13

Turing Machines

13.1. DESCRIPTION AND EXAMPLES OF TURING MACHINES

In Chapter 12 we showed that the fundamental, intuitively obvious requirements to which any algorithm must conform are those of determinacy, generality, and applicability (efficacy). In addition, the result of an algorithmic procedure must be completely independent of the person executing it. The executor merely acts like a machine: there is no "creative" work involved here, because the executor needs only to follow instructions. If this is so, then why not delegate the execution of the algorithm to a machine? This chapter will present one class of machines capable of executing such tasks.

The above properties of an algorithm also pertain to a machine executing this algorithm. To begin with, such a machine must be fully determinate, operating within the specified rules. Second, it must allow the input of a variety of "initial data," that is, of a variety of individual problems from a given class of problems. Third, the specified operational rules for the machine and the class of problems which can be solved must be matched in such a way that the result of machine operation will always be "readable" (that is, the machine will give a useful result).

There are many constants capable of executing algorithms. The most graphic of these is the scheme proposed in 1936 by the English mathematician Turing. We shall now describe one of the possible variants of this machine.

The basic component of our Turing machine is an infinitely long tape divided lengthwise into squares. The tape extends in only one direction (to the right), so that we can meaningfully talk about a "leftmost" square. Each square may contain only one symbol s_i from a finite alphabet $\{s_0, \ldots, s_n\}$. We shall ascribe a special significance to the symbol s_0: its presence in a square shall denote that the square is blank. In any tape, the number of nonblank squares is always finite (but as large as desired), all the other squares being blank.

The second component of the Turing machine is a *read-erase-record head*. This special device can move along the tape, either to the left or to the right, one square at a time. Upon an external command, the head can erase a symbol present in the tape square that happens to face the head at a given moment, and it can print another one in its stead. The external commands causing these actions are issued by a *controller*, a device which is itself governed by the signals generated by the head (these signals indicate the presence of symbols s_i in a given tape square). The controller operates in discrete time $t = 0, 1, 2, \ldots$, and it may assume a finite number $m + 1$ of internal states $q_0, \ldots, q_m$. Its input consists of symbols of s_i read and generated by the head, while its output consists of commands to the head (these commands indicate what symbol, if any, should be printed in a given tape square, as well as the direction of motion of the head). For example, assume that at time t the head faces the lth square from the left, that this square contains the symbol s_i, and that the controller is in state q_j. The head reads the symbol s_i and generates a signal corresponding to it. In response to this, the controller generates a symbol s_k which causes the head to erase the old symbol s_i and print s_k on the tape. Then the controller produces one of the symbols R, L, S ("right," "left," "stop"), in compliance with which the head moves one square to the right or left or stays put. After this, the controller assumes a new state q_r, which is uniquely determined by the previous state q_j and the signal s_i. After the entire operation has been completed (at time $t + 1$), the lth square contains the symbol s_k, the controller is in state q_r, and the head is situated opposite either the $(l + 1)$st, the $(l - 1)$st, or the lth square (depending on whether the motion command was R, L, or S). Thus, the controller is a sequential machine with two output converters. Its inputs are symbols from the alphabet $\{s_0, \ldots, s_n\}$, received from the read-record head. Its states are symbols from the alphabet $\{q_0, \ldots, q_m\}$. Its first output is a signal commanding the head to print a symbol from the alphabet $\{s_0, \ldots, s_n\}$, whereas its second output is a signal commanding a shift of the head and belonging to the alphabet $\{R, L, S\}$. The operation of this s-machine can be specified by means of three tables—those for an automaton and for two converters. However, it is customary to combine these into one basic table. Thus the automaton Table 13.1, the first converter Table 13.2, and the second converter Table 13.3 may all be combined, in that order, into Table 13.4, which fully describes the operation of this Turing machine. Again, if the basic table of a Turing machine is given, then its operation is uniquely specified.

Table 13.1

	s_0	s_1	s_2
q_0	q_0	q_1	q_0
q_1	q_0	q_0	q_1

Table 13.2

	s_0	s_1	s_2
q_0	s_2	s_0	s_1
q_1	s_1	s_1	s_2

Table 13.3

	s_0	s_1	s_2
q_0	R	R	L
q_1	S	L	R

Table 13.4

	s_0	s_1	s_2
q_0	$q_0 s_2 \mathrm{R}$	$q_1 s_0 \mathrm{R}$	$q_0 s_1 \mathrm{L}$
q_1	$q_0 s_1 \mathrm{S}$	$q_0 s_1 \mathrm{L}$	$q_1 s_2 \mathrm{R}$

State of the controller shall denote the *rest state* of the Turing machine; that is, row q_0 of the basic table has the following properties: (1) The first symbol in every square of this row shall always be q_0 (never q_j if $j \neq 0$); (2) The second symbol of each square will be s_i, the same symbol as in the respective *column* heading (never s_k if $k \neq i$); (3) The third symbol of every square shall be the symbol S (never R or L). For an illustration of row q_0, see Tables 13.4 and 13.5.

Table 13.5

	s_0	s_1	s_2
q_0	$q_0 s_0 \mathrm{S}$	$q_0 s_1 \mathrm{S}$	$q_0 s_2 \mathrm{S}$
q_1	$q_1 s_2 \mathrm{L}$	$q_1 s_0 \mathrm{R}$	$q_0 s_1 \mathrm{S}$

Now, if the controller is at any time t in state q_0, then whatever the position of the head, and whatever the symbol in the corresponding tape square, the controlling device will remain in state q_0, the head will not move, and the tape entries will remain the same as before. To simplify the basic table, we shall therefore omit row q_0 (see Table 13.6).

