The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual

Chapter 5, Problem 5.16

1. $S(y)$ is r.e., but not recursive; in consequence, its complement is not r.e. Rice's theorem applies directly to $S(y)$: the set is characterized by input-output behavior and is neither empty nor the universe; thus $S(y)$ is not recursive. Intuitively, $S(y)$ is r.e. because we can try all functions for all arguments for all steps, printing the function index whenever that function stops for that argument and that number of steps and returns y. Formally, for each fixed y, we can define a function

$$\theta(x) = \mu z[\text{step}(x, \Pi_1(z), \Pi_2(z)) = \text{Succ}(y)]$$

   For a given value of x, our function converges if and only if, for some argument and some number of steps, $\theta(x)$ returned y; thus the domain of θ is exactly $S(y)$.

2. By Rice's theorem, this set is not recursive: the property is a property of functions, not a property of programs, and is non-trivial (there are both injective and non-injective functions). It follows by closure that the complement is also non-recursive. We need to be a bit more specific about injectivity before we decide whether or not the set or its complement is r.e.; the definition of injectivity normally given is $\forall x, y, (x \neq y) \Longrightarrow (f(x) \neq f(y))$, which does not carry over well to non-total functions. We can simply ignore all arguments on which the function is not defined and state that ϕ is injective if, whenever $\phi(x)\downarrow$ and $\phi(y)\downarrow$, then $(x \neq y) \Longrightarrow (\phi(x) \neq \phi(y))$, or we can state that only total functions can be injective. We look at both interpretations below. Under the first interpretation (only defined values count), we claim that the complement is r.e., so that the set itself cannot be r.e. To see that the complement is r.e., note that placing some program ϕ_i in this complement set can be done just by identifying two arguments x and y such that $x \neq y$ and yet $\phi_i(x) = \phi_i(y)$. We can do this by quintuple dovetailing, on all programs, all pairs of arguments, and all steps for each argument; whenever we find a z such that

- $\text{step}(\Pi_1^5(z), \Pi_2^5(z), \Pi_3^5(z)) \neq 0$ and
- $\text{step}(\Pi_1^5(z), \Pi_4^5(z), \Pi_5^5(z)) \neq 0$ and
- $\Pi_2^5(z) \neq \Pi_4^5(z)$ and
- $\text{step}(\Pi_1^5(z), \Pi_2^5(z), \Pi_3^5(z)) = \text{step}(\Pi_1^5(z), \Pi_4^5(z), \Pi_5^5(z))$,

we print $\Pi_1^5(z)$.

Under the second interpretation, however, neither the set nor its complement is r.e. Thus we must do some extra work. We reduce $\overline{K}$ in turn to the set and to its complement. To reduce $\overline{K}$ to the set of (total) injective functions, we define

$$\theta(x, y) = \begin{cases} y & \text{step}(x, x, y) = 0 \\ \phi_j(y) & \text{otherwise} \end{cases}$$

where ϕ_j is the totally undefined function. This new function is clearly partial recursive (a valid definition by cases), so we can write $\theta(x, y) = \phi_i(x, y)$ for some index i. Using the s-m-n construction, we now get $\phi_{s(i,x)}(y) = \theta(x, y)$. But note that $\phi s(i, x)$ is the identity function whenever $x \in \overline{K}$ and is undefined on at least one value of y otherwise. Hence $s(i, x)$ is injective whenever $x \in \overline{K}$ and is not injective (because it is partial) otherwise. Thus we have reduced $\overline{K}$ to our set of injective functions, thereby showing that our set is not r.e. It remains to show that the complement of our set is not r.e. either; for that we need to reduce $\overline{K}$ directly to the complement. We define a new function $\zeta(x, y) = y + \text{Zero}(\phi_{\text{univ}}(x, x))$. Again, this is a partial recursive function, so that there exists an index k such that $\phi_k(x, y) = \zeta(x, y)$; again, the s-m-n theorem tells us that we can write $\phi_{s(k,x)}(y) = \phi_k(x, y) = \zeta(x, y)$. But note that $\phi_{s(k,x)}(y)$ is totally undefined when $x \in \overline{K}$ and is the identity function otherwise; that is, $s(k, x)$ is in the complement of our set whenever $x \in \overline{K}$, so that we have a reduction from $\overline{K}$ to the complement of our set, thereby proving it to be non-r.e.

