When it comes to primitive recursion, we finally need to do some work. We will have to proceed in stages. As before, on the assumption that we already have terms $\overline{g}$ and $\overline{h}$ representing functions $g$ and $h$, respectively, we want a term $\overline{f}$ representing the function $f$ defined by

$$f(0, \overline{z}) = g(\overline{z})$$
$$f(x + 1, \overline{z}) = h(z, f(x, \overline{z}), \overline{z}).$$

So, in general, given lambda terms $G'$ and $H'$, it suffices to find a term $F$ such that

$$F(\overline{0}, \overline{z}) \equiv G'(\overline{z})$$
$$F(\overline{n + 1}, \overline{z}) \equiv H'(\overline{n}, F(\overline{n}, \overline{z}), \overline{z})$$

for every natural number $n$; the fact that $G'$ and $H'$ represent $g$ and $h$ means that whenever we plug in numerals $\overline{m}$ for $\overline{z}$, $F(\overline{n + 1}, \overline{m})$ will normalize to the right answer.

But for this, it suffices to find a term $F$ satisfying

$$F(\overline{0}) \equiv G$$
$$F(\overline{n + 1}) \equiv H(\overline{n}, F(\overline{n}))$$

for every natural number $n$, where

$$G = \lambda \overline{z}. G'(\overline{z})$$
$$H(u, v) = \lambda \overline{z}. H'(u, v(u, \overline{z}), \overline{z}).$$

In other words, with lambda trickery, we can avoid having to worry about the extra parameters $\overline{z}$—they just get absorbed in the lambda notation.

Before we define the term $F$, we need a mechanism for handling ordered pairs. This is provided by the next lemma.

**Lemma lam.1.** There is a lambda term $D$ such that for each pair of lambda terms $M$ and $N$, $D(M, N)(\overline{0}) \triangleright M$ and $D(M, N)(\overline{1}) \triangleright N$.

**Proof.** First, define the lambda term $K$ by

$$K(y) = \lambda x. y.$$  

In other words, $K$ is the term $\lambda y. \lambda x. y$. Looking at it differently, for every $M$, $K(M)$ is a constant function that returns $M$ on any input.

Now define $D(x, y, z)$ by $D(x, y, z) = z(K(y))x$. Then we have

$$D(M, N, \overline{0}) \triangleright \overline{0}(K(N))M \triangleright M$$
$$D(M, N, \overline{1}) \triangleright \overline{1}(K(N))M \triangleright K(N)M \triangleright N,$$

as required.  

The idea is that $D(M,N)$ represents the pair $(M,N)$, and if $P$ is assumed to represent such a pair, $P(0)$ and $P(1)$ represent the left and right projections, $(P)_0$ and $(P)_1$. We will use the latter notations.

**Lemma lam.2.** The lambda representable functions are closed under primitive recursion.

**Proof.** We need to show that given any terms, $G$ and $H$, we can find a term $F$ such that

\[
F(0) \equiv G \\
F(n + 1) \equiv H(n, F(n))
\]

for every natural number $n$. The idea is roughly to compute sequences of pairs $\langle 0, G \rangle, \langle 1, F(1) \rangle, \ldots$, using numerals as iterators. Notice that the first pair is just $\langle 0, G \rangle$. Given a pair $\langle \pi, F(\pi) \rangle$, the next pair, $\langle n + 1, F(n + 1) \rangle$ is supposed to be equivalent to $\langle n + 1, H(n, F(n)) \rangle$. We will design a lambda term $T$ that makes this one-step transition.

The details are as follows. Define $T(u)$ by

\[
T(u) = \langle S((u)_0), H((u)_0, (u)_1) \rangle.
\]

Now it is easy to verify that for any number $n$,

\[
T(\langle \pi, M \rangle) \equiv \langle n + 1, H(\pi, M) \rangle.
\]

As suggested above, given $G$ and $H$, define $F(u)$ by

\[
F(u) = (u(T, (\bar{0}, G)))_1.
\]

In other words, on input $\pi$, $F$ iterates $T$ $n$ times on $\langle 0, G \rangle$, and then returns the second component. To start with, we have

1. $\bar{0}(T, (\bar{0}, G)) \equiv (\bar{0}, G)$
2. $F(\bar{0}) \equiv G$

By induction on $n$, we can show that for each natural number one has the following:

1. $\overline{n + 1}(T, \langle 0, G \rangle) \equiv \langle n + 1, F(n + 1) \rangle$
2. $F(n + 1) \equiv H(\pi, F(\pi))$
For the second clause, we have

\[
F(n + 1) \triangleright (n + 1)(T, (\emptyset, G)))_1 \\
≡ (T(n(T, (\emptyset, G))))_1 \\
≡ (T((\pi, F(\pi))))_1 \\
≡ ((n + 1, H(n, F(\pi))))_1 \\
≡ H(\pi, F(\pi)).
\]

Here we have used the induction hypothesis on the second-to-last line. For the first clause, we have

\[
\overline{n + 1}(T, (\emptyset, G)) \equiv T(\pi(T, (\emptyset, G))) \\
≡ T((\pi, F(\pi))) \\
≡ (n + 1, H(\pi, F(\pi))) \\
≡ (n + 1, F(n + 1)).
\]

Here we have used the second clause in the last line. So we have shown \(F(\emptyset) \equiv G\) and, for every \(n\), \(F(n + 1) \equiv H(\pi, F(\pi))\), which is exactly what we needed.

\[\square\]

Photo Credits

Bibliography