For simplicity, we shall also assume that the alphabet $\{s_i\}$ consists of only two symbols; blank (that is, 0) and nonblank (that is, 1).

Table 13.6

	s_0	s_1	s_2
q_1	$q_1 s_2 \mathrm{L}$	$q_1 s_0 \mathrm{R}$	$q_0 s_1 \mathrm{S}$

Table 13.7

Machine A

	0	1
q_1	q_0 1S	q_1 1R

Now let us discuss a few simple Turing machines.

1) *Machine A* (Table 13.7). This machine operates as follows. Assume that at $t = 0$, the controller is in state q_1, and the head faces a nonblank square. The machine then "looks" for the first blank square to the right, prints the symbol 1 in it, and stops. If, however, the head faces a blank square at $t = 0$, then the machine prints 1 in that square, and stops (no motion of the head). Tables 13.8 and 13.9 illustrate two possible modes of operation of this machine. (A bar above a tape square indicates that the head faces that square

Table 13.8

Time	Tape printout
0	. . . 1 $\overline{q_1}$ 1 1 1 0 0 . . .
1	. . . 1 1 $\overline{q_1}$ 1 1 0 0 . . .
2	. . . 1 1 1 $\overline{q_1}$ 1 0 0 . . .
3	. . . 1 1 1 1 $\overline{q_1}$ 0 0 . . .
4	. . . 1 1 1 1 1 $\overline{q_1}$ 0 . . .
5	. . . 1 1 1 1 1 $\overline{q_0}$ 0 . . .

Table 13.9

Time	Tape printout
0	. . . 1 0 $\overline{q_1}$ 0 0 1 . . .
1	. . . 1 0 $\overline{q_1}$ 0 1 . . .

at that time, and the symbol q shows the state of the controller.) The dots represent those tape squares where the symbols are known to remain unchanged throughout the operating time considered (the head does not reach them).

Table 13.10

Machine B

	0	1
q_1	$q_1 0 L$	$q_0 0 L$

2) *Machine B* (Table 13.10). This machine can also assume only one state (aside from state q_0). If the head faces, at $t = 0$, a nonblank square, it erases the symbol 1 in it, moves one square to the left, and stops. If the head faces, at $t = 0$, a blank square, it moves to the first nonblank square on the left, erases the symbol 1 in it, and moves one additional square to the left (see Table 13.11).

Table 13.11

Time	Tape printout
0	$\ldots 0\,1\,1\,1\,0\,\overset{q_1}{\overline{0}}\ldots$
1	$\ldots 0\,1\,1\,\overset{q_1}{\overline{1}}\,0\,0\ldots$
2	$\ldots 0\,1\,\overset{q_1}{\overline{1}}\,0\,0\,0\ldots$
3	$\ldots 0\,\overset{q_0}{\overline{1}}\,0\,0\,0\,0\ldots$

3) *Machine C* (Table 13.12). At $t = 0$, this machine may face either a blank or a nonblank square. The head then moves to the right until it finds the first group of symbols 1 after a group of

Table 13.12

Machine C

	0	1
q_1	$q_2 0 R$	$q_1 1 R$
q_2	$q_2 0 R$	$q_3 1 R$
q_3	$q_0 0 L$	$q_3 1 R$

zeros, and stops at the last 1 of that group. One version of this machine is shown in Table 13.13.

Table 13.13

Time	Tape printout
0	q_1 . . . 0 1 1 0 0 1 1 0 0 . . .
1	q_1 . . . 0 1 1 0 0 1 1 0 0 . . .
2	q_1 . . . 0 1 1 0 0 1 1 0 0 . . .
3	q_2 . . . 0 1 1 0 0 1 1 0 0 . . .
4	q_2 . . . 0 1 1 0 0 1 1 0 0 . . .
5	q_3 . . . 0 1 1 0 0 1 1 0 0 . . .
6	q_3 . . . 0 1 1 0 0 1 1 0 0 . . .
7	q_0 . . . 0 1 1 0 0 1 1 0 0 . . .

In some cases the Turing machine may be incompletely specified, in that some of the squares of the basic table contain no symbols. This is permissible in those cases where one can predict that these combinations of machine states and tape symbols will never occur. Consider an example.

Table 13.14

Machine D

	0	1
q_1		$q_2$1R
q_2	$q_3$1R	$q_2$1R
q_3	$q_3$1R	$q_4$1L
q_4		$q_0$0L

4) *Machine D* (Table 13.14). This machine searches for the group of zeros between the first two groups of ones to the right of the position of the head at $t = 0$. It then replaces all but one of these zeros with ones. If the combinations q_1, 0 and q_4, 0 are avoided at $t = 0$, they will not occur in the future because state q_1 will never occur, whereas the machine will assume state q_4 only after the last symbol 1 has been printed. One variant of this machine is shown in Table 13.15.

We shall sometimes deal with machines having not one, but several rest states (q_0', q_0'', and so on). Consider a typical example.

5) *Machine E* (Table 13.16). At $t = 0$, the head of this machine always faces a nonblank square. Then, depending on whether the next square to the left contains 0 or 1, the machine assumes either state q_0' or q_0'', and stops facing the initial square. Variants of this machine are shown in Tables 13.17 and 13.18.

