lam.1 Representability by Lambda Terms

cmp:lam:rep:sec How can the lambda calculus serve as a model of computation? At first, it is not even clear how to make sense of this statement. To talk about computability on the natural numbers, we need to find a suitable representation for such numbers. Here is one that works surprisingly well.

Definition lam.1. For each natural number n, define the *numeral* $\overline{n}$ to be the lambda term $\lambda x.\, \lambda y.\, (x(x(x(\dots x(y)))))$, where there are n x's in all.

The terms $\overline{n}$ are "iterators": on input f, $\overline{n}$ returns the function mapping y to $f^n(y)$. Note that each numeral is normal. We can now say what it means for a lambda term to "compute" a function on the natural numbers.

Definition lam.2. Let $f(x_0, \dots, x_{n-1})$ be an n-ary partial function from $\mathbb{N}$ to $\mathbb{N}$. We say a lambda term X *represents* f if for every sequence of natural numbers $m_0, \dots, m_{n-1}$,

$$X\overline{m_0}\,\overline{m_1}\dots\overline{m_{n-1}} \triangleright \overline{f(m_0, m_1, \dots, m_{n-1})}$$

if $f(m_0, \dots, m_{n-1})$ is defined, and $X\overline{m_0}\,\overline{m_1}\dots\overline{m_{n-1}}$ has no normal form otherwise.

cmp:lam:rep:thm:lambda-rep **Theorem lam.3.** *A function f is a partial computable function if and only if it is represented by a lambda term.*

This theorem is somewhat striking. As a model of computation, the lambda calculus is a rather simple calculus; the only operations are lambda abstraction and application! From these meager resources, however, it is possible to implement any computational procedure. explanation

Photo Credits

Bibliography