9 Turing Machines and Linear Bounded Automata

In the early 1930s, mathematicians were trying to define effective computation. Alan Turing in 1936, Alanzo Church in 1933, S.C. Kleene in 1935, Schonfinkel in 1965 gave various models using the concept of Turing machines, λ-calculus, combinatory logic, post-systems and μ-recursive functions. It is interesting to note that these were formulated much before the electro-mechanical/electronic computers were devised. Although these formalisms, describing effective computations, are dissimilar, they turn to be equivalent.

Among these formalisms, the Turing's formulation is accepted as a model of algorithm or computation. The Church–Turing thesis states that any algorithmic procedure that can be carried out by human beings/computer can be carried out by a Turing machine. It has been universally accepted by computer scientists that the Turing machine provides an ideal theoretical model of a computer.

Turing machines are useful in several ways. As an automaton, the Turing machine is the most general model. It accepts type-0 languages. It can also be used for computing functions. It turns out to be a mathematical model of partial recursive functions. Turing machines are also used for determining the undecidability of certain languages and measuring the space and time complexity of problems. These are the topics of discussion in this chapter and some of the subsequent chapters.

For formalizing computability, Turing assumed that, while computing, a person writes symbols on a one-dimensional paper (instead of a two-dimensional paper as is usually done) which can be viewed as a tape divided into cells.

One scans the cells one at a time and usually performs one of the three simple operations, namely (i) writing a new symbol in the cell being currently scanned, (ii) moving to the cell left of the present cell, and (iii) moving to the cell right of the present cell. With these observations in mind, Turing proposed his 'computing machine.'

9.1 TURING MACHINE MODEL

The Turing machine can be thought of as finite control connected to a R/W (read/write) head. It has one tape which is divided into a number of cells. The block diagram of the basic model for the Turing machine is given in Fig. 9.1.

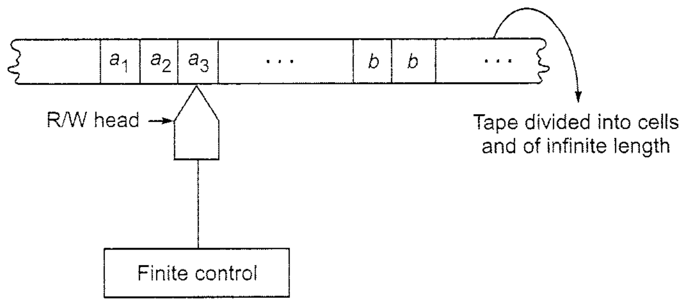

Fig. 9.1 Turing machine model.

Each cell can store only one symbol. The input to and the output from the finite state automaton are effected by the R/W head which can examine one cell at a time. In one move, the machine examines the present symbol under the R/W head on the tape and the present state of an automaton to determine

(i) a new symbol to be written on the tape in the cell under the R/W head,
(ii) a motion of the R/W head along the tape: either the head moves one cell left (L), or one cell right (R),
(iii) the next state of the automaton, and
(iv) whether to halt or not.

The above model can be rigorously defined as follows:

Definition 9.1 A Turing machine M is a 7-tuple, namely $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$, where

1. Q is a finite nonempty set of states,
2. Γ is a finite nonempty set of tape symbols,
3. $b \in \Gamma$ is the blank,
4. Σ is a nonempty set of input symbols and is a subset of Γ and $b \notin \Sigma$,
5. δ is the transition function mapping (q, x) onto (q', y, D) where D denotes the direction of movement of R/W head: $D = L$ or R according as the movement is to the left or right.
6. $q_0 \in Q$ is the initial state, and
7. $F \subseteq Q$ is the set of final states.

Notes: (1) The acceptability of a string is decided by the reachability from the initial state to some final state. So the final states are also called the accepting states.

(2) δ may not be defined for some elements of $Q \times \Gamma$.

9.2 REPRESENTATION OF TURING MACHINES

We can describe a Turing machine employing (i) instantaneous descriptions using move-relations, (ii) transition table, and (iii) transition diagram (transition graph).

9.2.1 REPRESENTATION BY INSTANTANEOUS DESCRIPTIONS

'Snapshots' of a Turing machine in action can be used to describe a Turing machine. These give 'instantaneous descriptions' of a Turing machine. We have defined instantaneous descriptions of a pda in terms of the current state, the input string to be processed, and the topmost symbol of the pushdown store. But the input string to be processed is not sufficient to be defined as the ID of a Turing machine, for the R/W head can move to the left as well. So an ID of a Turing machine is defined in terms of the entire input string and the current state.

Definition 9.2 An ID of a Turing machine M is a string $\alpha\beta\gamma$, where β is the present state of M, the entire input string is split as $\alpha\gamma$, the first symbol of γ is the current symbol a under the R/W head and γ has all the subsequent symbols of the input string, and the string α is the substring of the input string formed by all the symbols to the left of a.

EXAMPLE 9.1

A snapshot of Turing machine is shown in Fig. 9.2. Obtain the instantaneous description.

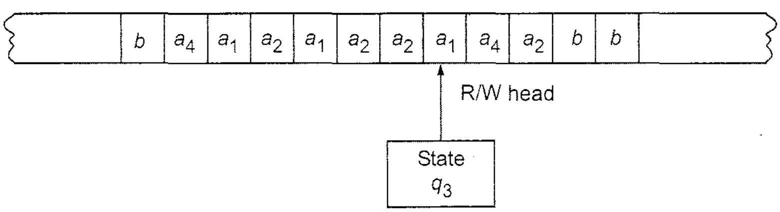

Fig. 9.2 A snapshot of Turing machine.

Solution

The present symbol under the R/W head is a_1. The present state is q_3. So a_1 is written to the right of q_3. The nonblank symbols to the left of a_1 form the string $a_4a_1a_2a_1a_2a_2$, which is written to the left of q_3. The sequence of nonblank symbols to the right of a_1 is a_4a_2. Thus the ID is as given in Fig. 9.3.

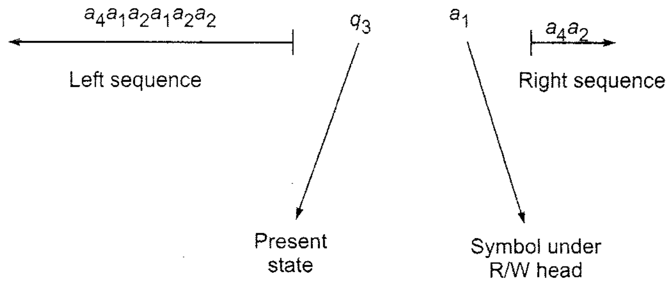

Fig. 9.3 Representation of ID.

Notes: (1) For constructing the ID, we simply insert the current state in the input string to the left of the symbol under the R/W head.

(2) We observe that the blank symbol may occur as part of the left or right substring.

Moves in a TM

As in the case of pushdown automata, $\delta(q, x)$ induces a change in ID of the Turing machine. We call this change in ID a move.

Suppose $\delta(q, x_i) = (p, y, L)$. The input string to be processed is $x_1x_2 \ldots x_n$, and the present symbol under the R/W head is x_i. So the ID before processing x_i is

$$x_1x_2 \ldots x_{i-1}qx_i \ldots x_n$$

After processing x_i, the resulting ID is

$$x_1 \ldots x_{i-2}\, px_{i-1}yx_{i+1} \ldots x_n$$

This change of ID is represented by

$$x_1x_2 \ldots x_{i-1}\, q\, x_i \ldots x_n \vdash x_i \ldots x_{i-2}\, px_{i-1}\, yx_{i+1} \ldots x_n$$

If $i = 1$, the resulting ID is $p\, y\, x_2x_3 \ldots x_n$.

If $\delta(q, x_i) = (p, y, R)$, then the change of ID is represented by

$$x_1x_2 \ldots x_{i-1}q\, x_i \ldots x_n \vdash x_1x_2 \ldots x_{i-1}\, y\, px_{i+1} \ldots x_n$$

If $i = n$, the resulting ID is $x_1x_2 \ldots x_{n-1}\, y\, p\, b$.

We can denote an ID by I_j for some j. $I_j \vdash I_k$ defines a relation among IDs. So the symbol $\overset{*}{\vdash}$ denotes the reflexive–transitive closure of the relation $\vdash$. In particular, $I_j \overset{*}{\vdash} I_j$. Also, if $I_1 \overset{*}{\vdash} I_n$, then we can split this as $I_1 \vdash I_2 \vdash I_3 \vdash \ldots \vdash I_n$ for some IDs, $I_2, \ldots, I_{n-1}$.

