CHAPTER 6

TURING MACHINES

6.1 INTRODUCTION

In this chapter we investigate a third type of recognizing device, the Turing machine. The Turing machine has been proposed as a mathematical model for describing procedures. Since our intuitive notion of a procedure as a finite sequence of instructions which can be mechanically carried out is not mathematically precise, we can never hope to show formally that it is equivalent to the precise notion of a Turing machine. However, from the definition of a Turing machine, it will be readily apparent that any computation that can be described by means of a Turing machine can be mechanically carried out. Thus the definition is not too broad. It can also be shown that any computation that can be performed on a modern-day digital computer can be described by means of a Turing machine. Thus if one ever found a procedure that fitted the intuitive notions, but could not be described by means of a Turing machine, it would indeed be of an unusual nature since it could not possibly be programmed for any existing computer. Many other formalizations of a procedure have been proposed, and they have been shown to be equivalent to the Turing machine formalization. This strengthens our belief that the Turing machine is general enough to encompass the intuitive notion of a procedure. It has been hypothesized by Church that any process which could naturally be called a procedure can be realized by a Turing machine. Subsequently, computability by a Turing machine has become the accepted definition of a procedure. We shall accept Church's hypothesis and simply substitute the formal definition of a Turing machine for the intuitive notion of a procedure.

6.2 DEFINITIONS AND NOTATION

Specifications for the Turing machine have been given in various ways in the literature. We begin with the discussion of a basic model, as shown in Fig. 6.1. Later we investigate other models of the Turing machine, and show that all these models are equivalent. The basic model has a *finite control*, an input tape which is divided into cells, and a *tape head* which scans one cell of the tape at a time. The tape has a leftmost cell but is infinite to the right. Each cell of the tape may hold exactly one of a finite number of *tape*

symbols. Initially, the n leftmost cells, for some finite n, hold the *input*, a string of symbols chosen from a subset of the tape symbols called the *input symbols*. The remaining infinity of cells hold the *blank*, a special tape symbol which is not an input symbol.

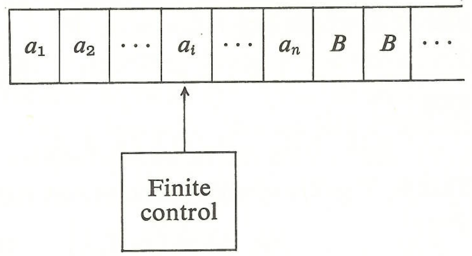

Fig. 6.1. Basic Turing machine.

In a *move* of the Turing machine, depending upon the symbol scanned by the tape head and the state of the finite control, the machine:

1. changes state.
2. prints a nonblank symbol on the tape cell scanned, replacing what was written there.
3. moves its head left or right one cell.

Note that the difference between a Turing machine and a two-way finite automaton lies in the former's ability to change symbols on its tape.

Formally, a Turing machine (Tm) is denoted $T = (K, \Sigma, \Gamma, \delta, q_0, F)$, where:

K is the finite set of *states*.

Γ is the finite set of allowable *tape symbols*. One of these, usually denoted B, is the *blank*.

Σ, a subset of Γ not including B, is the set of *input symbols*.

δ is the *next move function*, a mapping from $K \times \Gamma$ to $K \times (\Gamma - \{B\}) \times \{L, R\}$.† δ may, however, be undefined for some arguments.

q_0 in K is the *start state*.

$F \subseteq K$ is the set of *final states*.

We denote a *configuration* of the Turing machine T by (q, α, i). Here q, the current state of T, is in K. α is a string in $(\Gamma - \{B\})^*$ and is the nonblank portion of the tape. Note that if the tape head ever leaves a cell, it

† We have not allowed a Tm to print a blank for simplicity in defining the configurations. However, a Tm could have another symbol which is treated exactly as if it were the blank except for the fact that the Tm is allowed to print this pseudo blank symbol. Thus, no extra power results if we allow blanks to be printed. In informal discussion, we often allow the printing of a blank, knowing that one could use a different, but equivalent, symbol instead.

must print a nonblank symbol on the cell, so the tape of T will always consist of a block of nonblank symbols (here α is that block), with an infinity of blanks to the right. Finally, i is an integer, the distance of the tape head of T from the left end of α.

We define a *move* of T as follows. Let $(q, A_1A_2\ldots A_n, i)$ be a configuration of T, where $1 \leqq i \leqq n + 1$. If

$$1 \leqq i \leqq n \qquad \text{and} \qquad \delta(q, A_i) = (p, A, R),$$

then

$$(q, A_1A_2\ldots A_n, i) \vdash_T (p, A_1A_2\ldots A_{i-1}AA_{i+1}\ldots A_n, i + 1).$$

That is, T prints symbol A and moves right. If

$$\delta(q, A_i) = (p, A, L) \qquad \text{and} \qquad 2 \leqq i \leqq n,$$

then

$$(q, A_1A_2\ldots A_n, i) \vdash_T (p, A_1A_2\ldots A_{i-1}AA_{i+1}\ldots A_n, i - 1).$$

Here T prints A and moves left, but not off the left end of the tape. If $i = n + 1$, the tape head is scanning the blank, B. If $\delta(q, B) = (p, A, R)$, then

$$(q, A_1A_2\ldots A_n, n + 1) \vdash_T (p, A_1A_2\ldots A_nA, n + 2).$$

If, instead, $\delta(q, B) = (p, A, L)$, then

$$(q, A_1A_2\ldots A_n, n + 1) \vdash_T (p, A_1A_2\ldots A_nA, n).$$

If two configurations are related by $\vdash_T$, we say that the second results from the first by one move. If one configuration results from another by some finite number of moves, including zero moves, they are related by the relation $\vdash^{*}_{T}$.