Table 13.15

Time	Tape printout
0	... 0 q_1 1 1 0 0 0 1 1 0 0 ...
1	... 0 1 q_2 1 0 0 0 1 1 0 0 ...
2	... 0 1 1 q_2 0 0 0 1 1 0 0 ...
3	... 0 1 1 1 q_2 0 0 1 1 0 0 ...
4	... 0 1 1 1 1 q_3 0 1 1 0 0 ...
5	... 0 1 1 1 1 1 q_3 1 1 0 0 ...
6	... 0 1 1 1 1 1 1 q_3 1 0 0 ...
7	... 0 1 1 1 1 1 q_4 1 1 0 0 ...
8	... 0 1 1 1 1 q_0 0 1 1 0 0

In concluding this section we shall present, without special explanations, a few Turing machines which we shall need at a later stage.

6) *Machine F* (Table 13.19). This machine searches for the nearest group of 1's which follow a group of zeros to the left of the position of the head at $t = 0$.

Table 13.16

Machine E

	0	1
q_1		$q_2 1 L$
q_2	$q_0' 0 R$	$q_0'' 1 R$

Table 13.17

Time	Tape printout
	q_1
0	... 0 0 1 $\overline{1}$ 1 ...
	q_2
1	... 0 0 $\overline{1}$ 1 1 ...
	q_0''
2	... 0 0 1 $\overline{1}$ 1 ...

Table 13.18

Time	Tape printout
	q_1
0	... 0 0 $\overline{1}$ 1 ...
	q_2
1	... 0 $\overline{0}$ 1 1 ...
	q_0'
2	... 0 0 $\overline{1}$ 1 ...

Table 13.19

Machine F

	0	1
q_1	$q_2 0 L$	$q_1 1 L$
q_2	$q_2 0 L$	$q_0 1 S$

Table 13.20

Machine G

	0	1
q_1	$q_0 0 S$	$q_1 0 L$

Table 13.21

Machine H

	0	1
q_1	$q_0 1 S$	$q_2 1 R$
q_2	$q_1 0 R$	$q_2 1 R$

7) *Machine G* (Table 13.20). This machine erases all the 1's (if such symbols are present) to the left of the position of the head at $t = 0$, and continues doing so until it encounters a 0.

8) *Machine H* (Table 13.21). It differs from *Machine A* only in that it prints the symbol 1 not in the first but in the second blank square on the right.

9) *Machine I* (Table 13.22). This machine starts at a nonblank square, erases the symbol 1 in it, and transfers it to the nearest blank square on the left (in other words, it shifts a group of ones one square to the left of the starting position).

10) *Machine K* (Table 13.23). It erases the symbol 1 in the square on the right of the initial one (if that square contains a 1).

Table 13.22
Machine I

	0	1
q_1		$q_2 0 L$
q_2	$q_0 1 L$	$q_2 1 L$

Table 13.23
Machine K

	0	1
q_1	$q_2 0 R$	$q_2 1 R$
q_2	$q_0 0 S$	$q_0 0 S$

11) *Machine L* (Table 13.24). It starts at a nonblank square, moves to the left, and stops at the second blank square to the left of the first group of ones.

Table 13.24
Machine L

	0	1
q_1	$q_0 0 L$	$q_1 1 L$

Table 13.25
Machine M

	0	1
q_1	$q_0' 0 S$	$q_0'' 1 S$

12) *Machine M* (Table 13.25). It can assume two rest states. Depending on whether it faces a blank or nonblank square, it assumes state q_0' or q_0''.

13.2. THE COMPOSITION OF TURING MACHINES

As we have just seen, what a Turing machine does is uniquely determined by the controller functioning in accordance with a basic table. We shall assume that the machine always starts from an initial state denoted by q_1, and that it assumes the rest state q_0 when it ceases to work. Now we can define the operations on Turing machines so that we can derive new basic tables from the given ones. Thus imagine that we have two machines T_1 and T_2, and that at $t = 0$ the collection of symbols on the tape is such that T_1 starts operating. At this point, T_1 is in state q_1^1, and the head is opposite the l_0th square. Then, at $t = t_0^1$ machine T_1 assumes the rest state q_0^1, and the head stops opposite the l_0^1th square. Now machine T_1 shuts off, and machine T_2 takes over, starting from state q_1^2, the head of T_2 at $t = t_0^1$ facing the same square l_0^1 at which T_1 ceased operating. Then, at $t = t_0^2$, T_2 shuts off and assumes the rest state q_0^2, while its head stops opposite square l_0^2. This consecutive operation of machines T_1 and T_2, is equivalent to the operation of a single Turing machine

T, the basic table of which is synthesized according to the following rule: if the controllers of T_1 and T_2 can assume k_1 and k_2 states*, respectively, then the controller of T can have $k_1 + k_2$ states, the initial and rest states of T being q_1^1 and q_0^2, respectively. The basic table of T consists of two parts, of which the top describes T_1, and the bottom T_2. The rest state q_0^1 of T_1 is the initial state q_1^2 of T_2. For example, if T_1 is machine F (Table 13.19) and T_2 is machine G (Table 13.20), then machine T (Table 13.26) will have a table with $2 + 1 = 3$ states, where $q_0 = q_0^2$ is the rest state of G. If we recode the states of T, Table 13.26 will take the form of Table 13.27.

