4 Finite Automata

'Welcome to the wonderful world of finite state machines.'

Introduction

Computation is a concept common to all computing machines, regardless of the messy details associated with their hardware implementation. However, actual computing machines/computers are too complicated (due to the several constraints caused by physical reality) for a manageable mathematical theory to be ascribed to them. Therefore, in order to fully understand the power and limitation of real machines, idealised computers or computational models are designed and studied. These idealised computers may be accurate in some ways but perhaps not in others.

There are several computational models and the purpose of a computational model is to capture the computational aspects that are relevant to the particular problem under consideration while hiding the other unimportant aspects. Thus, a computational model can be thought of as a custom machine designed to suit particular needs. Some of the important computational models are – *deterministic finite automaton* (DFA), *the non-deterministic finite automaton* (NFA), *the deterministic pushdown automaton* (DPDA), *the nondeterministic pushdown automation* (NPDA), *the deterministic Turing machine* (DTM) *and the nondeterministic Turing machine* (NTM). Undoubtedly each of these models has a special significance in the theory of computation. The most basic computational model is the deterministic finite automaton (DFA).

4.1 Finite Automata

As discussed in chapter 1, finite automaton is a mathematical model of a system with discrete inputs and outputs. Such a system can be in any one of the finite number of internal configurations or 'states' and each state of the system provides sufficient information concerning the past inputs so that the behaviour of the system could be studied on the provision of subsequent inputs. There are many examples of finite automaton like, the control mechanism of an elevator, the controller for an automatic door, human brain, controllers of various household appliances, digital watches, calculators and many examples in computer

science. In fact, the computer itself can be viewed as a finite state system. Switching circuits, which are the control units of a computer, are designed in such a way that their logical design is separated from the electronic implementation and hence they can be viewed as finite state systems. Also, certain commonly used programs like text editors, lexical analysers etc., which are found in most of the compilers, are finite state systems.

In order to locate the strings of characters corresponding to identifiers, reserved words, numerical constants etc., a lexical analyser scans the symbols in a computer program. Thus, in the design of efficient string processors, the theory of finite automata is extensively used.

Theoretically, the state of the *CPU*, main memory and auxiliary storage at any given time correspond to a very large but finite number of states. However, finite automata are indeed good models for computers with an extremely limited amount of memory. Before giving a formal definition of finite automaton, an example is considered to illustrate its salient features.

EXAMPLE 4.1.1: (Automatic door Controller)

Consider the controller for an automatic door which is often found at a supermarket entrance. These doors automatically swing open on sensing the approach of a person. A top view of an automatic door is presented in figure 4.1. There are two pads–one in front of the door and the other located at the rear of the doorway.

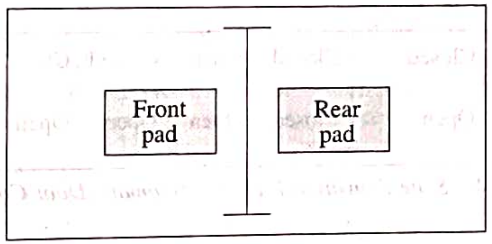

Figure 4.1. *A Top View of an Automatic Door*

The presence of a person, who is about to walk through the doorway, is detected from the front pad while the rear pad makes the controller hold the door open, long enough for not only the person to pass through but also to avoid hitting the persons standing behind the door.

In this system, the controller is in either of *two states* 'open' or 'closed' representing the corresponding conditions of the door at any time. The four possible input conditions are:

a. *Front* → a person standing on the front pad.
b. *Rear* → a person standing on the rear pad.
c. *Both* → people are standing on both the pads.
d. *Neither* → No one on either of the pads.

In figure 4.2 and Table 4.1, the state diagram and the transition table for automatic door controller are presented.

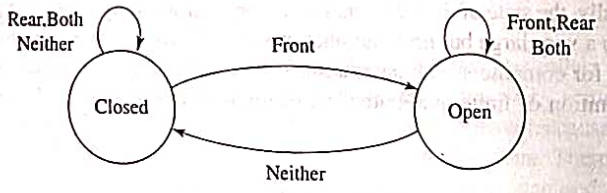

Figure 4.2. *State Diagram for Automatic Door Controller*

State \ Input Signal	Neither	Front	Rear	Both
Closed	Closed	Open	Closed	Closed
Open	Closed	Open	Open	Open

Table 4.1 *State Transition Table for Automatic Door Controller*

The controller moves from one state to the other, depending upon the input it receives. For example, a controller which is initially in 'closed' state, after receiving the input signals: *Front, Rear, Neither, Front, Both, Neither, Rear, Neither*, would move through the series of states: *open, open, closed, open, open, closed, closed, closed.*

Clearly, the controller considered here is a computer with just a single bit of memory, capable of recording the two states. But in the case of an elevator, the controller requires several bits of memory to keep track of the information with regard to the floor. Thus, the design of such finite state systems requires a thorough knowledge of the methodology and terminology of finite automata.

4.1.1 *Finite Automata from a Mathematical Perspective*

In order to develop a precise definition of a finite automaton, the associated terminology for describing and manipulating finite automata, the theoretical results that provide the complete description of their power and limitations and a good mathematical theory of finite automata are needed.

Consider a finite automaton M as depicted in figure 4.3.

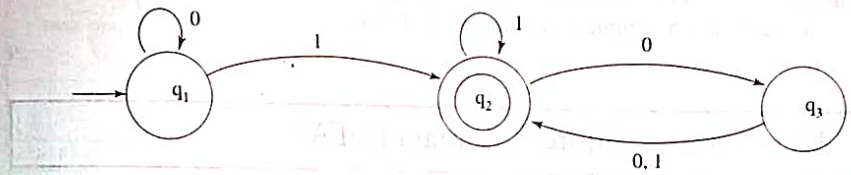

Figure 4.3. *A Finite Automaton M with Three States q_1, q_2 and q_3*

Figure 4.3 is the *state diagram* of M which has

a. three states q_1, q_2 and q_3, where q_1 → *start state*, q_2 → *accept state*
b. the *transitions*–represented by arrows
c. the input string 1101
d. the output, which is either accept or reject.

The processing of the input begins in M at the start state. As it receives the symbols from the string one by one from left to right, M moves from one state to another as directed by the transitions. When the last symbol is read, M produces the output 'accept' if M is in the accept state otherwise the output is 'reject'. The step-by-step process can be described as follows:

Step 1: Start state q_1
Step 2: read 1; transition from q_1 to q_2
Step 3: read 1; transition from q_2 to q_2
Step 4: read 0; transition from q_2 to q_3
Step 5: read 1; transition from q_3 to q_2.

Output is 'accept', since M is in the accept state q_2 at the end of the output.

A careful experimentation with the machine M with a variety of input strings reveals that the following are 'accept' strings:

1, 01, 11, 0101010101, 100, 0100, 110000 and 0101000000;

The 'reject' strings are : 0, 10, 101000.

4.1.2 *Formal Definition of a Finite Automaton*

A formal definition is needed for two specific reasons:

a. A formal definition is precise in nature and resolves any ambiguities involved.
b. A formal definition provides a good notation to think and express clearly.

According to the formal definition, a finite automation is a list of five objects: set of states, input alphabet, rules for moving, start and accept states. Thus a mathematical definition of a finite automaton is a five-tuple consisting of the five objects.

4.2 Deterministic Finite Automata (DFA)

Definition
A deterministic finite automaton is a finite state machine where for each pair of states and input symbol there is a unique next state.

