Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lukas Stevens

Lambda Calculus
Winter Term 2022/23
Solutions to Exercise Sheet 2

Exercise 1 (β-reduction)

A term t is in β-normal form if there is no term t' such that $t \to_\beta t'$. List all terms t such that:

$$(\lambda x.\ (\lambda x\ y.\ x)\ z\ y)\ ((\lambda x.\ x\ x)\ (\lambda x.\ x\ x)\ ((\lambda x\ y.\ x)\ y)) \to_\beta^* t$$

Which are normal forms?

Solution

Let $\omega := \lambda x.\ x\ x$.

- $(\lambda x.\ (\mathsf{true}\ z)\ y)\ (\omega\ \omega\ (\mathsf{true}\ y))$
- $(\lambda x.\ (\lambda x.\ z)\ y)\ (\omega\ \omega\ (\mathsf{true}\ y))$
- $(\lambda x.\ z)\ (\omega\ \omega\ (\mathsf{true}\ y))$
- $(\lambda x.\ (\mathsf{true}\ z)\ y)\ (\omega\ \omega\ (\lambda x.\ y))$
- $(\lambda x.\ (\lambda x.\ z)\ y)\ (\omega\ \omega\ (\lambda x.\ y))$
- $(\lambda x.\ z)\ (\omega\ \omega\ (\lambda x.\ y))$
- $(\mathsf{true}\ z)\ y$
- $(\lambda x.\ z)\ y$
- z

Only z is a normal form.

Exercise 2 (Lists in λ-calculus)

Specify λ-terms for nil, cons, hd, tl and null, that encode lists in the λ-calculus. Show that your terms satisfy the following conditions:

$$\begin{array}{llll} \mathsf{null}\ \mathsf{nil} & \to_\beta^* \mathsf{true} & \mathsf{hd}\ (\mathsf{cons}\ x\ l) & \to_\beta^* x \\ \mathsf{null}\ (\mathsf{cons}\ x\ l) & \to_\beta^* \mathsf{false} & \mathsf{tl}\ (\mathsf{cons}\ x\ l) & \to_\beta^* l \end{array}$$

Hint: Use pairs.

Solution

A list is represented as a pair of its head and its tail. We define an encoding of pairs and booleans as below:

$$\begin{array}{lll} \mathsf{true} := \lambda x\ y.\ x & \mathsf{false} := \lambda x\ y.\ y & \\ \mathsf{pair} := \lambda a\ b\ f.\ f\ a\ b & \mathsf{fst} := \lambda p.\ p\ (\lambda x\ y.\ x) & \mathsf{snd} := \lambda p.\ p\ (\lambda x\ y.\ y) \end{array}$$

We only specify the solution without tagging here:

$$\begin{array}{rlrl} \mathsf{nil} & := \mathsf{false} & \mathsf{cons} & := \mathsf{pair} \\ \mathsf{hd} & := \mathsf{fst} & \mathsf{tl} & := \mathsf{snd} \\ \mathsf{null} & := \lambda l.\ l\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true} & & \end{array}$$

We can now show the above conditions:

$$\begin{array}{rcl} \mathsf{null\ nil} & = & (\lambda l.\ l\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true})\ \mathsf{false} \\ & \to_\beta^* & \mathsf{false}\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true} \\ & = & (\lambda x\ y.\ y)\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true} \\ & \to_\beta^* & \mathsf{true} \\ \mathsf{null}\ (\mathsf{cons}\ x\ l) & = & (\lambda l.\ l\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true})\ (\mathsf{pair}\ x\ l) \\ & =_\alpha & (\lambda l'.\ l'\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true})\ (\mathsf{pair}\ x\ l) \\ & \to_\beta^* & (\mathsf{pair}\ x\ l)\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true} \\ & = & ((\lambda a\ b\ f.\ f\ a\ b)\ x\ l)\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true} \\ & \to_\beta^* & (\lambda f.\ f\ x\ l)\ (\lambda h\ t\ d.\ \mathsf{false})\ \mathsf{true} \\ & \to_\beta^* & (\lambda h\ t\ d.\ \mathsf{false})\ x\ l\ \mathsf{true} \\ & \to_\beta^* & \mathsf{false} \end{array}$$

