Modèles de Calcul
Contrôle Continu - λ-calcul
L3 Informatique
CORRIGÉ

25/02/2014

Durée : 1h30 sans document.

1 Questions de cours

Q.1 Donnez la définition et un exemple de terme divergent.

Réponse : Un terme est divergent s'il n'a pas de forme normale.

exemple : $(\lambda x.(x\ x)\ \lambda x.(x\ x))$

Q.2 Donnez la définition et un exemple de terme faiblement normalisant.

Réponse : Un terme est faiblement normalisant s'il existe une suite finie de β-réduction issue de ce terme qui aboutit sur une forme normale.

exemple : $(\lambda z.y\ (\lambda x.(x\ x)\ \lambda x.(x\ x)))$

Q.3 Donnez la définition et un exemple de terme fortement normalisant. **Réponse : Un terme est fortement normalisant si toute suite de β-réduction issue de ce terme est finie.**

exemple : $(\lambda x.x\ z)$

Q.4 Donnez une définition inductive de $=_\beta$.

Réponse : $=_\beta$ est inductivement défini par :

- **Base :**
  R1 Si $M \equiv N$ alors $M =_\beta N$.
  R2 Si $M \to_\beta N$ alors $M =_\beta N$.
- **Induction :**
  R3 Si $N =_\beta M$ alors $M =_\beta N$.
  R4 Si $M =_\beta P$ et $P =_\beta N$ alors $M =_\beta N$.

Q.5 Démontrez par induction structurelle que si $M =_\beta M'$ alors il existe un λ-terme L tel que $M \to_\beta^* L$ et $M' \to_\beta^* L$.

Réponse : la démonstration par induction structurelle sur la définition de $=_\beta$.

Si $M =_\beta N$ alors on a les cas suivants :

- **Base :**

R1 Soit $M \equiv N$ alors $L \equiv M$ convient.

R2 Soit $M \to_\beta N$ alors $L \equiv N$ convient.

– **Induction :**

R3 Soit $N =_\beta M$ alors on peut appliquer l'hypothèse d'induction ce qui permet de conclure.

R4 Soit il existe P tel que $M =_\beta P$ et $P =_\beta N$ alors on peut appliquer deux fois l'hypothèse d'induction, donc il existe L_1, L_2 tels que $M, P \to_\beta^* L_1$ et $P, N \to_\beta^* L_2$, on peut appliquer le lemme de Church-Rosser à P, L_1, L_2 pour en conclure qu'il existe L tel que $L_1, L_2 \to_\beta^* L$. Or $M \to_\beta^* L_1 \to_\beta^* L$ et $N \to_\beta^* L_1 \to_\beta^* L$, ce qui conclut la démonstration.

2 Modélisation en λ-calcul

Dans cet exercice on se propose d'étudier la modélisation de la structure de donnée arborescente. On veut pouvoir manipuler des arbres binaires dont le contenu des feuilles sont des λ-termes.
Pour cela on considère le codage des constructeurs, $\mathcal{F}$euille et $\mathcal{N}$oeud suivants :

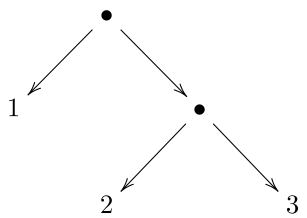
$$\begin{array}{lcl} \mathcal{F} & \equiv & \lambda ngy.(y\ n) \\ \mathcal{N} & \equiv & \lambda abgy.(g\ (a\ g\ y)\ (b\ g\ y)) \end{array}$$

On rappelle le codage des entiers

$$\lceil 0 \rceil \equiv \lambda fx.x$$

$$\lceil n \rceil \equiv \lambda fx.(\overbrace{f\ \ldots (f}^{n}\ x))$$

Ainsi l'arbre suivant :



est représenté par le λ-terme $A \equiv (\mathcal{N}\ (\mathcal{F}\ \lceil 1 \rceil)\ (\mathcal{N}\ (\mathcal{F}\ \lceil 2 \rceil)\ (\mathcal{F}\ \lceil 3 \rceil)))$

Q.6 Donnez la forme normale de A.

