5

Finite State Machines

OBJECTIVES

- Introduction
- Finite State Machines
- Finite State Machine As Models of Physical System
- Equivalence of Finite State Machines
- Finite State Machines as Language Recognizers

5.1 Introduction

A device that receives a set of input signals and produces the corresponding set of output signals is termed as information-processing machine. A machine with a finite number of states is called a finite state machine while a machine with an infinite number of states is called an infinite state machines. In this chapter, we shall restrict our study to finite state machines.

5.2 Finite State Machines

A finite state machine is an abstract model of machine with a primitive internal memory. A finite state machine M is specified by:

1. A finite set of input symbols $I = \{i_1, i_2, \ldots\}$
2. A finite set of internal states $S = \{s_0, s_1, s_2, \ldots\}$
3. A finite set of output symbols $O = \{o_1, o_2, \ldots\}$
4. An initial state s_0 in S
5. A next-state (transition) function $f : S{\times}I \to S$
6. An output function $g : S{\times}I \to O$.

A finite state machine M is denoted by $M = M(I, S, O, s_0, f, g)$.

Example 1. Suppose $I = \{a, b\}$, $S = \{s_0, s_1, s_2\}$, $O = \{x, y, z\}$, Initial state: s_0.

transition function $f : S{\times}I \to S$ defined as

$$f(s_0, a) = s_1, f(s_1, a) = s_2, f(s_2, a) = s_0,$$

$$f(s_0, b) = s_2, f(s_1, b) = s_1, f(s_2, b) = s_1$$

Output function $g : S{\times}I \to O$ defined by

$$g(s_0, a) = x, g(s_1, a) = x, g(s_2, a) = y$$

$$g(s_0, b) = y, g(s_1, b) = z, g(s_2, b) = x$$

Then $M = M(I, S, O, s_0, f, g)$ is a finite state machine.

5.2.1 Transition Table and Transition Diagram

1. Transition (or state) table: The functions f and g can be represented by a table called transition or state table. For example 1, the transition table is

	f		g	
S \ I	a	b	a	b
s_0	s_1	s_2	x	y
s_1	s_2	s_1	x	z
s_2	s_0	s_1	y	x

2. Transition (or state) diagram: A transition of a finite state machine M is a labelled directed graph in which there is a node for each state symbol in S and each node is labelled by a state symbol with which it is associated. The initial state s_0 is indicated by an arrow. Moreover, if $f(s_i, a_j) = s_k$ and $g(s_i, a_j) = O_r$ then there is an arrow (arc) from s_i to s_k which is labelled with the pair (a_j, O_r). We usually put the input symbol a_j near the base of the arrow (near s_i) and the output symbol O_r, near the centre of the arrow. We can also represent it by a/O_i near the centre of the arrow. Therefore, the transition diagram of the finite machine in example 1 is as given below:

