TD 1 : λ-calcul et codage des types de données

lionel.rieg@ens-lyon.fr

Exercice 1 *λ-calcul*

Le λ-calcul est défini par la syntaxe

$$M, N \;:=\; x \quad|\quad \lambda x.\, M \quad|\quad M\, N \qquad\qquad \text{où } x \text{ est une variable}$$

et la règle de réécriture (la β-réduction)

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

La clôture symétrique réflexive transitive de $\to$ est notée $=_\beta$. Voici quelques termes classiques :

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

1. Réduire les λ-termes $\Delta\, I\, I$ et Ω.
2. Donner les graphes de réduction des λ-termes $S\ K\, K$, $\Delta\,(I\, I)$ et $K\, I\, \Omega$.

Exercice 2 *Couples et types somme*

Définissez des λ-termes pour

$\langle \_, \_ \rangle$	le constructeur de couples	ι_1	la première injection
π_1	la première projection	ι_2	la seconde injection
π_2	la seconde projection	case	le filtrage

tels que

$$\pi_1\, \langle x,\, y\rangle =_\beta x \quad \text{et} \quad \pi_2\, \langle x,\, y\rangle =_\beta y \qquad\qquad \text{case}\,(\iota_1\, x)\, f\, g =_\beta f\, x \quad \text{et} \quad \text{case}\,(\iota_2\, x)\, f\, g =_\beta g\, x\;.$$

Exercice 3 *λ-calcul (bis)*

1. Caractériser les λ-termes en forme β-normale.
2. Restreindre la β-réduction pour implémenter l'appel par nom et l'appel par valeur. Trouver un λ-terme qui distingue ces deux stratégies de réduction.

Exercice 4 *Entiers de Church*

La représentation de Church d'un entier n est le λ-terme $\overline{n} := \lambda f x. f^n x$, c'est à dire n itérations de la fonction f en x.

1. Écrire $\overline{0}$ et $\overline{3}$.
2. Écrire une fonction successeur : $\mathrm{S}\, \overline{n} =_\beta \overline{n+1}$.
3. Écrire un itérateur, c'est-à-dire c'est-à-dire un terme Iter tel que pour tous termes M, N, on ait

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

4. Écrire des λ-termes représentant l'addition et la multiplication.
5. Quelle fonction représente le terme $\overline{n}\,\overline{m}$?

On représente les booléens par $\mathrm{T} := \lambda xy.x$ et $\mathrm{F} := \lambda xy.y$.

6. Donner une représentation de `if then else`.
7. Comment représenter les couples ?
8. Proposer un λ-terme représentant le prédécesseur.

Exercice 5 *Entiers de Barendregt*

Les entiers de Barendregt $\ulcorner n \urcorner$ sont définis par

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

1. Proposer une implémentation du successeur, du prédécesseur et de la conditionnelle.
2. Proposer une implémentation de l'addition.

Exercice 6 *Listes et arbres*

On s'intéresse ici à l'encodage des listes en λ-calcul. On va écrire les listes sous la forme $\lambda c.\lambda n.M[c, n]$. Moralement, une liste est une fonction qui attend un constructeur, une liste vide et qui les manipule. Par exemple, la liste `["Salade"; "Tomate"; "Oignon"]` sera représentée par

$$\lambda c.\lambda n.(c\ \texttt{"Salade"}\ (c\ \texttt{"Tomate"}\ (c\ \texttt{"Oignon"}\ n)))\ .$$

1. Écrire les opérateurs nil et cons.
2. Écrire un *itérateur* ListIt tel que

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

3. Écrire la concaténation et le miroir.
4. Proposer un encodage pour les arbres binaires.

Exercice 7 *Combinateurs de point fixe*

Un λ-terme M est un point fixe d'un λ-terme F si $M =_\beta FM$.
Un terme C est un combinateur de point fixe si pour tout terme F, CF est un point fixe de F.
On définit les termes suivants :

$$p := \lambda fx.f(xx) \qquad Y := \lambda f.pf(pf) \qquad q := \lambda xy.y(xxy) \qquad \Theta := qq$$

1. Montrer que Y et Θ sont des combinateurs de point fixe. Le combinateur Y est dû à Haskell Curry ; Θ est dû à Alan Turing. Ce dernier est en fait légèrement plus fort que Y.
2. Comparer Y et Θ vis-à-vis de la β-réduction.
3. Montrer qu'il existe un λ-terme G tel que $YG \to^* \Theta$.

Nous allons maintenant voir quelques propriétés communes aux combinateurs de point fixe.

4. Soit C un combinateur de point fixe, montrer qu'il existe x et P tels que $C =_\beta \lambda x.P$.
5. Montrer que $C =_\beta \lambda x.x(Cx)$.
6. En déduire une caractérisation des combinateurs de point fixe en fonction de G.