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

jean-marie.madiot@ens-lyon.org

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

Exercice 3 *Entiers et booléens de Church*

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

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

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

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

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 4 *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 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* fold tel que

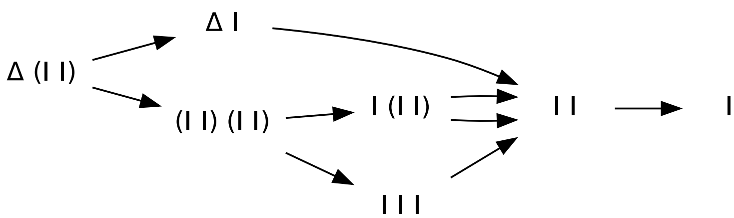
$$\mathrm{fold} \;\; f \;\; u \;\; \mathrm{nil} =_\beta u \qquad \text{et} \qquad \mathrm{fold} \;\; f \;\; u \; (\mathrm{cons} \;\; a \;\; l) =_\beta f \;\; a \; (\mathrm{fold} \;\; f \;\; u \;\; l) \; .$$

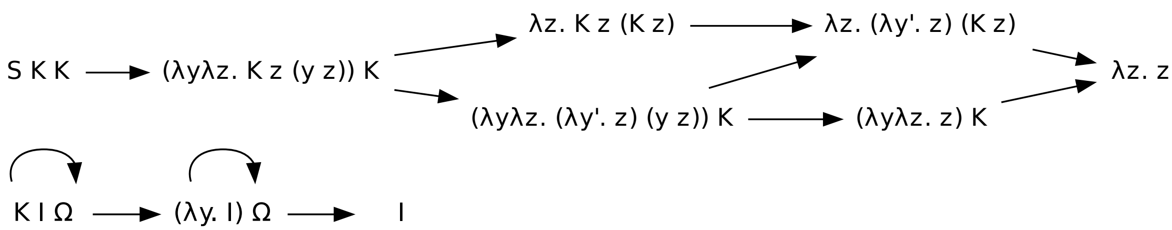
3. Écrire la concaténation et le miroir.
4. Proposer un encodage pour les arbres binaires.

Solutions

▶ Exercice 1

1. $\underline{\Delta\, I}\, I \to \underline{I\, I}\, I \to \underline{I\, I} \to I$
   $\Omega \to \Omega$



2. 

▶ Exercice 2

$$\langle \_, \_ \rangle := \lambda ab.\lambda f.fab \qquad \iota_1 := \lambda x.\lambda fg.fx$$
$$\pi_1 := \lambda p.p\,\mathrm{T} \qquad \iota_2 := \lambda y.\lambda fg.gy$$
$$\pi_2 := \lambda p.p\,\mathrm{F} \qquad \mathrm{case} := I$$

▶ Exercice 3

1. $\bar{0} := \lambda fx.x$ et $\bar{3} := \lambda fx.f(f(fx))$
2. $\mathrm{S} := \lambda nfx.\bar{n}f(fx)$ ou $\mathrm{S} := \lambda nfx.f(\bar{n}fx)$
3. $\mathrm{Iter} := \lambda MNn.\ n\,N\,M$
4. $$\begin{aligned} n \mathbin{\bar{+}} m &:= \lambda fx.\ \bar{n}f\,(\bar{m}\,f\,x) \\ n \mathbin{\bar{*}} m &:= \lambda fx.\ \bar{n}\,(\bar{m}\,f)\,x \\ n^{\bar{m}} &:= \bar{m}\,\bar{n} \end{aligned} \qquad \text{ou plus simplement} \qquad \begin{aligned} n \mathbin{\bar{+}} m &:= \mathrm{Iter}\,\bar{n}\,S\,\bar{m} \\ n \mathbin{\bar{*}} m &:= \mathrm{Iter}\,\bar{n}\,plus\,\bar{m} \\ n^{\bar{m}} &:= \mathrm{Iter}\,\bar{m}\,mult\,\bar{n} \end{aligned}$$
5. $\texttt{if}\ b\ \texttt{then}\ M\ \texttt{else}\ N = b\ M\ N$
6. cf exercice 4 : $\langle a,\, b\rangle := \lambda f.f\,a\,b$
7. On utilise le codage des couples. Si on itère $n+1$ fois la fonction $(x,y) \mapsto (x+1,x)$, représentée par $\lambda p.\lambda k.k(S(p\,\mathrm{T}))(p\,\mathrm{T})$ à partir de (0,0), alors on obtient (n+1,n) D'où pred $\bar{n} := \bar{n}\ (\lambda p.\lambda k.\ k(S(p\,\mathrm{T}))\,(p\,\mathrm{T}))(\lambda k.k\,\bar{0}\,\bar{0})$ F

▶ Exercice 4

1. $\overrightarrow{\lambda x}.\ x\,\overrightarrow{N}$ i.e. une suite d'abstraction, puis une variable appliquée à des termes β-réduits.

2. Il suffit de limiter le passage au contexte de la β-réduction. Pour l'appel par valeurs, définissons donc les valeurs :

$$V := \quad x \quad | \quad \lambda x.\, M \ .$$

Puis on définit la sémantique opérationnelle du λ-calcul en limitant le passage au contexte :

$$\textbf{call by value} \qquad \frac{}{(\lambda x.\, M)\, V \to M[V/x]} \qquad \frac{M \to M'}{M\, N \to M'\, N} \qquad \frac{N \to N'}{V\, N \to V\, N'}$$

$$\textbf{call by name} \qquad \frac{}{(\lambda x.\, M)\, N \to M[N/x]} \qquad \frac{M \to M'}{M\, N \to M'\, N}$$

▶ Exercice 5

1. $$S := \lambda nk.k\,\mathrm{F}\,n = \lambda n.\,\langle \mathrm{F},\, n\rangle \qquad P := \lambda n.n\,\mathrm{F} \qquad Z := \lambda n.n\,\mathrm{T}$$

2. $$Add := Y(\lambda axy.Z\ x\ y\ (S\ (a\ (P\ x)\ y)))$$

▶ Exercice 6

1. $\mathrm{nil} := \lambda cn.\, n$ et $\mathrm{cons}\, x\, l := \lambda cn.\, c\, x\, (l\, c\, n)$
2. $\mathrm{fold}\, f\, u\, l := l\ f\ u$
3. $l_1 @\, l_2 := \lambda cn.\, l_1\, c\, (l_2\, c\, n)$ et $\mathrm{rev}\, l := l\, (\lambda x l'.\, l' @\, (\mathrm{cons}\, x\, \mathrm{nil}))\, \mathrm{nil}$
4. $\mathrm{leaf} := \lambda nl.\, l$ et $\mathrm{node}\, a\, t_1\, t_2 := \lambda nl.n\ a\, t_1\, t_2$