18 Turing Machine Computable = Partial Recursive

(Optional)

In Chapter 10 we referred to the Most Amazing Fact that all systems which have been proposed as formalizations of the notion of computability have been shown to be equivalent. In this chapter we will prove the equivalence of two of them. Moreover, we will do it constructively: Given a definition of a partial recursive function φ, we will produce a Turing machine which calculates that function; and given a Turing machine which calculates a function φ, we will produce a partial recursive definition of φ. Nothing later in this text depends on this chapter.

A. Partial Recursive Implies Turing Machine Computable

In order to make the proof in this section go more smoothly we are going to show something a bit stronger than the title of this section. We say that a Turing machine that computes a function *uses a one-way tape* if at no point in any of its computations on any input does it go to the left of the first blank at the left of the input. That is, the initial configuration for input $(n_1, \ldots, n_k)$ is

$$\overset{\substack{\text{never goes to the left of here}\\ \downarrow}}{0}\,1^{n_1+1}\,0 \cdots 0\,1^{n_k+1}\;\overset{\substack{\text{blank squares can be added here}\\ \downarrow}}{}$$

THEOREM 1 Every definition of a partial recursive function can be effectively converted into a Turing machine that computes that function and which uses a one-way tape and never halts in a nonstandard configuration on an input of the correct number of variables.

Proof: First note that all the machines of Chapter 9.C use one-way tapes, and the machines you defined as answers to the exercises in Chapter 9 can be modified to run on a one-way tape if they don't already.

Hence, we can conclude that the initial partial recursive functions (successor, the projections (Exercise 9.2), zero) as well as the equality function (Exercise 9.6), addition, and multiplication can all be computed on TM's that use a one-way tape. Moreover, for these total functions the machines always halt in standard configuration. We now need to show that the class of functions computed by TM's that use one-way tapes and never halt in a nonstandard configuration is closed under composition, μ-operator, and primitive recursion.

Composition We showed this for functions of one variable in Chapter 9.C. Now suppose that $\varphi_1, \ldots, \varphi_m$ are functions of k variables, ψ is a function of m variables, and we have a machine for each which computes it using a one-way tape and which never halts in a nonstandard configuration. We will describe the operation of a machine that computes the composition of these, $\psi(\varphi_1, \ldots, \varphi_m)$, and leave to you the actual definition of the machine as a set of quadruples (no, we don't think it's easy to do that, but it's not very instructive either).

For input $\vec{x} = (n_1, \ldots, n_k)$, we begin with the tape configuration $01^{n_1+1}\, 0 \ldots 0\, 1^{n_k+1}$, which we'll call $1^{\vec{x}+1}$. Here are the successive contents of the tape.

1. $0\,1^{\vec{x}+1}$
2. $0\,1^{\vec{x}+1}\,0\,0\,0\,1^{\vec{x}+1}$
3. $0\,1^{\vec{x}+1}\,0\,0 \cdots 0\,1^{\varphi_1(\vec{x})}$

We aren't operating the machine T_{φ_1} on a badly configured tape here. What we are doing is adding quadruples that allow for the simulation of the machine at the appropriate stage in the computation, and we know that the machine will be simulated correctly because it never goes to the left of the first blank to the left of its input.

Similarly, we continue,

4. $0\,1^{\vec{x}+1}\,0\,0\,0\,1^{\varphi_1(\vec{x})+1}$
5. $0\,1^{\vec{x}+1}\,0\,0\,0\,1^{\varphi_1(\vec{x})+1}\,0\,0\,1^{\vec{x}+1}$
6. $0\,1^{\vec{x}+1}\,0\,0\,0\,1^{\varphi_1(\vec{x})+1}\,0\,0\,1^{\varphi_2(\vec{x})+1}$
7. $0\,1^{\vec{x}+1}\,0\,0\,0\,1^{\varphi_1(\vec{x})+1}\,0\,0\,1^{\varphi_2(\vec{x})+1}\,0\,0\,1^{\vec{x}+1}$
8. $0\,1^{\vec{x}+1}\,0\,0\,0\,1^{\varphi_1(\vec{x})+1}\,0\,0\,1^{\varphi_2(\vec{x})+1}\,0\,0 \cdots 0\,0\,1^{\varphi_m(\vec{x})+1}$
9. $0\,1^{\varphi_1(\vec{x})+1}\,0\,0\,1^{\varphi_2(\vec{x})+1}\,0\,0 \cdots 0\,0\,1^{\varphi_m(\vec{x})+1}$
10. $1^{\psi[\varphi_1(\vec{x})+1,\ \varphi_2(\vec{x})+1,\ \varphi_m(\vec{x})+1]}$