4.2.1 *Elements of DFA*

The deterministic finite automata exhibits the following five characteristics:

a. a finite set of *states* Q,
b. an *alphabet* Σ of possible input symbols,
c. a transition function δ such that $\delta(x, 1) = y$ where $x, y \in Q$ and $1 \in \Sigma$,
d. the *initial state* $q0 \in Q$,
e. the set of final states (F), where $F \subseteq Q$.

Note–1:
The term deterministic refers to the fact that on each input, there is one and only one state to which the automaton can transit from its current state.

4.2.2 *Operation of DFA*

Initially, the DFA is assumed to be in the initial state q_0 with its read head on the left most symbol of the input string. During each move of DFA, the read head moves one position to the right. Thus, each move consumes one input symbol. When the end of string is reached, the string is accepted if DFA is in one of its final states, else rejected. The working of DFA is demonstrated in examples 4.2.1 and 4.2.2.

EXAMPLE 4.2.1: (DFA that accepts the input string *abba*)

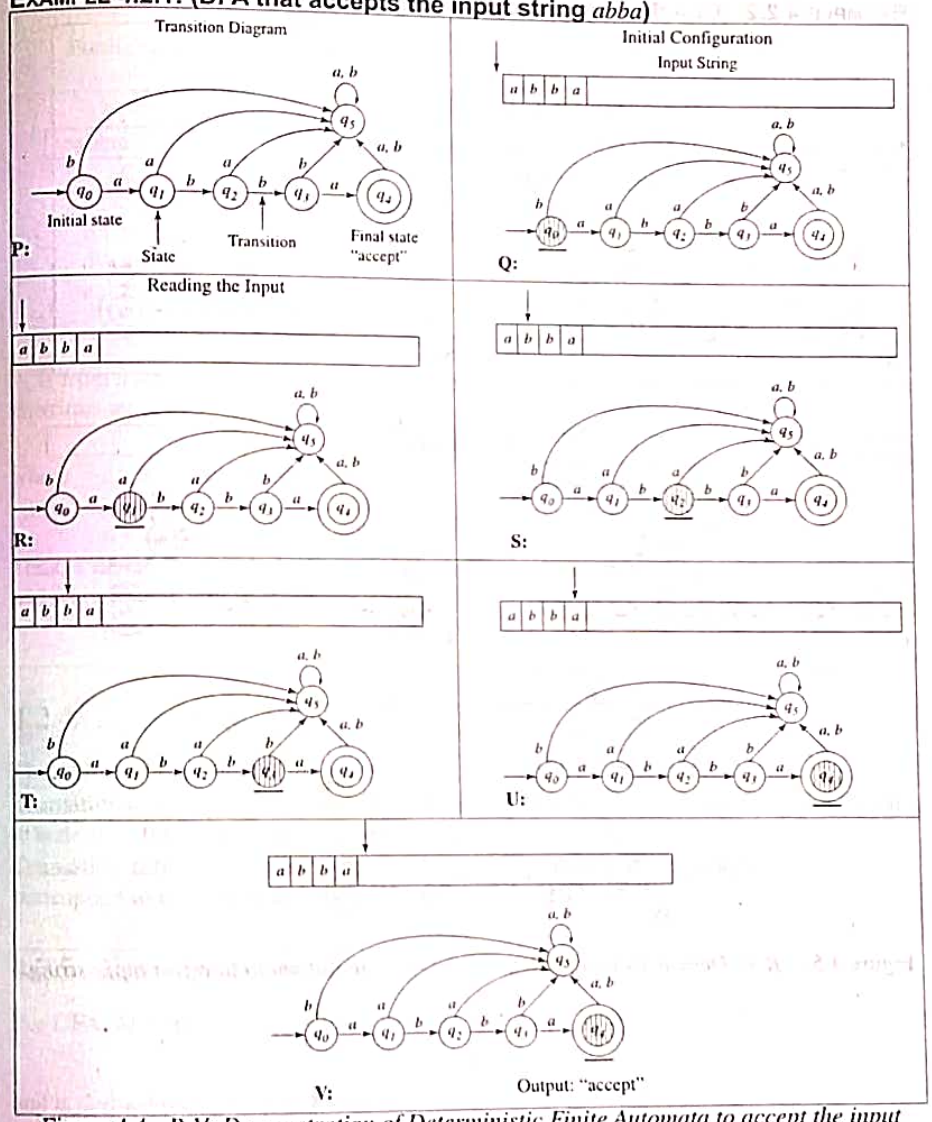

Figure 4.4. *P-V: Demonstration of Deterministic Finite Automata to accept the input string.*

EXAMPLE 4.2.2: (DFA that rejects input string *aba*)

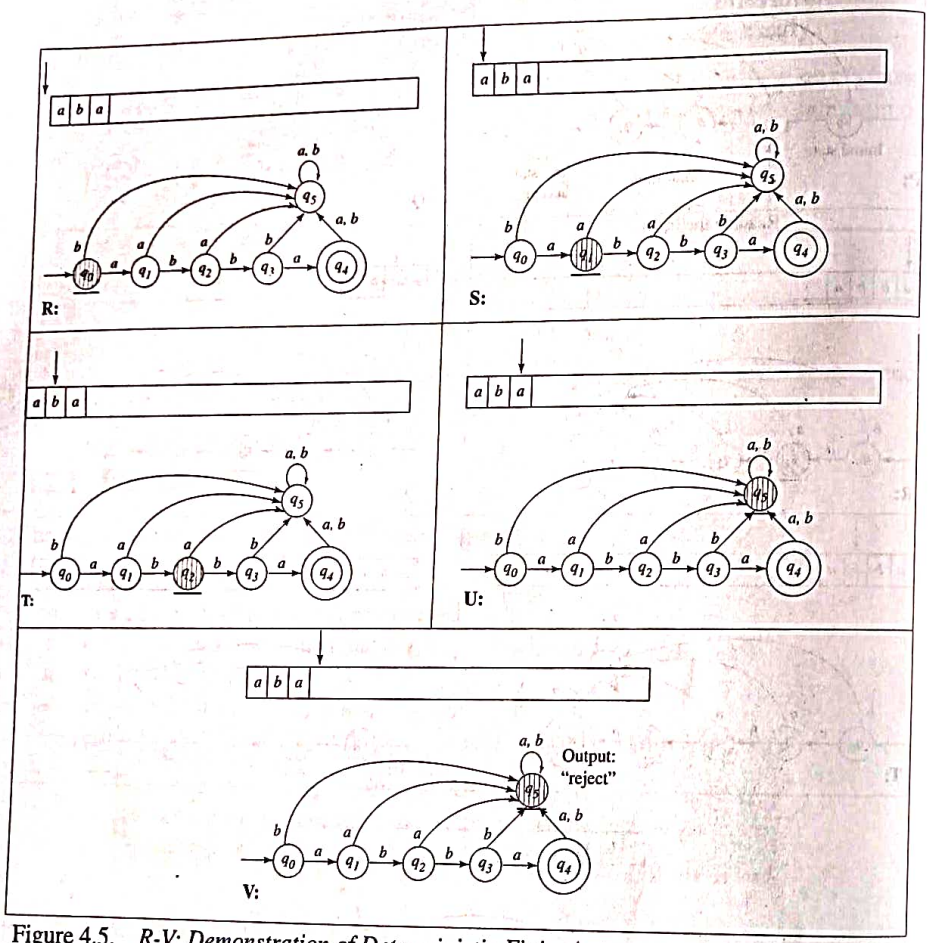

Figure 4.5. *R-V: Demonstration of Deterministic Finite Automata to reject input string.*

4.2.3 *Ordered Quintuple Specification of DFA:*

Formally, a DFA, M is a five-tuple

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

where

a. Q is a finite set of *states* of finite automata,
b. Σ is a finite set of input symbols called the *alphabet*,
c. $\delta : Q \times \Sigma \to Q$ is the transition function,
d. $q_0 \in Q$ is the start state,
e. $F \subseteq Q$ is the set of accept states.

