λ-Definable Arithmetical Functions

Proposition rep.1. The successor function succ is λ-definable.

Proof. A term that λ-defines the successor function is
\[ \text{Succ} \equiv \lambda a. \lambda fx. f(afx). \]

Given our conventions, this is short for
\[ \text{Succ} \equiv \lambda a. \lambda f. \lambda x. (f((af)x)). \]

Succ is a function that accepts as argument a number \( a \), and evaluates to another function, \( \lambda fx.(afx) \). That function is not itself a Church numeral. However, if the argument \( a \) is a Church numeral, it reduces to one. Consider:
\[
(\lambda a. \lambda fx.(afx)) \pi \rightarrow \lambda fx.f(\pi fx).
\]
The embedded term \( \pi fx \) is a redex, since \( \pi \) is \( \lambda fx. f^n x \). So \( \pi fx \rightarrow f^n x \) and so, for the entire term we have
\[
\text{Succ} \pi \rightarrow \lambda fx.f(f^n x),
\]
i.e., \( n + 1 \).

Example rep.2. Let’s look at what happens when we apply Succ to \( \bar{0} \), i.e., \( \lambda fx. x \). We’ll spell the terms out in full:
\[
\text{Succ} \bar{0} \equiv (\lambda a. \lambda fx.(afx))((\lambda fx.(afx))(\lambda x.x))
\rightarrow \lambda fx.(f((\lambda fx.(afx))(\lambda x.x)))
\rightarrow \lambda fx.(f((\lambda x.x)(\lambda x.x)))
\rightarrow \lambda fx.(fx) \equiv \bar{1}
\]

Problem rep.1. The term
\[ \text{Succ}' \equiv \lambda n. \lambda fx. nf(fx) \]
λ-defines the successor function. Explain why.

Proposition rep.3. The addition function \( \text{add} \) is λ-definable.

Proof. Addition is λ-defined by the terms
\[ \text{Add} \equiv \lambda ab. \lambda fx. af(bfx) \]
or, alternatively,

\[ \text{Add}' \equiv \lambda ab. a \text{ Succ } b. \]

The first addition works as follows: Add first accept two numbers \( a \) and \( b \). The result is a function that accepts \( f \) and \( x \) and returns \( af(bfx) \). If \( a \) and \( b \) are Church numerals \( \overline{a} \) and \( \overline{b} \), this reduces to \( f^{n+m}(x) \), which is identical to \( f^n(f^m(x)) \). Or, slowly:

\[
\begin{align*}
(\lambda ab. \lambda fx. af(bfx))\overline{a}\overline{b} & \rightarrow \lambda fx. \overline{a} f(\overline{b}fx) \\
& \rightarrow \lambda fx. \overline{a} f(\overline{b}f^m x) \\
& \rightarrow \lambda fx. f^n(\overline{b}f^m x) \equiv \overline{n} + \overline{m}.
\end{align*}
\]

The second representation of addition \( \text{Add}' \) works differently: Applied to two Church numerals \( \overline{n} \) and \( \overline{m} \),

\[ \text{Add}' \overline{n} \overline{m} \rightarrow \overline{n} \text{ Succ } \overline{m}. \]

But \( \overline{af}x \) always reduces to \( f^n(x) \). So,

\[ \overline{n} \text{ Succ } \overline{m} \rightarrow \text{ Succ } ^n \overline{m}. \]

And since \( \text{Succ} \) \( \lambda \)-defines the successor function, and the successor function applied \( n \) times to \( m \) gives \( n + m \), this in turn reduces to \( n + m \).

\[ \boxed{\text{Proposition rep.4. \ Multiplication is } \lambda \text{-definable by the term} \quad \text{Mult} \equiv \lambda ab. \lambda fx. a(bfx)} \]

Proof. To see how this works, suppose we apply \( \text{Mult} \) to Church numerals \( \overline{n} \) and \( \overline{m} \). \( \text{Mult} \overline{n} \overline{m} \) reduces to \( \lambda fx. \overline{n}(\overline{m}f)x \). The term \( \overline{mf} \) defines a function which applies \( f \) to its argument \( m \) times. Consequently, \( \overline{n}(\overline{mf})x \) applies the function “apply \( f \) \( m \) times” itself \( n \) times to \( x \). In other words, we apply \( f \) to \( x \), \( n \cdot m \) times. But the resulting normal term is just the Church numeral \( \overline{nm} \).

We can actually simplify this term further by \( \eta \)-reduction:

\[ \text{Mult} \equiv \lambda ab. \lambda f. a(\overline{bf})x \]

But then we first have to explain \( \eta \)-reduction.

\[ \boxed{\text{Problem rep.2. \ Multiplication can be } \lambda \text{-defined by the term} \quad \text{Mult}' \equiv \lambda ab. \text{ Succ } \overline{a} \overline{b}} \]

Explain why this works.
The definition of exponentiation as a $\lambda$-term is surprisingly simple:

$$\text{Exp} \equiv \lambda be. eb.$$  

The first argument $b$ is the base and the second $e$ is the exponent. Intuitively, $ef$ is $f^e$ by our encoding of numbers. If you find it hard to understand, we can still define exponentiation also by iterated multiplication:

$$\text{Exp}' \equiv \lambda be. e(\text{Mult } b)1.$$  

Predecessor and subtraction on Church numeral is not as simple as we might think: it requires encoding of pairs.

Photo Credits

Bibliography