CS520, Handout 05, 19 Sept 2006

More on Recursion in the Lambda Calculus

Reading: [Pierce, pages 65-67, pages 143-145]

Motivating Example

In [Pierce, page 61] the term `plus` simulates addition on the Church numerals.

Suppose we want a term `add` that performs addition directly, i.e., on

$$0 \qquad \mathtt{succ}\,0 \qquad \mathtt{succ}\,(\mathtt{succ}\,0) \qquad \mathtt{succ}\,(\mathtt{succ}\,(\mathtt{succ}\,0)) \qquad \ldots$$

rather than on the Church numerals $\mathtt{c}_0, \mathtt{c}_1, \mathtt{c}_2, \ldots$.

For convenience, define $1 \triangleq \mathtt{succ}\,0$, $2 \triangleq \mathtt{succ}\,(\mathtt{succ}\,0)$, etc. We are now working in $\boldsymbol{\lambda}\mathbf{NB}$ not in $\boldsymbol{\lambda}$. Let m and n be metavariables ranging over $\{0, 1, 2, \ldots\}$.

If we succeed in defining the desired term `add`, it will behave as follows:

$$\mathtt{add}\; m\; n \Rightarrow^{*} \underbrace{\mathtt{succ}\,(\mathtt{succ}\,(\cdots(\mathtt{succ}}_{m \text{ times}}\; n)\cdots))$$

This will be achieved if add can be evaluated as follows:

$$\mathtt{add}\ m\ n \Rightarrow^* \mathtt{if}\ (\mathtt{iszero}\ m)\ \mathtt{then}\ n\ \mathtt{else}\ (\mathtt{succ}\ (\underset{\uparrow}{\mathtt{add}}\ (\mathtt{pred}\ m)\ n))$$

Note the **recursive call** to add. This corresponds to the usual (recursive) definition of "addition" using "successor" and "predecessor":

$$m + n = \begin{cases} n & \text{if } m = 0, \\ \mathtt{succ}\ (\mathtt{pred}\ m + n) & \text{if } m > 0. \end{cases}$$

Allowing evaluation under λ, we want the following behavior:

$$\mathtt{add} \Rightarrow^* \lambda \mathtt{x}.\ \lambda \mathtt{y}.\ \mathtt{if}\ (\mathtt{iszero}\ \mathtt{x})\ \mathtt{then}\ \mathtt{y}\ \mathtt{else}\ (\mathtt{succ}\ (\underset{\uparrow}{\mathtt{add}}\ (\mathtt{pred}\ \mathtt{x})\ \mathtt{y}))$$

A Special Magic Term

Suppose we can define a fixed term Y which, for an arbitrary term t, behaves as follows:

$$\begin{aligned} \mathrm{Y}\ t &\Rightarrow \langle t \rangle \\ &\Rightarrow^* \underbrace{t\ (t\ (\cdots (t}_{m \geqslant 0}\ \langle t \rangle) \cdots)) \end{aligned}$$

where we can make m arbitrarily small or large, and $\langle t \rangle$ is another term produced by applying Y to t.

Such a term Y is called a fixed-point combinator.

Armed with such magic Y, we can now return to the problem of defining add.

The following reduction [this handout, page 3]:

$$\mathtt{add} \Rightarrow^* \lambda\mathtt{x}.\, \lambda\mathtt{y}.\, \mathtt{if}\ (\mathtt{iszero}\,\mathtt{x})\ \mathtt{then}\ \mathtt{y}\ \mathtt{else}\ (\mathtt{succ}\,(\underset{\uparrow}{\mathtt{add}}\ (\mathtt{pred}\ \mathtt{x})\ \mathtt{y}))$$

suggests the following definition:

$$\mathtt{addfn} \triangleq \lambda\, \underset{\uparrow}{\mathtt{f}}\ \lambda\mathtt{x}.\, \lambda\mathtt{y}.\, \mathtt{if}\ (\mathtt{iszero}\,\mathtt{x})\ \mathtt{then}\ \mathtt{y}\ \mathtt{else}\ (\mathtt{succ}\,(\underset{\uparrow}{\mathtt{f}}\ (\mathtt{pred}\ \mathtt{x})\ \mathtt{y}))$$

$$\mathtt{add} \triangleq \mathtt{Y}\,\mathtt{addfn}$$

Verify that $\mathtt{add}\, 2\, 3 \Rightarrow^* 5$

How to Define a Fixed-Point Combinator

There are different ways of defining a fixed-point combinator Y. For definiteness here, we use the following definition of Y [Handout 04, page 9, Pierce, page 65]:

$$\boxed{\mathtt{Y} \triangleq \lambda\mathtt{f}.\, (\lambda\mathtt{x}.\, \mathtt{f}\ (\mathtt{x}\ \mathtt{x})\,)(\lambda\mathtt{x}.\, \mathtt{f}\ (\mathtt{x}\ \mathtt{x})\,)}$$

For an arbitrary term t, it is easy to check that:

$$\begin{aligned} \mathtt{Y}\ t &\Rightarrow \langle t\rangle \\ &\Rightarrow^* \underbrace{t\ (t\ (\cdots (t}_{m\geqslant 0}\ \langle t\rangle)\cdots)) \qquad \text{where} \end{aligned}$$

$$\langle t\rangle \triangleq (\lambda\mathtt{x}.\, t\ (\mathtt{x}\ \mathtt{x})\,)\,(\lambda\mathtt{x}.\, t\ (\mathtt{x}\ \mathtt{x})\,)$$