If from a state p, there exists a transition going to state q on an input symbol a, then this is written as

$$\delta(p, a) = q$$

where δ - is a function whose domain is a set of ordered pairs (p, a)
p - a state
a - input symbol.

Thus, δ defines a mapping whose domain will be a set of ordered pairs of the form (p, a) and whose range will be a set of states i.e.,

$$\delta : Q \times \Sigma \to Q.$$

4.2.4 *Description of a DFA*

The *transitions of DFA* can be represented using transition diagram or table.

Transition diagram: if $\delta(p, a) = q$, then the arrow goes from the vertex which corresponds to state p, to the vertex that corresponds to state q labelled by a.

Transition table: Rows correspond to states and columns correspond to inputs. Entries correspond to next states to indicate the transition of DFA.

4.2.5 *Extended transition function for DFA*

For DFA, $M = (Q, \Sigma, \delta, q_0, F)$ the function 'δ' is extended as

$$\delta : Q \times \Sigma^* \to Q$$

and is defined recursively as follows:

a. For any state q of Q

$$\delta(q, \epsilon) = q$$

This means that DFA stays in the same state q when it reads an empty string at q.

b. For any state q of Q, any string $x \epsilon \Sigma^*$ with a as the last symbol of x and $a \epsilon \Sigma$

$$\delta(q, xa) = \delta(\delta(q, x), a)$$

4.2.6 *Language accepted by DFA*

The language accepted by a DFA, $M = (Q, \Sigma, \delta, q_0, F)$ is the set of all strings on Σ accepted by M i.e.,

$$L(M) = \{W \in \Sigma^* | \delta(q_0, W) \in F\}$$

A language is said to be rejected by DFA if $M = (Q, \Sigma, \delta, q_0, F)$ such that

$$L(M) = \{W \in \Sigma^* | \delta(q_0, W) \notin F\}$$

Note–2: A machine may accept several strings but it recognises only one language.

4.3 Design of DFAs

The basic design strategy for DFA is as follows:

a. Understand the language properties for which the DFA has to be designed.
b. Determine the state set required.
c. Identify the initial, accepting and dead state of DFA.
d. For each state, decide on the transition to be made for each character of the input string.

e. Obtain the transition table and diagram for DFA.
f. Test the DFA obtained on short strings.

EXAMPLE 4.3.1: To design a DFA that accepts set of all strings that contain 0's or 1's and end in 00.

Solution: We are required to design a DFA for the regular expression $r = (0 + 1)^*00\ldots$ In other words the DFA shoud be designed to accept the language of r,

i.e., $L(M) = \{00, 100, 1100, 0000, 111000, \ldots.\}$

where M is a DFA.

Consider, $\Sigma = \{0, 1\}$
$Q = \{q_0, q_1, q_2\}$
q_0 = initial state
q_2 = final state and $\delta : Q \times \Sigma \to Q$ is given by

Transition diagram:

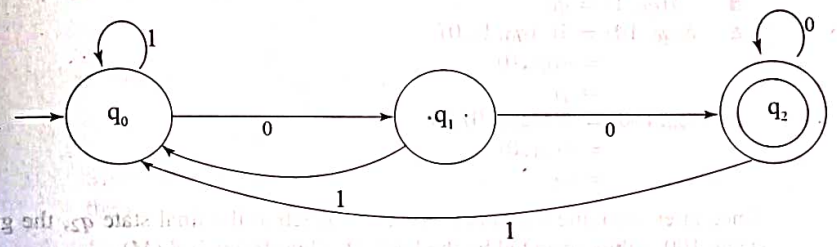

Figure 4.6. *State Transition to represent* ($L(M) = \{W \in \Sigma^* | W$ *ends with* 00$\}$)

Transition table:

States	Σ / Q	Present Inputs 0	1
→	q_0	q_1	q_0
	q_1	q_2	q_0
*	q_2	q_2	q_0

Table 4.2 *State Transition Table*

DFA action for the input string

1. To show that the string 100 is accepted by DFA:
   i. *Using sequence state diagram*

Figure 4.7. *Sequence State Diagram for 100*

Since we encounter the end of the input and we are in the final state, we say that string is accepted by machine M. Thus 100 is in $L(M)$.

ii. *Using extended transition function*
Consider,

- $\delta(q_0, 1) = q_0$
- $\delta(q_0, 10) = \delta(\delta(q_0, 1), 0)$
  $= \delta(q_0, 0)$
  $= q_1$
- $\delta(q_0, 100) = \delta(\delta(q_0, 10), 0)$
  $= \delta(q_1, 0)$
  $= q_2$

Since after scanning the entire string, we reach at the final state q_2, the given string 100 is thus accepted by the DFA M. Thus 100 is in $L(M)$.

iii. *Using vdash function* ($\vdash$)

- An input string a is accepted by $\vdash M$ iff $(W \in \Sigma^* | \vdash (q, a) \in F)$.
- $\vdash: Q \times \Sigma \to Q$
- (q, e) means 'reached end of input'.

Consider,

$$(q_0, 100) = \vdash M(q_0, 00)$$
$$\vdash M(q_1, 0)$$
$$\vdash M(q_2, e)$$

Since we reached the end of input (e) and we are in the final state q_2, hence 100 is accepted by M. Thus 100 is in $L(M)$.

2. To show that the string 010 is rejected by DFA:
   i. *Using sequence state diagram*

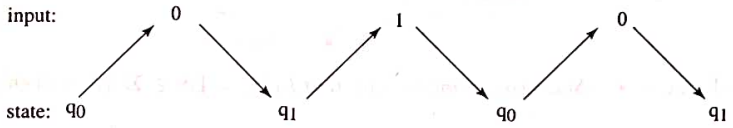

Figure 4.8. *Sequence State Diagram for 010*

Since we encounter the end of input and q_1 is not the final state, we say that the string 010 is rejected by the machine.

ii. *Using extended transition function*

- $\delta(q_0, 0) = q_1$
- $\delta(q_0, 01) = \delta(\delta(q_0, 0), 1)$
  $= \delta(q_1, 1)$
  $= q_0$
- $\delta(q_0, 010) = \delta(\delta(q_0, 01), 0)$
  $= \delta(q_0, 0)$
  $= q_1$

Since after scanning the entire string, we did not reach the final state q_2, hence the string 010 is rejected by DFA 'M'.

iii. *Using vdash function*

$$(q_0, 010) = \vdash M(q_1, 10)$$
$$\vdash M(q_0, 0)$$
$$\vdash M(q_1, e).$$

Since we reached the end of input (e) and we are not in final state q_2, hence 010 is rejected by M. Thus 010 is not in $L(M)$.

EXAMPLE 4.3.2: To design a DFA that accepts a set of even number of a's.

