Exercises - Bonus

λ-calculus

The syntax of λ-calculus is in general given in Backus–Naur form:

$$t, s ::= x \parallel t@s \parallel \lambda x.t$$

meaning that:

- λ-terms will be denoted t and s,
- a variable x is a λ-term,
- the result of applying a λ-term t to another λ-term s is a λ-term[1],
- the result of abstracting a variable x from a term t is a λ-term.

In this definition, x is more of a meta-variable: y, z, w, … as well as $\lambda y.t$, $\lambda z.t$, …, are valid λ-terms. This is enforced by α-conversion, which let e.g., $\lambda x.x =_\alpha \lambda y.y$ since those terms are "essentially the same" (in this case, they both capture the identity function, denoted id).

We follow those conventions

- Outermost parentheses are omitted, so (t) is t.
- Applications associate to the left; thus, $t@s@u$ means $(t@s)@u$.
- The body of a lambda abstraction (the part after the dot) extends as far to the right as possible. In particular, $\lambda x.t@u$ means $\lambda x.(t@u)$, and not $(\lambda x.t)@u$.
- Multiple lambda abstractions can be contracted; thus $\lambda xyz.t$ will abbreviate $\lambda x.\lambda y.\lambda z.t$.

The best places to learn more about this are B. Pierce's textbook [1], and if you want to become an expert, Barendregts's books [2–4].

Exercise (inspired by [5, Exercise 3]):

1. Write the following terms with as few parenthesis as possible, without changing the meaning or structure of the terms:

   - $(\lambda x.(\lambda y.(\lambda z.((x@z)@(y@z)))))$,
   - $(((a@b)@(c@d))@((e@f)@(g@h)))$,
   - $(\lambda x.((\lambda y.(y@x))@(\lambda v.v)@z)@u)@(\lambda w.w)$,

2. Restore all the dropped parentheses in the following terms, without changing the meaning or structure of the terms:

   - $x@x@x@x$,
   - $\lambda x.x@\lambda y.y$,
   - $\lambda x.(x@\lambda y.y@x@x)@x$

Read [5, Section 2.4] for a gentle introduction to β-reduction and one example.

Definition We let true and false be $\text{tru} = \lambda xy.x$ and $\text{fls} = \lambda xy.y$.
Exercise Write all the possible β-reductions from fls@fls@(fls@tru) to id.

[1] The "application symbol" @ is often omitted.

References

[1] B.C. Pierce, Types and programming languages, MIT Press, London, England, 2002.

[2] H.P. Barendregt, The lambda calculus: Its syntax and semantics, revised, Elsevier, 1984.

[3] H.P. Barendregt, Lambda calculus with types, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Handbook of Logic in Computer Science, Oxford University Press, 1992: pp. 117–309.

[4] H.P. Barendregt, G. Manzonetto, A lambda calculus satellite, College Publications, 2023.

[5] P. Selinger, Lecture notes on the lambda calculus, (2013). https://arxiv.org/pdf/0804.3434.