Modeling Computation

Computers can perform many tasks. Given a task, two questions arise.

The first is: Can it be carried out using a computer?

The second is: How can the task be carried out?

Models of computation are used to help answer these questions.

Types of structures used in models of computation are,

- Grammars,
- Finite-state machines,
- Turing machines.

Grammars are used to generate the words of a language and to determine whether a word is in a language. Formal languages, which are generated by grammars, provide models for both natural languages, such as English, and for programming languages, such as Pascal, Fortran, Prolog, C, and Java. In particular, grammars are extremely important in the construction and theory of compilers.

Various types of finite-state machines are used in modeling. All finite-state machines have a set of states, including a starting state, an input alphabet (a finite set of elements), and a transition function that assigns a next state to every pair of a state and an input. The states of a finite-state machine give it limited memory capabilities. Some finite-state machines produce an output symbol for each transition; these machines can be used to model many kinds of machines, including vending machines, delay machines, binary adders, and language recognizers. There are finite-state machines that have no output but do have final states. Such machines are extensively used in language recognition. The strings that are recognized are those that take the starting state to a final state. The concepts of grammars and finite-state machines can be tied together.

Turing machines can be used to recognize sets and to compute number-theoretic functions.

Finite-State Machines with Output

Finite-state machines can be used to model a vending machine. A vending machine accepts nickels (5 cents), dimes (10 cents), and quarters (25 cents). When a total of 30 cents or more has been deposited, the machine immediately returns the amount more than 30 cents. When 30 cents has been deposited and any excess refunded, the customer can push an orange button and receive an orange juice or push a red button and receive an apple juice. We can describe how the machine works by specifying its states, how it changes states when input is received, and the output that is produced for every combination of input and current state.

The machine can be in any of seven different states s_i , i = 0, 1, 2, ... , 6, where s_i is the state where the machine has collected 5i cents. The machine starts in state s_0, with 0 cents received. The possible inputs are 5 cents, 10 cents, 25 cents, the orange button (O), and the red button (R). The possible outputs are nothing (n), 5 cents, 10 cents, 15 cents, 20 cents, 25 cents, an orange juice, and an apple juice.

We illustrate how this model of the machine works with this example. Suppose that a student puts in a dime followed by a quarter, receives 5 cents back, and then pushes the orange button for an orange juice. The machine starts in state s_0. The first input is 10 cents, which changes the state of the machine to s_2 and gives no output. The second input is 25 cents. This changes the state from s_1 to s_6, and gives 5 cents as output. The next input is the orange button, which changes the state from s_6 back to s_0 (because the machine returns to the start state) and gives an orange juice as its output.

We can display all the state changes and output of this machine in a table. To do this we need to specify for each combination of state and input the next state and the output obtained. Table 1 shows the transitions and outputs for each pair of a state and an input.

Another way to show the actions of a machine is to use a directed graph with labeled edges, where each state is represented by a circle, edges represent the transitions, and edges are labeled with the input and the output for that transition. Figure 1 shows such a directed graph for the vending machine.

TABLE 1 State Table for a Vending Machine.

	Next State					Output				
	Input					*Input*				
State	**5**	**10**	**25**	***O***	***R***	**5**	**10**	**25**	***O***	***R***
s_0	s_1	s_2	s_5	s_0	s_0	n	n	n	n	n
s_1	s_2	s_3	s_6	s_1	s_1	n	n	n	n	n
s_2	s_3	s_4	s_6	s_2	s_2	n	n	5	n	n
s_3	s_4	s_5	s_6	s_3	s_3	n	n	10	n	n
s_4	s_5	s_6	s_6	s_4	s_4	n	n	15	n	n
s_5	s_6	s_6	s_6	s_5	s_5	n	5	20	n	n
s_6	s_6	s_6	s_6	s_0	s_0	5	10	25	OJ	AJ