Solution: We are required to design a DFA, M for the regular expression $r = (aa)^*$. In other words, DFA should be designed to accept the language of r,

$$\text{i.e., } L(M) = \{aa, aaaa, aaaaaa, \ldots.\}$$

Consider: $\Sigma = \{a\}$, $Q = \{q_0, q_1\}$, q_0 = initial state, q_1 = final state and δ is given by:

Transition diagram:

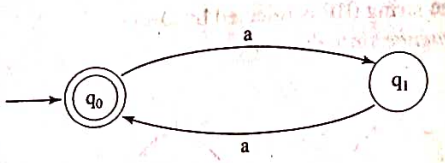

Figure 4.9. *State Transition to represent* $L(M) = \{W \in \Sigma^* | W \text{ is even}\}$

Transition Table:

Q \ Σ	Present Inputs
	a
→ *q_0	q_1
q_1	q_0

Table 4.3 *State Transition Table*

DFA action for the input string

i. To show that the string *aaaa* is accepted by DFA:
Consider,

$$\begin{aligned}(q_0, aaaa) &\vdash M(q_1, aaa)\\ &\vdash M(q_0, aa)\\ &\vdash M(q_1, a)\\ &\vdash M(q_0, e)\end{aligned}$$

Since we reached end of input (e) and we are in the final state q_0, hence *aaaa* is accepted by M. Thus *aaaa* is in $L(M)$.

ii. To show that the string *aaaaa* is rejected by DFA:
Consider,

$$\begin{aligned}(q_0, aaaaa) &\vdash M(q_1, aaaa)\\ &\vdash M(q_0, aaa)\\ &\vdash M(q_1, aa)\\ &\vdash M(q_0, a)\\ &\vdash M(q_1, e).\end{aligned}$$

Since we reached the end of input and we are not in the final state q_0, hence *aaaaa* is not accepted by M.

EXAMPLE 4.3.3: To design DFA that accepts odd number of 1's.

Solution: We are required to design a DFA, M for the regular expression $r = (11)^*1$ or $r = 1(11)^*$. In other words, DFA should be designed to accept the language of r,

$$\text{i.e., } L(M) = \{1, 111, 11111, \ldots.\}.$$

Consider, $\Sigma = \{1\}$, $Q = \{q_0, q_1, q_2\}$, q_0 = initial state, q_1 = final state and δ is given by:

Transition diagram:

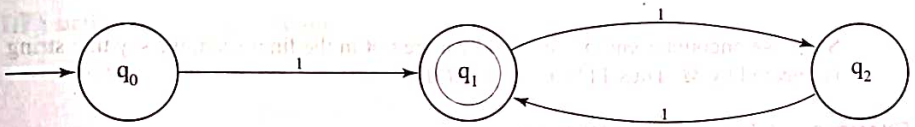

Figure 4.10. *Transition Diagram for* $L(M) = \{W \in \Sigma^* | W \text{ is Odd Number of } 1\text{'s}\}$

Transition Table:

Q \ Σ	Present Inputs
	1
→ q_0	q_1
q_1	q_2
* q_2	q_1

Table 4.4 *State Transition Table*

DFA action for the input string

i. To show that 11111 is accepted by DFA:

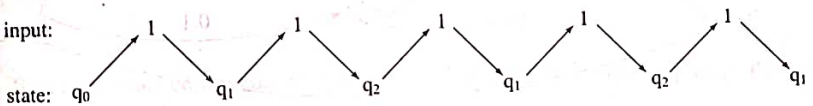

Figure 4.11. *State Sequence Diagram for 11111*

Since we encounter end of input and we are in the final state, we say that the string is accepted by M. Thus, 11111 is in $L(M)$.

ii. To show that the string 1111 is rejected by DFA:

Figure 4.12. *State Sequence Diagram for 1111*

Since we encounter end of input and we are not in the final state, we say that string is rejected by M. Thus 1111 is not in $L(M)$.

EXAMPLE 4.3.4: To design a DFA that

- starts with 0 and has odd number of 0s
- starts with 1 and has even number of 1s.

Solution: We are required to design a DFA, M for the regular expression

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

i.e $L(M) = \{01100,\ 110011, \ldots\ldots\}$.

Consider, $\Sigma = \{0,1\}$, $Q = \{q_0, q_1, q_2\}$, q_0 = initial state, q_1 = final state and δ is given by:

Transition diagram:

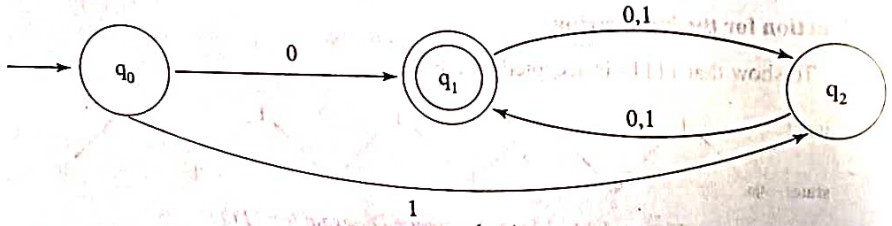

Figure 4.13. *Transition Diagram for* $L(M) = \left\{ W \;\middle|\; \begin{array}{l} W \text{ starts with 0 and has odd 0's} \\ W \text{ starts with 1 and has even 1's} \end{array} \right\}$

Transition Table:

Q \ Σ	Present Inputs 0	1
→ q0	q1	q2
* q1	q2	q2
q2	q1	q1

Table 4.5 *State Transition Table*

DFA action for the input string

i. To show that 110 011 is accepted by DFA:

Consider, $(q_0, 110011) \vdash M(q_2, 10011)$
$\vdash M(q_1, 0011)$
$\vdash M(q_2, 011)$
$\vdash M(q_1, 11)$
$\vdash M(q_2, 1)$
$\vdash M(q_1, e)$

Since we reached the end of input (e) and we are in the final state q_1, hence 110011 is accepted by M.

ii. To show that the string 011000 is rejected by DFA:

Consider, $(q_0, 011000) \vdash M(q_1, 11000)$
$\vdash M(q_2, 1000)$
$\vdash M(q_1, 000)$
$\vdash M(q_2, 00)$
$\vdash M(q_1, 0)$
$\vdash M(q_2, e)$

Since we reached the end of input (e) and not in the final state q_1, hence 011000 is rejected by $L(M)$.

EXAMPLE 4.3.5: To design a DFA to accept even number of a's and b's.

Solution: This is to design a DFA, M for the regular expression $r = (aa)^*(bb)^*$

i.e $L(M) = \{aa, bb, aabb, aaaabbbb, aaaabb, \ldots.\}$

Consider, $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$ q_0 = initial and final states and δ is given by:

Transition diagram:

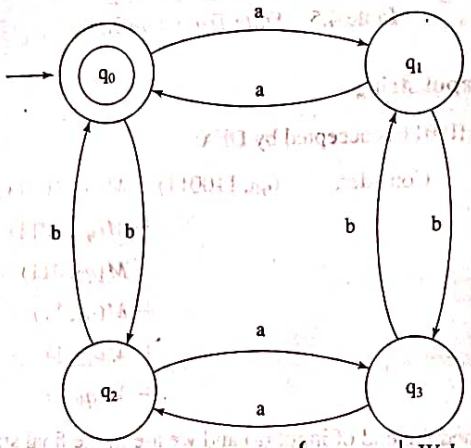

Figure 4.14. *Transition Diagram for* $L(M) = \left\{ W \in \Sigma^* \;\middle|\; \begin{array}{l} W \text{ has even} \\ \text{length of a's and b's} \end{array} \right\}$

Transition Table:

Q \ Σ	Present Inputs a	b
→ * q0	q1	q2
q1	q0	q3
q2	q3	q0
q3	q2	q1

Table 4.6 *State Transition Table*

DFA action for the input string

i. To show that *aabb* is accepted by DFA:

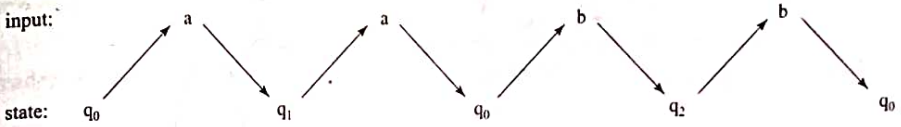

Figure 4.15. *State Sequence Diagram for aabb*

Since we encounter the end of input and we are in the final state, we say that the string is accepted by M.

ii. To show that *aabba* is rejected by DFA:

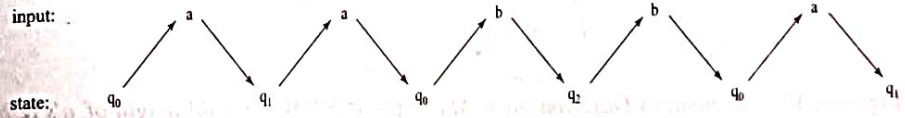

Figure 4.16. *State Sequence Diagram for aabba*

Since we encounter the end of input and we are not in the final state we say that the string is not accepted by M hence *aabba* is not in $L(M)$.