Réponse :

$$\begin{array}{rcl} A & \equiv & (\mathcal{N}\ (\mathcal{F}\ \lceil 1 \rceil)\ (\lambda abgy.(g\ (a\ g\ y)\ (b\ g\ y))\ (\mathcal{F}\ \lceil 2 \rceil)\ (\mathcal{F}\ \lceil 3 \rceil))) \\ & \to_\beta & (\mathcal{N}\ (\mathcal{F}\ \lceil 1 \rceil)\ (\lambda bgy.(g\ ((\mathcal{F}\ \lceil 2 \rceil)\ g\ y)\ (b\ g\ y))\ (\mathcal{F}\ \lceil 3 \rceil))) \\ & \to_\beta & (\mathcal{N}\ (\mathcal{F}\ \lceil 1 \rceil)\ \lambda gy.(g\ ((\mathcal{F}\ \lceil 2 \rceil)\ g\ y)\ ((\mathcal{F}\ \lceil 3 \rceil)\ g\ y))) \\ & \equiv & (\lambda abg'y'.(g'\ (a\ g'\ y')\ (b\ g'\ y'))\ (\mathcal{F}\ \lceil 1 \rceil)\ \lambda gy.(g'\ ((\mathcal{F}\ \lceil 2 \rceil)\ g\ y)\ ((\mathcal{F}\ \lceil 3 \rceil)\ g\ y))) \\ & \to_\beta & (\lambda bg'y'.(g'\ ((\mathcal{F}\ \lceil 1 \rceil)\ g'\ y')\ (b\ g'\ y'))\ \lambda gy.(g'\ ((\mathcal{F}\ \lceil 2 \rceil)\ g\ y)\ ((\mathcal{F}\ \lceil 3 \rceil)\ g\ y))) \\ & \to_\beta & \lambda g'y'.(g'\ ((\mathcal{F}\ \lceil 1 \rceil)\ g'\ y')\ (\lambda gy.(g\ ((\mathcal{F}\ \lceil 2 \rceil)\ g\ y)\ ((\mathcal{F}\ \lceil 3 \rceil)\ g\ y))\ g'\ y')) \\ & \to_\beta & \lambda g'y'.(g'\ ((\mathcal{F}\ \lceil 1 \rceil)\ g'\ y')\ (\lambda y.(g'\ ((\mathcal{F}\ \lceil 2 \rceil)\ g'\ y)\ ((\mathcal{F}\ \lceil 3 \rceil)\ g'\ y))\ y')) \\ & \to_\beta & \lambda g'y'.(g'\ ((\mathcal{F}\ \lceil 1 \rceil)\ g'\ y')\ (g'\ ((\mathcal{F}\ \lceil 2 \rceil)\ g'\ y')\ ((\mathcal{F}\ \lceil 3 \rceil)\ g'\ y'))) \\ & \overset{\textbf{def}}{\equiv} & B \end{array}$$

Or

$$\begin{array}{lcl} ((\mathcal{F}\ \lceil n \rceil)\ x\ z) & \equiv & (\lambda ngy.(y\ n)\ \lceil n \rceil\ x\ z) \\ \to_\beta & (\lambda gy.(y\ \lceil n \rceil)\ x\ z) & \\ \to_\beta & (\lambda y.(y\ \lceil n \rceil)\ z) & \\ \to_\beta & (z\ \lceil n \rceil) & \end{array}$$

Donc on obtient la forme normale suivante:

$$A \to_\beta^* B \to_\beta^* \lambda g'y'.(g'\ (y'\ \mathcal{F}\ \lceil 1 \rceil)\ (g'\ (y'\ \lceil 2 \rceil)\ (y' \lceil 3 \rceil)))$$

Q.7 Donnez un terme qui appliqué à tout arbre calcule la somme des entiers contenus dans ses feuilles.

On pourra utiliser ADD pour l'addition des entiers. Indication : il n'est pas nécessaire d'utiliser le combinateur de point fixe ni d'une fonction qui teste si un arbre est une feuille ou non. La structure de l'arbre permet un traitement direct. **Réponse : Il suffit de remplacer les noeuds par l'addition et les feuilles par l'identité. La structure de l'arbre suffira pour mener à bien les calculs (pas besoin de récursivité). Le term SUM définit comme suit suffit donc :**

$$SUM \stackrel{\textbf{def}}{=} \lambda a.(a\ ADD\ \lambda x.x)$$

Q.8 Donnez le codage d'un terme LF tel qu'appliqué à un arbre ce dernier produise la liste des feuilles contenus dans l'arbre. Par exemple $(LF\ A) \to_\beta^* [\lceil 1 \rceil; \lceil 2 \rceil; \lceil 3 \rceil]$.

