Chapter 9

The Kleene Recursion Theorem

Kill Grates at Macrohard Corporation wants to corner the market with programming system $\eta_{\_}$. Since Kill knows that $\eta_{\_}$ isn't very good (it re-uses all the previous system's code and takes twice as much disk space), he has to resort to cunning rather than to quality. His plan is to wipe the competing programming system $\phi_{\_}$ off of the planet by releasing MH-Virus into the ambient computing environment. The MH-Virus design team has been commanded to find a computer virus that screws up the performance of every program in the $\phi_{\_}$ system. That is, MH-Virus is supposed to compute a total recursive function v such that for each n, k,

$$\phi_n^k \neq \phi_{v(n)}^k.$$

You are agent Gödel number 7 and your job is to foil Grates' plan. Your first question is: can such a plan possibly succeed?

Here is an analogy from analysis. Suppose you are looking at the unit line and you are challenged to find a continuous transformation of the unit line that alters every point. In other words, you have been challenged to find a continuous $f : [0, 1] \to [0, 1]$ such that for each $x \in [0, 1], f(x) \neq f$. This amounts to drawing a line (without gaps) across the unit square without touching the $x = y$ diagonal of the square (try it). For if the graph of f touches the $x = y$ diagonal, then at that point $f(x) = x$. so point x has not been altered. But that isn't possible, because the diagonal is "in the way", no matter how you try to draw the connected line from one side of the square to the other. If $f(x) = x$ then x is called a **fixed point** of f.

Now if Kill Grates' virus is viewed as a computable distortion of the partial recursive functions, we might expect that the diabolical scheme must fail in the (not very useful) sense that at least one program's input-output behavior is unaffected by the virus. That is precisely what happens. In fact, the construction is literally analogous to trying to avoid the diagonal of a square when you try to draw a continuous line across it, as I shall try to bring out in the following proof.

The Kleene Recursion Theorem Let ϕ_n^k be an acceptable numbering of *Part*.

$$\forall k\ \forall \text{ total recursive } f\ \exists n\ (\phi_n^k = \phi_{f(n)}^k).$$

Proof. I'll do it for the case of $k = 1$ and drop the annoying superscripts. The proof of this theorem is facilitated by a convention. Consider the embedded expression

$$\phi_{\phi_n(\vec{x})}(\vec{y}).$$

It is pretty clear that if

$$\phi_n(\vec{x}) \simeq z$$

then

$$\phi_{\phi_n(\vec{x})} = \phi_z.$$

But what if $\phi_n(\vec{x})$ is undefined? Then our acceptable numbering isn't "given a number to interpret", so $\phi_{\phi_n(\vec{x})}(\vec{y})$ does not denote a function. But there is another way to look at it. Let u be a universal index for acceptable numbering ϕ_-^-. Now we have that for each y,

$$\phi_u(\phi_n(\vec{x}), \langle\vec{y}\rangle) \uparrow,$$

where $\uparrow$ means "undefined". Thus, we may *also* think of the whole expression as denoting the everywhere undefined function

$$\phi_{\phi_n(\vec{x})} = \emptyset.$$

Now let an arbitrary, total recursive "virus" f be given. Think of a two-dimensional table in which the cell $T[n, m]$ is filled by the function $\phi_{\phi_n(m)}$. The table looks like:

$$\begin{array}{cccc}
\phi_{\phi_{\mathbf{0}}(\mathbf{0})} & \phi_{\phi_0(1)} & \phi_{\phi_0(2)} & \cdots \\
\phi_{\phi_1(0)} & \phi_{\phi_{\mathbf{1}}(\mathbf{1})} & \phi_{\phi_1(2)} & \cdots \\
\phi_{\phi_2(0)} & \phi_{\phi_2(1)} & \phi_{\phi_{\mathbf{2}}(\mathbf{2})} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{array}$$

(Think of this table as standing in for the unit square of reals in our analogy). Every cell of the table is filled with a partial recursive function because of our convention for dealing with the case in which $\phi_n(m)$ is undefined. Consider the (bold-face) diagonal of the table. We will now see, remarkably enough, that the diagonal of the table

$$\phi_{\phi_0(0)}, \phi_{\phi_1(1)}, \phi_{\phi_2(2)}, \cdots$$

is also a row of the table

$$\phi_{\phi_z(0)}, \phi_{\phi_z(1)}, \phi_{\phi_z(2)}, \cdots$$

where the function ϕ_z generating the row is a total recursive function. We do this using the universal and *s-m-n* properties as follows. Using a universal index u of numbering ϕ^{-}_{-}, define partial recursive function

$$\begin{aligned} \psi(n, x) &\simeq \phi_u(\phi_n(n), \langle x \rangle) \\ &\simeq \phi_{\phi_n(n)}(x) \end{aligned}$$