EXAMPLE 4.3.6: To design a DFA to accept odd number of a's and, followed by b's.

Solution: This is to design a DFA, M for the regular expression $r = (aa)^*a.(bb)^*b$

i.e $L(M) = \{ab, aaabbb, abbbbb, ababab, aabbab, \ldots.\}$.

Consider, $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$ q_0 = initial state, q_3 = final state and δ is given by:

Transition Table:

Q \ Σ	Present Inputs a	b
→ q0	q1	q2
q1	q0	q3
q2	q3	q0
* q3	q2	q1

Table 4.7 *State Transition Table*

Transition diagram:

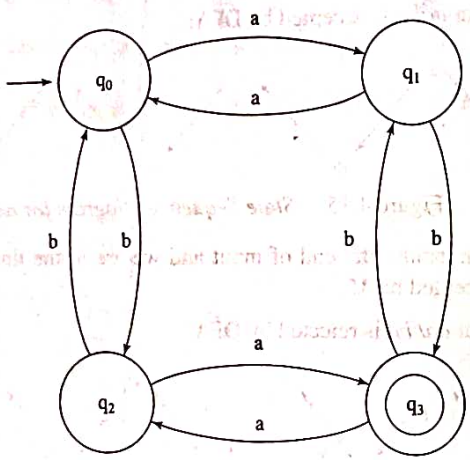

Figure 4.17. *Transition Diagram for* $L(M) = \{W \in \Sigma^* | W$ *has odd length of a's & b's*$\}$

DFA action for the input string

i. To show that *ababab* is accepted by DFA:

$$\begin{aligned} \text{Consider,} \quad (q_0, ababab) &\vdash M(q_1, babab) \\ &\vdash M(q_3, abab) \\ &\vdash M(q_2, bab) \\ &\vdash M(q_0, ab) \\ &\vdash M(q_1, b) \\ &\vdash M(q_3, e) \end{aligned}$$

Since we reached the end of input(e), and we are in the finalstate, hence *ababab* is accepted by$M$.

ii. To show that *bbabb* is rejected by DFA:

$$\begin{aligned} \text{Consider,} \quad (q_0, bbabb) &\vdash (q_2, babb) \\ &\vdash M(q_0, abb) \\ &\vdash M(q_1, bb) \\ &\vdash M(q_3, b) \\ &\vdash M(q_1, e) \end{aligned}$$

Since we reached the end of input (e) and not in the final state, hence *bbabb* is rejected by DFA.

EXAMPLE 4.3.7: To design a DFA to accept odd number of a's and even number of b's

Solution: This is to design a DFA, M for the regular expression $r = (aa)^*a.(bb)^*$.

i.e $L(M) = \{aaa, aaabb, a, abb, abbbb, aaabbbb, \ldots\}$

Consider, $\Sigma = \{a, b\}, Q = \{q_0, q_1, q_2, q_3\}, q_0$ = initial state, q_1 = final state and δ is given by:

Transition diagram:

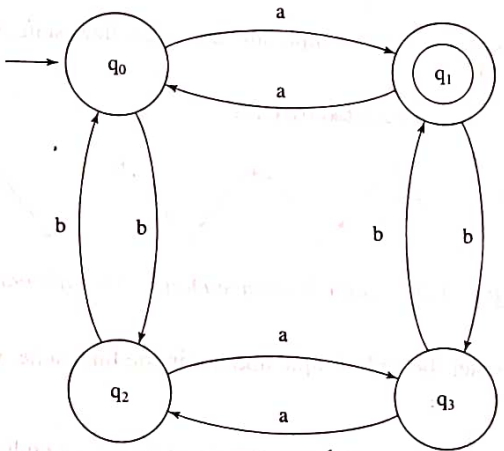

Figure 4.18. Transition Diagram for $L(M) = \left\{ W \in \Sigma^* \;\middle|\; \begin{array}{l} W \text{ has odd no of } a\text{'s} \\ \text{and even no of } b\text{'s} \end{array} \right\}$

Transition Table:

Q \ Σ	Present Inputs a	b
→ q0	q1	q2
* q1	q0	q3
q2	q3	q0
q3	q2	q1

Table 4.8 *State Transition Table*

DFA action for the input string

i. To show that *ababa* is accepted by DFA:

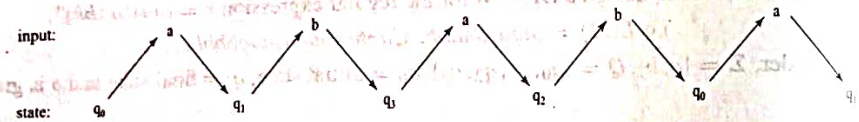

Figure 4.19. *State Sequence Diagram for ababa*

Since we encounter the end of input and are in the final state, we say that the string is accepted by M.

ii. To show that *ababaa* is rejected by DFA:

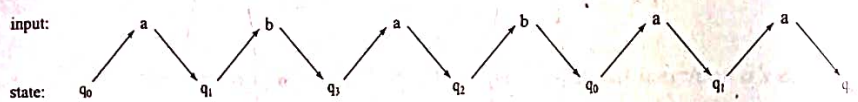

Figure 4.20. *State Sequence Diagram for ababaaa*

Since we encounter the end of input and not in the final state, we say that the string is rejected by M.

EXAMPLE 4.3.8: To design a DFA that contains set of all strings ending with 3 consecutive zeros, over the alphabet $\{0, 1\}$.

Transition diagram:

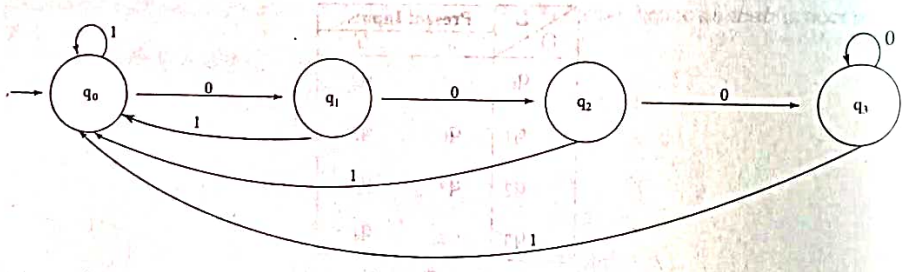

Figure 4.21. *Transition Diagram for* $L(M) = \{W \in \Sigma^* \{W$ *is String Ending with 3 Consecutive Zeros*$\}$

Solution: This is to design a DFA, M for the regular expression $r = (0 + 1)^*000$ or $r = 1^*(01)^*(001)^*(0001)^*.000$

i.e $L(M) = \{000, 1000, 11000, \ldots.\}$

Consider, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_3\}$, q_0 = initial state, q_3 = final state and δ is shown in figure 4.21. and Table 4.9.