Table 13.26

	0	1
q_1'	$q_2'0L$	$q_1'1L$
q_2'	$q_2'0L$	$q_1''1S$
q_1''	$q_0''0S$	$q_1''0L$

Table 13.27

	0	1
q_1	q_20L	q_11L
q_2	q_20L	q_31S
q_3	q_00S	q_30L

Table 13.28

	0	1
q_1	q_20S	q_10L
q_2	q_30L	q_21L
q_3	q_30L	q_01S

Machine T so obtained is the *product* of T_1 and T_2; that is, $T = T_1 \cdot T_2$. The operation of deriving a third machine from two given ones is the multiplication of machines. Thus Tables 13.26 and 13.27 are tables of machine $F \cdot G$. Multiplication of machines is obviously a noncommutative operation: $T_1 \cdot T_2 \neq T_2 \cdot T_1$ (Table 13.28 shows the product $G \cdot F$, and it obviously differs from Table 13.27). However, multiplication is associative; that is, with three machines T_1, T_2, and T_3, we have $(T_1 \cdot T_2) \cdot T_3 = T_1 \cdot (T_2 \cdot T_3)$.** Accordingly, no parentheses are used in writing the product of several machines.

The operation of *raising to a power* is defined in the usual way: the nth power of machine T is the product obtained by multiplying T by itself n times.

So far we have discussed the multiplication of machines with one rest state. If one of the machines of the product has two or more rest states (for example, if it is machine E or M of the preceding section), the multiplication is the same, but one must indicate which of the rest states of the first constituent machine shall be the initial state of the second machine. For example, if T_1 has two rest states,

*Henceforth, the number of states shall not include the rest states, of which there may be several, as in machine M above.

**From now on, we shall omit the dots in the product of machines.

we shall write the product of T_1 and T_2 as $T=T_1\begin{cases}(1) & T_2\\ (2)\end{cases}$, or as $T=T_1\begin{cases}(1),\\ (2) & T_2\end{cases}$, depending on whether the initial state of T_2 is the first or the second rest state of T_1. Machine T also has two rest states, the first of which is one of the rest states of T_1, while the second is the rest state of T_2.

Now, the meaning of an expression such as

$$T=T_1\begin{cases}(1) & T_2\,T_3,\\ (2) & T_4\end{cases}$$

is also clear here. Here, there are two independent multiplications, involving the first and the second rest states of machine T_1.

There also exists the operation of iteration of a single machine. Thus let machine T_1 have s rest states. We select its rth rest state and make it the initial state of machine T, which is then shown as

$$T=\dot{T}_1\begin{cases}(1),\\ \cdot\\ \cdot\\ \cdot\\ (\dot{r}),\\ \cdot\\ \cdot\\ \cdot\\ (s)\end{cases}$$

This machine is the result of iteration of T_1. Here, the dots above the letters indicate that the rth rest state is made the initial state of the iterating machine T_1. If T_1 has only one rest state, then iteration yields a machine with no rest states.

Henceforth, we shall use the following notation. If we perform iteration on a machine which itself is the result of multiplication and iteration of other machines, then we place corresponding number of dots above those machines whose states (rest or initial) are used in the new machine. For example, the expression

$$T=\dot{T}_1\ddot{T}_2\begin{cases}(1) & \dot{T}_3,\\ (2) & T_4\,\ddot{T}_5,\\ (3) & T_6\end{cases}$$

means that the rest state of machine T_3 is made the initial state of machine T_1 and that the rest state of T_5 is made the initial state of T_2.

Now let us synthesize a machine by means of multiplication and iteration. Let our constituent machines be C, G, L, and M, described in Section 13.1, and let us use them to synthesize a machine N by

Machine N

	0	1
q_1	$q_2$0L	$q_1$1L
q_2	$q_3$0S	$q_4$1S
q_3	$q_5$0R	$q_3$1R
q_4	$q_1$0S	$q_4$0L
q_5	$q_5$0R	$q_6$1R
q_6	$q_0$0L	$q_6$1R

means of the above rules. Our machine N

$$N = \dot{L}M \begin{cases} (1) & C, \\ (2) & \dot{G} \end{cases}$$

will have the basic Table 13.29 (obtained from Tables 13.12, 13.20, 13.23, and 13.25). At $t = 0$, machine N is in state q_1, and its head is opposite a nonblank tape square. It then proceeds to erase all the symbols 1 to the left of its initial position, and continues doing so until it encounters two consecutive blank squares. At this point, the head returns to the right and stops opposite the extreme right nonblank square in that group of 1's opposite it at start of the operation. Table 13.30 shows one variant of N, showing only those tape conditions at which the machine assumes a new state (to reduce clutter, state symbols q_i are indicated only by their ordinal numbers i).

Here the use of iteration yielded a machine N repeating the operation of erasing groups of 1's until it is given a specified command to cease.

13.3. COMPUTATION ON TURING MACHINES

We shall show that whatever the algorithm, there will always be a Turing machine capable of executing this algorithm. To formulate this statement in precise terms, we must first formalize the concept of an algorithm in some manner. Here, we shall use Church's thesis according to which every algorithm is merely a computation of a recursive function. Because of that, we must first define what we mean by "computing" of arithmetical functions on a Turing machine.

To start with, let us specify the representation of natural numbers and zero on the tape of a Turing machine. We shall use a code in which numbers are written in the natural ("unary") system of notation, so that a number n is represented by $n+1$ symbols 1 located in consecutive squares of the tape. Thus zero is represented by a single symbol 1, unity—by two symbols 1, etc.*

*We could represent n by n consecutive symbols 1, but then we would have to represent zero by a blank square. This would interfere with our scheme, in which we need blank squares both for separating numbers and for carrying out computations.

