Primitive recursion in pure λ-calculus

Viktor Engelmann
Viktor.Engelmann@RWTH-Aachen.de

August 30, 2009

Abstract

1 Definitions

... For example the addition of two numbers n and m can be achieved by applying the *succ* function n times to m by using

$$\begin{aligned} & n\ succ\ m \\ = \ & (\lambda f.\lambda x.\underbrace{f(f(\cdots(f}_{n \text{ times}}\ x)\cdots)))\ succ\ m \\ = \ & (\lambda x.\underbrace{succ(succ(\cdots(succ}_{n \text{ times}}\ x)\cdots)))\ m \\ = \ & \underbrace{succ(succ(\cdots(succ}_{n \text{ times}}\ m)\cdots)) \end{aligned}$$

...

2 Theorem

A primitive recursion $\overline{f}$ is presentable by a λ-term of the form

$$\lambda i.\lambda x_1.\cdots\lambda x_n.(\underbrace{\lambda g.g(\lambda y_1.\lambda y_2.y_2)}_{sel_2^2})\ (i\ h\ (g\ x_1\ \cdots\ x_n))$$

Proof:
Let $\overline{f}$ a primitive recursion with

$$\begin{aligned} \overline{f}(0,x_1,\cdots,x_n) &= \overline{g}(x_1,\cdots,x_n) \\ \overline{f}(i,x_1,\cdots,x_n) &= \overline{h}(x_1,\cdots,x_n,i-1,\overline{f}(i-1,x_1,\cdots,x_n)) \text{ for } i>0 \end{aligned}$$

and let g and h be λ-terms that calculate $\overline{g}$ and $\overline{h}$. if you apply such a function to some values of i, you notice a simple pattern

i	$f(i, x_1, \cdots, x_n)$
0	$\overline{g}(x_1, \cdots, x_n)$
1	$\overline{h}(x_1, \cdots, x_n, 0, \overline{g}(x_1, \cdots, x_n))$
2	$\overline{h}(x_1, \cdots, x_n, 1, \overline{h}(x_1, \cdots, x_n, 0, \overline{g}(x_1, \cdots, x_n)))$
3	$\overline{h}(x_1, \cdots, x_n, 2, \overline{h}(x_1, \cdots, x_n, 1, \overline{h}(x_1, \cdots, x_n, 0, \overline{g}(x_1, \cdots, x_n))))$

Keep in mind that in the λ-calculus the result of $h\ x_1\ \cdots\ x_n\ (i-1)$ is a function, which is applied to the result of the recursive call. Let's look at the previous table in terms of the λ-calculus.

i	$f\ i\ x_1\ \cdots\ x_n$
0	$g\ x_1\ \cdots\ x_n$
1	$(h\ x_1\ \cdots\ x_n\ 0)\ (g\ x_1\ \cdots\ x_n)$
2	$(h\ x_1\ \cdots\ x_n\ 1)\ ((h\ x_1\ \cdots\ x_n\ 0)\ (g\ x_1\ \cdots\ x_n))$
3	$(h\ x_1\ \cdots\ x_n\ 2)\ ((h\ x_1\ \cdots\ x_n\ 1)\ ((h\ x_1\ \cdots\ x_n\ 0)\ (g\ x_1\ \cdots\ x_n)))$

Now imagine that h didn't get the $i-1$ parameter. The λ-terms then would look like this

i	$f\ i\ x_1\ \cdots\ x_n$
0	$g\ x_1\ \cdots\ x_n$
1	$(h\ x_1\ \cdots\ x_n)\ (g\ x_1\ \cdots\ x_n)$
2	$(h\ x_1\ \cdots\ x_n)\ ((h\ x_1\ \cdots\ x_n)\ (g\ x_1\ \cdots\ x_n))$
3	$(h\ x_1\ \cdots\ x_n)\ ((h\ x_1\ \cdots\ x_n)\ ((h\ x_1\ \cdots\ x_n)\ (g\ x_1\ \cdots\ x_n)))$

so if h didn't get the $i-1$ parameter, $f\ i\ x_1\ \cdots\ x_n$ would be equivalent to the i-fold application of $(h\ x_1\ \cdots\ x_n)$ to $(g\ x_1\ \cdots\ x_n)$.

In the pure λ-calculus the natural number i is represented by the λ-term

$$\lambda f.\lambda x.\underbrace{f(f(\cdots(f}_{i \text{ times}}\ x)\cdots))$$

which is the i-fold application of the first parameter to the second. Therefore f could be written as

$$f = \lambda i.\lambda x_1.\cdots.\lambda x_n.i\ (h\ x_1\ \cdots\ x_n)\ (g\ x_1\ \cdots\ x_n)$$

However h still has to get the $i-1$ parameter, but in the pure λ-terms, i doesn't have to be passed as a parameter from outside of the functioncall. We can just define functions $\dot{g}$ and $\dot{h}$ which return a pair containing the result and also the $i-1$ parameter for the outer applications of h.

$$\begin{aligned} \dot{g}\ x_1\ \cdots\ x_n &= (0, g\ x_1\ \cdots\ x_n) \\ \dot{h}\ x_1\ \cdots\ x_n\ \underbrace{r}_{\text{recursion result}} &= (1 + (sel_1^2\ r), h\ x_1\ \cdots\ x_n\ (\underbrace{sel_1^2\ r}_{i-1})\ (sel_2^2\ r)) \end{aligned}$$

using these definitions, we get

$$f = \lambda i.\lambda x_1.\cdots\lambda x_n.sel_2^2\ (i\ (\dot{h}\ x_1\ \cdots\ x_n)(\dot{g}\ x_1\ \cdots\ x_n))$$

but moreover, using the above idea, we don't have to pass $x_1\ \cdots\ x_n$ to $\dot{h}$ directly, either. Instead we can define functions $\ddot{g}$ and $\ddot{h}$ which return an $n+2$-tuple, that also contains the $x_1\ \cdots\ x_n$ parameters:

$$\begin{aligned}
\ddot{g}\ x_1\ \cdots\ x_n &= (x_1, \cdots, x_n, 0, \overbrace{g\ x_1\ \cdots\ x_n}) \\
\ddot{h}\ r &= (\underbrace{sel_1^{n+2}\ r}_{x_1}, \cdots, \underbrace{sel_n^{n+2}\ r}_{x_n}, \underbrace{(1 + sel_{n+1}^{n+2}\ r)}_{1+(i-1)}, \overbrace{h\ \underbrace{(sel_1^{n+2}\ r)}_{x_1}\ \cdots\ \underbrace{(sel_n^{n+2}\ r)}_{x_n}\ \underbrace{(sel_{n+1}^{n+2}\ r)}_{i-1}(\ \underbrace{sel_{n+2}^{n+2}\ r}_{recursionresult}\ )})
\end{aligned}$$

so now we have

$$f = \lambda i.\lambda x_1.\cdots\lambda x_n.sel_{n+2}^{n+2}(i\ \ddot{h}\ (\ddot{g}\ x_1\ \cdots\ x_n))$$

which only has to be brought into pure λ-form, but that is also possible.

One important observation is that this gives us a new characterization for the borderline between primitive-recursive and μ-recursive functions: in the pure λ-calculus, primitive recursive functions can be computed without using fixedpoint-combinators, while μ-recursive functions in general require them. Also, since every computable set can be computed using the μ-operatior only once[1], this also shows that every computable set can be computed in pure λ-calculus using only one fixedpoint-combinator.

[1] see http://s-inf.de/Skripte/HS/Rekursionstheorie.2003-SS-Thomas.(JRe).Skript.ps page 16