Transition Table:

Q \ Σ	Present Inputs 0	1
→ q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
$*q_3$	q_3	q_0

Table 4.9 *State Transition Table*

DFA action for the input string

To show that 01011000 is accept by DFA:

$$
\begin{aligned}
\text{Consider,} \quad (q_0, 01011000) &\vdash M(q_1, 1011000) \\
&\vdash M(q_0, 011000) \\
&\vdash M(q_1, 11000) \\
&\vdash M(q_0, 1000) \\
&\vdash M(q_0, 000) \\
&\vdash M(q_1, 00) \\
&\vdash M(q_2, 0) \\
&\vdash M(q_3, e).
\end{aligned}
$$

Since we reached the end of input(e) and are in the final state q_3, hence 01011000 is accepted by M.

EXAMPLE 4.3.9: To design a DFA, to accept the language $L = \{awa | W \in (a + b)^*\}$ over alphabets $\Sigma = \{a, b\}$.

Solution: This is to design a DFA, M for the regular expression $r = a(a+b)^*a$ or $r = ab^*(ab)^*a^*$ i.e. a machine M to accept all strings starting with 'a', followed by any number of a's and b's and ending at a

i.e., $L(M) = \{aa, aba, aaaa, abbbaaa, \ldots.\}$.

Consider, $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$ q_0 = initial state, q_2 = final state and δ is given by:

Transition diagram:

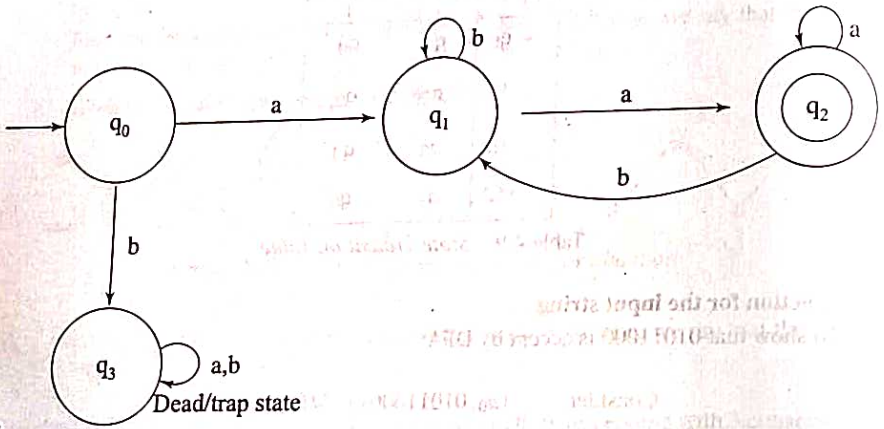

Figure 4.22. *Transition Diagram for* $L(M) = \{W \in \Sigma^* | W \in a(a+b)^*a\}$

Transition Table:

Q \ Σ	Present Inputs a	b
→q0	q1	q3
q1	q2	q1
*q2	q2	q1
q3	q3	q3

Table 4.10 *State Transition Table*

DFA action for the input string

i. To show that *ababa* is accepted by DFA:

$$\begin{aligned} \text{Consider} \quad (q_0, ababa) &\vdash M(q_1, baba) \\ &\vdash M(q_1, aba) \\ &\vdash M(q_2, ba) \\ &\vdash M(q_1, a) \\ &\vdash M(q_1, e). \end{aligned}$$

Since we reached the end of input (e) and are in the final state, hence *ababa* is accepted.

ii. To show that *abababb* is rejected by DFA:

$$\begin{aligned} \text{Consider} \quad (q_0, abababb) &\vdash M(q_1, bababb) \\ &\vdash M(q_1, ababb) \\ &\vdash M(q_2, babb) \\ &\vdash M(q_1, abb) \\ &\vdash M(q_2, bb) \\ &\vdash M(q_1, b) \\ &\vdash M(q_1, e). \end{aligned}$$

Since we reached the end of input and are not in the final state, hence *abababb* is rejected by 'M'.

EXAMPLE 4.3.10: To design a DFA to accept the set of all strings of a and b starting with the string *ab*.

Solution: This is to design DFA, M for the regular expression $r = ab\,(a+b)^*$

i.e., $L(M) = \{ab, abaa, abaaa, abbbb, \ldots.\}$.

Consider, $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$, q_0 = initial state, q_2 = final state and δ is given as follows:

Transition diagram:

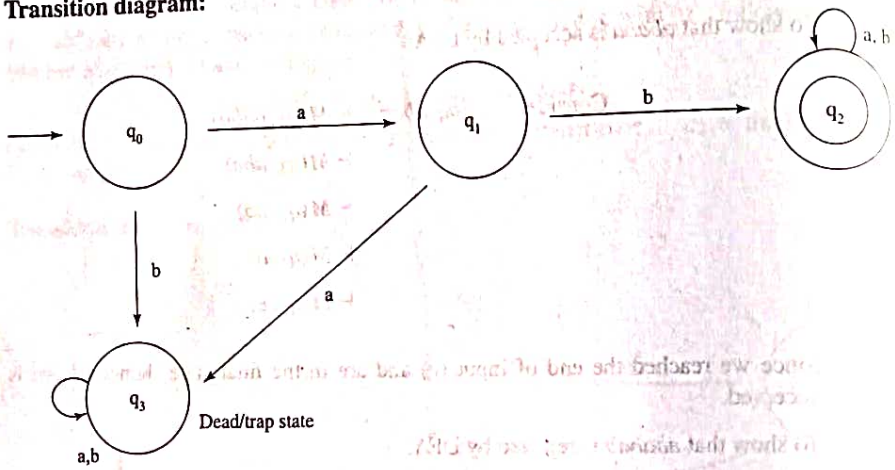

Figure 4.23. *Transition Diagram for* $L(M) = \{W \in \Sigma^* | W \text{ start with } ab\}$

Transition Table:

Q \ Σ	Present Inputs a	b
→ q_0	q_1	q_3
q_1	q_3	q_2
* q_2	q_2	q_2
q_3	q_3	q_3

Table 4.11 *State Transition Table*

DFA action for the input string

i. To show that the string *ababa* is accepted by DFA:

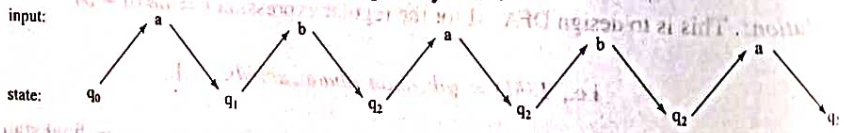

Figure 4.24. *State Sequence Diagram for ababa*

Since we encounter the end of input and we are in the final state, we say that the string *ababa* is accepted by DFA.

ii. To show that the string *aaab* is rejected by DFA.

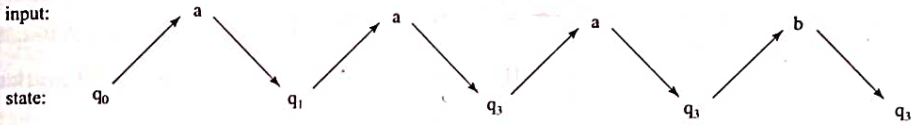

Figure 4.25. *State Sequence Diagram for aaab*

Since we encounter the end of input and q_3 is not the final state, we say that the string *aaab* is rejected by DFA.

