TD 7: Codage des types de données dans système F

lionel.rieg@ens-lyon.fr

Définition ***Système F à la Church***

Types $A, B \quad := \quad \alpha \quad | \quad A \to B \quad | \quad \forall \alpha. A$

Termes $M, N \quad := \quad x \quad | \quad \lambda x : A. M \quad | \quad M\ N \quad | \quad \Lambda \alpha. M \quad | \quad M\ A$

Exercice 1 *Algèbre libre engendrée par une signature*

Vous avez vu en cours comment coder certaines des structures de données que nous avons vu au premier TD pour le λ-calcul non typé : les entiers de Church, les couples et les types sommes. Nous allons voir à présent le schéma général et les deux exemples non encore traités : les listes et les arbres.

On part d'un inductif I à m paramètres p_i, k constructeurs c_j qui chacun ont n_j arguments x_l, c'est-à-dire le type Coq suivant :

Inductive I $p_1 \dots p_m$ **:=**
| $c_1\ x_1 \dots x_{n_1}$
⋮
| $c_k\ x_1 \dots x_{n_k}$

On traduit chaque constructeur c_j en le terme :

$$\underbrace{\Lambda p_1 \dots \Lambda p_m}_{\text{paramètres}} \underbrace{\lambda x_1 \dots \lambda x_{n_j}}_{\text{arguments}} \Lambda \alpha\ \underbrace{\lambda e_1 \dots \lambda e_k}_{\text{éliminateurs}} .\ e_j\ x_1 \dots x_{n_j}$$

1. Quelle est la forme du codage d'un terme de type I $p_1 \dots p_m$?
2. Avec ce codage, comment implémenter le filtrage ?
3. Donner le codage des listes et des arbres binaires (y compris leurs types).

Exercice 2 *Codage du récurseur*

1. Rappeler le codage de la paire et des entiers de Church.
2. Donner sans justification les termes clos en forme normale de type Nat.
3. Donner un terme clos Iter, de type $\forall \alpha. \alpha \to (\alpha \to \alpha) \to \text{Nat} \to \alpha$ et tel que pour tout type T et tous termes $u : T$, $v : T \to T$ et n : Nat, on ait

$$\text{Iter}\ T\ u\ v\ 0 =_\beta u \qquad \text{et} \qquad \text{Iter}\ T\ u\ v\ (S\ n) =_\beta v\ (\text{Iter}\ T\ u\ v\ n)\ .$$

On va maintenant voir comment coder un récurseur, c'est-à-dire un terme Rec de type $\forall \alpha. \alpha \to (\text{Nat} \to \alpha \to \alpha) \to \text{Nat} \to \alpha$ tel que pour tout type T, pour tous termes $u : T$, $v : \text{Nat} \to T \to T$ et pour tout terme clos n : Nat, on ait

$$\text{Rec}\ T\ u\ v\ 0 =_\beta u \qquad \text{et} \qquad \text{Rec}\ T\ u\ v\ (S\ n) =_\beta v\ n\ (\text{Rec}\ T\ u\ v\ n) \tag{1}$$

La difficulté vient de la seconde équation (celle qui diffère de l'itérateur Iter) :

$$\text{Rec}\ T\ u\ v\ (S\ n) =_\beta v\ n\ (\text{Rec}\ T\ u\ v\ n) \tag{2}$$

4. Montrer que tout terme clos n de type Nat est β-équivalent à $\mathrm{Iter}\, S\ 0\, n$.

La première idée est alors de réécrire l'équation (2) comme :

$$\mathrm{Rec}\, T\, u\, v\, (S\ n) =_B v'\, \langle \mathrm{Iter}\, T\, S\ 0\, n,\ \mathrm{Rec}\, T\, u\, v\, n \rangle$$

où $v' := \lambda p.\, v\, (\pi_1\ p)\, (\pi_2\ p)$. La seconde idée est de remplacer la paire

$$\langle \mathrm{Iter}\, T\, S\ 0\, n,\ \mathrm{Rec}\, T\, u\, v\, n \rangle$$

par un terme de la forme

$$\mathrm{Iter}\, (\mathrm{Nat} \times T)\ \langle 0,\ u \rangle\ (P\, S\ v)\, n$$

où P est un terme clos choisi afin d'obtenir, pour tout terme clos n : Nat,

$$\mathrm{Iter}\, (\mathrm{Nat} \times T)\ \langle 0,\ u \rangle\ (P\, S\ v)\, n =_\beta \langle n,\ \mathrm{Rec}\, T\, u\, v\, n \rangle \tag{3}$$

5. En supposant donné un terme clos Rec vérifiant (1), proposer un terme clos P tel que pour tout terme clos n : Nat, l'équation (3) est vérifiée.
6. Proposer un terme clos Rec implémentant le récurseur au sens de (1).
7. Proposer un codage du prédécesseur sur les entiers de Church.