Note that if for some i the machine that calculates $\varphi_i(\vec{x})$ does not halt, then the composition machine does not halt on $\vec{x}$.

The μ-operator Suppose we have a machine that computes a function φ that uses a one-way tape and never halts in a nonstandard configuration. We will describe the operation of a machine that computes $\mu y\,[\,\varphi(\vec{x}, y) = 0\,]$, the μ-operator applied to φ, and leave to you to define the machine as a set of quadruples. Here are the successive contents of the tape.

1. $0\,1^{\vec{x}+1}$
2. $0\,1^{\vec{x}+1}\,0\,1$
3. $0\,1^{\vec{x}+1}\,0\,1\,0\,1^{\varphi(\vec{x},0)+1}\,0\,1$
4. Use the equality machine, T_E, to determine if $\varphi(\vec{x},0) = 0$ by applying it to the string beginning to the right of $1^{\vec{x}+1}\,0\,1\,0$ (i.e., insert the appropriately relabeled quadruples of T_E).
5. If equal, erase the tape and stop.
6. If not equal, erase the tape back to $0\,1^{\vec{x}+1}\,0\,1$ and add a 1 to the right, $0\,1^{\vec{x}+1}\,0\,11$.

   ⋮
7. $0\,1^{\vec{x}+1}\,0\,1^{n+1}$
8. $0\,1^{\vec{x}+1}\,0\,1^{n+1}\,0\,1^{\varphi(\vec{x},n)+1}$
9. $0\,1^{\vec{x}+1}\,0\,1^{n+1}\,0\,1^{\varphi(\vec{x},n)+1}\,0\,1$
10. Use the equality machine, T_E, to determine if $\varphi(\vec{x}, n) = 0$ by applying it to the string beginning to the right of $0\,1^{\vec{x}+1}\,0\,1^{n+1}\,0$.
11. If equal, erase all but 1^n and stop.
12. If not equal, erase the tape back to $0\,1^{\vec{x}+1}\,0\,1^{n+1}$ and add a 1 to the right, $0\,1^{\vec{x}+1}\,0\,1^{n+2}$, and repeat the process.

Primitive recursion This is difficult, so difficult that we're tempted to leave it to you. But there's a way out. In Chapter 22.A we prove that the partial recursive functions comprise the smallest class containing the zero function, the successor function, the projections, addition, multiplication, and the characteristic function for equality and which is closed under composition and the μ-operator. That's just what we need to complete the proof here. You can read that section with the background you already have. ■

B. Turing Machine Computable Implies Partial Recursive

THEOREM 2 If a Turing machine calculates a function φ then the set of quadruples of the machine can be effectively converted into a partial recursive definition of φ.

Proof: Let M be a Turing machine that calculates the function φ. By coding what the machine does at each step of its computation, we will be able to derive a partial recursive definition of φ.

First we assign the following numbers to the operations:

Delete the current symbol, if any	0
Write the symbol 1	1
Move one square to the right	2
Move one square to the left	3

Let the highest numbered state of M be n. The quadruples of M can be seen as a function g from (states, symbols) to (operations, states). For example, $q_1\, 0\, R\, q_2$ can be written as $g[(q_1,0)] = (R,q_2)$. If we assign the number j to state q_j, we can express this function numerically as

$$d(x) = \begin{cases} g[(x)_0,(x)_1] & \text{if } g \text{ is defined for this input} \\ 47 & \text{otherwise} \end{cases}$$

We'll let you make this more formal, but it should be clear that d is (primitive) recursive since it's a finite table.

