Université Paris 7 - MPRI M2 Année 2006-2007

TD de *Modèles des langages de programmation: domaines, catégories, jeux*
n° 2
(Correction)

Sur la λ-définissabilité

Un problème de définissabilité, relatif à un language L et à un modèle M de L, est la donné d'un élément $f \in M^{\sigma_1 \to \sigma_2 \to \ldots \to \sigma_m \to \sigma}$.

Une solution du problème est la donné:

- soit d'un terme (clos) $t : \sigma_1 \to \sigma_2 \to \ldots \to \sigma_m \to \sigma$ de L : tel que $[\![t]\!] = f$,
- soit d'une preuve qu'un tel terme n'existe pas.

Une preuve de non-définissabilité peut utiliser:

- la méthode des *relations logiques*: on définit une relation n-aire $R_\tau \subseteq (M^\tau)^n$ pour chaque τ type de base de L; pour les types d'ordre > 0, on définit $R^{\sigma \to \tau}$ par $(f_1,\ldots,f_n) \in R^{\sigma \to \tau}$ si pour tout $(x_1,\ldots,x_n) \in R^\sigma$, $(f_1 x_1,\ldots,f_n x_n) \in R^\tau$;
  il faut que la famille de relations $\{R^\sigma\}$ définie ainsi *fixe* les constantes de L (c'est à dir que, pour toute constante $c : \delta$ de L, le n-uplet $([\![c]\!]^M,\ldots,[\![c]\!]^M)$ est un élélment de R^δ).
  Puis on exhibe m n-uplet:
  $(x_1^1,\ldots,x_1^n) \in R^{\sigma_1}$
  $(x_2^1,\ldots,x_2^n) \in R^{\sigma_2}$
  ...
  $(x_m^1,\ldots,x_m^n) \in R^{\sigma_m}$
  tels que
  $((f x_1^1 x_2^1 .. x_m^1,)(f x_1^2 x_2^2 \ldots x_m^2),\ldots,(f x_1^n x_2^n .. x_m^n)) \notin R^\sigma$.
  Par le lemme fondamentale des relations logiques, ceci permet de conclure que f n'est pas définissable.
- la méthode de la *définissabilité relative*:
  soit $g \in M^\delta$ un élément du modèle, et
  $t : (\sigma_1 \to \sigma_2 \to \ldots \to \sigma_m \to \sigma) \to \delta$
  un terme tel que $[\![t]\!] f = g$.
  On dit dans ce cas que g est définissable relativement a f, ou simplement que g est f-définissable.
  Si g n'est pas définissable, et g est f-définissable, alors f n'est pas définissable (car si $[\![t']\!] = f$ alors $[\![tt']\!] = g$).

Exercice 1 Resoudre les problèmes de définissabilité suivants:

Langage	Modèle	Élément
PCF	C	$pand \in C^{bool \to bool \to bool}$
PCF	C	$pif \in C^{bool \to bool \to bool \to bool}$
PCF	C	$lor \in C^{bool \to bool \to bool}$
PCF	C	$ror \in C^{bool \to bool \to bool}$
PCF	C	$F \in C^{(bool \to bool \to bool) \to bool}$
$\lambda_{\to}$	F_2	$perm \in F_2^{o \to o}$
PCF$^-$	C	$if\_then\_else \in C^{bool \to bool \to bool \to bool}$

C est le modèle continu de PCF et F_2 est la structure applicative construite sur un seul type de base o telle que F_2^o a deux éléments, telle que $F_2^{\sigma \to \tau}$ est l'ensemble de toutes les fonction de F_2^σ vers F_2^τ.

Le langage $\lambda_{\to}$ est le λ-calcul simplement typé, sans constantes.

$$pand\ x\ y = \begin{cases} tt & \text{si } x = tt \text{ et } y = tt \\ ff & \text{si } x = ff \text{ ou } y = ff \\ \bot & \text{sinon} \end{cases}$$

$$pif\ x\ y\ z = \begin{cases} y & \text{si } x = tt \\ z & \text{si } x = ff \\ y & \text{si } x = \bot \text{ et } y = z \\ \bot & \text{sinon} \end{cases}$$

$$lor\ x\ y = \begin{cases} tt & \text{si } x = tt \\ \bot & \text{sinon} \end{cases} \qquad ror\ x\ y = \begin{cases} tt & \text{si } y = tt \\ \bot & \text{sinon} \end{cases}$$

$$F\ f = \begin{cases} tt & \text{si } f \geq lor \text{ ou } f \geq ror \\ \bot & \text{sinon} \end{cases}$$

$perm \in F_2^{2 \to 2}$ est la fonction qui echange les deux éléments de F_2^2.

PCF$^-$ est le langage obtenu en éliminant de PCF le type `nat` et la constante `if then else`.

Correction : 1.

$not = [\![\lambda x : bool.if\ x\ then\ false\ else\ true]\!]$

2. La non définissabilité de *pand* peut etre vue en utilisant la 2eme methode (definiss. rel.): on sait que *por* n'est pas PCF def. Or

$[\![\lambda f : bool \to bool \to bool \lambda x, y : bool.\mathtt{not}(f(\mathtt{not}x)(\mathtt{not}y))]\!] pand = por$

verif. table de vérité.

Si on veut une relation logique pour eliminer pand, on s''apercoit que la meme que pour *por* marche (en changeant ff et tt).

3.

Pour pif aussi, definssabilité relative de *por*:

$[\![\lambda f : bool \to bool \to\ bool \to bool \lambda x, y : bool.f\ x\ tt\ y]\!] pif = por$

vérif: table de vérité.

4.

$lor = [\![\lambda x, y : bool.if\ x\ then\ true\ else\ \Omega]\!]$

verif: table de verite.

5.

$ror : [\![\lambda x, y : bool. if\ y\ then\ true\ else\ \Omega]\!]$

6.

$F = [\![\lambda f : bool \to bool \to bool. f(ftt\Omega)(f\Omega tt)]\!]$

F envoie tout majorant de lor et ror sur tt. Par contre, il faut éliminer les résultats ff, on définit $\bar{t} = if\ tthen\ true\ else\ \Omega$ alors

$F = [\![\lambda f : bool \to bool \to bool. \overline{f(\overline{ftt\Omega)(f\Omega tt)}}]\!]$

7. Il faut une rel. log. Comme il n'y a aucune constante, toute rel binaire au type de base est ok, en particulier $R^o = \{(a,b)\}$ (a,b etant les 2 elts du type de base. on a $(echa, echb) = (b,a) \notin R$, donc ech n'est pas définissable.

8. Comme pour 7, la seule diff. est que la rel au type bool doit contenir les e n-uplet constants. On choisit $n = 2$, avec $R^{bool} = \{(tt,tt),(ff,ff),(\perp,\perp),(tt,ff)\}$ on a $(if\ tt\ ff\ tt, if\ ff\ ff\ tt) \notin R$.