**Proposition rep.1.** The successor function \( \text{succ} \) is \( \lambda \)-definable.

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

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

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

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

**Problem rep.1.** The term
\[
\text{Succ’} \equiv \lambda n. \lambda f x. n f (f x)
\]
\( \lambda \)-defines the successor function. Explain why.

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

*Proof.* Addition is \( \lambda \)-defined by the terms
\[
\text{Add} \equiv \lambda a b. \lambda f x. a f (b f x)
\]
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 \( \overline{f}^{n+m}(x) \), which is identical to \( \overline{f}^n(f^m(x)) \). Or, slowly:

\[
\begin{align*}
(\lambda ab. \lambda fx. af(bfx))\overline{a}\overline{b} \rightarrow & \lambda fx. \overline{a} f(\overline{b} f x) \\
\rightarrow & \lambda fx. \overline{a} f(\overline{b} f^2 x) \\
\rightarrow & \lambda fx. \overline{a} f^3(f^2 x) \equiv n + m.
\end{align*}
\]

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 \( \overline{f}x \) 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 \( n + m \).

\[ \square \]

**Proposition rep.4.** Multiplication is \( \lambda \)-definable by the term

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

**Proof.** To see how this works, suppose we apply Mult to Church numerals \( \overline{n} \) and \( \overline{m} \). Mult \( \overline{n} \overline{m} \) reduces to \( \lambda fx. \overline{n}(\overline{m} f)x \). The term \( \overline{m} f \) defines a function which applies \( f \) to its argument \( m \) times. Consequently, \( \overline{n}(\overline{m} f)x \) 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 \( \overline{nm} \).

\[ \square \]

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

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

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

\[ \square \]

**Problem rep.2.** Multiplication can be \( \lambda \)-defined by the term

\[ \text{Mult'} \equiv \lambda ab. a(\text{Add } a)\overline{0}. \]

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

$$\text{Exp} \equiv \lambda b e. eb.$$  

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:

$$\text{Exp'} \equiv \lambda b e. e(\text{Mult } b)\mathsf{T}.$$  

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

**Photo Credits**

**Bibliography**