Note: The description of moves by IDs is very much useful to represent the processing of input strings.

9.2.2 REPRESENTATION BY TRANSITION TABLE

We give the definition of δ in the form of a table called the transition table. If $\delta(q, a) = (\gamma, \alpha, \beta)$, we write $\alpha\beta\gamma$ under the α-column and in the q-row. So if we get $\alpha\beta\gamma$ in the table, it means that α is written in the current cell, β gives the movement of the head (L or R) and γ denotes the new state into which the Turing machine enters.

Consider, for example, a Turing machine with five states $q_1, \ldots, q_5$, where q_1 is the initial state and q_5 is the (only) final state. The tape symbols are 0, 1 and b. The transition table given in Table 9.1 describes δ.

TABLE 9.1 Transition Table of a Turing Machine

Present state	*Tape symbol*		
	b	0	1
$\to q_1$	$1Lq_2$	$0Rq_1$	
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3		bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
q_5	$0Lq_2$		

As in Chapter 3, the initial state is marked with $\to$ and the final state with $\bigcirc$.

EXAMPLE 9.2

Consider the TM description given in Table 9.1. Draw the computation sequence of the input string 00.

Solution

We describe the computation sequence in terms of the contents of the tape and the current state. If the string in the tape is $a_1a_2 \ldots a_j a_{j+1} \ldots a_m$ and the TM in state q is to read a_{j+1}, then we write $a_1a_2 \ldots a_j q\, a_{j+1} \ldots a_m$.

For the input string 00b, we get the following sequence:

$$
\begin{aligned}
& q_100b \vdash 0q_10b \vdash 00q_1b \vdash 0q_201 \vdash q_2001 \\
\vdash\ & q_2b001 \vdash bq_3001 \vdash bbq_401 \vdash bb_0q41 \vdash bb_01q_4b \\
\vdash\ & bb010q_5 \vdash bb01q_200 \vdash bb0q_2100 \vdash bbq_20100 \\
\vdash\ & bq_2b0100 \vdash bbq_30100 \vdash bbbq_4100 \vdash bbb_1q_400 \\
\vdash\ & bbb10q_40 \vdash bbb100q_4b \vdash bbb1000q_5b \\
\vdash\ & bbb100q_200 \vdash bbb10q_2000 \vdash bbb1q_20000 \\
\vdash\ & bbbq_210000 \vdash bbq_2b10000 \vdash bbbq_310000 \vdash bbbbq_50000
\end{aligned}
$$

9.2.3 REPRESENTATION BY TRANSITION DIAGRAM

We can use the transition systems introduced in Chapter 3 to represent Turing machines. The states are represented by vertices. Directed edges are used to represent transition of states. The labels are triples of the form (α, β, γ), where $\alpha, \beta \in \Gamma$ and $\gamma \in \{L, R\}$. When there is a directed edge from q_i to q_j with label (α, β, γ), it means that

$$\delta(q_i, \alpha) = (q_j, \beta, \gamma)$$

During the processing of an input string, suppose the Turing machine enters q_i and the R/W head scans the (present) symbol α. As a result, the symbol β is written in the cell under the R/W head. The R/W head moves to the left or to the right, depending on γ, and the new state is q_j.

Every edge in the transition system can be represented by a 5-tuple $(q_i, \alpha, \beta, \gamma, q_j)$. So each Turing machine can be described by the sequence of 5-tuples representing all the directed edges. The initial state is indicated by $\rightarrow$ and any final state is marked with $\circ$.

EXAMPLE 9.3

M is a Turing machine represented by the transition system in Fig. 9.4. Obtain the computation sequence of M for processing the input string 0011.

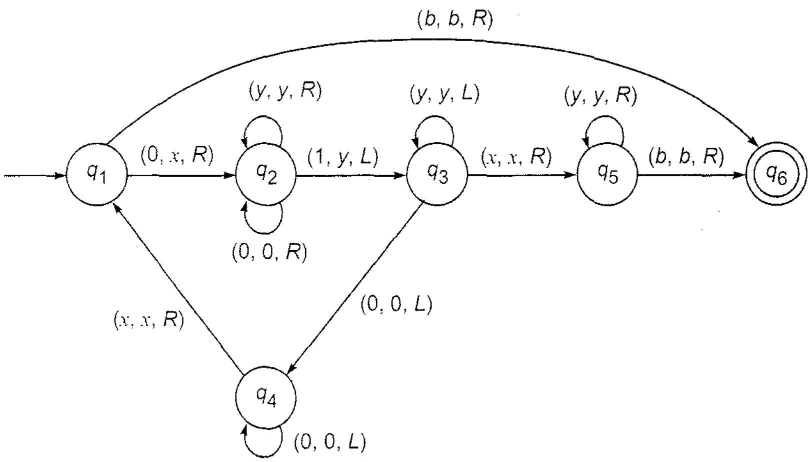

Fig. 9.4 Transition system for M.

Solution

The initial tape input is $b0011b$. Let us assume that M is in state q_1 and the R/W head scans 0 (the first 0). We can represent this as in Fig. 9.5. The figure can be represented by

$$\begin{array}{c}\downarrow\\ b0011b\\ q_1\end{array}$$

From Fig. 9.4 we see that there is a directed edge from q_1 to q_2 with the label $(0, x, R)$. So the current symbol 0 is replaced by x and the head moves right. The new state is q_2. Thus, we get

$$\begin{array}{c}\downarrow\\ bx011b\\ q_2\end{array}$$

The change brought about by processing the symbol 0 can be represented as

$$\underset{q_1}{b\overset{\downarrow}{0}011b} \xrightarrow{(0,x,R)} \underset{q_2}{bx\overset{\downarrow}{0}11b}$$

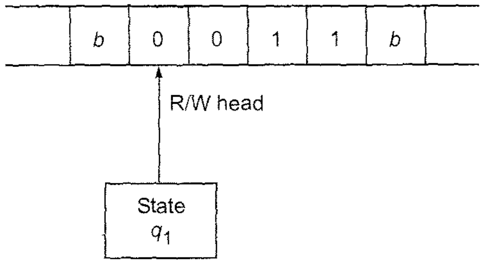

Fig. 9.5 TM processing 0011.

The entire computation sequence reads as follows:

$$\underset{q_1}{b\overset{\downarrow}{0}011b} \xrightarrow{(0,x,R)} \underset{q_2}{bx\overset{\downarrow}{0}11b} \xrightarrow{(0,0,R)} \underset{q_2}{bx0\overset{\downarrow}{1}1b}$$

$$\xrightarrow{(1,y,L)} \underset{q_3}{bx\overset{\downarrow}{0}y1b} \xrightarrow{(0,0,L)} \underset{q_4}{b\overset{\downarrow}{x}0y1b} \xrightarrow{(x,x,R)} \underset{q_1}{bx\overset{\downarrow}{0}y1b}$$

$$\xrightarrow{(0,x,R)} \underset{q_2}{bxx\overset{\downarrow}{y}1b} \xrightarrow{(y,y,R)} \underset{q_2}{bxxy\overset{\downarrow}{1}b} \xrightarrow{(1,y,L)} \underset{q_3}{bxx\overset{\downarrow}{y}yb}$$

$$\xrightarrow{(y,y,L)} \underset{q_3}{bx\overset{\downarrow}{x}yyb} \xrightarrow{(x,x,R)} \underset{q_5}{bxx\overset{\downarrow}{y}yb} \xrightarrow{(y,y,R)} \underset{q_5}{bxxy\overset{\downarrow}{y}b}$$

$$\xrightarrow{(y,y,R)} \underset{q_5}{bxxyy\overset{\downarrow}{b}} \xrightarrow{(b,b,R)} \underset{q_6}{bxxyyb\overset{\downarrow}{b}}$$

9.3 LANGUAGE ACCEPTABILITY BY TURING MACHINES

Let us consider the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, b, F)$. A string w in Σ^* is said to be accepted by M if $q_0w \overset{*}{\vdash} \alpha_1 p \alpha_2$ for some $p \in F$ and $\alpha_1, \alpha_2 \in \Gamma^*$.

M does not accept w if the machine M either halts in a nonaccepting state or does not halt.

It may be noted that though there are other equivalent definitions of acceptance by the Turing machine, we will be not discussing them in this text.