Réponse : C'est la même technique que pour la question précédente. Il suffit de remplacer ADD par la concaténation de listes et pour les feuilles il faut créer une liste d'un seul élément. Cela se fait par :

$$LF \stackrel{\textbf{def}}{=} \lambda a.(a\ CONCAT\ \lambda x.(\texttt{cons}\ x\ \texttt{nil}))$$

avec $CONCAT$ défini par point fixe par

$$CONCAT \stackrel{\textbf{def}}{=} (Y \lambda c, l_1, l_2.(\texttt{if}\ (NULL?\ l_1)\ l_2 (\texttt{cons}\ (\texttt{head}\ l_1)\ (c\ (\texttt{tail}\ l_1)\ l_2))))$$

3 Typage

On considère le λ-calcul simplement typé avec les entiers : $\Lambda_\mathcal{N} ::= \mathbf{S} \mid \mathbf{0} \mid \mathbf{x} \mid \mathbf{0} \mid \mathbf{S} \mid (\mathbf{\Lambda}_\mathcal{N}\ \mathbf{\Lambda}_\mathcal{N}) \mid \lambda \mathbf{x} : \tau.\mathbf{\Lambda}_\mathcal{N}$ Les types sont définis par $\tau ::= \mathcal{N} \mid \tau \to \tau$ où $\mathcal{N}$ est le seul type de base.

On type les constantes par $0 : \mathcal{N}$ et $S : \mathcal{N} \to \mathcal{N}$. L'entier n sera représenté par $(\overbrace{S(S \ldots (S}^{n} 0) \ldots))$.

Une fonction f de $\mathcal{N}^n$ dans $\mathcal{N}$ est dite β-exprimable s'il existe un terme t tel que pour tout n-uplet d'entiers $(a_1, \ldots, a_n)$ le terme $(t\ a_1 \ldots a_n)$ se β-réduise sur $f(a_1, \ldots, a_n)$.

Q.9 Montrez que les fonctions qui rendent une constante sont β-exprimables.

Réponse : Les fonctions $f(x_1, \ldots, x_n) = k$ sont β-exprimables par

$$\lambda x_1, \ldots, x_n : \mathcal{N}.(\overbrace{\mathbf{S}(\mathbf{S} \ldots (\mathbf{S}}^{k} 0) \ldots))$$

Q.10 Montrez que les fonctions qui ajoutent une constante à un de leurs arguments sont β-exprimables.

Réponse : Les fonctions $f(x_1, \ldots, x_n) = x_i + k$ sont β-exprimables par

$$\lambda x_1, \ldots, x_n : \mathcal{N}.(\overbrace{\mathbf{S}(\mathbf{S} \ldots (\mathbf{S}}^{k}\ x_i) \ldots))$$

Q.11 Montrer que réciproquement seules les fonctions constantes et celles qui ajoutent une constante à un de ses arguments sont β-exprimables.

Réponse : Soit f une fonction β-exprimable par le terme t, f est également β-exprimable par la forme normale de t (à cause des propriétés de préservation du typage par réduction et de normalisation forte dans le λ-calcul simplement typé).

On montre par récurrence sur la taille de t que f est une fonction constante ou une fonction qui ajoute une constante à l'un de ses arguments. Soit $t \equiv \lambda x_1, \ldots, x_n : \mathcal{N}.(x\ u_1\ \ldots\ u_k)$.

- **si $x \equiv x_i$ alors $k = 0$ et $t \equiv \lambda x_1, \ldots, x_n : \mathcal{N}.x_i$, cette fonction est la fonction qui ajoute 0 à son i^{eme} argument.**
- **Si $x \equiv \mathbf{0}$ alors $k = 0$ et $t \equiv \lambda x_1, \ldots, x_n : \mathcal{N}.\mathbf{0}$, cette fonction est la fonction constante égale à 0.**
- **Si $x \equiv \mathbf{S}$ alors $k = 0$ ou $k = 1$.**
  * **Si $k = 0$ alors $t \equiv \lambda x_1, \ldots, x_n : \mathcal{N}.(Su)$. Par hypothèse de récurrence le terme $t \equiv \lambda x_1, \ldots, x_n : \mathcal{N}.u$ exprime une fonction constante égale à r ou une fonction qui ajoute r à son i^{eme} argument. Dans le premier cas, la fonction f est égale à la fonction constante égale à $r + 1$, dans le second cas c'est la fonction qui ajoute $r + 1$ à son i^{eme} argument.**
  * **Si $k = 1$ alors $t \equiv \lambda x_1, \ldots, x_n : \mathcal{N}.\mathbf{S}$ et f est la fonction qui ajoute 1 à son $(n + 1)^{eme}$ argument.**