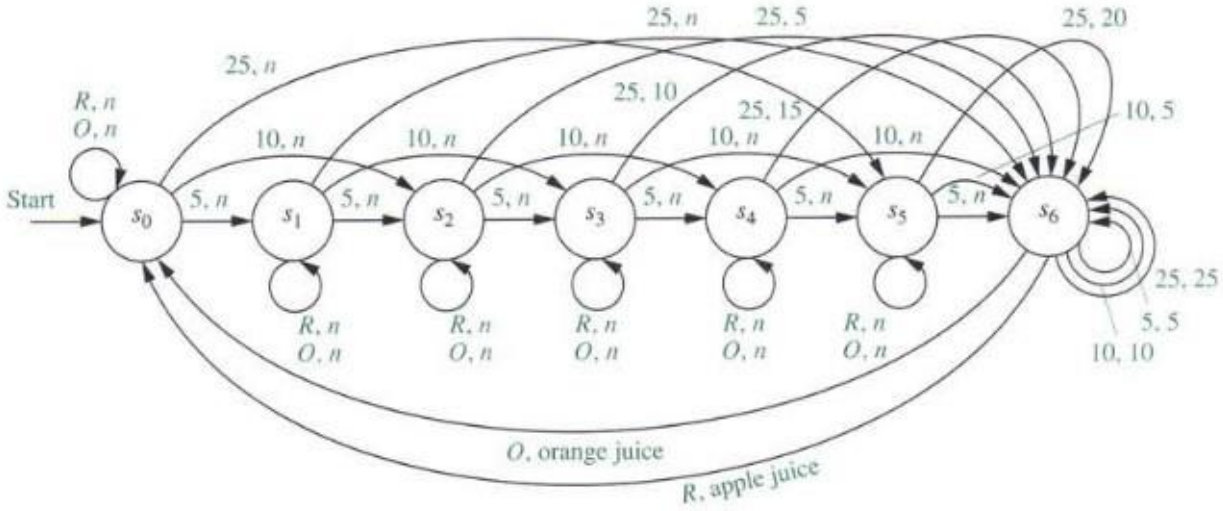


FIGURE 1 A Vending Machine.

Finite-State Machines with No Output (Finite-State Automata)

One of the most important applications of finite-state machines is in language recognition. This application plays a fundamental role in the design and construction of compilers for programming languages. Instead of producing output, these machines have final states. A string is recognized if and only if it takes the starting state to one of these final states.

Set of Strings

Before discussing finite-state machines with no output, we will introduce some important background material on sets of strings.

Definition - Alphabet (Vocabulary)

A vocabulary (or alphabet) V is a finite, nonempty set of elements called symbols. A word (or sentence) over V is a string of finite length of elements of V. The empty string or null string, denoted by λ, is the string containing no symbols. The set of all words over V is denoted by V*. A language over V is a subset of V*.

Examples of Alphabets:

1. Latin Alphabet (lower case) = {a, b, c, …, z}
2. Latin Alphabet (upper case) = {A, B, C, …, Z}
3. Decimal Alphabet = {0, 1, 2, 3, …, 9}
4. Binary Alphabet = {0, 1}

If an alphabet is defined by the set A, then a string is any concatenated sequence of symbols/elements from A.

From A, a set A* can be defined. A* is the set of all possible finite sequences (strings) from A including the empty string, λ.

From A, a set A^+ can be defined. A^+ is the set of all possible finite sequences (strings) from A without the empty string, λ.

1. Binary Alphabet A = {0, 1}

A* = { λ , 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, …}

A^+ = {0, 1, 00, 01, 10, 11, 000, 001, 010, 011, …}

2. A = {ab, bc}

A* = { λ , ab, bc, abab, ababab, abbc, bcab, bcbcab, …} abc ∉ A*

Definition 1

Suppose that A and B are subsets of V*, where V is a vocabulary. The *concatenation* of A and B, denoted by AB, is the set of all strings of the form xy, where x is a string in A and y is a string in B.

Example 1:

Let A = {0, 11} and B = {1, 10, 110}. Find AB and BA.

Solution: The set AB contains every concatenation of a string in A and a string in B. Hence, AB = {01, 010, 0110, 111, 1110, 11110}. The set BA contains every concatenation of a string in B and a string in A. Hence, BA = {10,111,100,1011,1100,11011}.

Note that it is not necessarily the case that AB = BA when A and B are subsets of V*, where V is an alphabet, as Example 1 illustrates.

From the definition of the concatenation of two sets of strings, we can define A^n (how many different strings can be made of length n), for n = 0, 1, 2,… This is done recursively by specifying that

$A^0 = \{\lambda\}$,

$A^{n+1} = A^nA$ for n = 0, 1, 2, ….

Example 2:

Let A = {1, 00}. Find A^n for n = 0, 1, 2, and 3.

Solution: We have $A^0 = \{\lambda\}$ and $A^1 = A^0A = \{\lambda\}A = (1, 00)$. To find A^2 we take concatenations of pairs of elements of A. This gives $A^2 = \{11, 100, 001, 0000\}$. To find A^3 we take concatenations of elements in A^2 and A; this gives $A^3 = \{111, 1100, 1001, 10000, 0011, 00100, 00001, 000000\}$.

Example 2.1:

Let A = {a, b}. Find A^n for n = 0, 1, 2, and 3.