EXAMPLE 9.4

Consider the Turing machine M described by the transition table given in Table 9.2. Describe the processing of (a) 011, (b) 0011, (c) 001 using IDs. Which of the above strings are accepted by M?

TABLE 9.2 Transition Table for Example 9.4

Present state	*Tape symbol*				
	0	1	x	y	b
$\rightarrow q_1$	xRq_2				bRq_5
q_2	$0Rq_2$	yLq_3		yRq_2	
q_3	$0Lq_4$		xRq_5	yLq_3	
q_4	$0Lq_4$		xRq_1		
q_5				$yxRq_5$	bRq_6
q_6					

Solution

(a) $q_1011 \vdash xq_211 \vdash q_3xy1 \vdash xq_5y1 \vdash xyq_51$

As $\delta(q_5, 1)$ is not defined, M halts; so the input string 011 is not accepted.

(b) $q_10011 \vdash xq_2011 \vdash x0q_211 \vdash xq_30y1 \vdash q_4x0y1 \vdash xq_10y1.$

$\vdash xxq_2y1 \vdash xxyq_21 \vdash xxq_3yy \vdash xq_3xyy \vdash xxq_5yy$

$\vdash xxyq_5y \vdash xxyyq_5b \vdash xxyybq_6$

M halts. As q_6 is an accepting state, the input string 0011 is accepted by M.

(c) $q_1001 \vdash xq_201 \vdash x0q_21 \vdash xq_30y \vdash q_4x0y$

$\vdash xq_10y \vdash xxq_2y \vdash xxyq_2$

M halts. As q_2 is not an accepting state, 001 is not accepted by M.

9.4 DESIGN OF TURING MACHINES

We now give the basic guidelines for designing a Turing machine.

(i) The fundamental objective in scanning a symbol by the R/W head is to 'know' what to do in the future. The machine must remember the past symbols scanned. The Turing machine can remember this by going to the next unique state.

(ii) The number of states must be minimized. This can be achieved by changing the states only when there is a change in the written symbol or when there is a change in the movement of the R/W head. We shall explain the design by a simple example.

EXAMPLE 9.5

Design a Turing machine to recognize all strings consisting of an even number of 1's.

Solution

The construction is made by defining moves in the following manner:

(a) q_1 is the initial state. M enters the state q_2 on scanning 1 and writes b.

(b) If M is in state q_2 and scans 1, it enters q_1 and writes b.

(c) q_1 is the only accepting state.

So M accepts a string if it exhausts all the input symbols and finally is in state q_1. Symbolically,

$$M = (\{q_1, q_2\}, \{1, b\}, \{1, b\}, \delta, q, b, \{q_1\})$$

where δ is defined by Table 9.3.

TABLE 9.3 Transition Table for Example 9.5

Present state	1
$\rightarrow (q_1)$	bq_2R
q_2	bq_1R

Let us obtain the computation sequence of 11. Thus, $q_1 11 \vdash bq_2 1 \vdash bbq_1$. As q_1 is an accepting state, 11 is accepted. $q_1 111 \vdash bq_2 11 \vdash bbq_1 1 \vdash bbbq_2$. M halts and as q_2 is not an accepting state, 111 is not accepted by M.

EXAMPLE 9.6

Design a Turing machine over $\{1, b\}$ which can compute a concatenation function over $\Sigma = \{1\}$. If a pair of words (w_1, w_2) is the input, the output has to be $w_1 w_2$.

Solution

Let us assume that the two words w_1 and w_2 are written initially on the input tape separated by the symbol b. For example, if $w_1 = 11$, $w_2 = 111$, then the input and output tapes are as shown in Fig. 9.6.

Fig. 9.6 Input and output tapes.

We observe that the main task is to remove the symbol b. This can be done in the following manner:

(a) The separating symbol b is found and replaced by 1.

(b) The rightmost 1 is found and replaced by a blank b.
(c) The R/W head returns to the starting position.

A computation is illustrated in Table 9.4.

TABLE 9.4 Computation for $11b111$

$$q_011b111 \vdash 1q_01b111 \vdash 11q_0b111 \vdash 111q_1111$$
$$\vdash 1111q_111 \vdash 11111q_11 \vdash 111111q_1b \vdash 11111q_21b$$
$$\vdash 1111q_31bb \vdash 111q_311bb \vdash 11q_3111bb \vdash 1q_31111bb$$
$$\vdash q_311111bb \vdash q_3b11111bb \vdash bq_f11111bb$$

From the above computation sequence for the input string $11b111$, we can construct the transition table given in Table 9.5.

For the input string $1b1$, the computation sequence is given as

$$q_01b1 \vdash 1q_0b1 \vdash 11q_11 \vdash 111q_1b \vdash 11q_2b \vdash 1q_31bb$$
$$\vdash q_311bb \vdash q_3b11bb \vdash bq_f11bb.$$

TABLE 9.5 Transition Table for Example 9.6

Present state	*Tape symbol*	
	1	b
$\to q_0$	$1Rq_0$	$1Rq_1$
q_1	$1Rq_1$	bLq_2
q_2	bLq_3	—
q_3	$1Lq_3$	bRq_f
q_f	—	—

EXAMPLE 9.7

Design a TM that accepts

$$\{0^n1^n \mid n \geq 1\}.$$

Solution

We require the following moves:

(a) If the leftmost symbol in the given input string w is 0, replace it by x and move right till we encounter a leftmost 1 in w. Change it to y and move backwards.
(b) Repeat (a) with the leftmost 0. If we move back and forth and no 0 or 1 remains, move to a final state.
(c) For strings not in the form 0^n1^n, the resulting state has to be nonfinal.

Keeping these ideas in our mind, we construct a TM M as follows:

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

where

$$Q = \{q_0, q_1, q_2, q_3, q_f)$$

$$F = \{q_f\}$$

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

$$\Gamma = \{0, 1, x, y, b\}$$

The transition diagram is given in Fig. 9.7. M accepts $\{0^n 1^n \mid n \geq 1\}$. The moves for 0011 and 010 are given below just to familiarize the moves of M to the reader.

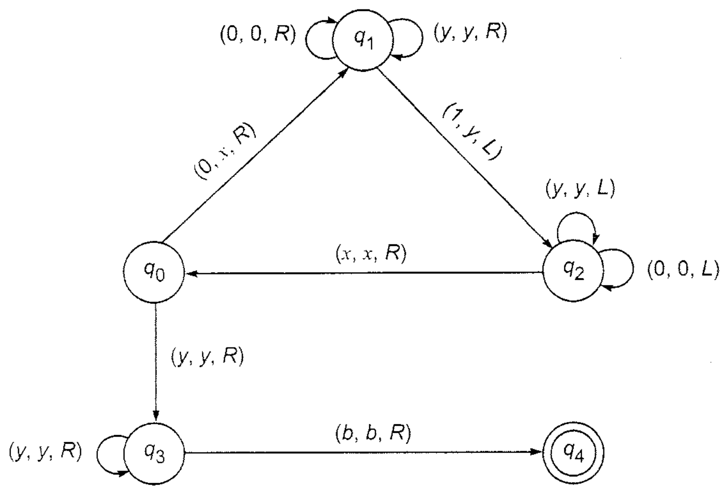

Fig. 9.7 Transition diagram for Example 9.7.

$$q_0 0011 \vdash xq_1 011 \vdash x0q_1 11 \vdash xq_2 0y1$$
$$\vdash q_2 x0y1 \vdash xq_0 0y1 \vdash xxq_1 y1 \vdash xxyq_1 1$$
$$\vdash xxq_2 yy \vdash xq_2 xyy \vdash xxq_0 yy \vdash xxyq_3 y$$
$$\vdash xxyyq_3 = xxyyq_3 b \vdash xxyybq_4 b$$

Hence 0011 is accepted by M.

$$q_0 010 \vdash xq_1 10 \vdash q_2 xy0 \vdash xq_0 y0 \vdash xyq_3 0$$

As $\delta(q_3, 0)$ is not defined, M halts. So 010 is not accepted by M.

EXAMPLE 9.8

Design a Turing machine M to recognize the language

$$\{1^n 2^n 3^n \mid n \geq 1\}.$$

Solution

Before designing the required Turing machine M, let us evolve a procedure for processing the input string 112233. After processing, we require the ID to be of the form $bbbbbbq_7$. The processing is done by using five steps:

Step 1 q_1 is the initial state. The R/W head scans the leftmost 1, replaces 1 by b, and moves to the right. M enters q_2.

