Proofs and Programs

TD 1 - Pure lambda-calculus

Philippe Audebaud, Aurore Alcolei

2 February 2018

HW- Short homeworks (labelled **HW**) are due at latest for the next Tuesday lecture, on a weekly basis.

Notations- As far as definitions or notations are concerned, always refer to *lecture notes*:

https://perso.ens-lyon.fr/philippe.audebaud/PnP/

Short reminder: Assume $\mathcal{X}$ a countable set of *variables*, λ-terms are generated by the grammar:

$$a, b, \ldots \in \Lambda \quad ::= \quad x \in \mathcal{X} \mid \lambda x.a \mid a\, b$$

λ-terms will always be considered up to α-*equivalence*, meaning for example that $\lambda x.x$ and $\lambda y.y$ are indistinguishable. Here are some common *combinators* (closed normal λ-terms):

$$\mathsf{I} \equiv \lambda x.x \quad \mathsf{T} \equiv \lambda x.\lambda y.x \quad \mathsf{F} \equiv \lambda x.\lambda y.y$$
$$\mathbf{\Delta} \equiv \lambda x.xx \quad \mathbf{\Omega} \equiv \mathbf{\Delta}\,\mathbf{\Delta} \quad \mathbf{\Upsilon} \equiv \lambda f.\,(\lambda x.f(xx))\,(\lambda x.f(xx))$$

In the following, $\twoheadrightarrow$ denotes the transitive closure of the β-reduction $\rightarrow_\beta$, and $=_\beta$ is the equivalence relation generated by β-reduction.

Exercice 1. (Warmup!)

a) Reduce the following terms to normal form:

$$\mathsf{I}\,\mathsf{I} \qquad \mathsf{T}\,\mathsf{I}$$
$$(\lambda f.\lambda g.f)\,g \qquad (\lambda x.\lambda y.xy)(\lambda x.x\,(\lambda y.y))(\lambda x.xx)$$

b) Decide whether the following β-equivalences hold:

$$\mathsf{I} =_\beta \mathsf{I}\,\mathsf{I} \qquad \mathbf{\Delta}\,\mathsf{I}\,\mathsf{I} =_\beta \mathsf{F}\,\mathsf{T}\,\mathsf{I}$$
$$x\,(\mathsf{I}\,\mathsf{I}) =_\beta x\,\mathsf{I} \qquad (\lambda b.\lambda x.\lambda y.byx)\,\mathsf{F} =_\beta (\lambda b.\lambda x.\lambda y.b(byx)(bxy))\,\mathsf{T}$$

Exercice 2 (Turing completeness)**.** The pure λ-calculus is Turing-complete as a programming langage! To prove this statement, it is sufficient to show that the following *features* can be encoded as λ-terms:

a) booleans and conditionals (exercise 3),

b) pairs and projections (exercise 4),

c) integers together with basic operations and recursion (exercices 5 and 6).

The key idea is to mimic the *operational behaviour* which is expected from each of these features... With these constructions, it is then easy to encode turing machines inside the λ-calculus. This is left as an exercise, or search references to Pablo Rauzy's *Le λ-calcul comme modèle de calculabilité*.

Exercice 3 (Booleans and conditionals)**.** Informally, the set of Booleans is the finite set $\{\mathbf{true}, \mathbf{false}\}$. Operationnally, their representative λ-terms (T for **true** and F for **false** – see above) behave as *selectors*.

a) If you are familiar with ML-like language, find a common type for both combinators $\mathsf{T}$and F;

b) Let $b, t, e \in \Lambda$ arbitrary, and let us consider **if** $b\ t\ e$ with the (expected) behaviour:

$$\textsf{\textbf{if}}\ \textsf{\textbf{T}}\ \ t\ e \twoheadrightarrow t \quad \text{and} \quad \textsf{\textbf{if}}\ \textsf{\textbf{F}}\ \ t\ e \twoheadrightarrow e$$

Which ML type whould you expect for **if**? Find a representation of **if** as a combinator, and check the above specification.

c) Define the combinators **or**, **not** and **xor**.

