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 $G$ and $H$ that $\lambda$-define functions $g$ and $h$, respectively, we want a term $H'$ that $\lambda$-defines the function $f$ defined by

\[
f(0, \vec{z}) = g(\vec{z}) \\
f(x + 1, \vec{z}) = h(z, f(x, \vec{z}), \vec{z}).
\]

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

\[
F(\overline{0}, \vec{z}) \equiv G(\vec{z}) \\
F(\vec{n} + 1, \vec{z}) \equiv H(\vec{n}, F(\vec{n}, \vec{z}), \vec{z})
\]

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

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

\[
F(\overline{0}) \equiv G \\
F(\vec{n} + 1) \equiv H(\vec{n}, F(\vec{n}))
\]

for every natural number $n$, where

\[
G = \lambda \vec{z}. G'(\vec{z}) \quad \text{and} \quad H(u, v) = \lambda \vec{z}. H'(u, v(u, \vec{z}), \vec{z}).
\]

In other words, with lambda trickery, we can avoid having to worry about the extra parameters $\vec{z}$—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 int.1.** There is a lambda term $D$ such that for each pair of lambda terms $M$ and $N$, $D(M, N)(\overline{0}) \Rightarrow M$ and $D(M, N)(\overline{1}) \Rightarrow N$.

**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

\[
D(M, N, \overline{0}) \Rightarrow \overline{0}(K(N))M \Rightarrow M \quad \text{and} \quad D(M, N, \overline{1}) \Rightarrow \overline{1}(K(N))M \Rightarrow K(N)M \Rightarrow N,
\]

as required. \(\blacksquare\)
The idea is that $D(M, N)$ represents the pair $\langle M, N \rangle$, and if $P$ is assumed to represent such a pair, $P(\overline{0})$ and $P(\overline{1})$ represent the left and right projections, $(P)_0$ and $(P)_1$. We will use the latter notations.

**Lemma int.2.** The $\lambda$-definable 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

\[
F(\overline{0}) \equiv G \\
F(\overline{n+1}) \equiv H(\pi, F(\pi))
\]

for every natural number $n$. The idea is roughly to compute sequences of pairs $\langle \overline{0}, F(\overline{0}) \rangle, \langle \overline{1}, F(\overline{1}) \rangle, \ldots$, using numerals as iterators. Notice that the first pair is just $\langle \overline{0}, G \rangle$. Given a pair $\langle \pi, F(\pi) \rangle$, the next pair, $\langle \overline{n+1}, F(\overline{n+1}) \rangle$ is supposed to be equivalent to $\langle \overline{n+1}, H(\pi, F(\pi)) \rangle$. 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 $n$,

\[
T(\langle \pi, M \rangle) \rightarrow \langle \overline{n+1}, H(\pi, M) \rangle.
\]

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

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

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

1. $\overline{0}(T, \langle \overline{0}, G \rangle) \equiv \langle \overline{0}, G \rangle$

2. $F(\overline{0}) \equiv G$

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

1. $\overline{n+1}(T, \langle \overline{0}, G \rangle) \equiv \langle \overline{n+1}, F(\overline{n+1}) \rangle$

2. $F(\overline{n+1}) \equiv H(\pi, F(\pi))$

For the second clause, we have

\[
F(\overline{n+1}) \rightarrow (\overline{n+1}(T, \langle \overline{0}, G \rangle))_1 \equiv (T(\pi(T, \langle \overline{0}, G \rangle)))_1 \equiv (T(\langle \pi, F(\pi) \rangle))_1 \equiv ((\overline{n+1}, H(\pi, F(\pi))))_1 \equiv H(\pi, F(\pi)).
\]
Here we have used the induction hypothesis on the second-to-last line. For the first clause, we have

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

Here we have used the second clause in the last line. So we have shown \( F(0) \equiv G \) and, for every \( n \), \( F(n + 1) \equiv H(n, F(n)) \), which is exactly what we needed. \( \square \)

Photo Credits

Bibliography