Step 2 On scanning the leftmost 2, the R/W head replaces 2 by b and moves to the right. M enters q_3.

Step 3 On scanning the leftmost 3, the R/W head replaces 3 by b, and moves to the right. M enters q_4.

Step 4 After scanning the rightmost 3, the R/W heads moves to the left until it finds the leftmost 1. As a result, the leftmost 1, 2 and 3 are replaced by b.

Step 5 Steps 1–4 are repeated until all 1's, 2's and 3's are replaced by blanks.

The change of IDs due to processing of 112233 is given as

$$q_1112233 \vdash bq_212233 \vdash b1q_22233 \vdash b1bq_3233 \vdash b1b2q_333$$

$$\vdash b1b2bq_43 \vdash b1b_2q_5b3 \vdash b1bq_52b3 \vdash b1q_5b2b3 \vdash bq_51b2b3$$

$$\vdash q_6b1b2b3 \vdash bq_11b2b3 \vdash bbq_2b2b3 \vdash bbbq_22b3$$

$$\vdash bbbbq_3b3 \vdash bbbbbq_33 \vdash bbbbbbq_4b \vdash bbbbbq_7bb$$

Thus,

$$q_1112233 \overset{*}{\vdash} q_7bbbbbb$$

As q_7 is an accepting state, the input string 112233 is accepted.

Now we can construct the transition table for M. It is given in Table 9.6.

TABLE 9.6 Transition Table for Example 9.7

Present state	*Input tape symbol*			
	1	2	3	b
$\rightarrow q_1$	bRq_2			bRq_1
q_2	$1Rq_2$	bRq_3		bRq_2
q_3		$2Rq_3$	bRq_4	bRq_3
q_4			$3Lq_5$	bLq_7
q_5	$1Lq_6$	$2Lq_5$		bLq_5
q_6	$1Lq_6$			bRq_1
q_7				

It can be seen from the table that strings other than those of the form $0^n1^n2^n$ are not accepted. It is advisable to compute the computation sequence for strings like 1223, 1123, 1233 and then see that these strings are rejected by M.

9.5 DESCRIPTION OF TURING MACHINES

In the examples discussed so far, the transition function δ was described as a partial function (function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is not defined for all (q, x)) by spelling out the current state, the input symbol, the resulting state, the tape symbol replacing the input symbol and the movement of R/W head to the left or right. We can call this a formal description of a TM. Just as we have the machine language and higher level languages for a computer, we can have a higher level of description, called the *implementation description*. In this case we describe the movement of the head, the symbol stored etc. in English. For example, a single instruction like 'move to right till the end of the input string' requires several moves. A single instruction in the implementation description is equivalent to several moves of a standard TM (Hereafter a standard TM refers to the TM defined in Definition 9.1). At a higher level we can give instructions in English language even without specifying the state or transition function. This is called a *high-level description.*

In the remaining sections of this chapter and later chapters, we give implementation description or high-level description.

9.6 TECHNIQUES FOR TM CONSTRUCTION

In this section we give some high-level conceptual tools to make the construction of TMs easier. The Turing machine defined in Section 9.1 is called the standard Turing machine.

9.6.1 TURING MACHINE WITH STATIONARY HEAD

In the definition of a TM we defined $\delta(q, a)$ as (q', y, D) where $D = L$ or R. So the head moves to the left or right after reading an input symbol. Suppose, we want to include the option that the head can continue to be in the same cell for some input symbol. Then we define $\delta(q, a)$ as (q', y, S). This means that the TM, on reading the input symbol a, changes the state to q' and writes y in the current cell in place of a and continues to remain in the same cell. In terms of IDs,

$$wqax \vdash wq'yx$$

Of course, this move can be simulated by the standard TM with two moves, namely

$$wqax \vdash wyq''x \vdash wq'yx$$

That is, $\delta(q, a) = (q', y, S)$ is replaced by $\delta(q, a) = (q'', y, R)$ and $\delta(q'', X) = (q', y, L)$ for any tape symbol X.

Thus in this model $\delta(q, a) = (q', y, D)$ where $D = L$, R or S.

9.6.2 STORAGE IN THE STATE

We are using a state, whether it is of a FA or pda or TM, to 'remember' things. We can use a state to store a symbol as well. So the state becomes a pair (q, a) where q is the state (in the usual sense) and a is the tape symbol stored in (q, a). So the new set of states becomes $Q \times \Gamma$.

EXAMPLE 9.9

Construct a TM that accepts the language **0 1* + 1 0***.

Solution

We have to construct a TM that remembers the first symbol and checks that it does not appear afterwards in the input string. So we require two states, q_0, q_1. The tape symbols are 0, 1 and b. So the TM, having the 'storage facility in state', is

$$M = (\{q_0, q_1\} \times \{0, 1, b\}, \{0, 1\}, \{0, 1, b\}, \delta, [q_0, b], \{[q_1, b]\})$$

We describe δ by its implementation description.

1. In the initial state, M is in q_0 and has b in its data portion. On seeing the first symbol of the input sting w, M moves right, enters the state q_1 and the first symbol, say a, it has seen.
2. M is now in $[q_1, a]$. (i) If its next symbol is b, M enters $[q_1, b]$, an accepting state. (ii) If the next symbol is a, M halts without reaching the final state (i.e. δ is not defined). (iii) If the next symbol is $\overline{a}$ ($\overline{a} = 0$ if $a = 1$ and $\overline{a} = 1$ if $a = 0$), M moves right without changing state.
3. Step 2 is repeated until M reaches $[q_1, b]$ or halts (δ is not defined for an input symbol in w).

9.6.3 MULTIPLE TRACK TURING MACHINE

In the case of TM defined earlier, a single tape was used. In a multiple track TM, a single tape is assumed to be divided into several tracks. Now the tape alphabet is required to consist of k-tuples of tape symbols, k being the number of tracks. Hence the only difference between the standard TM and the TM with multiple tracks is the set of tape symbols. In the case of the standard Turing machine, tape symbols are elements of Γ; in the case of TM with multiple track, it is Γ^k. The moves are defined in a similar way.

9.6.4 SUBROUTINES

We know that subroutines are used in computer languages, when some task has to be done repeatedly. We can implement this facility for TMs as well.

First, a TM program for the subroutine is written. This will have an initial state and a 'return' state. After reaching the return state, there is a temporary halt. For using a subroutine, new states are introduced. When there is a need for calling the subroutine, moves are effected to enter the initial state for the subroutine (when the return state of the subroutine is reached) and to return to the main program of TM.

We use this concept to design a TM for performing multiplication of two positive integers.

EXAMPLE 9.10

Design a TM which can multiply two positive integers.

Solution

The input (m, n), m, n being given, the positive integers are represented by 0^m10^n. M starts with 0^m10^n in its tape. At the end of the computation, 0^{mn}(mn in unary representation) surrounded by b's is obtained as the ouput.

The major steps in the construction are as follows:

1. 0^m10^n1 is placed on the tape (the output will be written after the rightmost 1).
2. The leftmost 0 is erased.
3. A block of n 0's is copied onto the right end.
4. Steps 2 and 3 are repeated m times and 10^m10^{mn} is obtained on the tape.
5. The prefix 10^m1 of 10^m10^{mn} is erased, leaving the product mn as the output.

For every 0 in 0^m, 0^n is added onto the right end. This requires repetition of step 3. We define a subroutine called COPY for step 3.

For the subroutine COPY, the initial state is q_1 and the final state is q_5. δ is given by the transition table (see Table 9.7).

TABLE 9.7 Transition Table for Subroutine COPY

State	*Tape symbol*			
	0	1	2	*b*
q_1	q_22R	q_41L	—	—
q_2	q_20R	q_21R	—	q_30L
q_3	q_30L	q_31L	q_12R	—
q_4	—	q_51R	q_40L	—
q_5	—	—	—	—

The Turing machine M has the initial state q_0. The initial ID for M is $q_00^m10^n1$. On seeing 0, the following moves take place (q_6 is a state of M). $q_00^m10^n1 \vdash bq_60^{m-1}10^n1 \overset{*}{\vdash} b0^{m-1}q_610^n1 \vdash b0^{m-1}1q_10^n1$. q_1 is the initial state of COPY. The TM M_1 performs the subroutine COPY. The following moves take place for M_1: $q_1 0^n 1 \vdash 2q_2 0^{n-1} 1 \overset{*}{\vdash} 20^{n-1} 1 q_3 b \vdash 20^{n-1} q_3 10 \overset{*}{\vdash} 2q_1 0^{n-1} 10$. After exhausting 0's, q_1 encounters 1. M_1 moves to state q_4. All 2's are converted back to 0's and M_1 halts in q_5. The TM M picks up the computation by starting from q_5. The q_0 and q_6 are the states of M. Additional states are created to check whether each 0 in 0^m gives rise to 0^m at the end of the rightmost 1 in the input string. Once this is over, M erases $10^n 1$ and finds 0^{nm} in the input tape.

