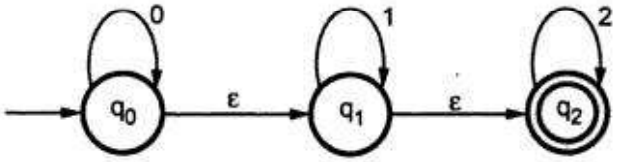


Fig. 2.24

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

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

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

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

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

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

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

$= \varepsilon\text{-closure}\{q_0\}$

$= \{q_0, q_1, q_2\}$ i.e. state A

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

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

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

$= \{q_1, q_2\}$ Call it as state B

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

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

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

$= \{q_2\}$ Call it as state C.

Thus we have obtained

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

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

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

i.e.

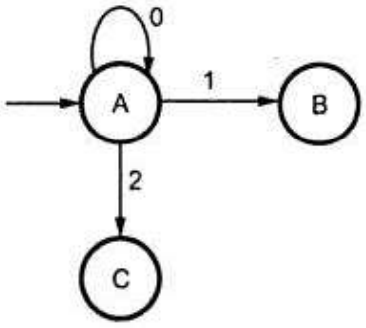


Fig. 2.25

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

Hence

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

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

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

Hence

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

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

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

The partial transition diagram will be

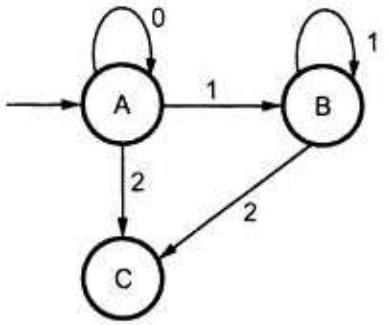


Fig. 2.26

Now we will obtain transitions for C :

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

Hence the DFA is

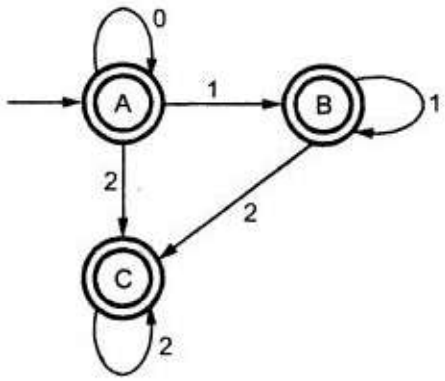


Fig. 2.27

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

Example 2.14 : *Convert the given NFA with ε to its equivalent DFA.*

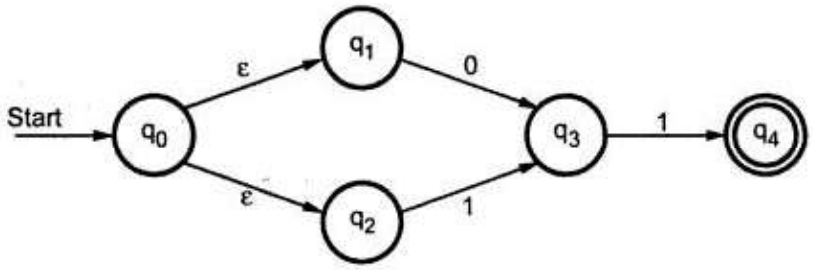


Fig. 2.28

Solution :

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

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

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

$$\varepsilon\text{-closure}\{q_3\} = \{q_3\}$$

$$\varepsilon\text{-closure}\{q_4\} = \{q_4\}$$

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

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

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

$= \varepsilon\text{-closure}\{q_3\}$

$= \{q_3\}$ call it as state B.

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

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

$= \varepsilon\text{-closure}\{q_3\}$

$= q_3 = B.$

The partial DFA will be

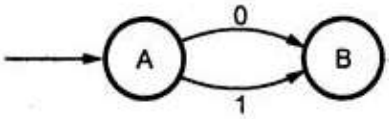


Fig. 2.29

Now,

$\delta'(B, 0) = \varepsilon\text{-closure}\{\delta(q_3, 0)\}$

$= \phi$

$\delta'(B, 1) = \varepsilon\text{-closure}\{\delta(q_3, 1)\}$

$$= \varepsilon\text{-closure}\{q_4\}$$

$$= \{q_4\} \text{ i.e. state C}$$

$$\delta'(C,0) = \varepsilon\text{-closure}\{\delta(q_4,0)\}$$

$$= \phi$$

$$\delta'(C,1) = \varepsilon\text{-closure}\{\delta(q_4,1)\}$$

$$= \phi$$

The DFA will be,

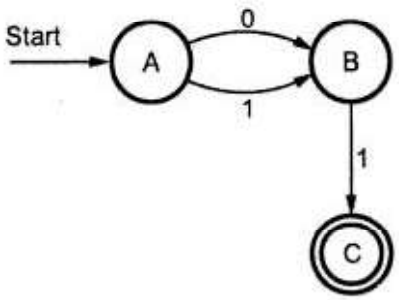


Fig. 2.30

Example 2.15 : *Consider following NFA with ε.*

→ State \ Input	a	b	c	ε
p	{p}	{q}	{r}	ϕ
q	{q}	{r}	ϕ	{p}
(r)	{r}	ϕ	{p}	{q}

Convert it to its equivalent DFA.

Solution : We will first compute ε - closure for start state p.

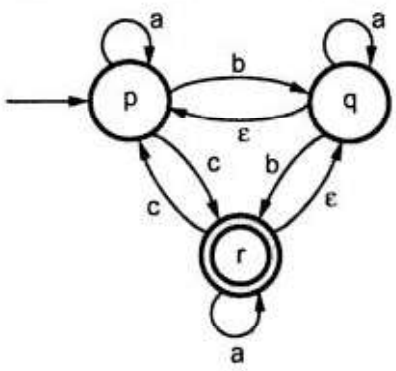


Fig. 2.31

ε - closure (p) = {p} call it as state A.

Now we will obtain δ transitions on state A.

