Examen du cours PROG1

27 octobre 2021

Durée : 1 heure 30 minutes.
À réaliser sans document ni machine.

1 Ça compte ou pas ?

On considère le code OCaml suivant :

```
let get =
  let c = ref 0 in
  fun () -> incr c ; !c

let map1 l = List.map get l
let map2 l = List.map (fun _ -> get ()) l
let map3 l = let g = get () in List.map (fun _ -> g) l
```

On rappelle l'extrait pertinent de la librairie standard :

— `val map : ('a -> 'b)-> 'a list -> 'b list`
`List.map f [a1; ...; aN]` applique la fonction `f` à `a1`, ..., `aN` et renvoie la liste des résultats renvoyés par `f`, c'est à dire `[f a1; ...; f an]`.

Question 1 Quel est le type de `get` ? Que fait cette fonction ?

Question 2 Pour chacune des fonctions `map1`, `map2` et `map3`, donner le type de la fonction et décrire ce qu'elle fait.

2 λ-calcul

Dans cet exercice on considère le λ-calcul simplement typé. La syntaxe des termes et des types est donc la suivante, où x dénote une variable et B un type de base :

$$M ::= x \mid M_1\ M_2 \mid \lambda x.M \qquad T ::= B \mid T_1 \to T_2$$

On utilisera la β-réduction (sémantique à petits pas) notée $\to_\beta$ comme dans le cours.

Question 1 On pose $M \stackrel{\text{def}}{=} (\lambda x.\ (\lambda y.\ (x\ y))\ z)\ (\lambda x.\ y)$. Quelles sont les variables libres de M ? Donner toutes les séquences de β-réductions possibles à partir de M.

Question 2 Soit M le terme $\lambda f.\lambda g.\lambda x.(f\ (g\ (f\ x)))$. Donner un type T tel que $\emptyset \vdash M : T$. (La dérivation détaillée n'est pas demandée, mais il est recommandé de la faire au brouillon.)

Question 3 Montrer que $\lambda x.\lambda y.\lambda z.(z\ (x\ y)\ (y\ x))$ n'est pas typable.

Question 4 Quizz : indiquer pour chaque affirmation ci-dessous si elle est vraie ou non, en justifiant.

a. Si un terme clos M se β-réduit en un terme M' fortement normalisant alors M est aussi fortement normalisant. (Rappel : un terme M est fortement normalisant s'il n'existe pas de séquence infinie de réductions partant de M.)

b. Si un terme M clos et fortement normalisant se β-réduit en un terme M' alors M' est aussi fortement normalisant.

c. Si un terme clos M se β-réduit en un terme M' typable, alors M est aussi typable.

d. Si un terme clos M typable se β-réduit en un terme M', alors M' est aussi typable.

3 Sémantique à grands pas

On considère dans cet exercice le langage PCF. Les types sont donc les types simples engendrés à partir de l'unique type de base Int, et les termes permettent les opérations arithmétiques, le test à zéro et la définition récursive de fonctions. Intuitivement, $\mathsf{fixfun}\ f\ x \to M$ décrit une fonction d'argument x dont le corps est donné par M, où f est utilisée pour se référer à la fonction elle-même (typiquement pour faire un appel récursif).

Une sémantique à grands pas est donnée en figure 1 pour les constructions utiles à cet exercice. Les valeurs (représentées avec la lettre v) sont soit des entiers (représentés avec les lettres n ou m) soit des clôtures notées $\langle E, M\rangle$ où M est un terme de la forme $\mathsf{fun}\ x \to N$ ou $\mathsf{fixfun}\ f\ x \to N$, et E est un environnement, c'est à dire une substitution associant des valeurs aux variables de son domaine. On note $E[x := v]$ l'environnement associant v à x et coïncidant avec E pour les autres variables du domaine de E.

$$\frac{}{E \vdash x \Downarrow E(x)} \qquad \frac{}{E \vdash n \Downarrow n} \qquad \frac{E \vdash M \Downarrow m \quad E \vdash N \Downarrow n}{E \vdash M - N \Downarrow m - n}$$

$$\frac{E \vdash M \Downarrow 0 \quad E \vdash N_1 \Downarrow v}{E \vdash \mathsf{ifz}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2 \Downarrow v} \qquad \frac{E \vdash M \Downarrow n \quad E \vdash N_2 \Downarrow v}{E \vdash \mathsf{ifz}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2 \Downarrow v}\ n \in \mathbb{Z}^*$$

$$\frac{}{E \vdash \mathsf{fixfun}\ f\ x \to M \Downarrow \langle E, \mathsf{fixfun}\ f\ x \to M\rangle}$$

$$\frac{E \vdash M \Downarrow \langle E', \mathsf{fixfun}\ f\ x \to M'\rangle \quad E \vdash N \Downarrow v \quad E'[f := \langle E', \mathsf{fixfun}\ f\ x \to M'\rangle, x := v] \vdash M' \Downarrow v'}{E \vdash M\ N \Downarrow v'}$$

FIGURE 1 – Une sémantique à grands pas pour PCF

Question 1 Cette sémantique est-elle en appel par valeur ou par nom ?

Question 2 Donner un terme clos M tel qu'on n'a pas $\emptyset \vdash (M - M) \Downarrow 0$.

Question 3 Le terme suivant est valide et admet le type $\mathsf{Int} \to \mathsf{Int}$:

$$F \stackrel{\text{def}}{=} \mathsf{fixfun}\ f\ x \to \mathsf{ifz}\ x\ \mathsf{then}\ x\ \mathsf{else}\ f\ (x - 1)$$

Spécifier, en fonction de $n \in \mathbb{Z}$, ce que calcule $F\ n$. Démontrer que c'est bien le cas : on attend une preuve formelle s'appuyant sur la sémantique donnée.