$A^0 = \{\lambda\}$, $A^1 = \{a, b\}$, $A^2 = \{aa, ab, ba, bb\}$ and $A^3 = \{aaa, aab, aba, abb, bbb, \ldots\}$

Definition 2

Suppose that A is a subset of V*. Then the *Kleene closure* of A, denoted by A*, is the set consisting of concatenations of arbitrarily many strings from A. That is, $A^* = \bigcup_{k=0}^{\infty} A^k$.

i:e: A* = {set of all strings of all lengths possible }

Example 3:

What are the Kleene closures of the sets A = {0}, B = {0, 1}, and C = {11}?

Solution: The Kleene closure of A is the concatenation of the string 0 with itself an arbitrary finite number of times. Hence, $A^* = \{0^n \mid n = 0, 1, 2, ...\}$. The Kleene closure of B is the concatenation of an arbitrary number of strings, where each string is either 0 or 1. This is the set of all strings over the alphabet V = {0, 1}. That is, B* = V*. Finally, the Kleene closure of C is the concatenation of the string 11 with itself an arbitrary number of times. Hence, C* is the set of strings consisting of an even number of 1s. That is, $C^* = (1^{2n} \mid n = 0, 1, 2, ...)$.

Finite-State Automata

We will now give a definition of a finite-state machine with no output. Such machines are also called finite-state automata, and that is the terminology we will use for them here. (Note: The singular of automata is automaton.) These machines do not produce output, but they do have a set of final states. As we will see, they recognize strings that take the starting state to a final state.

Definition 3

A finite-state automaton M = (S, I, *f*, s_0, F) consists of a finite set S of *states*, a finite *input alphabet* I, a *transition function f* that assigns a next state to every pair of state and input (so that $f : S \times I \rightarrow S$), an *initial or start state* s_0, and a subset F of S consisting of *final* (or *accepting states*).

(To denote the transition function, different symbols are used such as N, T instead of *f*. Then the transition function is defined as $N : S \times I \rightarrow S$ or $T : S \times I \rightarrow S$.)

We can represent finite-state automata using either state tables or state diagrams. Final states are indicated in state diagrams by using double circles.

Example 4:

Construct the state diagram for the finite-state automaton M = (S, I, *f*, s_0, F), where S = {s_0, s_1, s_2, s_3}, I = {0, 1}, F = {s_0, s_3}, and the transition function *f* is given in Table 1.

Solution: The state diagram is shown in Figure 1. Note that because both the inputs 0 and 1 take s_2 to s_0, we write 0,1 over the edge from s_2 to s_0.

TABLE 1

	f	
	Input	
State	**0**	**1**
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1

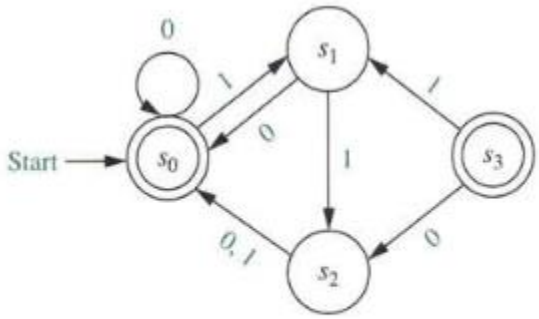


FIGURE 1 The State Diagram for a Finite-State Automaton.

Language Recognition by Finite-State Machines

Next, we define some terms that are used when studying the recognition by finite-state automata of certain sets of strings.

Definition 4

A string x is said to be *recognized* or *accepted* by the machine M = (S, I, *f*, s_0, F) if it takes the initial state s_0 to a final state, that is, f(s_0, x) is a state in F. The *language recognized* or *accepted* by the machine M, denoted by L(M), is the set of all strings that are recognized by M. Two finite-state automata are called equivalent if they recognize the same language.

Example 5:

Determine the languages recognized by the finite-state automata M1, M2, and M3 in Figure 2.

Solution: The only final state of M1 is s_0. The strings that take s_0 to itself are those consisting of zero or more consecutive 1s. Hence, L(M1) = {1^n | n = 0, 1, 2, ...}.

The only final state of M2 is s_2. The only strings that take s_0 to s_2 are 1 and 01. Hence, L(M2) = {1, 01}.

The final states of M3 are s_0 and s_3. The only strings that take s_0 to itself are λ, 0, 00, 000, …, that is, any string of zero or more consecutive 0s. The only strings that take s_0 to s_3 are a string of zero or more consecutive 0s, followed by 10, followed by any string. Hence L(M3) = {0^n , 0^n10x | n = 0, 1, 2, ... and x is any string}.