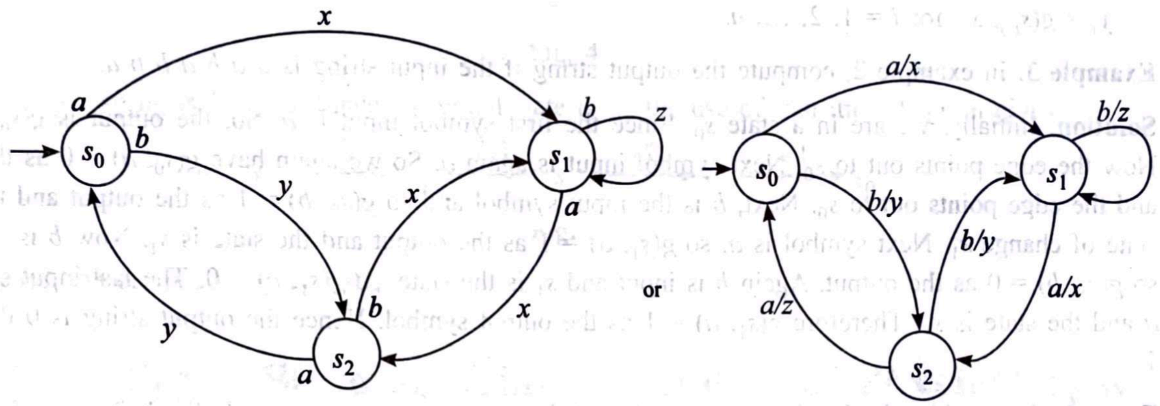


Fig. 1

Example 2. Given $I = \{a, b\}$, $O = \{0, 1\}$ and $S = \{s_0, s_1\}$. Let s_0 be the initial state. Define $f: S \times I \to S$ by

$f(s_0, a) = s_0$, $f(s_0, b) = s_1$, $f(s_1, a) = s_1$, $f(s_1, b) = s_1$ and $g: S \times I \to O$ by

$g(s_0, a) = 0$, $g(s_0, b) = 1$, $g(s_1, a) = 1$, $g(s_1, b) = 0$.

Here $M = M(I, S, O, s_0, f, g)$ is a finite state machine whose transition table is given below:

	f		g	
S \ I	a	b	a	b
s_0	s_0	s_1	0	1
s_1	s_1	s_1	1	0

The transition diagram for M is drawn as below:.

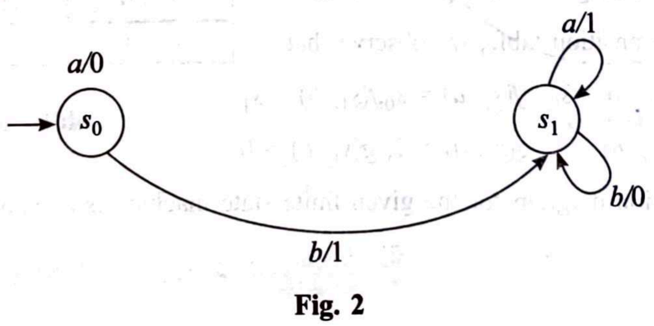


Fig. 2

5.2.2 Input and Output strings

Let, $M = M(I, S, O, s_0, f, g)$ be a finite state machine. An input string for M is a string over I.

The string $y_1, y_2, \ldots y_n$ is the output string for M corresponding to the input string $x_1, x_2, x_3, \ldots x_n$ if there exist states $s_0, s_1, \ldots, s_n \in S$ such that

$$s_i = f(s_{i-1}, x_i) \text{ for } i = 1, 2, \ldots, n,$$

$$y_i = g(s_{i-j}, x_i) \text{ for } i = 1, 2, \ldots, n.$$

Example 3. In example 2, compute the output string if the input string is $a\ a\ b\ a\ b\ b\ a$.

Solution. Initially, we are in a state s_0. Since the first symbol input is a. So, the output is $g(s_0, a) = 0$. Now the edge points out to s_0. Next symbol input is again a. So we again have $g(s_0, a) = 0$ as the output and the edge points out to s_0. Next, b is the input symbol and so $g(s_0, b) = 1$ as the output and there is a state of change s_1. Next symbol is a, so $g(s_1, a) = 1$ as the output and the state is s_1. Now b is input and so $g(s_1, b) = 0$ as the output. Again b is input and s_1 is the state, so $g(s_1, b) = 0$. The last input symbol is a and the state is s_1. Therefore $g(s_1, a) = 1$ as the output symbol. Hence the output string is 0 0 1 1 0 0 1.

Example 4. Consider the FSM of example 1. Let the input string be $a\ b\ a\ a\ b$. We being by taking s_0 as the initial state. Using diagram, we have

$$s_0 \xrightarrow{a/x} s_1 \xrightarrow{b/z} s_1 \xrightarrow{a/x} s_2 \xrightarrow{a/z} s_0 \xrightarrow{b/y} s_2$$

Fig. 3

Hence, the output string is *xzxzy*.

