Preuves Constructives
Examen
DEA de Programmation

18 décembre 2001

1 Classique (sans tiers-exclu)

Dans la théorie de Martin-Löf, on a fabriqué un terme t de type

$$\Pi x : \text{nat}.\Sigma y : \text{nat}.(x = (\text{mult } 2\ y)) + (x = (mult\ 3\ y)) + (x = (S\ (mult\ 2\ y)))$$

où 2 et 3 signifient respectivement $(S(S\ 0))$ et $(S\ 2)$, etc.

Quelles sont les formes normales possibles de :

$\pi_1(t\ 4)$

$\pi_1(t\ 5)$

$\pi_1(t\ 9)$.

2 Peano revisited

On modifie un peu la définition de l'arithmétique de Peano en déduction modulo. Pour cela, on étend le langage par une fonction pred et un prédicat Null. Plus précisément, on a les symboles de fonction :

$$0, S, +, *, \mathsf{pred}$$

les prédicats :

$=$ d'arité 2,

Null d'arité 1.

On ne garde que les axiomes de l'égalité et de récurrence :

$$\forall x\ x = x$$
$$\forall x \forall y\ x = y \wedge P(x) \Rightarrow P(y)$$
$$P(0) \wedge (\forall x\ P(x) \Rightarrow P(S(x))) \Rightarrow \forall n\ P(n)$$

Enfin on prend les règles de réécriture usuelles pour $+$ et $*$ (par exemple $0 + x \rhd x$) que l'on étend avec les trois règles suivantes :

$$\begin{aligned} \mathsf{Null}(0) &\rhd \top \\ \mathsf{Null}(S(x)) &\rhd \bot \\ \mathsf{pred}(S(x)) &\rhd x \end{aligned}$$

Prouvez, dans ce cadre, les autres axiomes de Peano, à savoir :

(a) $\forall x\ \neg 0 = S(x)$

(b) $\forall x\ \forall y\ S(x) = S(y) \Rightarrow x = y$

3 Des petits programmes

3.1 un minimum d'ordre

Définissez dans le système T une fonction is_inf de type nat → nat → nat qui rende 0 si le premier argument est inférieur ou égal au second, et 1 sinon.

3.2 small is beautifull

(a) Définissez dans le système T une fonction min de type nat → nat → nat tel que le résultat soit le plus petit de ses deux arguments.
(b) Quelle est la complexité de cette fonction ?
(c) Cette définition peut être transposée dans la théorie des types de Martin-Löf. Dans cette théorie, comment décriveriez-vous cette fonction ? (écrivez la proposition formelle correspondante).

4 Imprédicativité

4.1 Que j'itère

On est dans le calcul des constructions. On rappelle le codage des entiers naturels :

$$\begin{aligned} \text{nat} &\equiv \Pi X : \text{Type}.X \to (X \to X) \to X \\ 0 &\equiv \lambda X : \text{Type}.\lambda x : X.\lambda f : X \to X.x \\ S &\equiv \lambda n : \text{nat}.\lambda X : \text{Type}.\lambda x : X.(f(nXxf)) \end{aligned}$$

On écrira 1, 2, etc pour les termes $(S\ 0), (S\ (S\ 0))\ldots$
On définit :

$$\mathsf{myst} \equiv \lambda n : \text{nat}.\Pi P : \text{nat} \to \text{Type}.(P\ 1) \to (\Pi x : \text{nat}.(P\ x) \to (P\ (\text{mult}\ 2\ x))) \to (P\ n)$$

(a) Quel est le type de myst ?
(b) Donnez des termes-preuves de (myst 1) et (myst 2).
(c) Donnez une définition mathématiquement équivalente mais plus lisible de myst.
(d) On veut un terme

$$div2 : \Pi n : \text{nat}.(\mathsf{myst}\ n) \to \text{nat}$$

tel que $(div2\ n\ p)$ se réduise vers $log_2(n)$. Proposez un tel terme, aussi simple que possible (en utilisant la *première* définition de myst).

4.2 Une ténébreuse affaire

Un jeune utilisateur de Coq va voir un héritier de Church. Il lui reproche qu'en théorie des types simples, on ne sait pas parler de listes. Aidons le jeune Church a réfuter l'argument du petit coq.
(a) On dit qu'un entier code une paire s'il est de la forme 2^p3^q (il code alors la paire (p, q)). Définissez la relation *exp* telle que $(exp\ a\ b\ c)$ signifie $a = b^c$. Définissez ensuite la propriété "être le code d'une paire".
(b) L'idée de Church Jr. est la suivante :
- 0 va être le code de la liste vide
- si n est le code d'une liste l, alors (a, l) est le code de la liste $n :: l$.

Définissez le prédicat "être le code d'une liste" en théorie des types simples.
(c) Comment définissez-vous la propriété : "être une liste de premier élément non nul" ? "être une liste sans élément nul" ?