Table 13.30

Time	Tape printout
0	1 1 0 0 1 1 1 0 1 1 0 1 1 $\overline{1}$ 1 0
. . . .	. . . . .
	1 1 0 0 1 1 1 0 1 1 $\overline{0}$ 1 1 1 1 0
	1 1 0 0 1 1 1 0 1 $\overline{1}$ 0 1 1 1 1 0
	1 1 0 0 1 1 1 0 1 $\overline{1}$ 0 1 1 1 1 0
. . . .	. . . . .
	1 1 0 0 1 1 1 $\overline{0}$ 0 0 0 1 1 1 1 0
	1 1 0 0 1 1 1 $\overline{0}$ 0 0 0 1 1 1 1 0
	1 1 0 0 1 1 $\overline{1}$ 0 0 0 0 1 1 1 1 0
	1 1 0 0 1 1 $\overline{1}$ 0 0 0 0 1 1 1 1 0
. . . .	. . . . .
	1 1 0 $\overline{0}$ 0 0 0 0 0 0 0 1 1 1 1 0
	1 1 0 $\overline{0}$ 0 0 0 0 0 0 0 1 1 1 1 0
	1 1 $\overline{0}$ 0 0 0 0 0 0 0 0 1 1 1 1 0
	1 1 $\overline{0}$ 0 0 0 0 0 0 0 0 1 1 1 1 0
	1 1 0 $\overline{0}$ 0 0 0 0 0 0 0 1 1 1 1 0
. . . .	. . . . .
	1 1 0 0 0 0 0 0 0 0 0 $\overline{1}$ 1 1 1 0
	1 1 0 0 0 0 0 0 0 0 0 1 $\overline{1}$ 1 1 0
. . . .	. . . . .
	1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 $\overline{0}$
	1 1 0 0 0 0 0 0 0 0 0 1 1 1 $\overline{1}$ 0

Two numbers are said to be located next to each other if their coded expressions are separated by a single blank square. Thus Table 13.31 contains consecutively (from the left) the numbers 3, 0, and 2.

Table 13.31

0 0 1 1 1 1 0 1 0 1 1 1 0

Now let us specify the recording and readout of arguments and values of functions on the tape. Thus assume we want machine T to compute the value of the function $\varphi(x_1, \ldots, x_n)$ of n variables $x_1, \ldots, x_n$ if the values of the arguments are $x_1 = a_1, \ldots, x_n = a_n$. Recall that the leftmost tape square of our Turing machine is considered the first one. Let us leave the first two squares blank (as in Table 13.31). Then, beginning with the third square, we record (in our code) the n consecutive numbers $a_1, \ldots, a_n$, pertaining to the independent variables $x_1, x_2, \ldots, x_n$. Table 13.32 illustrates this for $n = 3$, where $x_1 = 2$, $x_2 = 1$, and $x_3 = 3$.

Table 13.32

$$0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ \overset{q_1}{\overline{1}}\ 0$$

From now on if there is no risk of confusion, we shall use $x_1, \ldots, x_n$ to denote both the independent variables (arguments) and the specific numerical values assumed by these variables.

We shall also say that the read-record head senses the system of numbers $(x_1, \ldots, x_n)$ in the *standard position* if these numbers are consecutively recorded on the tape, and if the head is opposite the rightmost square involved in the representation of the last number (x_n). For example, see Table 13.32, where the position of the head is, as usual, denoted by a bar.

Now we can define what we mean by computation on a Turing machine. Assume we have machine T and that the following holds at $t = 0$:

a) T is in the initial state q_1;

b) a system of n numbers $(x_1, \ldots, x_n)$ is represented on the tape, and the head senses it in the standard position;

c) all squares to the right of the one opposite the head are blank.

Now T starts working. We shall say that T computes the function $x = \varphi(x_1, \ldots, x_n)$ of n variables if, regardless of what kind of numbers $x_1, \ldots, x_n$ are involved, there arrives a time when

d) T assumes the rest state q_0;

e) the tape represents $(n + 1)$ numbers $x_1, \ldots, x_n, x$ [where $x = \varphi(x_1, \ldots, x_n)$], and the head again senses the entire number system in the standard position;

f) all squares to the right of that opposite the head are blank.

If however, the machine never stops, or if it does stop but condition (e) or (f) are not fulfilled, then this machine does not compute function φ for this particular set of arguments $(x_1, \ldots, x_n)$.

For example, to illustrate a Turing machine which does compute, let $n = 3$ and $\varphi(x_1, x_2, x_3) = x_1 + x_2 + x_3$. Then, for $x_1 = 2$, $x_2 = 1$, and $x_3 = 3$, the starting tape is that of Table 13.32, while the final tape (after the machine has stopped) is that of Table 13.33.

Table 13.33

$$0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,1\,\overset{q_0}{1}\,0\,0$$

Now we shall introduce a few specialized Turing machines.

Machine P is synthesized by multiplication and iteration of machines I, M, C, K, and A of Section 13.1:

$$P = \dot{I}M\begin{cases}(1)\ \dot{C}\\(2)\ KCA.\end{cases}$$

This machine places numbers next to each other. It does this by transposing to the left the nonblank squares in the tape representation of a given number (see Table 13.36). The basic table is shown in Fig. 13.34. However, analysis shows that the same result can be obtained with the machine of Table 13.35, which has half the number of states.