State A

$$\delta(A, a) = \varepsilon\text{ - closure } (\delta(A, a))$$
$$= \varepsilon\text{ - closure}(\delta(p, a))$$
$$= \varepsilon\text{ - closure } (p)$$
$$= \{p\} \text{ i.e. state A only.}$$
$$\delta(A, b) = \varepsilon\text{ - closure } (\delta(A, b))$$
$$= \varepsilon\text{ - closure}(\delta(p, b))$$
$$= \varepsilon\text{ - closure } (q)$$
$$= \{q, p\} \text{ i.e. } \{p, q\} \text{ Let us call it state B.}$$
$$\delta(A, b) = B$$
$$\delta(A, c) = \varepsilon\text{ - closure } (\delta(A, c))$$
$$= \varepsilon\text{ - closure } (\delta(p, c))$$
$$= \varepsilon\text{ - closure } (r) = \{q, r\}. \text{ Call it as state C.}$$

State B = {p, q}

$$\delta(B, a) = \varepsilon\text{ - closure } (\delta(B, a))$$
$$= \varepsilon\text{ - closure}(\delta(p, q), a)$$
$$= \varepsilon\text{ - closure } (\delta(p, a) \cup \delta(q, a))$$
$$= \varepsilon\text{ - closure } (p \cup q)$$
$$= \varepsilon\text{ - closure } (p, q)$$
$$= \varepsilon\text{ - closure}(p) \cup \varepsilon\text{ - closure}(q)$$
$$= \{p\} \cup \{q\} = \{p, q\} \text{ i.e. state B only.}$$

$\therefore$ $\delta(B, a) = B.$

$$\delta(B, b) = \varepsilon\text{ - closure } (\delta(B, b))$$
$$= \varepsilon\text{ - closure}(\delta(p, q), b)$$
$$= \varepsilon\text{ - closure } (\delta(p, b) \cup \delta(q, b))$$
$$= \varepsilon\text{ - closure } (q \cup r)$$
$$= \varepsilon\text{ - closure } (q, r)$$
$$= \varepsilon\text{ - closure}(q) \cup \varepsilon\text{ - closure}(r)$$
$$= \{p, q\} \cup \{q, r\} = \{p, q, r\} \text{ i.e. state D.}$$

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

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

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

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

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

$= \varepsilon$ - closure (r)

$= \{q, r\}$

$\delta(B, c) = C$

State C = {q, r}

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

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

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

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

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

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

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

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

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

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

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

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

$= \varepsilon$ - closure (r)

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

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

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

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

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

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

$= \varepsilon$ - closure (p)

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

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

State D = {p, q, r}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\delta(D, c) = D$

The transition table from above calculations can be obtained as

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

As state A = {p} it is a start state and states C and D contain final state r, hence these are final states. The transition diagram for the DFA is

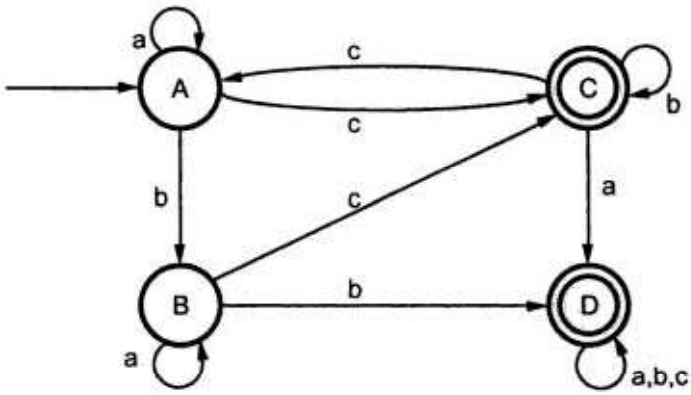


Fig. 2.32

2.6 Minimization of FSM

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

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

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

Method for Construction of Minimum State Automata :

Step 1 : We will create a set Π_0 as $\Pi_0 = \{ Q_1^0, Q_2^0 \}$ where Q_1^0 is set of all final states and $Q_2^0 = Q - Q_1^0$ where Q is a set of all the states in DFA.

Step 2 : Now we will construct Π_{K+1} from Π_K. Let Q_i^K be any subset in Π_K. If q_1 and q_2 are in Q_i^K they are (K + 1) equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are K equivalent. Find out whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are residing in the same equivalence class Π_K. Then it is said that q_1 and q_2

are (K + 1) equivalent. Thus Q_i^K is further divided into (K + 1) equivalence classes. Repeat step 2 for every Q_i^K in Π_K and obtain all the elements of Π_{K+1}.

Step 3 : Construct Π_n for n = 1, 2, ... until $\Pi_n = \Pi_{n+1}$.

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

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

Example 2.16 : *Construct the minimum state automaton for the following transition diagram.*

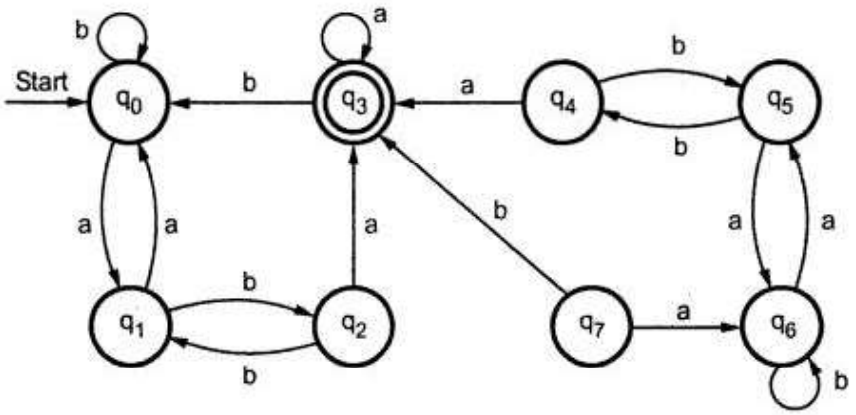


Fig. 2.33

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

We will start constructing equivalence classes.