3. This is a rather tricky question. It depends on what is understood by being primitive recursive. If what is meant is "the program was defined using only the three base functions and the operations of substitution and primitive recursion," then the set is certainly r.e. and in fact recursive—we defined a total recursive predicate in a previous homework. In that case, of course, the complement of the set is also recursive.

   However, if what is meant is "the program computes a function that can be implemented by a primitive recursive program," then we have an I/O property (the set of all programs that compute the same function as any of the primitive recursive programs) and thus our set is not recursive by Rice's theorem. Rather worse, our set is then not even r.e., intuitively because it would require us to decide of two programs whether they compute the same function (one being the program in hand and the other some program defined in a primitive recursive fashion). Consider the simple function

$$\theta(x, y) = \begin{cases} 0 & \text{step}(x, x, y) = 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

and use the s-m-n theorem to obtain $\phi_{s(x)}(y) = \theta(x, y)$. Note that $\phi_{s(x)}(y)$ need not be a primitive recursive function as defined. However, if $x \in \overline{K}$, then $\phi_{s(x)}(y)$ is the constant function 0, which is a primitive recursive function (defined by Zero(y)), whereas, if $x \in K$, then $\phi_{s(x)}(y)$ is mostly undefined and thus not primitive recursive. Thus we have reduced $\overline{K}$ to the set of primitive recursive functions.

The complement of the set (the set of non-primitive recursive functions) is equally non-r.e. We prove it by reduction from $\overline{K}$. Define $\theta(x, y) = \text{Zero}(\phi_{\text{univ}}(x, x))$ and use the s-m-n theorem to obtain $\phi_{s(x)}(y) = \theta(x, y)$. If $x \in \overline{K}$, then $\phi_{s(x)}(y)$ is the totally undefined function, which is not primitive recursive and thus is in the complement, while, if $x \in K$, $\phi_{s(x)}(y)$ is the constant function 0, which is primitive recursive.

4. This is an I/O property—it concerns the values returned by the program for increasingly larger arguments; thus the set is not recursive by Rice's theorem. Intuitively, the set is non-r.e., because verifying a growth rate requires verification for all possible arguments. To be more precise, we need to define carefully what is meant by "grow as fast as". If we mean that the function should be at least as large as n^2 everywhere, then the set is clearly non-r.e., since it requires first that the function be total (an easy reduction from $\overline{K}$ is to use the step function mechanism and define the new function to be n^3 when $\text{step}(x, x, xy) = 0$).

   The complement of the set (the functions that on at least one value are smaller than n^2 or are undefined) is also non-r.e., simply because the totally undefined function is part of that set, so that a trivial reduction of the type

$$\theta(x, y) = n^3 + \text{Zero}(\phi_{\text{univ}}(x, x))$$

will work—if $x \in \overline{K}$, then the function is totally undefined and thus on the complement, whereas, if $x \in K$, the function is just n^3, which grows faster than n^2 and thus is not in the complement.

5. Recall that an r.e. set is the range (or domain) of a partial recursive function. Thus our set could equally well have been defined as the set of all partial recursive functions that have at least three elements in their range (or domain). Since having a range (or domain) of size at least three is an I/O property, this set is not recursive by Rice's theorem. The set is r.e., however: intuitively, we can dovetail on all functions, all arguments, and all numbers of steps, keeping track of what converged and printing out a function index as soon as it has converged on at least three distinct arguments (for domain definitions, but range definitions are similar). Formalizing the notion is not hard (we basically want to run $\text{step}(\Pi_1^3(z), \Pi_2^3(z), \Pi_3^3(z))$, recording $\Pi_1^3(z)$ and one of $\Pi_2^3(z)$ (for domains) or $\text{step}(\Pi_1^3(z), \Pi_2^3(z), \Pi_3^3(z))$ (for ranges) for which the function is non-zero, and printing $\Pi_1^3(z)$ as soon as three distinct values of $\Pi_2^3(z)$ (for domains) or of $\text{step}(\Pi_1^3(z), \Pi_2^3(z), \Pi_3^3(z))$ (for ranges) are found that pair with the same $\Pi_1^3(z)$ value.

   Since the set is non-recursive and r.e., its complement is not r.e. This is an r.e. set. Informally, since we can enumerate all Turing machines and enumerate all possible input strings, we can start a triple dovetailing process, on all machines, all inputs, and all steps. In the process, we every now and then discover a pair (machine,input) such that the machine accepts the string; we then "chalk one up" for that machine. As soon as we have chalked up three for a machine, we print it (its index). In that way, we correctly enumerate all machines that accept at least three strings.