To code the tape descriptions, suppose that after t steps in the computation on input $\vec{x}$ the machine M is in state $q(t)$ and the symbol being observed is

$$s(t) = \begin{cases} 0 & \text{if blank} \\ 1 & \text{if the symbol is 1} \end{cases}$$

And suppose that we have the following configuration:

$$\overset{\substack{\text{blank from this point on}\\ \downarrow}}{\cdots\, b_s} \mid b_{s-1} \mid \cdots \mid b_1 \mid b_0 \mid \underset{\substack{\uparrow\\ q(t)}}{s(t)} \mid c_0 \mid c_1 \mid \cdots \mid c_{r-1} \mid \overset{\substack{\text{blank from this point on}\\ \downarrow}}{c_r\, \cdots}$$

We should indicate that both s and q depend on $\vec{x}$, but for the sake of legibility we will suppress that until the end of the proof. And, as usual, we will notate that b_i or c_i is blank with a 0. (Note that neither c_r nor b_s can be 0 since they mark the point beyond which the tape becomes blank.)

We may describe this configuration by coding up the contents of the tape to the left and the right of the square being observed

$$b(t) = \langle b_0, b_1, \dots, b_s \rangle$$
$$c(t) = \langle c_0, c_1, \dots, c_r \rangle$$

where these are 0 if the tape is blank in that direction.

On input $\vec{x}$, the machine begins on a tape containing only $1^{\vec{x}+1}$ (see the proof of Theorem 1 for notation). So $b(0) = 0$, $s(0) = 1$, $q(0) = 1$; we'll let you write down what $c(0)$ is. Then at step $t+1$, call $a = d(\langle q(t), s(t) \rangle)$, and we have $q(t+1) = (a)_1$ and one of the following cases:

i. The machine moves right at this step, that is, $(a)_0 = 2$

$$s(t+1) = (c(t))_0$$
$$b(t+1) = \langle s(t), b_0, b_1, \dots, b_s \rangle$$
$$c(t+1) = \langle c_1, \dots, c_r \rangle$$

ii. The machine moves left, that is, $(a)_0 = 3$

$$s(t+1) = (b(t))_0$$
$$b(t+1) = \langle b_1, \dots, b_s \rangle$$
$$c(t+1) = \langle s(t), c_0, c_1, \dots, c_r \rangle$$

iii. The machine writes or deletes, that is, $(a)_0 = 0$ or 1

$$s(t+1) = (a)_0$$
$$b(t+1) = b(t)$$
$$c(t+1) = c(t)$$

We leave to you to confirm that b, c, q, and s are primitive recursive functions (see Chapter 11.D.2 and D.7). Indeed, they are elementary.

To determine at stage t whether the machine has halted in a standard configuration we need to know at most how many squares of tape have been used up to that stage. Since at each stage the machine can add no more than one new square, an upper bound is $c(0) + t$. Then at stage t the machine has halted in standard configuration iff

$$d(\langle q(t), s(t) \rangle) = 47$$
$$b(t) = 0$$

and

$$\forall i < c(0)+t\ [\neg[(p_i)^2 \mid c(t)] \rightarrow \neg[(p_{i+1})^2 \mid c(t)]]$$

This is a primitive recursive condition, indeed elementary; call its characteristic function h. If the machine does halt in standard configuration then the output is $lh(c(t))$. Remembering now that each of the functions we have defined depends on $\vec{x}$, we have

$$\varphi(\vec{x}) = lh(c(\vec{x}, \mu t\ [h(\vec{x}, t) = 1])) \qquad \blacksquare$$

Combining Theorems 1 and 2 we have the following.

COROLLARY 3 A function is TM computable
iff it is partial recursive
iff it is computable on a TM that uses a one-way tape and never halts in a nonstandard configuration.

These computable correspondences allow us to translate facts about partial recursive functions into facts about Turing machines. For example, from Theorem 16.4 we can derive the following.

COROLLARY 4 There is a universal Turing machine.

In Exercise 9.8 we defined the halting problem for Turing machines and sketched a proof that it was not Turing machine computable. Now we can conclude that directly from Corollary 15.3.

COROLLARY 5 (The Halting Problem for Turing Machines)
The halting problem for Turing machines is not Turing machine computable.