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

There is only one final state i.e. q_3. Hence we can partition Q as Q_1^0 and Q_2^0.

$$Q_1^0 = F = \{q_3\} \text{ As } \{q_3\} \text{ cannot be partitioned further.}$$

$$Q_2^0 = Q - Q_1^0$$

$$= \{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}$$

Now, we will compare q_0 with q_2.

	a	b
q_0	q_1	q_0
q_2	q_3	q_1

Under a - column of q_0 and q_2 we get q_1 and q_3 respectively. But q_1 and q_3 lie in different states. Hence they are not 1 - equivalent.

Similarly consider q_0 and q_4.

	a	b
q_0	q_1	q_0
q_4	q_3	q_5

Under a - column of q_0 and q_4 we get q_1 and q_3 which lie in different sets. Hence q_0 is not 1 - equivalent to q_4. Similarly q_0 is not 1 - equivalent to q_7 (observe b-column of q_0 and q_7). But q_0 is 1 - equivalent to q_1, q_5 and q_6.

$$Q_1' = \{q_3\}$$

$$Q_2' = \{q_0, q_1, q_5, q_6\}$$

q_2 is 1 - equivalent to q_4. Hence

$$Q_3' = \{q_2, q_4\}$$

Since under a-column we get q_3 only for q_2 and q_4. And under b - column of q_2 and q_4 we get q_1 and q_5. But q_1 and q_5 lie in one set. Thus q_2 and q_4 are 1 - equivalent.

The only element left over in Q_2^0 is q_7.

$\therefore$ $Q_4' = \{q_7\}$ The equivalence class is -

$$\Pi_1 = \{\{q_3\}, \{q_0, q_1, q_5, q_6\}, \{q_2, q_4\}, \{q_7\}\}$$

$$Q_1^2 = \{q_3\}$$

Now q_0 is 2 - equivalent to q_6 because

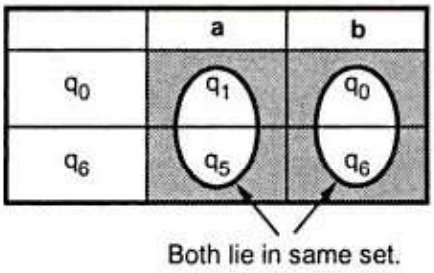


Fig. 2.34 (a)

But q_0 is not 2 - equivalent to q_1 and q_5.

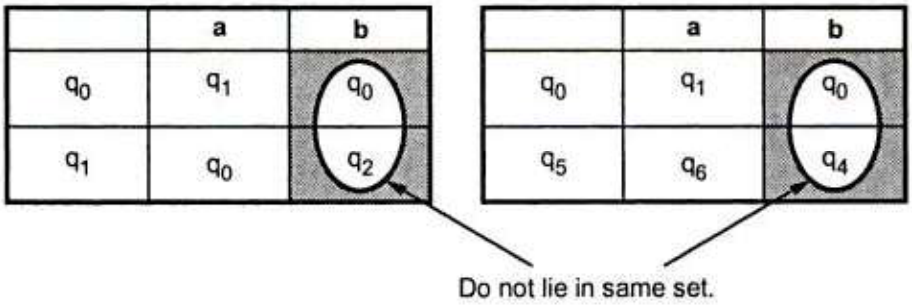


Fig. 2.34 (b)

Hence $Q_2^2 = \{q_0, q_6\}$

But q_1 is 2 - equivalent to q_5.

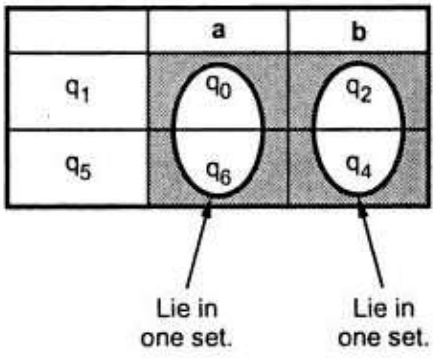


Fig. 2.34 (c)

Hence $Q_3^2 = \{q_1, q_5\}$

Similarly q_2 is 2 - equivalent to q_4.

$Q_4^2 = \{q_2, q_4\}$

$Q_5^2 = \{q_7\}$

Thus, $\Pi_2 = \{\{q_3\}, \{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_7\}\}$

$Q_1^3 = \{q_3\}$

As q_0 is 3 - equivalent to q_6,

$Q_2^3 = \{q_0, q_6\}$

As q_1 is 3 - equivalent to q_5,

$Q_3^3 = \{q_1, q_5\}$

As q_2 is 3 - equivalent to q_4,

$Q_4^3 = \{q_2, q_4\}$

and $Q_5^3 = \{q_7\}$

$\therefore$ $\Pi_3 = \{\{q_3\}, \{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_7\}\}$

As $\Pi_3 = \Pi_2$ we have got equivalence class from Π_2. Hence we can write a transition table as

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

The transition diagram with minimized states is -

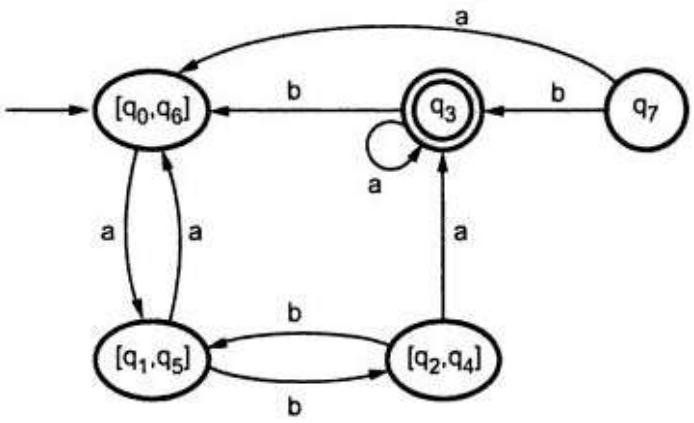


Fig. 2.35

Example 2.17 : *Construct a minimum state DFA for the following given automata -*