M can be defined by

$$M = (\{q_0, q_1, \ldots, q_{12}\}, \{0, 1\}, \{0, 1, 2, b\}, \delta, q_0, b, \{q_{12}\})$$

where δ is defined by Table 9.8.

TABLE 9.8 Transition Table for Example 9.10

	0	1	2	b
q_0	q_6bR	—	—	—
q_6	q_60R	q_11R	—	—
q_5	q_70L	—	—	—
q_7	—	q_81L	—	—
q_8	q_90L	—	—	$q_{10}bR$
q_9	q_90L	—	—	q_0bR
q_{10}	—	$q_{11}bR$	—	—
q_{11}	$q_{11}bR$	$q_{12}bR$	—	—

Thus M performs multiplication of two numbers in unary representation.

9.7 VARIANTS OF TURING MACHINES

The Turing machine we have introduced has a single tape. $\delta(q, a)$ is either a single triple (p, y, D), where $D = R$ or L, or is not defined. We introduce two new models of TM:

(i) a TM with more than one tape
(ii) a TM where $\delta(q, a) = \{(p_1, y_1, D_1), (p_2, y_2, D_2), \ldots, (p_r, y_r, D_r)\}$. The first model is called a multitape TM and the second a nondeterministic TM.

9.7.1 MULTITAPE TURING MACHINES

A multitape TM has a finite set Q of states, an initial state q_0, a subset F of Q called the set of final states, a set P of tape symbols, a new symbol b, not in P called the blank symbol. (We assume that $\Sigma \subseteq \Gamma$ and $b \notin \Sigma$.)

There are k tapes, each divided into cells. The first tape holds the input string w. Initially, all the other tapes hold the blank symbol.

Initially the head of the first tape (input tape) is at the left end of the input w. All the other heads can be placed at any cell initially.

δ is a partial function from $Q \times \Gamma^k$ into $Q \times \Gamma^k \times \{L, R, S\}^k$. We use implementation description to define δ. Figure 9.8 represents a multitape TM. A move depends on the current state and k tape symbols under k tape heads.

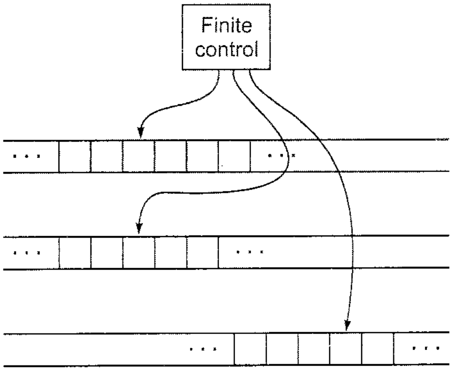

Fig. 9.8 Multitape Turing machine.

In a typical move:

(i) M enters a new state.
(ii) On each tape, a new symbol is written in the cell under the head.
(iii) Each tape head moves to the left or right or remains stationary. The heads move independently; some move to the left, some to the right and the remaining heads do not move.

The initial ID has the initial state q_0, the input string w in the first tape (input tape), empty strings of b's in the remaining $k - 1$ tapes. An accepting ID has a final state, some strings in each of the k tapes.

Theorem 9.1 Every language accepted by a multitape TM is acceptable by some single-tape TM (that is, the standard TM).

Proof Suppose a language L is accepted by a k-tape TM M. We simulate M with a single-tape TM with $2k$ tracks. The second, fourth, . . ., $(2k)$th tracks hold the contents of the k-tapes. The first, third, . . ., $(2k - 1)$th tracks hold a head marker (a symbol say X) to indicate the position of the respective tape head. We give an 'implementation description' of the simulation of M with a single-tape TM M_1. We give it for the case $k = 2$. The construction can be extended to the general case.

Figure 9.9 can be used to visualize the simulation. The symbols A_2 and B_5 are the current symbols to be scanned and so the headmarker X is above the two symbols.

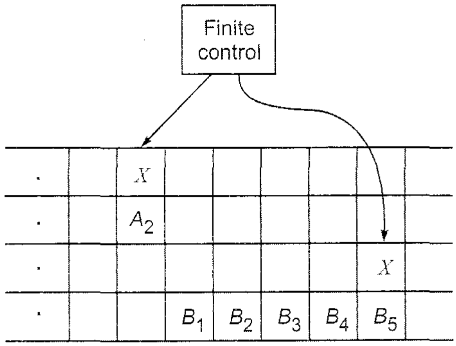

Fig. 9.9 Simulation of multitape TM.

Initially the contents of tapes 1 and 2 of M are stored in the second and fourth tracks of M_1. The headmarkers of the first and third tracks are at the cells containing the first symbol.

To simulate a move of M, the $2k$-track TM M_1 has to visit the two headmarkers and store the scanned symbols in its control. Keeping track of the headmarkers visited and those to be visited is achieved by keeping a count and storing it in the finite control of M_1. Note that the finite control of M_1 has also the information about the states of M and its moves. After visiting both head markers, M_1 knows the tape symbols being scanned by the two heads of M.

Now M_1 revisits each of the headmarkers:

(i) It changes the tape symbol in the corresponding track of M_1 based on the information regarding the move of M corresponding to the state (of M) and the tape symbol in the corresponding tape M.
(ii) It moves the headmarkers to the left or right.
(iii) M_1 changes the state of M in its control.

This is the simulation of a single move of M. At the end of this, M_1 is ready to implement its next move based on the revised positions of its headmarkers and the changed state available in its control.

M_1 accepts a string w if the new state of M, as recorded in its control at the end of the processing of w, is a final state of M.

Definition 9.3 Let M be a TM and w an input string. The running time of M on input w, is the number of steps that M takes before halting. If M does not halt on an input string w, then the running time of M on w is infinite.

Note: Some TMs may not halt on all inputs of length n. But we are interested in computing the running time, only when the TM halts.

Definition 9.4 The time complexity of TM M is the function $T(n)$, n being the input size, where $T(n)$ is defined as the maximum of the running time of M over all inputs w of size n.

Theorem 9.2 If M_1 is the single-tape *TM* simulating multitape TM M, then the time taken by M_1 to simulate n moves of M is $O(n^2)$.

Proof Let M be a k-tape TM. After n moves of M, the head markers of M_1 will be separated by $2n$ cells or less. (At the worst, one tape movement can be to the left by n cells and another can be to the right by n cells. In this case the tape headmarkers are separated by $2n$ cells. In the other cases, the 'gap' between them is less). To simulate a move of M, the TM M_1 must visit all the k headmarkers. If M starts with the leftmost headmarker, M_1 will go through all the headmarkers by moving right by at most $2n$ cells. To simulate the change in each tape, M_1 has to move left by at most $2n$ cells; to simulate changes in k tapes, it requires at most two moves in the reverse direction for each tape.

Thus the total number of moves by M_1 for simulating one move of M is atmost $4n + 2k$. ($2n$ moves to right for locating all headmarkers, $2n + 2k$ moves to the left for simulating the change in the content of k tapes.) So the number of moves of M_1 for simulating n moves of M is $n(4n + 2k)$. As the constant k is independent of n, the time taken by M_1 is $O(n^2)$.

9.7.2 NONDETERMINISTIC TURING MACHINES

In the case of standard Turing machines (hereafter we refer to this machine as deterministic TM), $\delta(q_1, a)$ was defined (for some elements of $Q \times \Gamma$) as an element of $Q \times \Gamma \times \{L, R\}$. Now we extend the definition of δ. In a nondeterministic TM, $\delta(q_1, a)$ is defined as a subset of $Q \times \Gamma \times \{L, R\}$.

Definition 9.5 A nondeterministic Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$ where

1. Q is a finite nonempty set of states
2. Γ is a finite nonempty set of tape symbols
3. $b \in \Gamma$ is called the blank symbol
4. Σ is a nonempty subset of Γ, called the set of input symbols. We assume that $b \notin \Sigma$.
5. q_0 is the initial state
6. $F \subseteq Q$ is the set of final states
7. δ is a partial function from $Q \times \Gamma$ into the power set of $Q \times \Gamma \times \{L, R\}$.