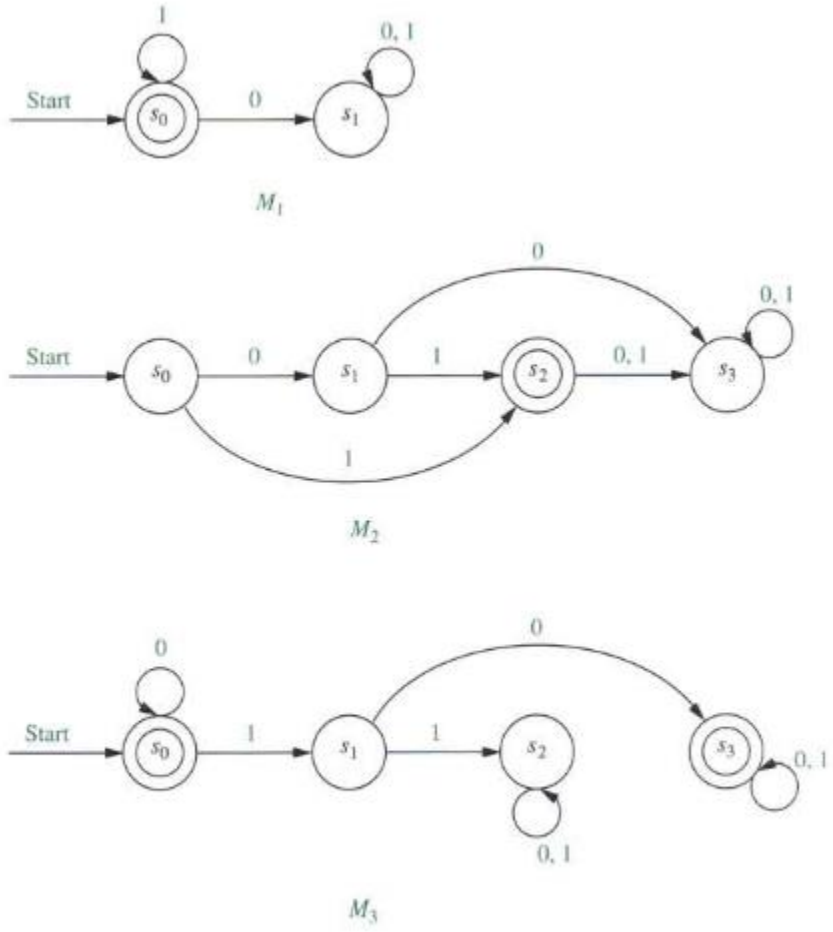


FIGURE 2 Some Finite-State Automata.

Equivalent Finite-State Automata

In definition 4 we specified that two finite-state automata are equivalent if they recognize the same language. Example 8 provides an example of two equivalent deterministic finite-state machines.

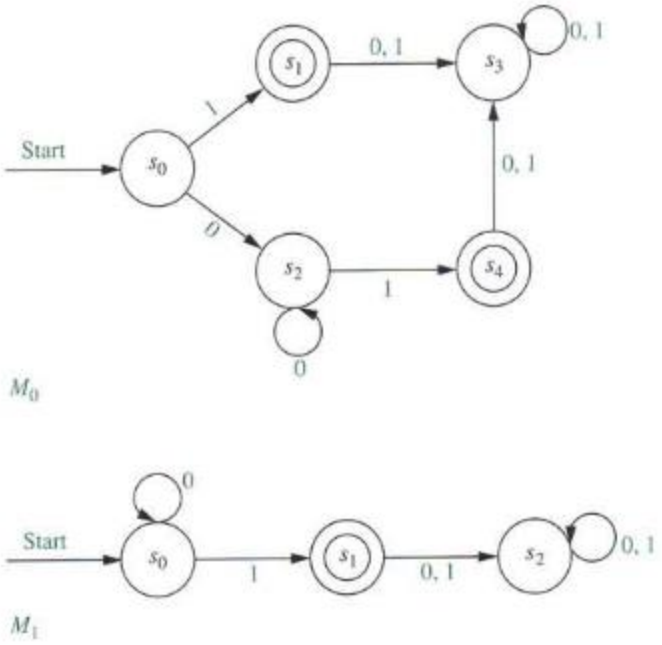


FIGURE 5 M_0 **and** M_1 **Are Equivalent Finite-State Automata.**

Example 8:

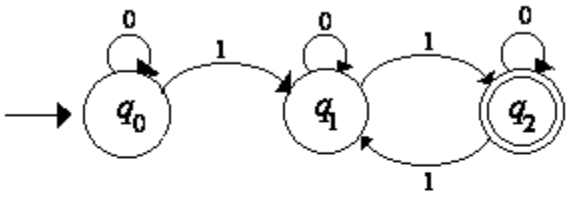
Show that the two finite-state automata M0 and M1 shown in Figure 5 are equivalent.