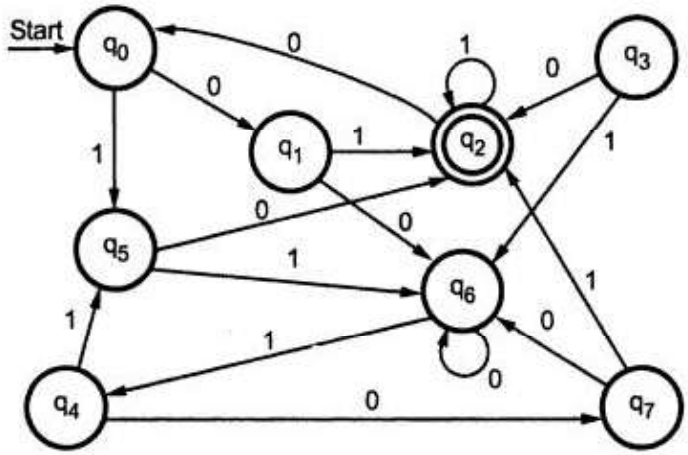


Fig. 2.36

Solution : We will first construct a transition table from given FA.

State \ Input	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Now we will first obtain Q_1^0.

$$Q_1^0 = F = \{q_2\} \rightarrow \text{Final state}$$

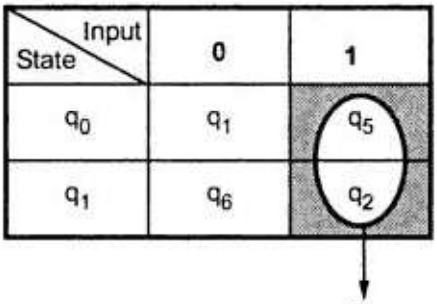
$$Q_2^0 = Q - Q_1^0$$

$$= \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$$

Hence $$\Pi_0 = \{\{q_2\}\ \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$$

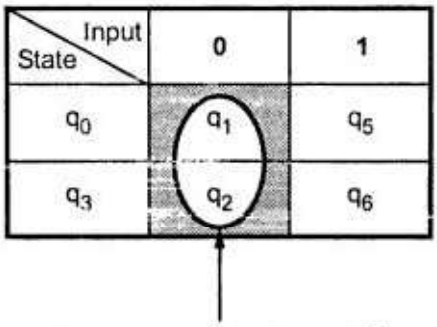
We cannot partition $\{q_2\}$ further. Hence $Q_1' = \{q_2\}$.

Now consider another set from Π_0 i.e. $\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$. We will now compare input entries of q_0 for all remaining states in this set. Consider



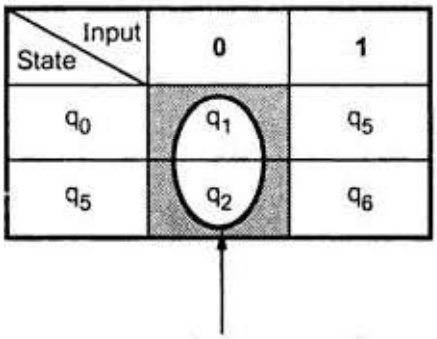
Here $q_5 \in Q_2^0$ and $q_2 \in Q_1^0$.
Hence they are not 1-equivalent.

Fig. 2.37 (a)



Here $q_1 \in Q_2^0$ and $q_2 \in Q_1^0$.
Hence they are not 1-equivalent.

Fig. 2.37 (b)



Here $q_1 \in Q_2^0$ and $q_2 \in Q_1^0$.
Hence they are not 1-equivalent.

Fig. 2.37 (c)

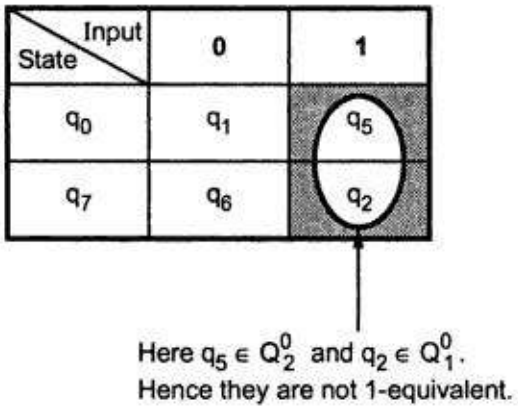


Fig. 2.37 (d)

Thus q_0 is not ρ - equivalent to q_3, q_5 and q_7. Now we will obtain the set of equivalent sets. Consider q_0 with q_4, q_6.

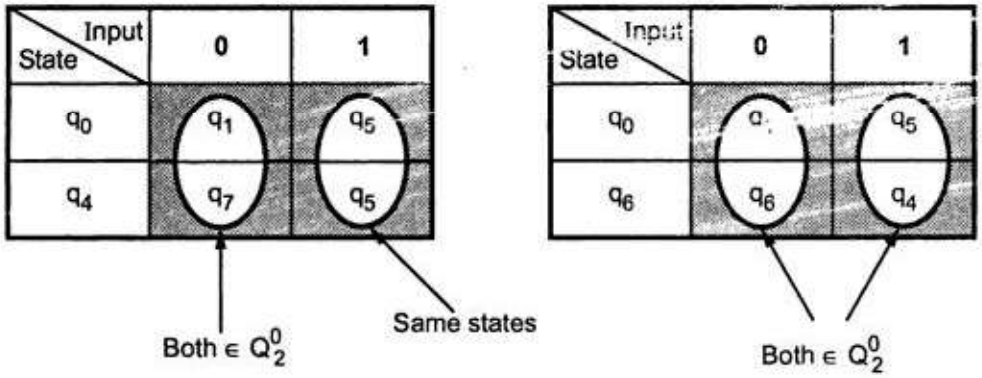


Fig. 2.37 (e)

Hence $\{q_0, q_4, q_6\}$ is one subset in Π_1.

Therefore $Q_2' = \{q_0, q_4, q_6\}$.

We will consider a subset $\{q_1, q_3, q_5, q_7\}$. Now we will find the 1 - equivalent subset from this subset. Hence we need to compare q_1 with q_3, q_5 and q_7 respectively.