Solutions

▸ Exercice 1

1. La traduction d'un terme de type I $p_1 \dots p_m$ est de la forme $\Lambda\alpha.\, \lambda e_1 \dots \lambda e_k.\, e_j\, x_1 \dots x_{n_j}$
2. La structure du filtrage `case` est la suivante : `case` $M\ e_1 \dots e_k$ où M est le terme à filtrer et les e_i sont les branches du filtrage, auquel il faut ajouter une annotation du type de retour α. Avec le codage, on constate que `case` := $\Lambda\alpha.\lambda x.x\,\alpha$, *i.e.* aux applications de type près, c'est l'identité.
3. Simple application de ce qui précède :
   - les listes :
     $$\begin{array}{lcl} \text{list}\,\alpha & := & \forall\beta.\, \beta \to (\alpha \to \beta \to \beta) \to \beta \\ \text{nil} & := & \Lambda\beta.\, \lambda e_{\text{nil}} : \beta.\, \lambda e_{\text{cons}} : \alpha \to \beta \to \beta.\, e_{\text{nil}} \\ \text{cons} & := & \Lambda\beta.\, \lambda a : \alpha.\, \lambda l : \text{list}\,\alpha.\, \lambda e_{\text{nil}} : \beta.\, \lambda e_{\text{cons}} : \alpha \to \beta \to \beta.\, e_{\text{cons}}\ a\,(l\, e_{\text{nil}}\, e_{\text{cons}}) \end{array}$$
   - les arbres :
     $$\begin{array}{lcl} \text{tree}\,\alpha\,\beta & := & \forall\gamma.\, (\alpha \to \gamma) \to (\beta \to \gamma \to \gamma \to \gamma) \to \gamma \\ \text{leaf} & := & \Lambda\gamma.\, \lambda a : \alpha.\, \lambda e_L : \alpha \to \beta.\, \lambda e_N : \beta \to \gamma \to \gamma \to \gamma.\, e_L\ a \\ \text{node} & := & \Lambda\alpha.\, \Lambda\beta.\, \Lambda\gamma.\, \lambda b : \beta.\, \lambda t_1 : \text{tree}\,\alpha\,\beta.\, \lambda t_2 : \text{tree}\,\alpha\,\beta.\, \lambda e_L : \alpha \to \beta.\, \lambda e_N : \beta \to \gamma \to \gamma \to \gamma. \\ & & e_N\ b\,(t_1\, e_L\, e_N)\,(t_2\, e_L\, e_N) \end{array}$$

▸ Exercice 2

1. Voici les codage dans système F :
   - la paire
     $$\begin{array}{lcl} \alpha * \beta & := & \forall\gamma.\, (\alpha \to \beta \to \gamma) \to \gamma \\ \langle \_, \_ \rangle & := & \Lambda\alpha.\, \Lambda\beta.\, \Lambda\gamma.\, \lambda a : \alpha.\, \lambda b : \beta.\, \lambda f : \alpha \to \beta \to \gamma.\, f\, a\, b \\ \pi_1 & := & \Lambda\alpha.\, \Lambda\beta.\, \lambda p : \alpha * b.\, p\,\alpha\,(\lambda x : \alpha.\, \lambda b : \beta.\, a) \\ \pi_2 & := & \Lambda\alpha.\, \Lambda\beta.\, \lambda p : \alpha * b.\, p\,\beta\,(\lambda x : \alpha.\, \lambda b : \beta.\, b) \end{array}$$
   - les entiers de Church :
     $$\begin{array}{lcl} \text{Nat} & := & \forall\alpha.\, (\alpha \to \alpha) \to \alpha \to \alpha \\ 0 & := & \Lambda\alpha.\, \lambda f : \alpha \to \alpha.\, \lambda x : \alpha.\, x \\ S & := & \Lambda\alpha.\, \lambda n : \text{Nat}\,.\lambda f : \alpha \to \alpha.\, \lambda x : \alpha.\, f\,(n\, f\, x) \end{array}$$
2. Ce sont les termes de la forme $\Lambda\alpha.\lambda x : \alpha.\lambda f : \alpha \to \alpha.f^n x$ qui sont β-équivalents à $S^n\, 0$.
3. $\text{Iter} = \Lambda\alpha.\, \lambda u : \alpha.\, \lambda v : \alpha \to \alpha.\, \lambda n : \text{Nat}\,.\, n\,\alpha\, f\, x$
4. Puisqu'on travaille à β-équivalence près, on peut supposer que n est en forme normale, et même qu'il s'écrit $n = S^m\, 0$ où m est un méta-entier car cette forme est β-équivalente à la forme normale. Par récurrence sur m, on montre alors que $n =_\beta \text{Iter Nat}\ 0\, S\ n$.
   - $m = 0$: on a $n = 0$ et
     $$\text{Iter Nat}\ 0\, S\ n = \text{Iter Nat}\ 0\, S\ 0 =_\beta 0 = n$$
   - $m = m' + 1$: on a $n = S\, n'$ donc $n' = S^{m'}\, 0$ et
     $$\text{Iter Nat}\ 0\, S\ n = \text{Iter Nat}\ 0\, S\ (S\ n') =_\beta S\,(\text{Iter Nat}\ 0\, S\ n') =_{HI} S\,(S^{m'}\, 0) = S^{m'+1}\, 0 = S^m\, 0 = n$$
   - $P = \lambda S\ : \text{Nat} \to \text{Nat}\,.\, \lambda v : \text{Nat} \to T \to T.\lambda c : \text{Nat} * T.\ \langle S\ (\pi_1\, c),\ v\,(\pi_1\, c)\,(\pi_2\, c)\rangle$
   - $\text{Rec} = \Lambda\alpha.\, \lambda u : \alpha.\, \lambda v : \text{Nat} \to \alpha \to \alpha.\, \lambda n : \text{Nat}\,.\ \text{Iter}\,(\text{Nat} \times \alpha)\ \langle 0,\, u\rangle\ (P\, S\ v)\, n$
   - On a alors $\text{pred}\, n = \text{Rec Nat}\ 0\, \pi_1$.