3.3.18 Show that there are total recursive functions \( f \) and \( g \) such that
\[
\text{domain } \phi_{\text{def}(x,y)} = \text{domain } \phi_x \cap \text{domain } \phi_y
\]
and
\[
\text{domain } \phi_{\text{def}(x,y)} = \text{domain } \phi_x \cup \text{domain } \phi_y.
\]
(Thus the class of r.e. sets is closed under union and intersection. It is not closed under complementation. Why?)

3.3.19 For each of the two sets below, show whether or not it is recursive, whether or not it is r.e., and whether its complement is r.e.: 
\[
A = \{ x : \text{there are } y, z \text{ such that } \phi_x(y) \text{ and } \phi_x(z) \text{ are convergent} \};
\]
\[
B = \{ x : \text{there is a } y \text{ such that } \phi_x(y) \text{ is convergent and } \phi_y \text{ is total} \}.
\]

3.3.20 Let \( C = \{ x : \phi_x(0) = 0 \} \) and \( D = \{ x : \phi_x(0) = 1 \} \). Show that there is no recursive set \( R \) such that \( C \subseteq R \) and \( D \cap R = \emptyset \); i.e., show that \( C \) and \( D \) are recursively inseparable. (See Exercises 2.4.13 and 3.1.9.)

3.4 THE RECURSION THEOREM AND ROGERS’ ISOMORPHISM THEOREM

The Recursion Theorem is the “fixed point” theorem of the theory of the computable functions. It states that for any effective mapping of programs to programs, there is a “fixed point” program which is mapped to an equivalent program. The Recursion Theorem has many important applications, including the justification of general types of recursive definitions of functions, such as ALGOL-like recursive procedure definitions, in any acceptable programming system.

3.4.1 RECURSION THEOREM For every total recursive function \( f \) there is a natural number \( n \) (depending on \( f \)) such that
\[
\phi_n = \phi_{f(n)}.
\]

Proof There is a total recursive function \( g \) such that
\[
\phi_{g(x)} = \begin{cases} \phi_{\phi_x(x)} & \text{if } \phi_x(x) \text{ is convergent} \\ \emptyset & \text{otherwise,} \end{cases}
\]
where \( \emptyset \) is, of course, the totally undefined function. Intuitively, the program \( g(x) \) on input \( y \) first computes \( \phi_x(x) \) and if and when this computation halts proceeds to compute \( \phi_{\phi_x(x)}(y) \). To get the function \( g \), define
\[
\theta(x, y) = \phi_{\text{univ}}(\phi_{\text{univ}}(x, x), y)
\]
and continue with our standard s-m-n construction. Let \( m \) be a program such that \( f \circ g = \phi_m \), and let \( n = g(m) \). Then since \( \phi_m \) is total, \( \phi_m(m) \) is convergent and we have that
\[
\phi_n = \phi_{g(m)} = \phi_{\phi_m} = \phi_{f(g(m))} = \phi_{f(n)}
\]
Therefore, \( n \) is our required fixed point program.

As a very simple application of the Recursion Theorem we can show that there is an \( n \) such that \( \phi_n \) is the constant function with output \( n \). To do so we use our standard s-m-n construction to produce a total recursive function \( f \) such that \( \phi_{f(x)} \) is the constant function \( x \), and then we take \( n \) to be a fixed point for \( f \). Such a program \( n \) might be called a “self-reproducing” program. (You might try to write a FORTRAN program which prints itself, and nothing else.) As another application of the Recursion Theorem, we can give a simple proof of Rice’s Theorem. Let \( \emptyset \) be such that \( \emptyset \neq \mathcal{P} \neq \mathcal{N} \), and let \( j \in \mathcal{P} \) and \( k \in \mathcal{P} \). Define \( f(x) = k \) if \( x \in \mathcal{P} \) and \( f(x) = j \) if \( x \in \mathcal{N} \). If \( \mathcal{P} \) were recursive then \( f \) would be a total recursive function which would not have a fixed point (since \( x \in \mathcal{P} \) iff \( f(x) \in \mathcal{P} \) for all \( x \)), contradicting the Recursion Theorem.

The proof of the Recursion Theorem is actually somewhat stronger than the statement of the theorem. Although the Recursion Theorem we have stated is sufficient for most applications, there are some occasions when we need the stronger version below. The Extended Recursion Theorem asserts that fixed point programs can be found effectively from a program for the function \( f \).

3.4.2 EXTENDED RECURSION THEOREM There is a total recursive function \( n \) such that for all \( x \), if \( \phi_x \) is total, then
\[
\phi_{\phi_x(n(x))} = \phi_{n(x)}.
\]

Proof Let \( i \) be a program for the function \( g \) in the proof of the Recursion Theorem; that is, \( \phi_i = g \). Let \( c \) be a total recursive function for composition; that is, \( \phi_{c(x,y)} = \phi_x \circ \phi_y \). Then \( n(x) = g(c(x, i)) \) is the required fixed point function. □

The next proposition is an example of the use of the Recursion Theorem to justify a recursive definition of a function and it is an important result in its own right.

3.4.3 PROPOSITION Let \( h \) be any total recursive function, and let \( s \) be an s-m-n function. There is an \( i \) such that \( \phi_{h(i)} = \phi_{s(i, x)} \) for all \( x \) and such that \( s(i, 1, x) \) is one-to-one as a function of \( x \).