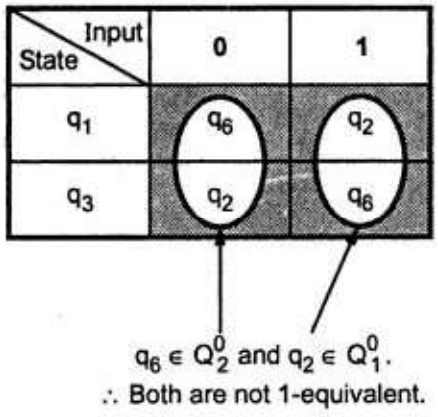


Fig. 2.37 (f)

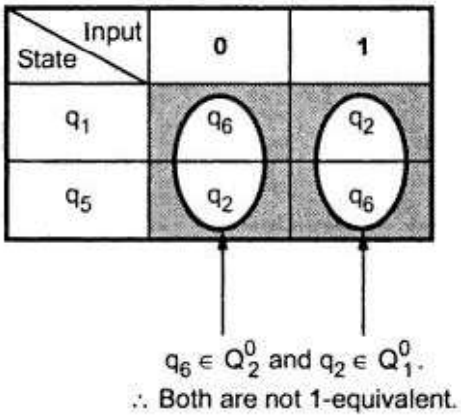


Fig. 2.37 (g)

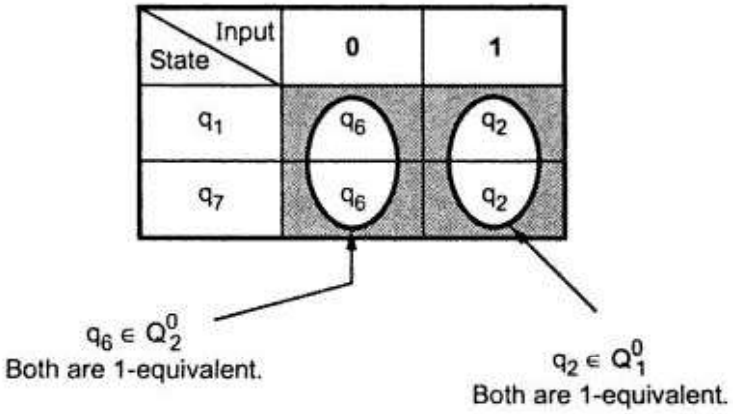


Fig. 2.37 (h)

i.e. q_1 is 1 - equivalent to q_7. Hence $\{q_1, q_7\}$ will be one subset. We cannot partition this subset further. Hence $Q_3' = \{q_1, q_7\}$.

Now we will compare q_3 with q_5.

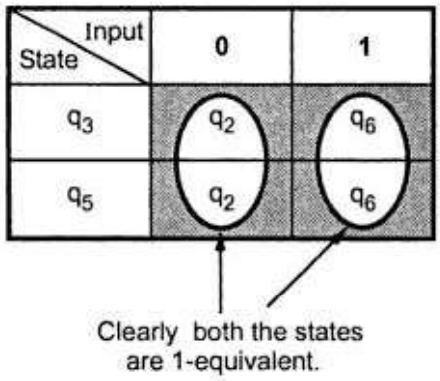


Fig. 2.37 (i)

$\therefore$ $Q'_4 = \{q_3, q_5\}$

Now,

$$\Pi_2 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$$

From Π_2 we can consider a subset $\{q_0, q_4, q_6\}$. We will again compare q_0 with q_4 and q_6.

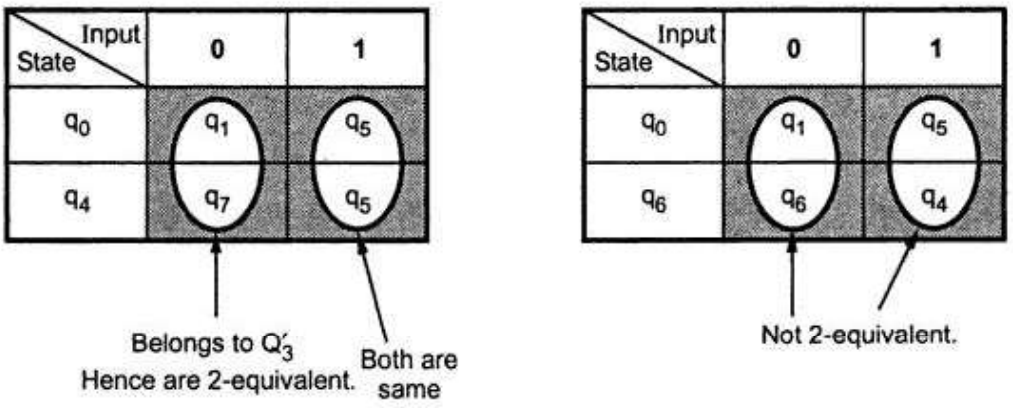


Fig. 2.37 (j)

Thus q_0 and q_4 and 2 - equivalent and q_0, q_6 are not 2 - equivalent.

Similarly q_3 and q_5 are 2 - equivalent.

Also q_1 and q_7 are 2 - equivalent.

Thus $\Pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

Now, q_0 and q_4 are 3 - equivalent.

q_1 and q_7 are 3 - equivalent.

q_3 and q_5 are 3 - equivalent.

As $\Pi_2 = \Pi_3$. The Π_2 gives equivalence classes. Therefore we can construct a finite automata with minimized state as

$$M' = (Q', \{0, 1\}, \delta', q'_0, F')$$

where Q' is a set of states in FA.

$$\text{i.e.} = \{[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]\}$$

q'_0 is initial state i.e. $[q_0, q_4]$

F' is a set of final states i.e. $[q_2]$

The δ' can be shown by following transition table.

