

## λ-calcul et logique informatique

baelde@lsv.ens-cachan.fr

### Exercice 1 — Encodages (suite et fin)

On propose le codage suivant pour les listes :

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

1. Donner un terme  $C$  qui encode l'ajout d'un élément à une liste, tel que  $C e_0 [e_1; \dots; e_n] = [e_0; e_1; \dots; e_n]$ .
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.

### Exercice 2 — 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  $\beta$ -redex que lorsque son argument est déjà 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  $\Lambda$  l'ensemble de tous les termes :

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

On définit formellement notre stratégie comme suit :

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

1. Utiliser le combinateur  $Y$  pour définir la fonction factorielle, de sorte que **fact**  $\bar{n} \rightarrow_{\beta}^* \bar{n}!$ . 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  $\triangleright^*$  ? Cela pose-t-il problème ? Si oui, proposer une solution.
3. Quelles sont les réductions de **fact**  $\bar{0}$  par  $\triangleright^*$  ? Cela pose-t-il problème ? Si oui, proposer une solution.

**Exercice 3 — Chemins sur  $\Lambda^*$**

On définit l'ensemble des chemins d'un terme  $u \in \Lambda$ , noté  $\mathcal{C}(u)$ , comme l'ensemble des objets  $p$  tels que  $p \prec u$  est dérivable dans le système suivant :

$$\frac{}{x \prec x} \quad \frac{P \prec M}{\mathbf{L} P \prec M N} \quad \frac{P \prec N}{\mathbf{R} P \prec M N} \quad \frac{P \prec M}{\lambda x.P \prec \lambda x.M}$$

On travaille modulo  $\alpha$ , en renommant implicitement les variables liées pour pouvoir appliquer la règle de passage sous l'abstraction. Par conséquent, deux termes  $\alpha$ -équivalents ont les mêmes chemins, et l'ensemble des chemins d'un terme est clos par  $\alpha$ -équivalence.

1. Donner les chemins des termes  $\lambda x. x y$ ,  $\lambda x. x x$ , et  $\lambda x. (\lambda y. y) x$ .
2. Que peut-on dire de deux termes qui ont les mêmes chemins ?
3. On étend notre définition de chemins à  $\Lambda^*$  en rajoutant la règle suivante :

$$\frac{P \prec M \quad Q \prec N}{P[x := Q] \prec (\lambda^*x. M) N}$$

Montrer que  $M \rightarrow N$  implique  $\mathcal{C}(M) = \mathcal{C}(N)$ , où  $\rightarrow$  est la réduction de  $\Lambda^*$  qui ne touche qu'aux rédexes annotés d'une étoile.

4. Montrer la réciproque :  $\mathcal{C}(M) = \mathcal{C}(N)$  implique  $M \leftrightarrow^* N$ , c'est à dire la convertibilité dans  $\Lambda^*$ .

**Exercice 4 — 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

$$\mathbf{K} M N \rightarrow M \quad \mathbf{S} M N P \rightarrow (M P) (N P)$$

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