Note: If $q \in Q$ and $x \in \Gamma$ and $\delta(q, x) = \{(q_1, y_1, D_1), (q_2, y_2, D_2), \ldots, (q_n, y_n, D_n)\}$ then the NTM can chose any one of the actions defined by (q_i, y_i, D_i) for $i = 1, 2, \ldots, n$.

We can also express this in terms of $\vdash$ relation. If $\delta(q, x) = \{(q_i, y_i, D_i) | i = 1, 2, \ldots, n\}$ then the ID $zqxw$ can change to any one of the n IDs specified by the n-element set $\delta(q, x)$.

Suppose $\delta(q, x) = \{(q_1, y_1, L), (q_2, y_2, R), (q_3, y_3, L)\}$. Then

$$z_1z_2 \ldots z_kqxz_{k+1} \ldots z_n \vdash z_1z_2 \ldots z_{k-1}q_1z_ky_1z_{k+1} \ldots z_n$$

or

$$z_1z_2 \ldots z_kqxz_{k+1} \ldots z_n \vdash z_1z_2 \ldots z_ky_2q_2z_{k+1} \ldots z_n$$

or

$$z_1z_2 \ldots z_kqxz_{k+1} \ldots z_n \vdash z_1z_2 \ldots z_{k-1}q_3z_ky_3z_{k+1} \ldots z_n.$$

So on reading the input symbol, the NTM M whose current ID is $z_1 z_2 \ldots z_k q x z_{k+1} \ldots z_n$ can change to any one of the three IDs given earlier.

Remark When $\delta(q, x) = \{(q_i, y_i, D_i) \mid i = 1, 2, \ldots, n\}$ then NTM chooses any one of the n triples totally (that is, it cannot take a state from one triple, another tape symbol from a second triple and a third $D(L$ or $R)$ from a third triple, etc.

Definition 9.6 $w \in \Sigma^*$ is accepted by a nondeterministic TM M if $q_0 w \overset{*}{\vdash} x q_f y$ for some final state q_f.

The set of all strings accepted by M is denoted by $T(M)$.

Note: As in the case of NDFA, an ID of the form xqy (for some $q \notin F$) may be reached as the result of applying the input string w. But w is accepted by M as long as there is some sequence of moves leading to an ID with an accepting state. It does not matter that there are other sequences of moves leading to an ID with a nonfinal state or TM halts without processing the entire input string.

Theorem 9.3 If M is a nondeterministic TM, there is a deterministic TM M_1 such that $T(M) = T(M_1)$.

Proof We construct M_1 as a multitape TM. Each symbol in the input string leads to a change in ID. M_1 should be able to reach all IDs and stop when an ID containing a final state is reached. So the first tape is used to store IDs of M as a sequence and also the state of M. These IDs are separated by the symbol $*$ (included as a tape symbol). The current ID is known by marking an x along with the ID-separator $*$ (The symbol $*$ marked with x is a new tape symbol.) All IDs to the left of the current one have been explored already and so can be ignored subsequently. Note that the current ID is decided by the current input symbol of w.

Figure 9.10 illustrates the deterministic TM M_1.

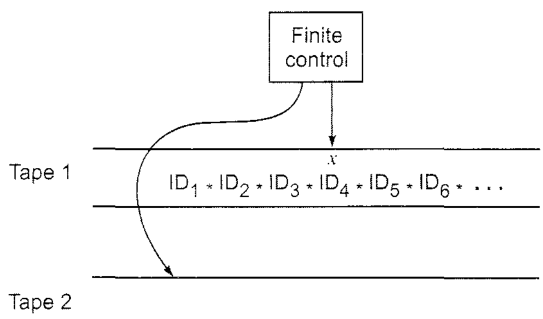

Fig. 9.10 The deterministic TM simulating M.

To process the current ID, M_1 performs the following steps.

1. M_1 examines the state and the scanned symbol of the current ID. Using the knowledge of moves of M stored in the finite control of M_1, M_1 checks whether the state in the current ID is an accepting state of M. In this case M_1 accepts and stops simulating M.

2. If the state q say in the current ID $xqay$ is not an accepting state of M_1 and $\delta(q, a)$ has k triples, M_1 copies the ID $xqay$ in the second tape and makes k copies of this ID at the end of the sequence of IDs in tape 2.
3. M_1 modifies these k IDs in tape 2 according to the k choices given by $\delta(q, a)$.
4. M_1 returns to the marked current ID, erases the mark x and marks the next ID-separator $*$ with x (to the $*$ which is to the left of the next ID to be processed). Then M_1 goes back to step 1.

M_1 stops when an accepting state of M is reached in step 1.

Now M_1 accepts an input string w only when it is able to find that M has entered an accepting state, after a finite number of moves. This is clear from the simulated sequence of moves of M_1 (ending in step 1)

We have to prove that M_1 will eventually reach an accepting ID (that is, an ID having an accepting state of M) if M enters an accepting ID after n moves. Note each move of M is simulated by several moves of M_1.

Let m be the maximum number of choices that M has for various (q, a)'s. (It is possible to find m since we have only finite number of pairs in $Q \times \Gamma$.) So for each initial ID of M, there are at most m IDs that M can reach after one move, at most m^2 IDs that M can reach after two moves, and so on. So corresponding to n moves of M, there are at most $1 + m + m^2 + \cdots + m^n$ moves of M_1. Hence the number of IDs to be explored by M_1 is at most nm^n.

We assume that M_1 explores these IDs. These IDs have a tree structure having the initial ID as its root. We can apply breadth-first search of the nodes of the tree (that is, the nodes at level 1 are searched, then the nodes at level 2, and so on.) If M reaches an accepting ID after n moves, then M_1 has to search atmost nm^n IDs before reaching an accepting ID. So, if M accepts w, then M_1 also accepts w (eventually). Hence $T(M) = T(M_1)$.

9.8 THE MODEL OF LINEAR BOUNDED AUTOMATON

This model is important because (a) the set of context-sensitive languages is accepted by the model, and (b) the infinite storage is restricted in size but not in accessibility to the storage in comparison with the Turing machine model. It is called the *linear bounded automaton* (LBA) because a linear function is used to restrict (to bound) the length of the tape.

In this section we define the model of linear bounded automaton and develop the relation between the linear bounded automata and context-sensitive languages. It should be noted that the study of context-sensitive languages is important from practical point of view because many compiler languages lie between context-sensitive and context-free languages.

A linear bounded automaton is a nondeterministic Turing machine which has a single tape whose length is not infinite but bounded by a linear function of the length of the input string. The models can be described formally by the following set format:

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

All the symbols have the same meaning as in the basic model of Turing machines with the difference that the input alphabet Σ contains two special symbols ₵ and \$. ₵ is called the left-end marker which is entered in the left-most cell of the input tape and prevents the R/W head from getting off the left end of the tape. \$ is called the right-end marker which is entered in the right-most cell of the input tape and prevents the R/W head from getting off the right end of the tape. Both the endmarkers should not appear on any other cell within the input tape, and the R/W head should not print any other symbol over both the endmarkers.

Let us consider the input string w with $|w| = n - 2$. The input string w can be recognized by an LBA if it can also be recognized by a Turing machine using no more than kn cells of input tape, where k is a constant specified in the description of LBA. The value of k does not depend on the input string but is purely a property of the machine. Whenever we process any string in LBA, we shall assume that the input string is enclosed within the endmarkers ₵ and \$. The above model of LBA can be represented by the block diagram of Fig. 9.11. There are two tapes: one is called the input tape, and the other, working tape. On the input tape the head never prints and never moves to the left. On the working tape the head can modify the contents in any way, without any restriction.