State \ Input	0	1
$[q_0, q_4]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_2]$	$[q_0, q_4]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

Here q_0 and q_4, q_1 and q_7, q_3 and q_5 are treated as one state. Now we can draw the transition diagram for minimized machine as -

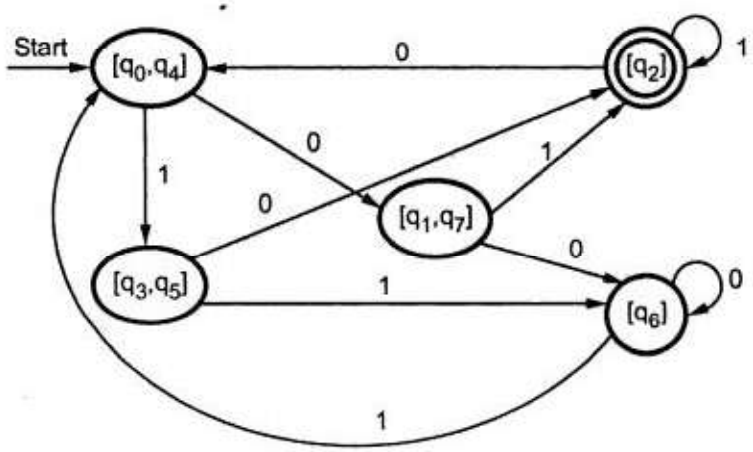


Fig. 2.38

Example 2.18 : *Construct a minimum state automaton for following DFA.*

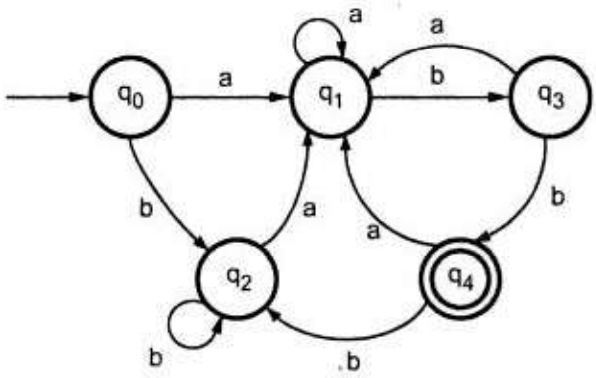


Fig. 2.39

Solution : It will be easier if we construct a transition table for given finite automata.

State \ Input	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
((q_4))	q_1	q_2

Now we will first obtain Q_1^0.

$\therefore$ $Q_1^0 = F = \{q_4\} \rightarrow$ Final state

$Q_2^0 = Q - Q_1^0$.

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

Hence $\Pi_0 = \{\{q_4\}, \{q_0, q_1, q_2, q_3\}\}$

As we cannot partition $Q_1^0 = \{q_4\}$ further we say $Q_1' = \{q_4\}$.

Now consider subset $\{q_0, q_1, q_2, q_3\}$. We will compare q_0 with q_1, q_2 and q_3.

State \ Input	a	b
q_0	q_1	q_2
q_1	q_1	q_3

Both are 1-equivalent.

(a)

State \ Input	a	b
q_0	q_1	q_2
q_2	q_1	q_2

Clearly q_0 and q_2 are equivalent.

(b)

State \ Input	a	b
q_0	q_1	q_2
q_3	q_1	q_4

$q_2 \in Q_2^0$ and

$q_4 \in Q_1^0$ Hence both are not 1-equivalent.

(c)

Fig. 2.40

$$Q'_2 = \{q_0, q_1, q_2\}$$

$$Q'_3 = \{q_3\}$$

Hence, $$\Pi_1 = \{\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}\}$$

Now we will compare q_0 with q_1 and q_2 respectively.

State \ Input	a	b
q_0	q_1	q_2
q_1	q_1	q_3

As $q_2 \in Q'_2$ and $q_3 \in Q'_3$ Both are not 2-equivalent

Fig. 2.40 (d)

State \ Input	a	b
q_0	q_1	q_2
q_2	q_1	q_2

Clearly both the states are equivalent.

Fig. 2.40 (e)

Hence we get, $$\Pi_2 = \{\{q_4\}, \{q_0, q_2\}, \{q_1\}, \{q_3\}\}$$

We can now state $\{q_0, q_2\}$ are 3 - equivalent. And we cannot partition $\{q_1\}$ and $\{q_3\}$ further. Hence

$$\Pi_3 = \{\{q_4\}, \{q_0, q_2\}, \{q_1\}, \{q_3\}\}$$

As $\Pi_3 = \Pi_2$ and Π_2 gives equivalence classes we can construct minimum state finite automata as -

	State \ Input	a	b
→	$[q_0, q_2]$	q_1	$[q_0, q_2]$
	q_1	$[q_1]$	$[q_3]$
	q_3	$[q_1]$	$[q_4]$
	(q_4)	$[q_1]$	$[q_0, q_2]$

The transition diagram can be -

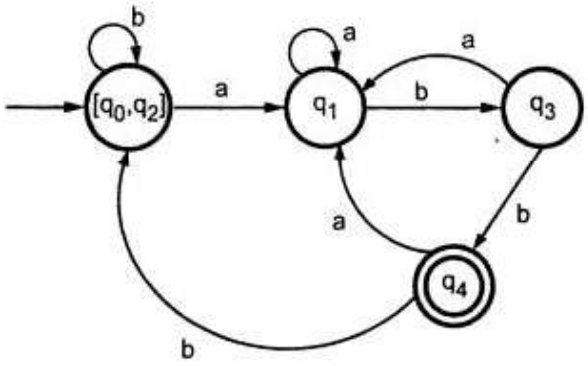


Fig. 2.41

Minimization of DFA (Alternate Method)

This method helps in finding equivalent states in one stroke. Hence it is an efficient method of minimization of given DFA. Let us understand this method with the help of some example.

