TD3 10 mars 2015

λ-calcul et logique informatique

baelde@lsv.ens-cachan.fr

Exercice 1 — Stratégie interne faible
La stratégie interne faible, aussi appelée *appel par valeur*, est aussi proche que possible des langages de programmation traditionnels. Elle ne réduit jamais sous les abstractions, et ne réduit un β-redex que lorsque son argument est déja complètement réduit — on dit alors que c'est une *valeur*.

On définit l'ensemble des valeurs V par la grammaire suivante, où $\mathcal{V}$ est l'ensemble des variables et Λ l'ensemble de tous les termes :

$$V := \mathcal{V}V \dots V \mid \lambda x.\Lambda$$

On définit formellement notre stratégie comme suit :

$$\frac{v \in V}{(\lambda x.u)v \rhd u[x := v]} \qquad \frac{u \rhd u'}{uv \rhd u'v} \qquad \frac{u \in V \quad v \rhd v'}{uv \rhd uv'}$$

1. Utiliser le combinateur Y pour définir la fonction factorielle, de sorte que **fact** $\overline{n} \to^*_\beta \overline{n!}$. On rappelle la définition :

$$Y = \lambda f.\ (\lambda x.\ f\ (x\ x))\ (\lambda x.\ f\ (x\ x))$$

2. Quelles sont les réductions de **fact** (sans argument) par $\rhd^*$? Cela pose-t-il problème ? Si oui, proposer une solution.
3. Quelles sont les réductions de **fact** $\overline{0}$ par $\rhd^*$? Cela pose-t-il problème ? Si oui, proposer une solution.

Exercice 2 — Combinateurs
On définit le calcul de combinateurs **SK** : les termes sont construits suivant la grammaire

$$M := \mathcal{V} \mid (M\ M) \mid \mathbf{S} \mid \mathbf{K}$$

et la réduction est plus petite congruence contenant

$$\mathbf{K}\ M\ N \to M \qquad \mathbf{S}\ M\ N\ P \to (M\ P)\ (N\ P)$$

1. On pose $\mathbf{I} := \mathbf{S}\ \mathbf{K}\ \mathbf{K}$. Réduire $\mathbf{I}\ M$ pour un terme M quelconque.
2. Construire une traduction de $\mathcal{C}$ dans Λ tel que $M \to N$ implique $[M] \to_\beta [N]$ pour tous $M, N \in \mathcal{C}$.

3. Définir une construction λ^*, prenant une variable et un terme de $\mathcal{C}$ et renvoyant un nouveau terme de $\mathcal{C}$ de sorte que $\lambda^*(x, M)N \to^* M[x := N]$ pour tout N. On procèdera par induction sur M, en commençant par les variables, et en utilisant le "distributeur" **S** pour l'application.
4. A-t-on $\mathbf{K} \leftrightarrow^* \lambda^*(x, \lambda^*(y, x))$?
5. Définir une traduction de Λ dans $\mathcal{C}$ telle que $u \to_\beta v$ implique $[u] \to^* [v]$.

Exercice 3 — Encodages (un dernier pour la route)
On propose le codage suivant pour les listes :

$$[e_1; \ldots; e_n] = \lambda f \lambda x.\ f\ e_1 (f\ e_2 \ldots (f\ e_n e)$$

1. Donner un terme C qui encode l'ajout d'un élément à une liste, tel que $C\ e_0\ [e_1; \ldots; e_n] = [e_0; e_1; \ldots; e_n]$.
2. Coder la concaténation, ...
3. le renversement, ...
4. et la fonction qui à une liste non vide associe sa queue.

En bonus, on pourra chercher la recette commune derrière les encodages des booléens, entiers naturels et listes, en passant par leur écriture comme des types sommes en Caml.