$$\begin{array}{rcl} \mathsf{hd}\ (\mathsf{cons}\ x\ l) & = & \mathsf{fst}\ (\mathsf{pair}\ x\ l) \\ & = & (\lambda p.\ p\ (\lambda x\ y.\ x))\ ((\lambda a\ b\ f.\ f\ a\ b)\ x\ l) \\ & \to_\beta^* & ((\lambda a\ b\ f.\ f\ a\ b)\ x\ l)\ (\lambda x\ y.\ x) \\ & \to_\beta^* & (\lambda f.\ f\ x\ l)\ (\lambda x\ y.\ x) \\ & \to_\beta^* & (\lambda x\ y.\ x)\ x\ l \\ & \to_\beta^* & x \\ \mathsf{tl}\ (\mathsf{cons}\ x\ l) & = & \mathsf{snd}\ (\mathsf{pair}\ x\ l) \\ & = & (\lambda p.\ p\ (\lambda x\ y.\ y))\ ((\lambda a\ b\ f.\ f\ a\ b)\ x\ l) \\ & \to_\beta^* & ((\lambda a\ b\ f.\ f\ a\ b)\ x\ l)\ (\lambda x\ y.\ y) \\ & \to_\beta^* & (\lambda f.\ f\ x\ l)\ (\lambda x\ y.\ y) \\ & \to_\beta^* & (\lambda x\ y.\ y)\ x\ l \\ & \to_\beta^* & l \end{array}$$

An alternative solution redefines nil and null:

$$\begin{array}{lll} \mathsf{nil} := \lambda f.\ \mathsf{true} & & \mathsf{null} := \lambda p.\ p\ (\lambda x\ l.\ \mathsf{false}) \end{array}$$

We only have to check if $\mathsf{null}\ \mathsf{nil} \to_\beta^* \mathsf{true}$ and $\mathsf{null}\ (\mathsf{cons}\ x\ l) \to_\beta^* \mathsf{false}$:

$$
\begin{array}{rcl}
\mathsf{null}\ \mathsf{nil} & = & (\lambda p.\ p\ (\lambda x\ l.\ \mathsf{false}))\ (\lambda f.\ \mathsf{true}) \\
 & \to_\beta^* & (\lambda f.\ \mathsf{true})\ (\lambda x\ l.\ \mathsf{false}) \\
 & \to_\beta^* & \mathsf{true} \\
\mathsf{null}\ (\mathsf{cons}\ x\ l) & = & (\lambda p.\ p\ (\lambda x\ l.\ \mathsf{false}))\ (\mathsf{pair}\ x\ l) \\
 & \to_\beta^* & (\mathsf{pair}\ x\ l)\ (\lambda x\ l.\ \mathsf{false}) \\
 & = & ((\lambda a\ b\ f.\ \ f\ a\ b)\ x\ l)\ (\lambda x\ l.\ \mathsf{false}) \\
 & \to_\beta^* & (\lambda f.\ \ f\ x\ l)\ (\lambda x\ l.\ \mathsf{false}) \\
 & \to_\beta^* & (\lambda x\ l.\ \mathsf{false})\ x\ l \\
 & \to_\beta^* & \mathsf{false}
\end{array}
$$

Homework 3 (Substitution Lemma)

Show that, given $x \neq y$ and $x \notin \mathrm{FV}(u)$:

$$s[t/x][u/y] = s[u/y][t[u/y]/x]$$

Homework 4 (Trees in λ-calculus)

Encode a datatype of binary trees in lambda calculus. Specify the operations tip and node that construct trees, as well as isTip, left, right, and value. Each tip should carry a value, whereas each node should consist of two subtrees.

Show that the following holds:

$$\begin{aligned}
\mathsf{isTip}\ (\mathsf{tip}\ a) &\to^*_\beta \mathsf{true} \\
\mathsf{isTip}\ (\mathsf{node}\ x\ y) &\to^*_\beta \mathsf{false} \\
\mathsf{value}\ (\mathsf{tip}\ a) &\to^*_\beta a \\
\mathsf{left}\ (\mathsf{node}\ x\ y) &\to^*_\beta x \\
\mathsf{right}\ (\mathsf{node}\ x\ y) &\to^*_\beta y
\end{aligned}$$

Homework 5 (Alternative Encoding of Lists)

In this exercise, we consider an alternative encoding of lists. The list $[x, y, z]$, for instance, will now be encoded as: $\lambda c n.\ c\,x\,(c\,y\,(c\,z\,n))$ (speaking in terms of functional programming, each list now encodes its corresponding *fold*). As in the tutorial, define the functions nil, cons, hd, and null for this encoding and show that they satisfy the same conditions. You do not need to define tl.

Homework 6 (Multiplication)

Define multiplication as a closed λ-term using add but no other helper functions and demonstrate its correctness on an example.