MPRI, cours 2-7-1, preuves constructives

examen du 20 novembre 2006

Corrigé

1

(a) $\pi_1(p) \rhd^* 3$

(b)

$$\begin{array}{rcl} \lambda a : A.\lambda b : B.(a,b) & : & A \to B \to \Sigma x : A.B \\ \lambda f.\lambda c.(f\ \pi_1(c)\ \pi_1(c)) & : & (A \to B \to C) \to (\Sigma x : A.B) \to C \\ \lambda f.\lambda a.\lambda b.(f\ (a,b)) & & \end{array}$$

2

(a) $P_{32} \equiv \lambda n : N.\forall P : N \to o.(P\ 1) \Rightarrow (\forall m.(P\ m) \Rightarrow (P\ 2m)) \Rightarrow (\forall m.(P\ m) \Rightarrow (P\ 3m)) \Rightarrow (P\ n)$

(b) Pareil, en remplaçant les $\forall$ par des Π.

$$p \equiv \lambda P.\lambda p_1.\lambda p_2.\lambda p_3.(p_2\ 4\ (p_2\ 2\ (p_2\ 1\ p_1)))$$

Le terme est de taille proportionelle à i (il itère i fois).

3

La manière la plus simple de définir "être une puissance de deux" est probablement de définir la fonction exponantiation puis de poser :

$$P_2 \equiv \lambda n : N.\Sigma i : N.n = (exp\ 2\ i)$$

(je donne des points à cette réponse)

Ici, on incite plutôt à passer par le test booléen correspondant.

Une possibilité est d'écrire la fonction qui rend le quotien par 2 (puisque pair rend ce qui correspond au reste).

On peut alors, par exemple, itérer n fois

4 Système F

$$test0 \equiv \lambda n.(n\ N\ 1\ \lambda p.0)$$

$$pair \equiv \lambda n.(n\ N\ 1\ test0)$$

5 Ecologie

P est de type $Arbre \to \mathsf{Type}$.

$$(RAb_P\ p_F\ p_N\ Feuille) \rhd p_F$$

$$(RAb_P\ p_F\ p_N\ (Noeud\ n\ T_1\ T_2)) \rhd p_N\ T_1\ T_2\ n\ (RAb_P\ p_F\ p_N\ T_1)\ (RAb_P\ p_F\ p_N\ T_2)$$

On construit d'abord la fonction qui compare un entier à la racine d'un arbre, si elle existe :

$$inf1 \equiv \lambda n.\lambda t.RAb_N\ 1\ \lambda m.\lambda\_\_.(inf\ n\ m)\ t$$

puis on construit le test pour les arbres :

$$is\_heap \equiv \lambda t.(RAb_N\ 1\ \lambda n\lambda T_1.\lambda T_2.\lambda r_1.\lambda r_2.r_1 \wedge_b r_2 \wedge_b (inf1\ n\ T_1) \wedge_b (inf1\ n\ T_2)$$

où $\wedge_b$ code le "et booléen".
Et si l'on veut le prédicat :

$$\lambda T : Arbre.(is\_heap\ T) = 1$$

6 Devinettes

(a) Non. Sinon on pourrait typer tous les lambda-termes. En particulier, on aurait

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

et donc aussi

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

ce qui contredit le théorème de normalisation.
(b) pour le premier, c'est facile, on prend

$$\lambda f : N \to N.(exi\ 0\ p)$$

où p est facile à construire.

Pour le second, c'est impossible, car une preuve constructive nous permettrait de décider si une suite à un majorant. Même pour une suite définie dans CC, c'est indécidable.