

Exercise 1 (β -reduction)

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

$$(\lambda x. (\lambda x y. x) z y) ((\lambda x. x x) (\lambda x. x x) ((\lambda x y. x) y)) \rightarrow_{\beta}^* t$$

Which are normal forms?

Solution

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

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

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

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:

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

We only specify the solution without tagging here:

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

We can now show the above conditions:

$$\begin{aligned} \text{null nil} &= (\lambda l. l (\lambda h t d. \text{false}) \text{true}) \text{false} \\ &\rightarrow_{\beta}^* \text{false} (\lambda h t d. \text{false}) \text{true} \\ &= (\lambda x y. y) (\lambda h t d. \text{false}) \text{true} \\ &\rightarrow_{\beta}^* \text{true} \\ \text{null (cons } x l) &= (\lambda l. l (\lambda h t d. \text{false}) \text{true}) (\text{pair } x l) \\ &=_{\alpha} (\lambda l'. l' (\lambda h t d. \text{false}) \text{true}) (\text{pair } x l) \\ &\rightarrow_{\beta}^* (\text{pair } x l) (\lambda h t d. \text{false}) \text{true} \\ &= ((\lambda a b f. f a b) x l) (\lambda h t d. \text{false}) \text{true} \\ &\rightarrow_{\beta}^* (\lambda f. f x l) (\lambda h t d. \text{false}) \text{true} \\ &\rightarrow_{\beta}^* (\lambda h t d. \text{false}) x l \text{true} \\ &\rightarrow_{\beta}^* \text{false} \\ \text{hd (cons } x l) &= \text{fst (pair } x l) \\ &= (\lambda p. p (\lambda x y. x)) ((\lambda a b f. f a b) x l) \\ &\rightarrow_{\beta}^* ((\lambda a b f. f a b) x l) (\lambda x y. x) \\ &\rightarrow_{\beta}^* (\lambda f. f x l) (\lambda x y. x) \\ &\rightarrow_{\beta}^* (\lambda x y. x) x l \\ &\rightarrow_{\beta}^* x \\ \text{tl (cons } x l) &= \text{snd (pair } x l) \\ &= (\lambda p. p (\lambda x y. y)) ((\lambda a b f. f a b) x l) \\ &\rightarrow_{\beta}^* ((\lambda a b f. f a b) x l) (\lambda x y. y) \\ &\rightarrow_{\beta}^* (\lambda f. f x l) (\lambda x y. y) \\ &\rightarrow_{\beta}^* (\lambda x y. y) x l \\ &\rightarrow_{\beta}^* l \end{aligned}$$

An alternative solution redefines nil and null:

$$\text{nil} := \lambda f. \text{true} \qquad \text{null} := \lambda p. p (\lambda x l. \text{false})$$

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

$$\begin{aligned} \text{null nil} &= (\lambda p. p (\lambda x l. \text{false})) (\lambda f. \text{true}) \\ &\rightarrow_{\beta}^* (\lambda f. \text{true}) (\lambda x l. \text{false}) \\ &\rightarrow_{\beta}^* \text{true} \\ \text{null (cons } x l) &= (\lambda p. p (\lambda x l. \text{false})) (\text{pair } x l) \\ &\rightarrow_{\beta}^* (\text{pair } x l) (\lambda x l. \text{false}) \\ &= ((\lambda a b f. f a b) x l) (\lambda x l. \text{false}) \\ &\rightarrow_{\beta}^* (\lambda f. f x l) (\lambda x l. \text{false}) \\ &\rightarrow_{\beta}^* (\lambda x l. \text{false}) x l \\ &\rightarrow_{\beta}^* \text{false} \end{aligned}$$

Homework 3 (Substitution Lemma)

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

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

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:

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

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: $\lambda cn. cx(cy(czn))$ (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.