Solution: For a string x to be recognized by M0, x must take us from s_0 to the final state s_1 or the final state s_4. The only string that takes us from s_0 to s_1 is the string 1. The strings that take us from s_0 to s_4 are those strings that begin with a 0, which takes us from s_0 to s_2, followed by zero or more additional 0s, which keep the machine in state s_2, followed by a 1, which takes us from state s_2 to the final state s_4. All other strings take us from s_0 to a state that is not final. We conclude that L(M0) is the set of strings of zero or more 0 bits followed by a final 1.

For a string x to be recognized by M1, x must take us from s_0 to the final state s_1. So, for x to be recognized, it must begin with some number of 0s, which leave us in state s_0, followed by a 1, which takes us to the final state s_1. A string of all zeros is not recognized because it leaves us in state s_0, which is not final. All strings that contain a 0 after 1 are not recognized because they take us to state s_2, which is not final. It follows that L(M1) is the same as L(M0). We conclude that M0 and M1 are equivalent.

Exercises

1.Consider the finite-state automaton ***M*** defined by the state diagram shown below:



a. What are the states (***Q***) of ***M***?

b. What is the alphabet (Σ) of ***M***?

c. What is the start state of ***M***?

d. What are the accept states (***F***) of ***M***?

e. Find $T(q_2, 1)$.

f. Create the transition table of ***M***.

Solution:

1.a. $Q = \{q_0, q_1, q_2\}$

b. $\Sigma = \{0, 1\}$ (i.e., **0** and **1** are the input symbols)

c. The start state of ***M*** is q_0.

d. $F = \{q_2\}$

e. $T(q_2, 1) = q_1$ (as there is an arrow from q_2 to q_1 labeled 1.)

f. The transition table of ***M*** is :

State/Input Symbol	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_1

2. Consider the finite-state automaton A defined by the following transition table.

States	Input Symbols: a	b	c
→ q_0	q_0	q_1	q_0
◎ q_1	q_0	q_1	q_2
◎ q_2	q_0	q_2	q_2

a. What are the states (Q) of A?

b. What is Σ (the alphabet of A)?

c. What is the start state of A?

d. What are the accept states (F) of A?

e. Find $T(q_2, b)$.

f. Draw a state diagram for A.

Solution:

2. a. $Q = \{q_0, q_1, q_2\}$

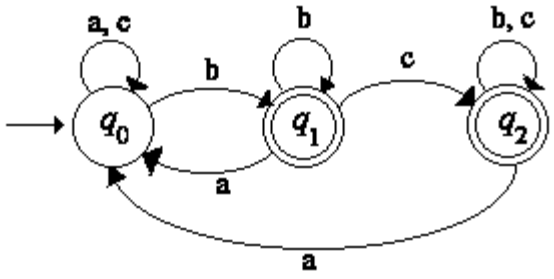
b. $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$

c. The start state of $A = \{q_0\}$

d. $F = \{q_1, q_2\}$

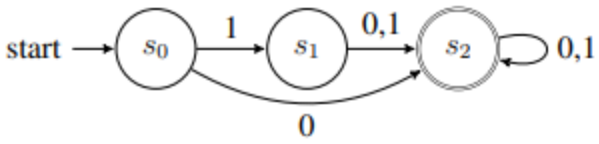
e. $T(q_2, \mathbf{b}) = q_2$

f. The state diagram for A can be drawn as follows:



3.

Find the language recognized by the given deterministic finite-state automaton.

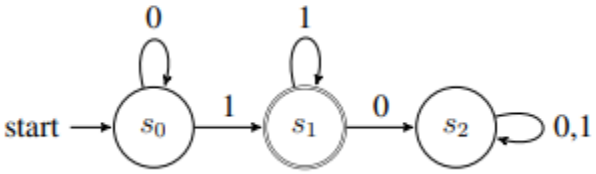


Method 1 - L(M) = {0x , 10x, 11x | n = 0, 1, 2, ... and x is any string}

Method 2 - The language is $\{0, 10, 11\}\{0, 1\}^*$

4.

Find the language recognized by the given deterministic finite-state automaton.



Method 1 - L(M) = $\{0^n11^n \mid n = 0, 1, 2, ...\}$

Method 2 - The language is $\{0\}^*\{1\}\{1\}^*$