Table 13.34

Machine P

	0	1
q_1		$q_2 0 L$
q_2	$q_3 1 L$	$q_2 1 L$
q_3	$q_4 0 S$	$q_7 1 S$
q_4	$q_5 0 R$	$q_4 1 R$
q_5	$q_5 0 R$	$q_6 1 R$
q_6	$q_1 0 L$	$q_6 1 R$
q_7	$q_8 0 R$	$q_8 1 R$
q_8	$q_9 0 S$	$q_9 0 S$
q_9	$q_{10} 0 R$	$q_9 1 R$
q_{10}	$q_{10} 0 R$	$q_{11} 1 R$
q_{11}	$q_{12} 0 L$	$q_{11} 1 R$
q_{12}	$q_0 1 S$	$q_{12} 1 R$

Table 13.35

Machine P

	0	1
q_1		$q_2 0 L$
q_2	$q_3 1 L$	$q_2 1 L$
q_3	$q_4 0 R$	$q_5 1 R$
q_4	$q_1 0 L$	$q_4 1 R$
q_5		$q_6 0 R$
q_6	$q_0 1 C$	$q_6 1 R$

Since we do not care what the actual logic of the machine is as long as we obtain the desired result, we shall understand that the machine performing the function of P can be either that of Table 13.34 or of Table 13.35. One variant of P (corresponding to Table 13.35) is shown in Table 13.36.

Machine R_m is a synthesis by multiplication and iteration of machines H, F, E, D, C, B, and A of Section 13.1:

$$R_m = H\dot{F}^m E \begin{cases} (1) & DC^m, \\ (2) & BC^m\dot{A}, \end{cases}$$

Table 13.36

Time	Tape printout
	1
0	0 1 1 1 0 0 0 1 1 1 $\overline{1}$ 0
	2
1	0 1 1 1 0 0 0 1 1 $\overline{1}$ 0 0
. . .	. . . . .
	2
	0 1 1 1 0 0 $\overline{0}$ 1 1 1 0 0
	3
	0 1 1 1 0 $\overline{0}$ 1 1 1 1 0 0
	4
	0 1 1 1 0 0 $\overline{1}$ 1 1 1 0 0
. . .	. . . . .
	4
	0 1 1 1 0 0 1 1 1 $\overline{0}$ 0
	1
	0 1 1 1 0 0 1 1 1 $\overline{1}$ 0 0
. . .	. . . . .
	1
	0 1 1 1 0 1 1 1 $\overline{1}$ 0 0 0
	2
	0 1 1 1 0 1 1 $\overline{1}$ 0 0 0 0
. . .	. . . . .
	2
	0 1 1 1 $\overline{0}$ 1 1 1 0 0 0 0
	3
	0 1 1 $\overline{1}$ 1 1 1 1 0 0 0 0
	5
	0 1 1 1 $\overline{1}$ 1 1 1 0 0 0 0
	6
	0 1 1 1 0 $\overline{1}$ 1 1 0 0 0 0
. . .	. . . . .
	6
	0 1 1 1 0 1 1 1 $\overline{0}$ 0 0 0
	0
	0 1 1 1 0 1 1 1 $\overline{1}$ 0 0 0

Table 13.37

Time	Tape printout	
	1	
0	0 0 1 1 $\overline{1}$ 0 0 0 0 0	
. . .	. . . . .	H
	0 0 1 1 1 0 $\overline{1}$ 0 0 0	
. . .	. . . . .	F
	0 0 1 1 $\overline{1}$ 0 1 0 0 0	
	0_2	E
	0 0 1 1 $\overline{1}$ 0 1 0 0 0	
	0 0 1 $\overline{1}$ 0 0 1 0 0 0	B
. . .	. . . . .	C
	0 0 1 1 0 0 $\overline{1}$ 0 0 0	
	0 0 1 1 0 0 1 $\overline{1}$ 0 0	A
. . .	. . . . .	
	0 0 1 0 0 0 1 1 $\overline{1}$ 0	
. . .	. . . . .	F
	0 0 $\overline{1}$ 0 0 0 1 1 1 0	
	0_1	E
	0 0 $\overline{1}$ 0 0 0 1 1 1 0	
. . .	. . . . .	D
	0 0 1 1 $\overline{1}$ 0 1 1 1 0	
. . .	. . . . .	C
	0	
	0 0 1 1 1 0 1 1 $\overline{1}$ 0	

where the superscript m indicates the power of a given machine. For example,

$$R_1 = H\dot{F}E \begin{cases} (1) & DC, \\ (2) & BC\dot{A}. \end{cases}$$

Machine R_m operates as follows: if the numbers $x_1, \ldots, x_m$ are represented on the tape in the standard position at $t = 0$, then R_m prints the first of these numbers (that is, x_1) to the right of the representation of $(x_1, \ldots, x_m)$ and stops; after this, the tape contains a system of $m + 1$ numbers $(x_1, \ldots, x_m, x_1)$ in standard position. However, if at $t = 0$ the tape contains, in standard position, the numbers $x_1, \ldots, x_n$, where $n > m$, then R_m prints the number x_{n-m+1} on the right-hand side of the representation of this system of numbers and stops; after this, the tape contains the number system $(x_1, x_2, \ldots, x_n, x_{n-m+1})$. An example of the operation of machine R_1 is shown in Table 13.37 (the letters on the right indicate which of the component machines of R_1 are responsible for a given step of the operation; the symbols 0_1 and 0_2 are the rest states of machine E).