Example 5. Draw the transition diagram of the finite state machine $M\ (I, S, O, s_0, f, g)$, where $I = \{a, b\}$, $S = \{s_0, s_1\}$, $O = \{0, 1\}$ and the transition table is as follows

	f		g	
$S \backslash I$	a	b	a	b
s_0	s_1	s_0	0	1
s_1	s_0	s_1	1	0

Also, find the output string for the input $b\ b\ a\ a$.

Solution. From the transition table, we observe that

$$f(s_0, a) = s_1,\ f(s_0, b) = s_0,\ f(s_1, a) = s_0, f(s_1, b) = s_1,$$

$$g(s_0, a) = 0,\ g(s_0, b) = 1\ \ g(s_1, a) = 1,\ g(s_1, b) = 0.$$

Therefore, the transition diagram for the given finite state machine is as shown below

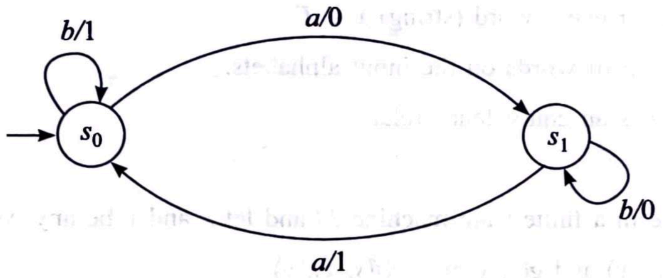


Fig. 4

The input string is *b b a a*. Since the initial state is s_0, the use of transition diagram gives

$$s_0 \xrightarrow{b/1} s_0 \xrightarrow{b/1} s_0 \xrightarrow{a/0} s_1 \xrightarrow{a/1} s_0$$

Fig. 5

Hence, the out put string is 1101.

5.3 Finite State Machine As Models of Physical System

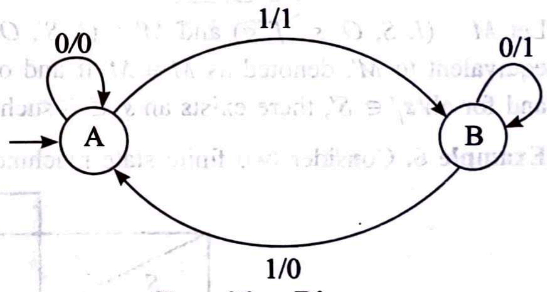
Psychologists, economists, sociologists, and scientists in many disciplines do use finite state machines to model systems they study. Here we will mention an example of finite state machine that can be used to model a physical system.

Consider a problem of designing a modulo 2 counter that receives a sequence of 0s and 1s as input and produces a sequence of 0s and 1s as output such that the output is equal to the modulo 2 sum of the digits in the input sequence.

Let initial state *A* corresponds to the case that the modulo 2 sum of all input digits is 0 and state *B* corresponds to the case that the modulo 2 sum of all input digits is 1. The transition table and the transition diagram for this model can be expressed as below.

	f		*g*	
S \ *I*	0	1	0	1
A	A	B	0	1
B	B	A	1	0

Transition Table



Transition Diagram

Fig. 6

5.4 Equivalent States

Let $M = M\{I, S, O, s_0, f, g\}$ be a finite state machine. Two states $s_i, s_j \in S$ are equivalent, denoted as $s_i \equiv s_j$, if and only if

$g(s_i, x) = g(s_j, x)$ for every word (string) $x \in I^*$,

where I^* denotes the set of words on the input alphabets.

Clearly, the relation $\equiv$ is an equivalence relation.

Properties:

1. Let s be any state in a finite state machine M and let x and y be any words. Then

   $f(s, x\,y) = f(f(s, x), y)$ and $g(s, x\,y) = g(f(s, x), y)$.

2. Let $M = M(I, S, O, s_0, f, g)$ be a finite state machine. If the states s_i and s_j are equivalent, then for any input sequence x, $f(s_i, x) \equiv f(s_j, x)$, that is, if two states are equivalent, then their next states are also equivalent.