Example 2.19 : *Minimize the DFA as given below.*

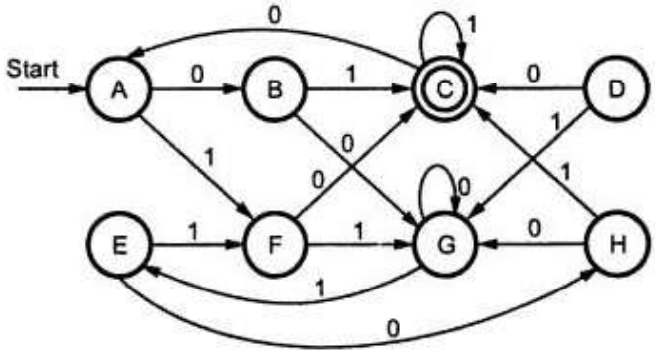
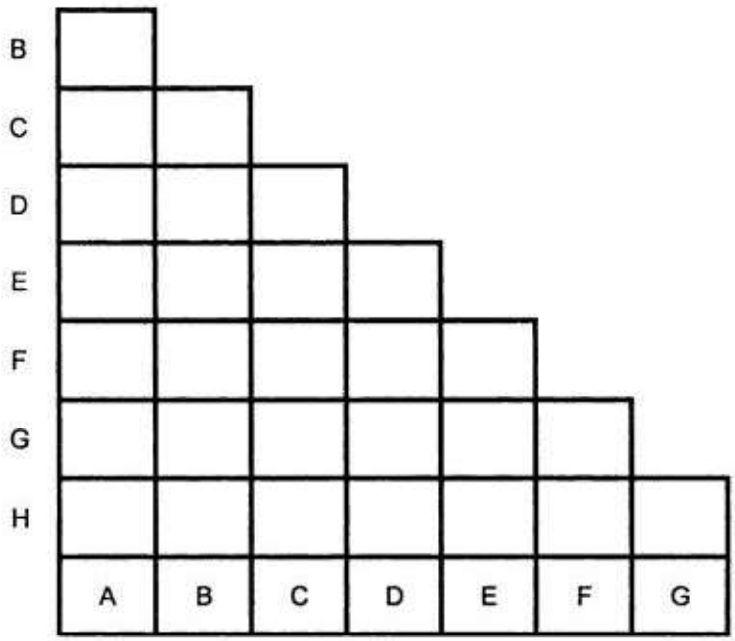


Fig. 2.42

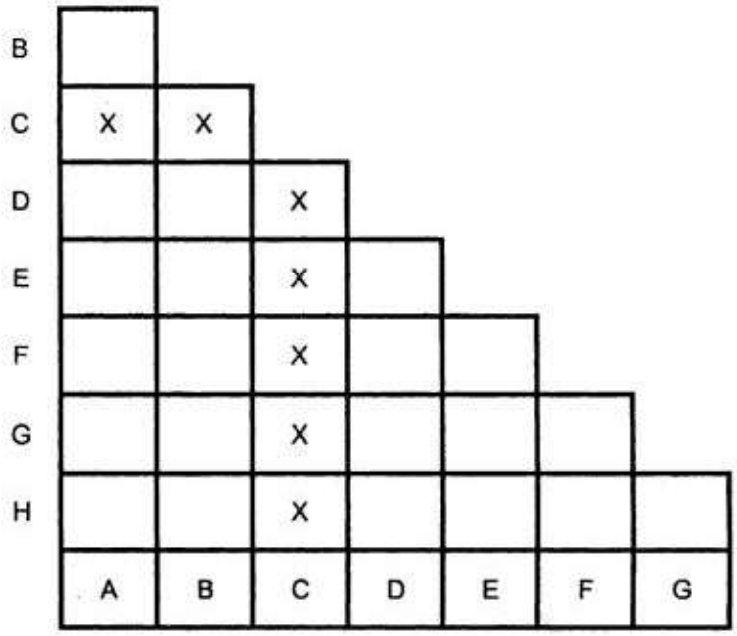
Solution : We will build the transitions table for given DFA as follows.

		0	1
→	A	B	F
	B	G	C
	C	A	C
	D	C	G
	E	H	F
	F	C	G
	G	G	E
	H	G	C

We will now construct a table for each pair of states.



We will mark X for all the final and non final states. Hence



Thus we have marked X for (A, C), (B, C), (C, D), (C, E), (C, F), (C, G), (C, H).

Now we will consider every pair form above table. Consider pair (G, H).

δ (G, 0) = G, δ (G, 1) = E

δ (H, 0) = G, δ (H, 1) = C

As for δ (G, 1) = E and δ (H, 1) = C. We can see X in (C, E) pair. Hence pair (G, H) is not equivalent.

Hence we will mark X in pair (G, H). Thus we will find the equivalent pairs.

Consider pair (B, H)

δ (B, 0) = G, δ (B, 1) = C

δ (H, 0) = G, δ (H, 1) = C

Thus the pair (B, H) is equivalent. Thus we obtain,

B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G	X	X	X	X	X	X	
H	X		X	X	X	X	X
	A	B	C	D	E	F	G

Thus we get equivalent pairs as (A, E), (B, H), (D, F). Hence the minimized DFA will be,

	0	1
→ A	B	D
B	G	C
(C)	A	C
D	C	G
G	G	A

2.7 Equivalence between Two FSM's

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

Method for Comparing two FAS

Let M and M' be two FAS and Σ is a set of input strings.

1 We will construct a transition table have pair wise entries (q, q') where $q \in M$ and $q' \in M'$ for each input symbol.

2. If we get in a pair as one final state and other nonfinal state then we terminate construction of transition table declaring that two FAs are not equivalent.

3. The construction of transition table gets terminated when there is no new pair appearing in the transition table.

Let us take one example to understand the technique of comparing two FA.