It can be shown that the mth power R_m^m of the machine R_m operating on numbers $(x_1, \ldots, x_m)$ in standard position, copies the entire system next to and on the right-hand side of the original representation, the final result being a system of $2m$ numbers $(x_1, \ldots, x_m, x_1, \ldots, x_m)$ in standard position. The starting and the final tapes of R_m^m at $m = 3$, $x_1 = 1$, $x_2 = 0$, and $x_3 = 2$ are shown in Table 13.38.

Table 13.38

Time	Tape printout
	1
0	0 0 1 1 0 1 0 1 1 $\bar{1}$ 0 0 0 0 0 0 0 0 0 0 0
. . .	. . . . .
	0
	0 0 1 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 $\bar{1}$ 0 0

Machine S_m does almost the same thing as R_m^m; it copies numbers $(x_1, \ldots, x_m)$ to the right of their original tape representation but not next to it (that is, there are two blank squares, instead of one, between x_m and x_1). The machine S_m is synthesized as follows:

$$S_m = AR_m^m BF^m BC^m.$$

The starting and final tapes of S_m at $m = 3$, $x_1 = 1$, $x_2 = 0$, and $x_3 = 2$ are shown in Table 13.39.

Now we can show *that for any recursive function there exists a Turing machine capable of computing it.* Our first proof will pertain to the basic recursive functions. These are successor, the constant and the identity functions.

I. The successor function $\varphi(x) = x'$ can be computed by machine R_1A. Indeed, if a number x is in the standard position at $t = 0$, the

Table 13.39

Time	Tape printout
	1
0	0 0 1 1 0 1 0 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0 0
. . .	. . . . .
	0
	0 0 1 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 1 $\overline{1}$ 0

machine will print $\varphi(x) = x + 1$ to the right of x, after which it will stop if the tape then contains the number system $(x, x + 1)$ in standard position. Thus R_1A computes $\varphi(x) = x'$ within our definition of "computation" (see above, p. 371). The operation of R_1A at $x = 3$ is shown in Table 13.40.

II. The constant function $\varphi(x_1, \ldots, x_n) = q$ can be computed by machine HA^q. If $n = 3$, $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, and $q = 2$, the tape is that of Table 13.41.

III. The identity function $\varphi(x_1, \ldots, x_n) = x_i$ can be computed by machine R_{n-i+1}. Indeed, this machine prints to the right of the number system $(x_1, \ldots, x_n)$ the $(n - i + 1)$st number from the right, that is, x_i.

The remainder of the proof on the existence of Turing machines is by induction on the depth of the recursive description of the recursive function. The depth of functions I, II, and III is, by definition, zero. The application of superposition, induction, and the

Table 13.40

Time	Tape printout	
	1	
0	0 0 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0	
. . .	. . . . .	R_1
	0 0 1 1 1 1 0 1 1 1 $\overline{1}$ 0 0	
	0	A
	0 0 1 1 1 1 0 1 1 1 1 $\overline{1}$ 0	

least number operator increases the maximum depth of any function subjected to these operations by 1. Since all recursive functions can be derived from the basic functions by finite repetition of these three operations, we need only to show that there exist Turing machines

Table 13.41

Time	Tape printout	
	1	
0	0 0 1 1 0 1 0 $\overline{1}$ 0 0 0 0 0	
. . .	. . . . .	H
	0 0 1 1 0 1 0 1 0 $\overline{1}$ 0 0 0	A
	0 0 1 1 0 1 0 1 0 1 $\overline{1}$ 0 0	
	0	A
	0 0 1 1 0 1 0 1 0 1 1 $\overline{1}$ 0	

capable of realizing the operations of superposition, induction and the least number operator.

For simplicity, we shall avoid details of synthesis of Turing machines realizing these operations and shall restrict ourselves to the simpler special cases. For example, instead of dealing with superposition with respect to n variables, we shall consider superposition with respect to only one such variable.* We thus obtain the following:

IV. *Superposition.* Let M_ψ and M_χ be Turing machines computing the recursive functions $\psi(x)$ and $\chi(x)$ whose depths are α and β, respectively. We desire a Turing machine M_φ computing the function $\varphi(x) = \chi(\psi(x))$ with a depth $\max(\alpha, \beta) + 1$. This will be accomplished by the machine

$$M_\varphi = S_1 M_\psi N M_\chi N P.$$

V. *Induction.* Let M_χ be a machine computing a recursive function $\chi(x)$ of depth α. We shall devise a machine M_φ computing the recursive function $\varphi(x)$ of depth $\alpha + 1$ specified by the induction scheme

$$\varphi(0) = q, \quad \varphi(x') = \chi(\varphi(x)).$$

This machine is given by

$$M_\varphi = S_1 A^{q+1} \dot{F} E \begin{cases} (1) & BCP, \\ (2) & BCM_\chi N\dot{P}. \end{cases}$$

VI. *The smallest number operator.* Let M_χ be a machine computing the function $\chi(x, y)$ of depth α. We shall devise a machine M_φ

*For a precise description of generalized Turing machines realizing these operations, see [42].

computing the function $\varphi(x) = \mu y[\chi(x, y) = 0]$ of depth $\alpha + 1$. This machine is

$$M_\varphi = S_1 G A M_\chi E \begin{cases} (1) & GFP, \\ (2) & GF\dot{A}. \end{cases}$$

Machine M_φ will stop only if x satisfies the relation $(\exists y)[\chi(x, y) = 0]$.

We shall now illustrate the computation by induction and the use of the smallest number operator so as to elucidate the operation of machines V and VI.