3. Let $M = (I, S, O, s_0, f, g)$ be a finite state machine. Then for some positive integer k, s_j is said to be k-equivalent to s_j, if and only if

   $g(s_i, x) = g(s_j, s)$ for all input sequence x such that $|x| \leq k$, $s \in S$.

4. If s_i and s_j are k-equivalent, and s_j and s_r are also k-equivalent, then s_i and s_r are also k-equivalent. We can also define an equivalence relation on the set of all states such that two states are related if they are k-equivalent. This relation generates a partition, say π_k, on the set of all states.

Theorem. Two states are in the same block in π_k if and only if they are in the same block in π_{k-1} and for any input letter, their successors are in the same block in π_{k-1}.

Proof. Let s_i and s_j are two k-equivalent states. Then they must have the same output and for any input letter, their successors are (k–1)-equivalent. Now by definition s_i and s_j are also (k–1)-equivalent. So we may conclude that two states are k-equivalent if and if only if they are (k–1)-equivalent and, for any input letter, their successors are also (k–1)-equivalent. Hence the theorem follows.

5.5 Equivalence of Finite State Machines

Let $M = (I, S, O, s_0, f, g)$ and $M' = (I, S', O, s_0', f', g')$ be to finite state machines. Then M is said to be equivalent to M', denoted as $M \equiv M'$ if and only if for all $s_i \in s$, there exists an $s_j' \in S'$ such that $s_i \equiv s_j'$ and for all $s_j' \in S'$, there exists an $s_i \in S$ such that $s_i \equiv s_j'$. Clearly, the relation $\equiv$ is an equivalence relation.

Example 6. Consider two finite state machines whose transition tables are as follows

S \ I	f: 0	f: 1	g: 0	g: 1
s_0	s_5	s_3	0	1
s_1	s_1	s_4	0	0
s_2	s_1	s_3	0	0
s_3	s_1	s_2	0	0
s_4	s_5	s_2	0	1
s_5	s_4	s_1	0	1

$M(I, S, O, s_0, f, g)$

and

S' \ I	f: 0	f: 1	g: 0	g: 1
s_0'	s_3'	s_2'	0	1
s_1'	s_1'	s_0'	0	0
s_2'	s_1'	s_2'	0	0
s_3'	s_0'	s_1'	0	1

We observe that $s_0' \equiv s_0$; $s_0' \equiv s_4$; $s_1' \equiv s_1$; $s_2' \equiv s_2$; $s_2' \equiv s_3$; $s_3' \equiv s_5$. Also the output functions g and g' are same for the indicated correspondence. Hence $M \equiv M'$.

5.6 Finite State Machines as Language Recognizers

A language is called finite state language if there is a finite state machine that accepts every sentence given in the language. Any given finite state machine defines a finite state language. A given language not necessarily a finite state language.

Example 7. Show that the language $L = \{a^k b^k \mid k \geq 1\}$ is not a finite state language.

Solution. If possible let us suppose that a finite state machine exists that accepts the sentences in L. Let this machine has N states. Clearly, the machine accepts the sentence $a^N b^N$. Starting from the initial state, the machine will visit N states after receiving the N a's in the input sequence as shown in figure 1(a), where s_{j0} is the initial state and $s_{j1}, s_{j2}, \ldots, s_{jN}$ are the states the machine is in after receiving the sequence a^N. Further, s_{j2N} is the state machine is in after receiving the sequence $a^N b^N$. Clearly, s_{j2N} is an accepting state. As per pigeonhole principle, among the $N + 1$ states $s_{j0}, s_{j1}, s_{j2}, \ldots, s_{jN}$, two of them are identical.