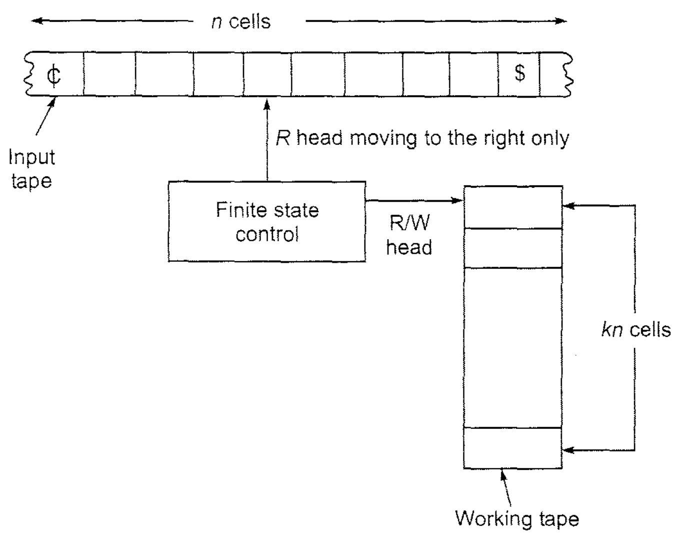

Fig. 9.11 Model of linear bounded automaton.

In the case of LBA, an ID is denoted by (q, w, k), where $q \in Q$, $w \in \Gamma$ and k is some integer between 1 and n. The transition of IDs is similar except that k changes to $k - 1$ if the R/W head moves to the left and to $k + 1$ if the head moves to the right.

The language accepted by LBA is defined as the set

$$\{w \in (\Sigma - \{¢, \$\})^* | (q_0, ¢w\$, 1) \overset{*}{\vdash} (q, \alpha, i)$$

for some $q \in F$ and for some integer i between 1 and $n\}$.

Note: As a null string can be represented either by the absence of input string or by a completely blank tape, an LBA may accept the null string.

9.8.1 RELATION BETWEEN LBA AND CONTEXT-SENSITIVE LANGUAGES

The set of strings accepted by nondeterministic LBA is the set of strings generated by the context-sensitive grammars, excluding the null strings. Now we give an important result:

If L is a context-sensitive language, then L is accepted by a linear bounded automaton. The converse is also true.

The construction and the proof are similar to those for Turing machines with some modifications.

9.9 TURING MACHINES AND TYPE 0 GRAMMARS

In this section we construct a type 0 grammar generating the set accepted by a given Turing machine M. The productions are constructed in two steps. In step 1 we construct productions which transform the string $[q_1 ¢ w\$]$ into the string $[q_2 b]$, where q_1 is the initial state, q_2 is an accepting state, $¢$ is the left-endmarker, and \$ is the right-endmarker. The grammar obtained by applying step 1 is called the *transformational grammar*. In step 2 we obtain inverse production rules by reversing the productions of the transformational grammar to get the required type 0 grammar G. The construction is in such a way that w is accepted by M if and only if w is in $L(G)$.

9.9.1 CONSTRUCTION OF A GRAMMAR CORRESPONDING TO TM

For understanding the construction, we have to note that a transition of ID corresponds to a production. We enclose IDs within brackets. So acceptance of w by M corresponds to the transformation of initial ID $[q_1\ ¢\ w\ \$]$ into $[q_2 b]$. Also, the 'length' of ID may change if the R/W head reaches the left-end or the right-end, i.e. when the left-hand side or the right-hand side bracket is reached. So we get productions corresponding to transition of IDs with (i) no change in length, and (ii) change in length. We assume that the transition table is given.

(D) The LBA productions are

$$\begin{array}{ll} ¢q_4\$ \to q_4\$, & ¢q_4\$ \to ¢q_4 \\ \$q_4\$ \to q_4\$, & ¢q_4 \to q_4 \\ 0q_4\$ \to q_4\$, & \\ 1q_4\$ \to q_4\$ & \end{array} \tag{9.12}$$

Step 2 The productions of the generative grammar are obtained by reversing the arrows of productions given by (9.5)–(9.12).

9.11 SUPPLEMENTARY EXAMPLES

EXAMPLE 9.13

Design a TM that copies strings of 1's.

Solution

We design a TM so that we have ww after copying $w \in \{1\}^*$. Define M by

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

where δ is defined by Table 9.11.

TABLE 9.11 Transition Table for Example 9.13

Present state	*Tape symbol*		
	1	b	a
q_0	q_0aR	q_1bL	—
q_1	q_11L	q_3bR	q_21R
q_2	q_21R	q_11L	—
q_3	—	—	—

The procedure is simple.

M replaces every 1 by the symbol a. Then M replaces the rightmost a by 1. It goes to the right end of the string and writes a 1 there. Thus M has added a 1 for the rightmost 1 in the input string w. This process can be repeated.

M reaches q_1 after replacing all 1's by a's and reading the blank at the end of the input string. After replacing a by 1, M reaches q_2. M reaches q_3 at the end of the process and halts. If $w = 1^n$, than we have 1^{2n} at the end of the computation. A sample computation is given below.

$$\begin{aligned} q_011 &\vdash aq_01 \vdash aaq_0b \vdash aq_1a \\ &\vdash a1q_2b \vdash aq_111 \vdash q_1a11 \\ &\vdash 1q_211 \vdash 11q_21 \vdash 111q_2b \\ &\vdash 11q_211 \vdash 1q_1111 \\ &\vdash q_11111 \vdash q_1b1111 \vdash q_31111 \end{aligned}$$

EXAMPLE 9.14

Construct a TM to accept the set L of all strings over $\{0,1\}$ ending with 010.

Solution

L is certainly a regular set and hence a deterministic automaton is sufficient to recognize L. Figure 9.12 gives a DFA accepting L.

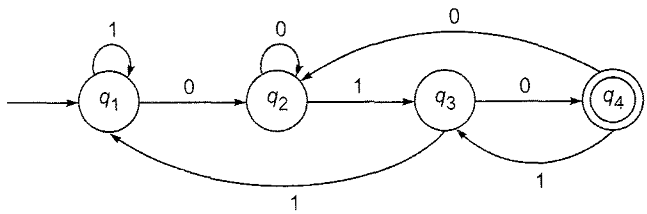

Fig. 9.12 DFA for Example 9.14.

Converting this DFA to a TM is simple. In a DFA M, the move is always to the right. So the TM's move will always be to the right. Also M reads the input symbol and changes state. So the TM M_1 does the same; it reads an input symbol, does not change the symbol and changes state. At the end of the computation, the TM sees the first blank b and changes to its final state. The initial ID of M_1 is q_0w. By defining $\delta(q_0, b) = (q_1, b, R)$, M_1 reaches the initial state of M. M_1 can be described by Fig. 9.13.

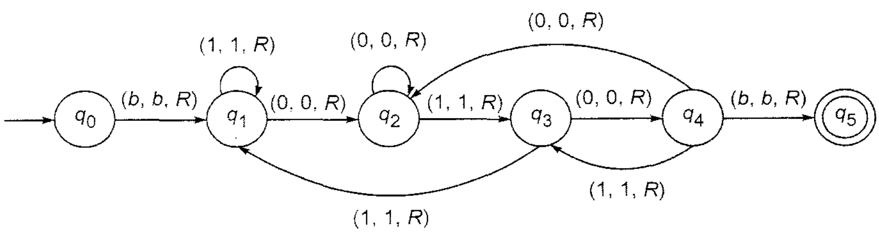

Fig. 9.13 TM for Example 9.14.

Note: q_5 is the unique final state of M_1. By comparing Figs. 9.12 and 9.13 it is easy to see that strings of L are accepted by M_1.

EXAMPLE 9.15

Design a TM that reads a string in $\{0, 1\}^*$ and erases the rightmost symbol.

Solution

The required TM M is given by

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, b\}, \delta, q_0, b, \{q_4\})$$

where δ is defined by

$$\delta(q_0, 0) = (q_1, 0, R) \qquad \delta(q_0, 1) = (q_1, 1, R) \qquad (\mathrm{R}_1)$$

$$\delta(q_1, 0) = (q_1, 0, R) \qquad \delta(q_1, 1) = (q_1, 1, R) \qquad (\mathrm{R}_2)$$

$$\delta(q_1, b) = (q_2, b, L) \qquad (\mathrm{R}_3)$$

$$\delta(q_2, 0) = (q_3, b, L) \qquad \delta(q_2, 1) = (q_3, b, L) \qquad (\mathrm{R}_4)$$

$$\delta(q_3, 0) = (q_3, 0, L) \qquad \delta(q_3, 1) = (q_3, 1, L) \qquad (\mathrm{R}_5)$$

$$\delta(q_3, b) = (q_4, b, R) \qquad (\mathrm{R}_6)$$

Let w be the input string. By (R_1) and (R_2), M reads the entire input string w. At the end, M is in state q_1. On seeing the blank to the right of w, M reaches the state q_2 and moves left. The rightmost string in w is erased (by (R_4)) and the state becomes q_3. Afterwards M moves to the left until it reaches the left-end of w. On seeing the blank b to the right of w, M changes its state to q_4, which is the final state of M. From the construction it is clear that the rightmost symbol of w is erased.