Since ϕ^{-}_{-} is onto $Part$, choose w such that

$$\phi_w(n, x) \simeq \psi(n, x).$$

Using the *s-m-n* property of numbering ϕ^{-}_{-}, we obtain a total recursive s such that

$$\phi_{s(w,n)}(x) \simeq \phi_w(n, x).$$

Now compose in a constant function to obtain a unary total recursive g such that for all n,

$$g(n) = s(w, n).$$

Let $\phi_j = g$. Unwinding the definitions, we obtain:

$$\begin{aligned} \phi_{\phi_j(n)}(x) &\simeq \phi_{g(n)}(x) \\ &\simeq \phi_{s(w,n)}(x) \\ &\simeq \phi_w(n, x) \\ &\simeq \psi(n, x) \\ &\simeq \phi_u(\phi_n(n), \langle x \rangle) \\ &\simeq \phi_{\phi_n(n)}(x). \end{aligned}$$

So as promised, the diagonal of the table is also the jth row table where $\phi_j = g$ is a total recursive function:

$$\phi_{\phi_j(n)} = \phi_{g(n)} = \phi_{\phi_n(n)}.$$

Now consider the total recursive virus f. Since g is total recursive, so is $\phi_n = C(f, g)$. Let so we also have that

$$\phi_{f(g(0))}, \phi_{f(g(1))}, \phi_{f(g(2))}, \ldots.$$

is the nth row of the table. Now (just as in our attempt to draw a line across the unit square), this row intersects the diagonal at $\phi_{\phi_n(n)} = \phi_{f(g(n))}$. Now let's check the effect of the virus f on the index $g(n)$, which exists because g is total:

$$\begin{aligned} \phi_{f(g(n))}(x) &\simeq \phi_{\phi_n(n)}(x) \\ &\simeq \phi_{g(n)}(x). \end{aligned}$$

So the behavior of the index $g(n)$ is unaltered by f. $\dashv$

Exercise 9.1 *Can the theorem be strengthened to a guarantee that the unaffected index is primitive recursive? Total? If your answer is affirmative carry out the proof. If it is negative, prove the negative claim and describe where Kleene's proof fails. You don't understand a theorem unless you do this. Relate the existence or nonexistence of acceptable numberings for classes of total recursive functions to your result.*

9.1 Good, Kleene Fun

We usually use the recursion theorem in tandem with the universal and *s-m-n* theorems. The recursion theorem can generate wonderful curiosities, like the self-printing program. At first it seems easy to make a self-printing program: something like

$$print(progam).$$

But that won't do because what is printed is *program*, not the actual program *print(program)*. Now we start to wonder if it is possible. It looks like there might be an infinite referential regress, in which the program tries forever to refer to itself but always misses the outermost "print" command in its own program. We would like to say

$$print(me),$$

but can programs be self-conscious? Some can. Let's construct one. The projection function p_2 is partial recursive. So let

$$\phi_n = p_2^2.$$

Now apply the *s-m-n* theorem to obtain a total recursive s such that for all x,

$$\begin{aligned} \phi_{s(n,x)}(y) &\simeq \phi_n(x,y) \\ &\simeq p_2^2(x,y). \end{aligned}$$

Now compose in a constant function and the appropriate projections to obtain a total recursive g such that

$$g(x) = s(n,x).$$

By the Kleene recursion theorem, we obtain an m such that

$$\phi_{g(m)} = \phi_m.$$

Thus, for each x:

$$\begin{aligned} \phi_m(x) &\simeq \phi_{g(m)}(x) \\ &\simeq \phi_{s(n,m)}(x) \\ &\simeq \phi_n(m,y) \\ &\simeq p_2^2(m,y) \\ &= m. \end{aligned}$$

Exercise 9.2 *Show that each partial recursive function ϕ_i has a finite variant ϕ_j that is "self-referential" in the sense that $\mu z.\phi_j \neq 0 = j$ and $\forall k > j, \phi_j(k) \simeq \phi_i(k)$. .*

Exercise 9.3 *Show that double recursion over the partial recursive functions yields a partial recursive function. Before we only said that it is "intuitively effective". By the Church-Turing thesis it follows that double-recursion is partial recursive. But we can now prove this fact formally, bypassing the Church-Turing thesis. Hint: follow the pattern of the preceding example. Write an expression for the recursion in which the recursive call is just a free variable. Apply* s-m-n *to this variable position and then apply Kleene's fixed point theorem. This is why it's called the "recursion theorem".*

The Kleene recursion theorem has far wider significance than these examples suggest. As we will see, it is a powerful way of turning purely computational problems into empirical problems.