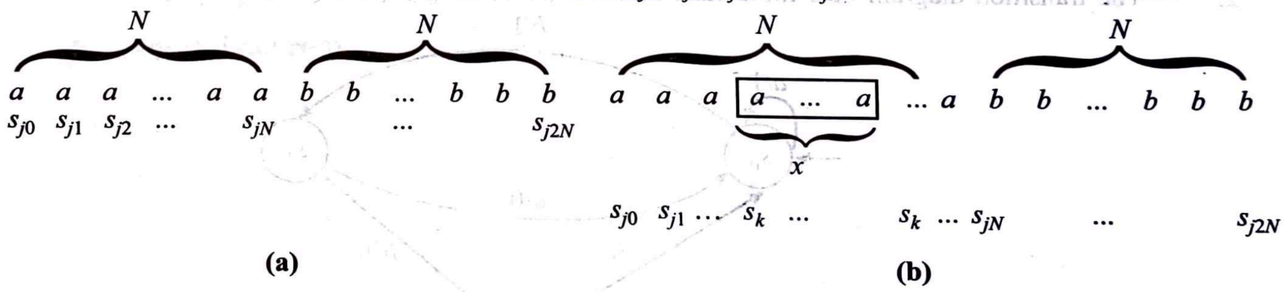


Fig. 7

Let, the machine visits state s_k twice as shown in figure 1(b) and there are x a's between the first and the second visit state s_k. Then the sequence $a^{N-x}b^N$ will also be accepted by the finite state machine which is a contradiction as it is not in L. Hence, we can conclude that language L is not a finite state language.

Example 8. Show that the language $L = \{a^k \mid k = i^2, i \geq 1\}$ is not a finite state language.

Solution. Suppose that there is a finite state machine that accepts language L. Let N denote the number of states in the machine and i be an integer that is sufficiently large such that $(i + 1)^2 - i^2 > N$.

Consider the case shown in figure 2. Since between the i^2th a and the $(i + 1)^2$th a, the finite state machine will visit a certain state s_k more than once, so deletion of the a's between these two visits will yield a sequence that will also be accepted by the machine, which is a contradiction as this sequence is not a sentence in the Language because it contains more than i^2 but less then $(i+1)^2$ a's. Hence L is not a finite state language.

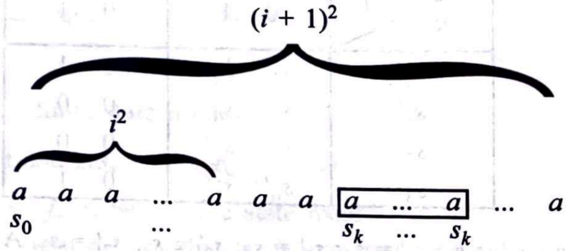


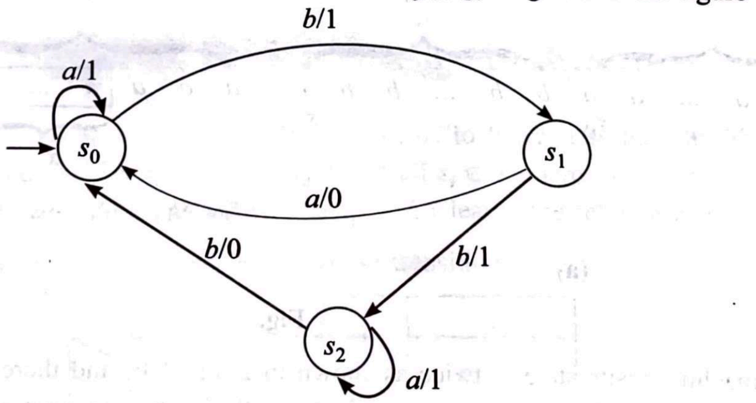
Fig. 8

EXERCISE 5

1. Draw the transition diagram of the finite state machine (I, O, S, s_0, f, g), where $I = \{a, b\}$; $O = \{0, 1\}$, $S = \{s_0, s_1, s_2, s_3\}$ and the transition table is

	f		g	
S \ I	a	b	a	b
s_0	s_1	s_2	0	0
s_1	s_0	s_2	1	0
s_2	s_3	s_0	0	1
s_3	s_1	s_3	0	0

2. The transition diagram of a finite state machine (I, O, S, s_0, f, g) is given in the figure



Determine I, O, S, s_0, and transition table for f and g.

□□□