6. Neither this set nor its complement is r.e. That neither is recursive is an immediate consequence of Rice's theorem, since being defined on a finite or infinite number of inputs is an I/O behavior. To show that neither S nor $\overline{S}$ is r.e., we reduce $\overline{K}$ to each in turn.

   To reduce $\overline{K}$ to S, we define the function $\theta(x, y) = \phi_{\text{univ}}(x, x)$; since this is a valid partial recursive function, it has some index ϕ_i and we can write $\phi_{s(x)}(y) = \phi_i(x, y) = \theta(x, y)$. Now, if $x \in \overline{K}$, then $\phi_{s(x)}(y)$ diverges for all y, and thus has empty (and hence finite) domain, and thus $s(x)$ belongs to S. On the other hand, if $x \in K$, then $\phi_{s(x)}(y) = c$, where $c = \phi_x(x)$ is a constant, so that $\phi_{s(x)}$ is total and thus has infinite domain, and hence $s(x) \notin S$, as desired.

To reduce $\overline{K}$ to $\overline{S}$, we define the function

$$\theta(x, y) = \begin{cases} \text{Zero}(y) & \text{step}(x, x, y) = 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

since this is a valid partial recursive function, it has some index ϕ_i and we can write $\phi_{s(x)}(y) = \phi_i(x, y) = \theta(x, y)$. Now, if $x \in \overline{K}$, then $\phi_{s(x)}(y) = \text{Zero}(y)$ and thus is total and has infinite domain, and thus $s(x)$ belongs to $\overline{S}$. On the other hand, if $x \in K$, then $\phi_{s(x)}(y)$ is undefined for all $y \geq y_0$ for some constant y_0, and thus has finite domain and therefore $s(x)$ does not belong to $\overline{S}$, as desired.

7. Exercise 5.15 gave us that all three sets are non-recursive.
   - The first set is the set of constant functions; a standard reduction according to Table 5.1 works directly, showing that this set is non-r.e.—define

     $$\theta(x, y) = \begin{cases} \text{Zero}(y) & \text{step}(x, x, y) = 0 \\ y & \text{otherwise} \end{cases}$$

     Then the resulting function is the constant 0 whenever x belongs to $\overline{K}$ and is a non-constant function otherwise, giving us the desired reduction. The complement of the set is the set of all non-constant functions; again, a direct reduction from $\overline{K}$ works easily—define

     $$\theta(x, y) = \text{Zero}(\phi_{\text{univ}}(x, x))$$

     Then the resulting function is the constant function 0 whenever x does not belong to $\overline{K}$ and the non-constant totally undefined function otherwise.
   - The second set is the set of functions with non-empty domain. This set is r.e.: intuitively, we can dovetail on all programs, all arguments, and all steps and output the index of a program as soon as it is observed to stop on an argument. The complement is thus automatically non-r.e.
   - The third set is the set of functions defined on at least one argument which happens to be the index of a total program. This set is non-r.e. (intuitively, we cannot verify that an index on which the function is defined characterizes a total program). To prove that this set is non-r.e., we reduce the set H of total

functions to it (a much easier reduction than reducing $\overline{K}$). Define

$$\theta(x, y) = \begin{cases} 0 & x = y \\ \text{undefined} & \text{otherwise} \end{cases}$$

Use the s-1-1 theorem to obtain a function $\phi_{g(x)}(y)$; note that this function is defined only at $y = x$. Thus $g(x)$ belongs to our set if and only if $y = x$ (its only convergence value) is the index of a total function, that is if and only if x belongs to H—the desired reduction.

The set's complement is the set

$$\{x \mid \forall y(\phi_x(y) \uparrow \text{ or } \phi_y \text{ is not total})\}$$

It is also a non-r.e. set (intuitively, we cannot verify the "for all" condition); note that it contains the totally undefined function. A simple reduction from $\overline{K}$ proves the case—define

$$\theta(x, y) = y + \text{Zero}(\phi_{\text{univ}}(x, x))$$

The resulting function is totally undefined whenever x belongs to $\overline{K}$ and is the function y otherwise—and that function converges everywhere and so in particular converges on the index of a total program.