TD 1: λ-calculus and encodings of data types

jean-marie.madiot@ens-lyon.org

Exercise 1 *λ-calculus*

The λ-calculus is defined by the following syntax:

$$M, N \ := \ x \quad | \quad \lambda x.\, M \quad | \quad M\, N \qquad\qquad \text{where } x \text{ is a variable}$$

and the following rewriting rule (β reduction):

$$(\lambda x.\, M)\, N \to M[N/x]\ .$$

The symmetric reflexive transitive closure of $\to$ is written $=_\beta$. Here are some usual terms:

$$I := \lambda x.\, x \qquad K := \lambda x \lambda y.\, x \qquad S := \lambda x \lambda y \lambda z.\, x\, z\, (y\, z) \qquad \Delta := \lambda x.\, x\, x \qquad \Omega := \Delta\, \Delta$$

1. Reduce the λ-terms $\Delta\, I\, I$ and Ω.
2. Give the reduction graphs of the terms: $S\, K\, K$, $\Delta\, (I\, I)$ and $K\, I\, \Omega$.

Exercise 2 *Pairs and sum types*

Give λ-terms for

$\langle \_, \_ \rangle$	pair constructor	ι_1	first injection
π_1	first projection	ι_2	second injection
π_2	second projection	case	matching

such that: $\pi_1\, \langle x, y \rangle =_\beta x \quad , \quad \pi_2\, \langle x, y \rangle =_\beta y \quad , \quad \mathrm{case}\, (\iota_1\, x)\, f\, g =_\beta f\, x \quad , \quad \mathrm{case}\, (\iota_2\, x)\, f\, g =_\beta g\, x\ .$

Exercise 3 *Church encodings*

The Church encoding of a natural number n is the term $\overline{n} := \lambda f x. f^n x$ (n iterations of the function f at x).

1. Write $\overline{0}$ and $\overline{3}$.
2. Write a successor function: $\mathrm{S}\, \overline{n} =_\beta \overline{n+1}$.
3. Write an iterator, i.e. a term Iter such that for all terms M, N, we have

$$\mathrm{Iter}\ M\ N\ \overline{0} =_\beta M \qquad \text{and} \qquad \mathrm{Iter}\ M\ N\ (\mathrm{S}\ \overline{n}) =_\beta N\ (\mathrm{Iter}\ M\ N\ \overline{n})\ .$$

4. Write terms encoding addition and multiplication.
5. Which function is represented by the term $\overline{n}\, \overline{m}$?

We represent booleans by $\mathrm{T} := \lambda xy.x$ and $\mathrm{F} := \lambda xy.y$.

6. Give an encoding of `if then else`.
7. How would you encode pairs?
8. Suggest a term encoding the predecessor function.

Exercise 4 *λ-calculus II*

1. Characterize λ-terms in β normal form.
2. Restrict β reduction in order to implement call by name and call by value. Find a λ-term distinguishing these two reduction strategies.

Exercise 5 *Barendregt natural numbers*

The Barendregt natural numbers $\ulcorner n \urcorner$ are defined by:

$$\ulcorner 0 \urcorner := I \qquad\qquad \ulcorner n+1 \urcorner := \lambda k.k\ \mathrm{F}\ \ulcorner n \urcorner$$

1. Implement the functions successor, predecessor, and conditional (if-zero).
2. Suggest an implementation of addition.

Exercise 6 *Lists and trees*

We want to encode lists in λ-calculus by terms of the form $\lambda c.\lambda n.M[c, n]$. Intuitively, a list is a function with two arguments: the first is a function in the case of a non-empty list, the second is some value in the case of the empty list. For instance the list `["Bacon"; "Lettuce"; "Tomato"]` will be represented by:

$$\lambda c.\lambda n.(c\ \texttt{"Bacon"}\ (c\ \texttt{"Lettuce"}\ (c\ \texttt{"Tomato"}\ n)))\ .$$

1. Write the operators nil and cons.
2. Write an *iterator* fold such that

$$\mathrm{fold}\ f\ u\ \mathrm{nil} =_\beta u \qquad \text{and} \qquad \mathrm{fold}\ f\ u\ (\mathrm{cons}\ a\ l) =_\beta f\ a\ (\mathrm{fold}\ f\ u\ l)\ .$$

3. Write terms for the concatenation and mirror functions.
4. Suggest an encoding for binary trees.