EXAMPLE 4.3.11: To design a DFA that contains strings of zeros and ones, with equal number of zeros and ones. No prefix of string should contain two more zeros than ones or two more ones than zeros.

Solution: This is to design a DFA, M for $L(M) = \{010101, 011001, \ldots.\}$.
Consider, $\Sigma = \{0, 1\}$, $Q = \{A, B, C\}$, $q_0 = A$, final state $F = \{A\}$ and δ is given by:

Transition diagram:

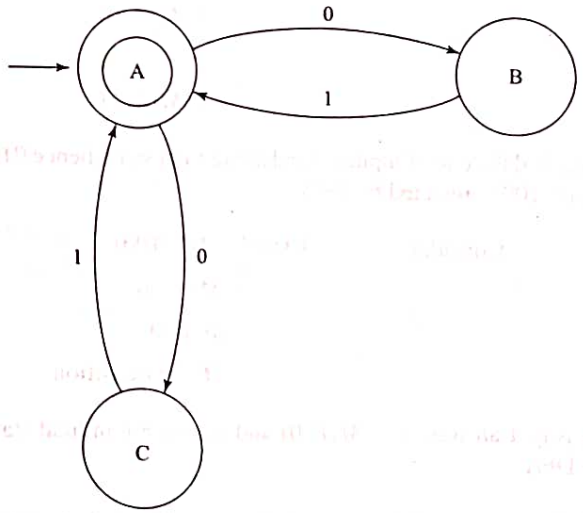

Figure 4.26. *Transition diagram for* $L(M) = \{W \in \Sigma^* | W \text{ contains equal no's of 0's and 1's}\}$

Transition Table:

Q \ Σ	Present Inputs 0	1
→* A	B	C
B	–	A
C	A	–

Table 4.12 *State Transition Table*

DFA action for the input string

i. To show that 011001 is accepted by DFA:

$$\begin{aligned}\text{Consider} \quad (A, 011001) &\vdash M(B, 11001)\\ &\vdash M(A, 1001)\\ &\vdash M(C, 001)\\ &\vdash M(A, 01)\\ &\vdash M(B, 1)\\ &\vdash M(A, e).\end{aligned}$$

Since we reached the end of input (e) and are in final state, hence 011001 is accepted.

ii. To show that 0100 is rejected by DFA:

$$\begin{aligned}\text{Consider} \quad (A, 0100) &\vdash M(B, 100)\\ &\vdash M(A, 00)\\ &\vdash M(B, 0)\\ &\vdash M(\text{No transition}).\end{aligned}$$

Since there is no transition at $\vdash M(B, 0)$ and we are not in final state, hence 0100 is rejected by DFA.

EXAMPLE 4.3.12: To design a DFA, over alphabet $\Sigma = \{0, 1\}$, that contains set of strings of 0's or 1's except those containing substring 110.

Solution: We need to design DFA, M for the regular expression $r = 0^* + 0^*(10)^* + 0^*(10)^*11^*$

i.e., $L(M) = \{00, 11, 0101, 00010010011, \ldots\ldots\}$.

Consider, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_3\}$ q_0 = initial state, $F = \{q_0, q_1, q_2\}$ as final states and δ is given by:

Transition Diagram:

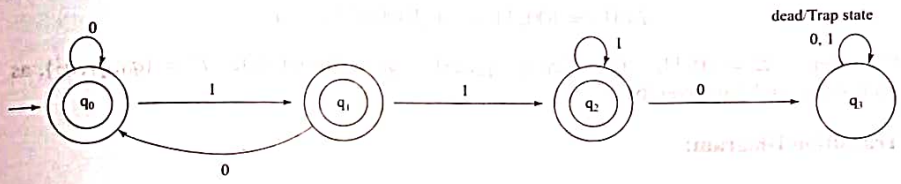

Figure 4.27. *Transition Diagram* $L(M) = \{W \in \Sigma * | W$ *doesn't contain substrings* 110$\}$

Transition Table:

Q \ Σ	Present Inputs 0	1
→* q0	q0	q1
* q1	q0	q2
* q2	q3	q2
q3	q3	q3

Table 4.13 *State Transition Table*

DFA action for the input string

To show that 0101101 is rejected by DFA:

$$\begin{aligned}\text{Consider,} \quad (q_0, 0101101) &\vdash M(q_0, 101101)\\ &\vdash M(q_1, 01101)\\ &\vdash M(q_0, 1101)\\ &\vdash M(q_1, 101)\\ &\vdash M(q_2, 01)\\ &\vdash M(q_3, 1)\\ &\vdash M(q_3, e).\end{aligned}$$

Since we reached the end of input (e) and are not in any of the final states (q_0, q_1, q_2), thus 0101101 is rejected by M.

EXAMPLE 4.3.13: To design a DFA over alphabet $\Sigma = \{0, 1\}$, that contains set of strings of 0's and 1's except those containing substring 001.

Solution: This is to design DFA, M for the language, such that

$$L(M) = \{00, 11, 0101, 1000000, \ldots..\}.$$

Consider, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_3\}$, q_0 = initial state, $F = \{q_0, q_1, q_2\}$, as final states and δ is given by:

Transition Diagram:

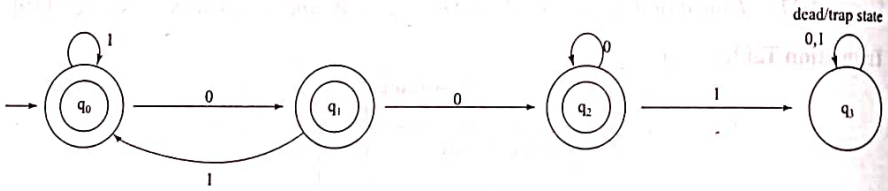

Figure 4.28. *Transition Diagram for $L(M) = \{W \in \Sigma^* | W$ does'nt contain substrings 001$\}$.*

Transition Table:

Q \ Σ	Present Inputs 0	1
→ * q0	q1	q0
* q1	q2	q0
* q2	q2	q3
q3	q3	q3

Table 4.14 *State Transition Table*

DFA action for the input string

To show that 100100 is rejected by DFA:

$$\begin{aligned} \text{Consider } (q_0, 100100) &\vdash M(q_0, 00100) \\ &\vdash M(q_1, 0100) \\ &\vdash M(q_2, 100) \\ &\vdash M(q_3, 00) \\ &\vdash M(q_3, 0) \\ &\vdash M(q_3, e). \end{aligned}$$

Since we reached the end of input (e) and we are not in the final state, hence the string 100100 is rejected by DFA 'M'.

EXAMPLE 4.3.14: To design a DFA over alphabet $\Sigma = \{0, 1\}$, that contains set of strings of 0's and 1's which contain substring

Solution: We are to design DFA, M for the regular expression $r = (0 + 1)^*0101(0 + 1)^*$

$$\text{i.e., } L(M) = \{00101,\ 10101,\ 11010100, \ldots..\}.$$

Consider, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, initial state = q_0, final state = $\{q_4\}$ and δ is given by:

Transition Diagram:

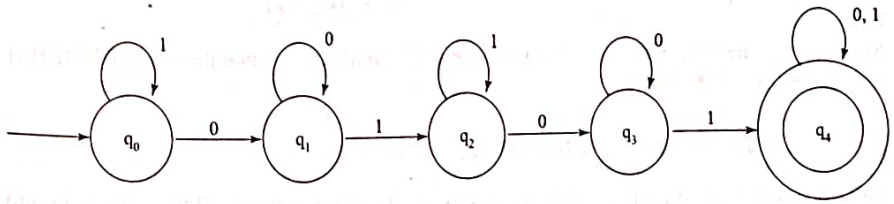

Figure 4.29. *Transition Diagram for $L(M) = \{W \in \Sigma^* | W$ is $(0 + 1)^*0101(0 + 1)^*\}$*

Transition Table:

Q \ Σ	Present Inputs 0	1
→ q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_4
* q_4	q_4	q_4

Table 4.15 *State Transition Table*

DFA action for the input string

To show that string 010010101 is accepted by DFA:

Consider, $(q_0, 010010101) \vdash M(q_1, 10010101)$

$\vdash M(q_2, 0010101)$

$\vdash M(q_3, 010101)$

$\vdash M(q_3, 10101)$

$\vdash M(q_4, 0101)$

$\vdash M(q_4, 101)$

$\vdash M(q_4, 01)$

$\vdash M(q_4, 1)$

$\vdash M(q_4, e)$

Since we reached the end of input (e) and are in the final state, hence the string 010010101 is accepted by DFA 'M'.