EXAMPLE 9.16

Construct a TM that accepts $L = \{0^{2^n} \mid n \geq 0\}$.

Solution

Let w be an input string in $\{0\}^*$. The TM accepting L functions as follows:

1. It writes b (blank symbol) on the leftmost 0 of the input string w. This is done to mark the left-end of w.
2. M reads the symbols of w from left to right and replaces the alternate 0's with x's.
3. If the tape contains a single 0 in step 2, M accepts w.
4. If the tape contains more than one 0 and the number of 0's is odd in step 2, M rejects w.
5. M returns the head to the left-end of the tape (marked by blank b in step 1).
6. M goes to step 2.

Each iteration of step 2 reduces w to half its size. Also whether the number of 0's seen is even or odd is known after step 2. If that number is odd and greater than 1, w cannot be 0^{2^n} (step 4). In this case M rejects w. If the number of 0's seen is 1 (step 3), M accepts w (In this case 0^{2^n} is reduced to 0 in successive stages of step 2).

We define M by

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_f, q_t\}, \{0\}, \{0, x, b\}, \delta, q_0, b, \{q_f\})$$

where δ is defined by Table 9.12.

TABLE 9.12 Transition Table for Example 9.16

Present state	Tape symbol		
	0	b	x
q_0	bRq_1	bRq_t	xRq_t
q_1	xRq_2	bRq_f	xRq_1
q_2	$0Rq_3$	bRq_4	xRq_2
q_3	xRq_2	bRq_6	xRq_3
q_4	$0Lq_4$	bRq_1	xLq_4
q_f	—	—	—
q_t	—	—	—

From the construction, it is apparent that the states are used to know whether the number of 0's read is odd or even.

We can see how M processes 0000.

$$\begin{aligned} q_00000 &\vdash bq_1000 \vdash bxq_200 \vdash bxq_30 \vdash bx0xq_2b \\ &\vdash bx0q_4xb \vdash bxq_40xb \vdash bq_4x0xb \vdash q_4bx0xb \\ &\vdash bq_1x0xb \vdash bxq_10xb \vdash bxxq_2xb \vdash bxxxq_2b \\ &\vdash bxxq_4xb \vdash bxq_4xxb \vdash bq_4xxxb \vdash q_4bxxxb \\ &\vdash bq_1xxxb \vdash bxq_1xxb \vdash bxxq_1xb \vdash bxxxq_1b \\ &\vdash bxxxbq_f. \end{aligned}$$

Hence M accepts w.

Also note that M always halts. If M reaches q_f, the input string w is accepted by M. If M reaches q_t, w is not accepted by M; in this case M halts in the trap state.

EXAMPLE 9.17

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

where δ is given by

$$\delta(q_0, 0) = (q_1, 1, R) \qquad (R_1)$$

$$\delta(q_1, 1) = (q_0, 0, R) \qquad (R_2)$$

$$\delta(q_1, b) = (q_2, b, R) \qquad (R_3)$$

Find $T(M)$.

Solution

Let $w \in T(M)$. As $\delta(q_0, 1)$ is not defined, w cannot start with 1. From (R_1) and (R_2), we can conclude that M starts from q_0 and comes back to q_0 after reaching 01.

So, $q_0(01)^n \overset{*}{\vdash} (10)^n q_0$. Also, $q_00b \vdash 1q_1b \vdash 1bq_2$.

So, $(01)^n0 \in T(M)$. Also, $(01)^n0$ is the only string that makes M move from q_0 to q_2. Hence, $T(M) = \{(01)^n0 \mid n \geq 0\}$.

SELF-TEST

Choose the correct answer to Questions 1–10:

1. For the standard TM:
(a) $\Sigma = \Gamma$
(b) $\Gamma \subseteq \Sigma$
(c) $\Sigma \subseteq \Gamma$
(d) Σ is a proper subset of Γ.

2. In a standard TM, $\delta(q, a)$, $q \in Q$, $a \in \Gamma$ is
(a) defined for all $(q, a) \in Q \times \Gamma$
(b) defined for some, not necessarily for all $(q, a) \in Q \times \Gamma$
(c) defined for no element (q, a) of $Q \times \Gamma$
(d) a set of triples with more than one element.

3. If $\delta(q, x_i) = (p, y, L)$, then
(a) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1x_2 \ldots x_{i-2}px_{i-1}yx_{i+1} \ldots x_n$
(b) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1x_2 \ldots x_{i-1}ypx_{i+1} \ldots x_n$
(c) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1 \ldots x_{i-3}px_{i-2}x_{i-1}yx_{i+1} \ldots x_n$
(d) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1 \ldots x_{i+1}pyx_{i+2} \ldots x_n$

4. If $\delta(q, x_i) = (p, y, R)$, then
(a) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1x_2 \ldots x_{i-1}ypx_{i+1} \ldots x_n$
(b) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1x_2 \ldots x_ipx_{i+1} \ldots x_n$
(c) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1x_2 \ldots x_{i-1}px_ix_{i+1} \ldots x_n$
(d) $x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \vdash x_1x_2 \ldots x_{i-1}ypx_{i+1} \ldots x_n$

5. If $\delta(q, x_1) = (p, y, L)$, then
(a) $qx_1x_2 \ldots x_n \vdash pyx_2 \ldots x_n$
(b) $qx_1x_2 \ldots x_n \vdash ypx_2 \ldots x_n$
(c) $qx_1x_2 \ldots x_n \vdash pbx_1 \ldots x_n$
(d) $qx_1x_2 \ldots x_n \vdash pbx_2 \ldots x_n$

6. If $\delta(q, x_n) = (p, y, R)$, then
(a) $x_1 \ldots x_{n-1}qx_n \vdash pyx_2x_3 \ldots x_n$
(b) $x_1 \ldots x_{n-1}qx_n \overset{*}{\vdash} pyx_2x_3 \ldots x_n$
(c) $x_1 \ldots x_{n-1}qx_n \vdash x_1x_2 \ldots x_{n-1}ypb$
(d) $x_1 \ldots x_{n-1}qx_n \overset{*}{\vdash} x_1x_2 \ldots x_{n-1}ypb$

7. For the TM given in Example 9.6:
(a) $q_01b11 \overset{*}{\vdash} bq_f11bb1$
(b) $q_01b11 \vdash bq_f11bb1$
(c) $q_01b11 \vdash 1q_0b111$
(d) $q_01b11 \vdash q_2b11bb1$

8. For the TM given in Example 9.4:
(a) 011 is accepted by M
(b) 001 is accepted by M
(c) 00 is accepted by M
(d) 0011 is accepted by M.

9. For the TM given in Example 9.5:
(a) 1 is accepted by M
(b) 11 is accepted by M
(c) 111 is accepted by M
(d) 11111 is accepted by M

10. In a standard TM $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$ the blank symbol b is
(a) in $\Sigma - \Gamma$
(b) in $\Gamma - \Sigma$
(c) $\Gamma \cap \Sigma$
(d) none of these

EXERCISES

9.1 Draw the transition diagram of the Turing machine given in Table 9.1.

9.2 Represent the transition function of the Turing machine given in Example 9.2 as a set of quintuples.

9.3 Construct the computation sequence for the input $1b11$ for the Turing machine given in Example 9.5.

9.4 Construct the computation sequence for strings 1213, 2133, 312 for the Turing machine given in Example 9.8.

9.5 Explain how a Turing machine can be considered as a computer of integer functions (i.e. as one that can compute integer functions; we shall discuss more about this in Chapter 11).

9.6 Design a Turing machine that converts a binary string into its equivalent unary string.

9.7 Construct a Turing machine that enumerates $\{0^n1^n \mid n \geq 1\}$.

9.8 Construct a Turing machine that can accept the set of all even palindromes over $\{0, 1\}$.

9.9 Construct a Turing machine that can accept the strings over $\{0, 1\}$ containing even number of 1's.

9.10 Design a Turing machine to recognize the language $\{a^nb^nc^m \mid n, m \geq 1\}$.

9.11 Design a Turing machine that can compute proper subtraction, i.e. $m \dot{-} n$, where m and n are positive integers. $m \dot{-} n$ is defined as $m - n$ if $m > n$ and 0 if $m \leq n$.