Exercice 4 (Pairs and projections)**.** Given $a, b \in \Lambda$, it is easy to pack them; for instance by building the λ-term $\lambda x.x\ a\ b$. Let us explore that path for building pairs:

a) Assuming a is given some type A, and b is given some type B by ML, which type would be given for $\lambda x.xab$? Find the "*most general*" ML type that it is possible to assign to $\lambda a.\lambda b.\lambda x.xab$?

b) Deduce from the previous analysis the existence of combinators **pair** (constructor), π_1 (first projection), and π_2 (second projection), with the expected operational behaviour:

$$\pi_1\,(\textsf{\textbf{pair}}\ \ a\ b) \twoheadrightarrow a \qquad \pi_2\,(\textsf{\textbf{pair}}\ \ a\ b) \twoheadrightarrow b$$

c) Let $f \in \Lambda$. Prove the existence of a λ-term t (depending on f) such that

$$t\ \ (\textsf{\textbf{pair}}\ \ a\ b) \quad \twoheadrightarrow \quad \textsf{\textbf{pair}}\ (f\ \ a)\ a$$

In the following, Φ will stand for the combinator corresponding to the curryfied version of the above construction. Make explicit the construction of Φ, and assign a most general ML-type to it.

Exercice 5 (Church numerals)**.** The very first idea to represent natural numbers operationally is as an *iterator*, very much like inside a for-loop. Hence, a Church numeral needs an initial seed x, and a function f which is expected to be iterated. Given $n \in \mathbb{N}$, let us denoted informally $f^0 x \equiv x$ and $f^{n+1} x \equiv f(f^n x)$.

a) Define formally the combinators representing zero (denoted **Z**) and the successor function (denoted **S**).

b) Find a combinator t such that $t\ \ \textsf{\textbf{Z}} \twoheadrightarrow \textsf{\textbf{T}}$ and $t\ (\textsf{\textbf{S}}\ \ n) \twoheadrightarrow \textsf{\textbf{F}}$ (test to zero);

c) Define the iterator **iter** as a combinator with the following operational behaviour:

$$\textsf{\textbf{iter}}\ \ a\,b\,\textsf{\textbf{Z}} =_\beta a \qquad \textsf{\textbf{iter}}\ \ a\,b\,(\textsf{\textbf{S}}\,n) =_\beta b\,(\textsf{\textbf{iter}}\ \ a\,b\,n)$$

And verify that if $\bar{n}$ is the Church representative for $n \in \mathbb{N}$, then $\textsf{\textbf{iter}}\ \ a\,b\,\bar{n} =_\beta b^n a$.

d) Explain the choice for the β-equivalence in place of the β-reduction.

e) **HW** Define the addition **add**. (The multiplication **mult** is as simple as it is tricky: any idea?)

f) **HW** The predecessor **pred** can be defined using the **iter** combinator. Informally, the idea is as follow:

  (a) the intial seed is the pair **pair Z Z**;
  (b) the iterated function is the λ-term Φ introduced is the exercise 4;
  (c) eventually, there remains to pick a projection...

Provide the complete definition of the combinator **pred**.

Exercice 6 (Recursion)**.** Recursion means being allowed to perform *unbounded iteration*. Coding the Russel paradox inside λ-calculus already provides the core idea:

a) **HW** Given $m \in \Lambda$, check that $\boldsymbol{\Upsilon}\,m =_\beta m\,(\boldsymbol{\Upsilon}\,m)$. Is it true that $\boldsymbol{\Upsilon}\,m \twoheadrightarrow m\,(\boldsymbol{\Upsilon}\,m)$?

b) **HW** Propose a closed λ-term **fact** such that, for all $n \in \mathbb{N}$, $\textsf{\textbf{fact}}\,\bar{n} =_\beta \overline{n!}$, and prove that $\textsf{\textbf{fact}}\bar{2} =_\beta \bar{2}$.

c) **HW** Let $\theta \equiv \lambda x.\lambda y.y\ \ (x\ x\ y)$. Prove that $\boldsymbol{\Theta} \equiv \theta\,\theta$ satisfies : for all e, $\boldsymbol{\Theta}\ e \twoheadrightarrow_\beta e\ \ (\boldsymbol{\Theta}\ e)$.