The *language accepted* by T is the set of those words in Σ^* which cause T to enter a final state when placed, justified at the left, on the tape of T, with T in state q_0, and the tape head of T at the leftmost cell. Formally, the language accepted by $T = (K, \Sigma, \Gamma, \delta, q_0, F)$ is

$$\{w \mid w \text{ in } \Sigma^* \text{ and } (q_0, w, 1) \vdash^{*}_{T} (q, \alpha, i) \text{ for some } q \text{ in } F,\ \alpha \text{ in } \Gamma^* \text{ and integer } i\}.$$

Given a Tm recognizing a language L, we assume without loss of generality that the Tm *halts*, i.e., has no next move whenever the input is accepted. However, for words not accepted, it is possible that the Tm will not halt.

Example 6.1. Consider the following Tm that recognizes the context-free language $L = \{0^n1^n \mid n \geqq 1\}$. Let $T = (K, \Sigma, \Gamma, \delta, q_0, F)$. Here,

$$K = \{q_0, q_1, \ldots, q_5\}, \quad \Sigma = \{0, 1\}, \quad \Gamma = \{0, 1, B, X, Y\}, \quad F = \{q_5\},$$

and δ is defined as follows.

1. $\delta(q_0, 0) = (q_1, X, R)$. ($T$ will alternately replace a 0 by X, then a 1 by Y. In state q_0, a 0 is replaced by an X, and T moves right in state q_1 looking for a 1.)
2. a) $\delta(q_1, 0) = (q_1, 0, R)$
   b) $\delta(q_1, Y) = (q_1, Y, R)$
   c) $\delta(q_1, 1) = (q_2, Y, L)$
   (T moves right in state q_1 (Rules 2a and 2b). When a 1 is found, it is changed to a Y, and the state becomes q_2 (Rule 2c). In q_2, we see that T moves left, looking for a 0 to convert to an X. Moving left, T will encounter a block of Y's, then, perhaps, a block of 0's, then an X.)
3. a) $\delta(q_2, Y) = (q_2, Y, L)$
   b) $\delta(q_2, X) = (q_3, X, R)$
   c) $\delta(q_2, 0) = (q_4, 0, L)$
   (T moves left, through Y's (3a). If T encounters an X while still in state q_2, there are no more 0's to convert. T goes to state q_3 to check that no

Configuration	Rule used	Configuration	Rule used
$(q_0, \overset{\downarrow}{0}00111, 1)$	start	$(q_4, X\overset{\downarrow}{X}0YY1, 2)$	3c
$(q_1, X\overset{\downarrow}{0}0111, 2)$	1	$(q_0, XX\overset{\downarrow}{0}YY1, 3)$	4b
$(q_1, X0\overset{\downarrow}{0}111, 3)$	2a	$(q_1, XXX\overset{\downarrow}{Y}Y1, 4)$	1
$(q_1, X00\overset{\downarrow}{1}11, 4)$	2a	$(q_1, XXXY\overset{\downarrow}{Y}1, 5)$	2b
$(q_2, X0\overset{\downarrow}{0}Y11, 3)$	2c	$(q_1, XXXYY\overset{\downarrow}{1}, 6)$	2b
$(q_4, X\overset{\downarrow}{0}0Y11, 2)$	3c	$(q_2, XXXY\overset{\downarrow}{Y}Y, 5)$	2c
$(q_4, \overset{\downarrow}{X}00Y11, 1)$	4a	$(q_2, XXX\overset{\downarrow}{Y}YY, 4)$	3a
$(q_0, X\overset{\downarrow}{0}0Y11, 2)$	4b	$(q_2, XX\overset{\downarrow}{X}YYY, 3)$	3a
$(q_1, XX\overset{\downarrow}{0}Y11, 3)$	1	$(q_3, XXX\overset{\downarrow}{Y}YY, 4)$	3b
$(q_1, XX0\overset{\downarrow}{Y}11, 4)$	2a	$(q_3, XXXY\overset{\downarrow}{Y}Y, 5)$	5a
$(q_1, XX0Y\overset{\downarrow}{1}1, 5)$	2b	$(q_3, XXXYY\overset{\downarrow}{Y}, 6)$	5a
$(q_2, XX0\overset{\downarrow}{Y}Y1, 4)$	2c	$(q_3, XXXYYY\overset{\downarrow}{\ }, 7)$	5a
$(q_2, XX\overset{\downarrow}{0}YY1, 3)$	3a	$(q_5, XXXYYYY\overset{\downarrow}{\ }, 8)$	5b

Fig. 6.2. Computation accepting 000111.