Example 2.20 : *Consider the DFAs given below are they equivalent.*

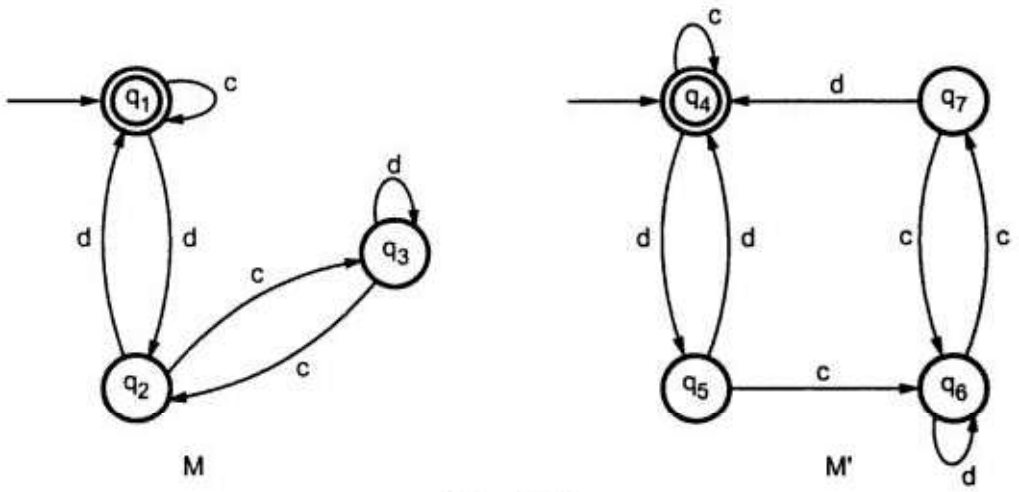


Fig. 2.43

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

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

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

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

Example 2.21 : *Following are two FAs check whether they are equivalent or not.*

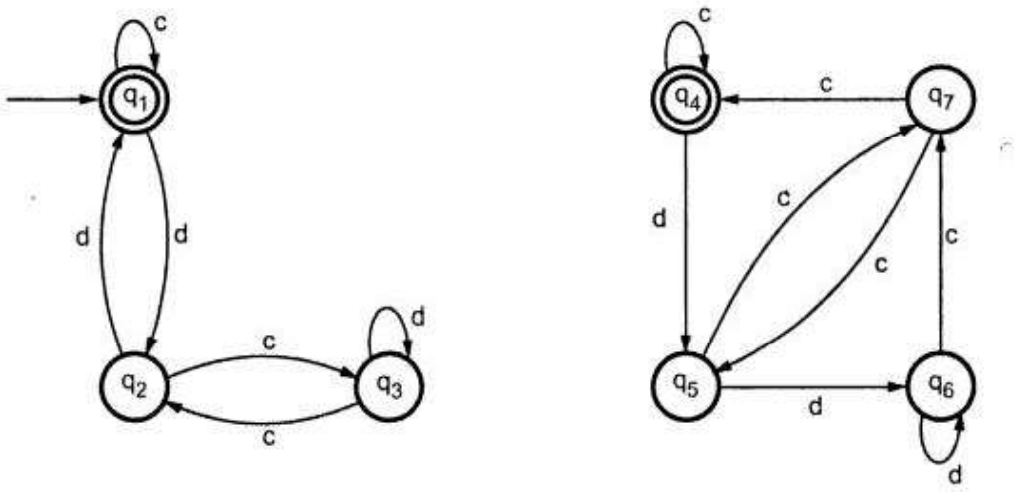


Fig. 2.44

Solution : We will design the transition table for each input symbol c and d as follows -

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

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

2.8 Finite Automata with Output

The finite automata is a collection of $(Q, \Sigma, \delta, q_0, F)$ where Q is a set of states including q_0 as a start state. In FA after reading the input string if we get final state then the string is said to be "acceptable". If we do not get final state then it is said that string is "rejected". That means there is no need of output for the finite Automata. The 'accept' or 'reject' acts like 'yes' or 'no' output for the machine. But if there is a need for specifying the output other than yes or no, then in such a case we require finite automata along with output. There are two types of FA with output and those are :

1) Moore machine

2) Mealy machine

1) Moore machine

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

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

Q is finite set of states.

Σ is finite set of input symbols.

Δ is an output alphabet.

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

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

q_0 is the initial state of machine.

For example :

Consider the Moore machine given below -

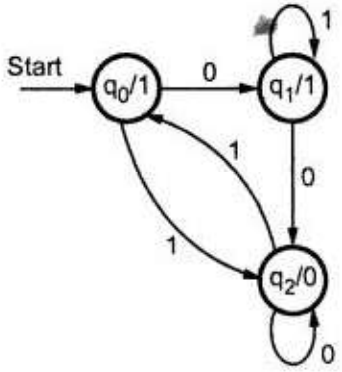


Fig. 2.45

The transition table will be -

Current state	Next state (δ)		Output (λ)
	0	1	
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

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

For the string 0110 then the output will be 11110.

2) Mealy machine

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

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

Q is finite set of states.

Σ is finite set of input symbols.

Δ is an output alphabet.

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

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

q_0 is initial state of machine.

For example :

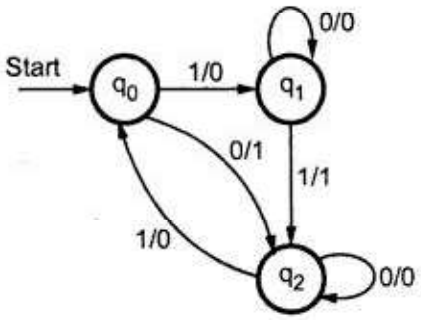


Fig. 2.46

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

Example 2.22 : *Design a Moore machine to generate 1's complement of given binary number.*

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

Hence the Moore machine will be,

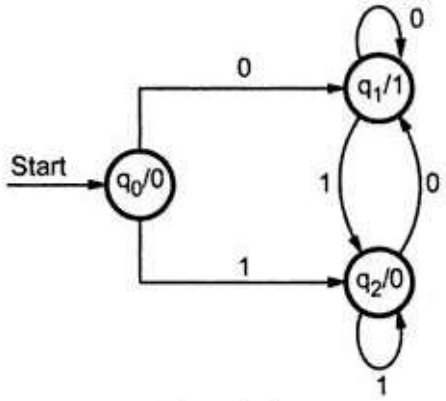


Fig. 2.47

For instance, take one binary number 1011 then

Input :		1	0	1	1
State :	q_0	q_2	q_1	q_2	q_2
Output :	0	0	1	0	0

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

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

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

The transition table shows the δ and λ functions.