EXAMPLE 4.3.15: *Construct a DFA that models the ATM*

Solution: Consider a customer who inserts his bank card into the ATM. It requests him to input his identification number (ID) and assumes that the ID is 234 (made up of 3 digits). The DFA to model this system needs the following states:

q_0 – Initial state, waiting for the first digit in the ID.
q_1 – If the first digit is correct; now waiting for the second digit.
q_2 – If the second digit is correct; now waiting for the third digit.
q_3 – If the third digit is correct; step the process and ID is valid.
q_4 – Trap states that captures all invalid ID's.

Thus, the DFA has $\Sigma = \{0 \cdot \cdot 9\}, Q = \{q_0, q_1, q_2, q_3, q_4\}, q_0 = q_0$ and $F = q_3$.
The transition diagram of DFA is shown in figure 4.30.

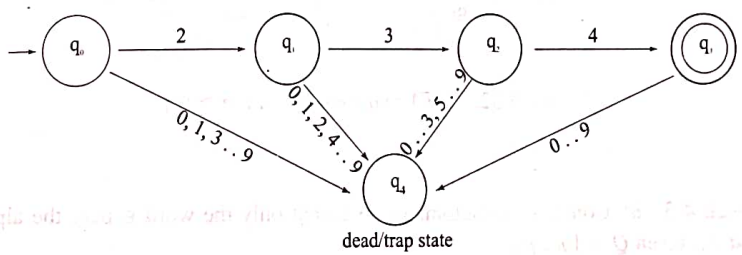

Figure 4.30. *DFA that models the ATM.*

EXAMPLE 4.3.16: Let $L = \{a^i b^j | i \geq 0, j \geq 0\}$ be a language over $\Sigma = \{a, b\}$ Show that there exists a DFA $D = (Q, \Sigma, q_0, \delta, F)$ such that $L(D) = L$.

Solution: Let $Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b\}$ and $F = \{q_0, q_1\}$. The transitions of D are described in the transition diagram shown in figure 4.31.

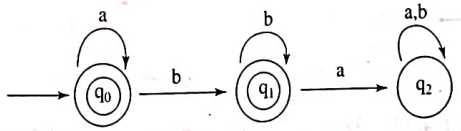

Figure 4.31. *DFA to accept* $L = \{a^i b^j | i \geq 0, j \geq 0\}$

It is clear from the transition diagram that $L(D) = \{a^i b^j | i \geq 0, j \geq 0.\}$

EXAMPLE 4.3.17: Construct a DFA to accept the set of all strings of the form $0^{2n}11$, $n \geq 0$.

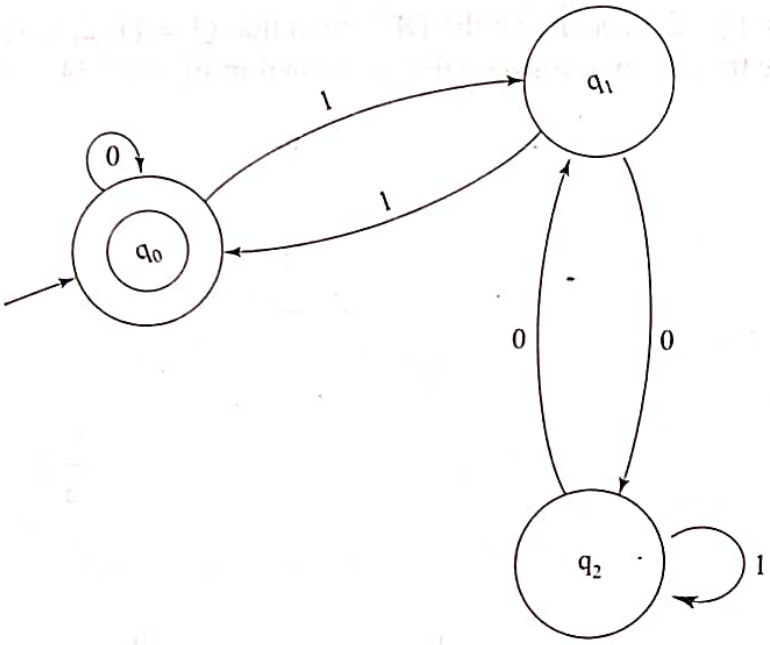

Figure 4.35. *Transition Diagram for* $L(M) = \{11, 110, 1001, \ldots.\}$ *or* $L(M) = \{W \in \Sigma^* | W$ *is divisible by* $3\}$

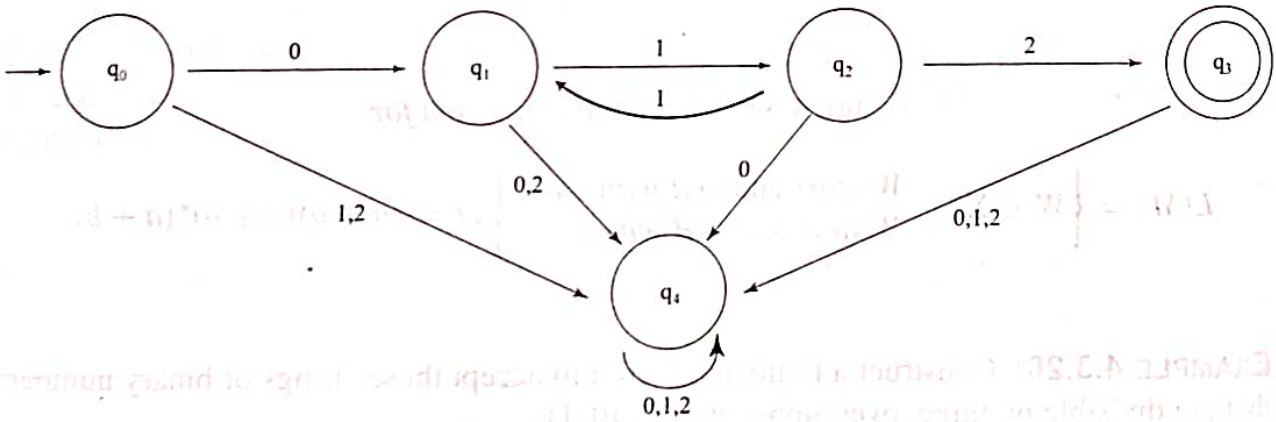

Figure 4.36. *Transition Diagram for* $L(M) = \{W | W \in 0(11)^*12\}$

4.4 Nondeterministic Finite Automata (NFA)

In this section, the concept of nondeterminism is discussed, which has had a great impact on the theory of computation. In the foregoing sections, it is observed, that every step of computation proceeds in a unique way from the preceding step. In other words,