Table 13.42

Time	Tape printout
	1
0	0 0 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
. . .	. . . . . S_1
	0 0 1 1 1 0 0 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0 0 0 0 0
. . .	. . . . . A^{q+}
	0 0 1 1 1 0 0 1 1 1 0 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0 0 0
. . .	. . . . . FE
	0_2
	0 0 1 1 1 0 0 1 1 $\overline{1}$ 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 B
	0 0 1 1 1 0 0 1 $\overline{1}$ 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
. . .	 C
	0 0 1 1 1 0 0 1 1 0 0 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0 0 0
. . .	. . . . . M
	0 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0
. . .	. . . . . N
	0 0 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0
. . .	. . . . . P
	0 0 1 1 1 0 0 1 1 0 1 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0 0
. . .	. . . . . FE
	0_2
	0 0 1 1 1 0 0 1 $\overline{1}$ 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 B
	0 0 1 1 1 0 0 $\overline{1}$ 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
. . .	 C
	0 0 1 1 1 0 0 1 0 0 1 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0
. . .	. . . . . M
	0 0 1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 $\overline{1}$ 0
. . .	. . . . . N
	0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 $\overline{1}$ 0
. . .	. . . . . P
	0 0 1 1 1 0 0 1 0 1 1 1 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0
. . .	. . . . . FE
	0_1
	0 0 1 1 1 0 0 $\overline{1}$ 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 B
	0 0 1 1 1 0 0 $\overline{0}$ 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
. . .	 C
	0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0
. . .	. . . . . P
	0
	0 0 1 1 1 0 1 1 1 1 1 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0 0

Table 13.43

Time	Tape printout	
	1	
0	0 0 1 1 $\overline{1}$ 0 0 0 0 0 0 0 0 0 0	
. . .	. . . . .	S_1GA
	0 0 1 1 1 0 0 $\overline{1}$ 0 0 0 0 0 0 0	
. . .	. . . . .	$M_\chi E$
	0_2	
	0 0 1 1 1 0 0 1 0 1 1 1 1 $\overline{1}$ 0	
. . .	. . . . .	G
	0 0 1 1 1 0 0 1 $\overline{0}$ 0 0 0 0 0 0	
	0 0 1 1 1 0 0 $\overline{1}$ 0 0 0 0 0 0 0	F
	0 0 1 1 1 0 0 1 $\overline{1}$ 0 0 0 0 0 0	A
. . .	. . . . .	$M_\chi E$
	0_2	
	0 0 1 1 1 0 0 1 1 0 1 1 1 $\overline{1}$ 0	
. . .	. . . . .	G
	0 0 1 1 1 0 0 1 1 $\overline{0}$ 0 0 0 0 0	
	0 0 1 1 1 0 0 1 1 $\overline{1}$ 0 0 0 0 0	F
	0 0 1 1 1 0 0 1 1 1 $\overline{1}$ 0 0 0 0	A
. . .	. . . . .	$M_\chi E$
	0_2	
	0 0 1 1 1 0 0 1 1 1 0 1 1 $\overline{1}$ 0	
. . .	. . . . .	G
	0 0 1 1 1 0 0 1 1 1 $\overline{0}$ 0 0 0 0	
	0 0 1 1 1 0 0 1 1 1 $\overline{1}$ 0 0 0 0	F
	0 0 1 1 1 0 0 1 1 1 1 $\overline{1}$ 0 0 0	A
. . .	. . . . .	M_L
	0 0 1 1 1 0 0 1 1 1 1 1 0 $\overline{1}$ 0	
	0_1	E
	0 0 1 1 1 0 0 1 1 1 1 1 0 $\overline{1}$ 0	G
	0 0 1 1 1 0 0 1 1 1 1 1 $\overline{0}$ 0 0	F
	0 0 1 1 1 0 0 1 1 1 1 $\overline{1}$ 0 0 0	
	0	P
	0 0 1 1 1 0 1 1 1 1 $\overline{1}$ 0 0 0 0	

Computation by induction. Let

$$\chi(y) = y + 3, \quad q = 1,$$

so that

$$\varphi(0) = 1, \quad \varphi(x') = \varphi(x+1) = \varphi(x) + 3.$$

Table 13.42 shows how machine M_φ computes the value of $\varphi(2)$. (Again, 0_1 and 0_2 denote states of machine E). The basic concept underlying this procedure consists in repeated computation of

function $\chi(y)$ while keeping a check on how many times this computation has been repeated.

The result of the computation: $\varphi(2) = 7$.

Selecting the operator of the smallest number. Let $\chi(x, y) = |x^2 - y|$, so that

$$\varphi(x) = \mu y[|x^2 - y| = 0].$$

We want to compute $\varphi(2)$.

The machine operation is shown in Table 13.43: here, we determine the consecutive values of $\chi(x, 0)$, $\chi(x, 1)$, and so on, until $\chi(x, y) = 0$. The result of the computations is $\varphi(2) = 4$.

Let us also point out that all machines synthesized according to schemes I - VI must be so designed that they will never go beyond the left-hand edge of the tape.

We have thus shown that any recursive function can be computed on a Turing machine. It can also be shown that only recursive functions may be computed on Turing machines. This is proved by means of gödelization of tape representations and verification of the fact that any change in such representations can be expressed by means of recursive functions (see the proof of this in [42]).

By virtue of the equivalence of the concepts of "recursive function" and of "function computable on a Turing machine", as well as by virtue of Church's thesis, we can now define an algorithm as follows: *An algorithm is any procedure which reduces to the computation of the values of an integer-valued function on an appropriate Turing machine.*