TD13 7 janvier 2013

Bonne année !

Logique & Calculabilité

Hubert Comon ∧ David Baelde
{comon,baelde}@lsv.ens-cachan.fr

Exercice 1
Montrer que l'addition, la multiplication et la division par deux sont récursives primitives.

Exercice 2 (Schéma de minimisation bornée)
Étant données ψ et ξ récursives primitives, on définit

$$\phi(\vec{n}) = \min_{m \leq \psi(\vec{n})} (\xi(\vec{n}, m) = 0)$$

Montrer que ϕ est récursive primitive – on pourra renvoyer une valeur quelconque quand le minimum n'est pas atteint.

Exercice 3 (Codage des paires)
Montrer qu'il existe $J : \mathbb{N}^2 \to \mathbb{N}$, et $K, L : \mathbb{N} \to \mathbb{N}$ telles que :

1. J, K et L sont récursives primitives,
2. J est une bijection et
3. on a $K(J(x, y)) = x$ et $L(J(x, y)) = y$ pour tous x, y.

Exercice 4
On considère une fonction, écrite dans un langage impératif quelconque, sous la forme suivante :

```
function f(x1,...,xN) {
  for i = 0 to F { G }
  return x1;
}
```

On suppose que F est une expression pure construite sur les variables x_i à partir des opérations arithmétiques usuelles.

1. On suppose que le bloc G est une suite d'affectations $x_i := E$ où E est une expression arithmétique construite sur les x_j. Montrer qu'on peut compiler f en une fonction récursive primitive.
2. De même si l'on autorise des tests dans G, avec des conditions construites à l'aide des connecteurs booléens et des comparaisons arithmétiques usuelles.

Exercice 5 (Fibonacci)
Montrer que la fonction définie comme suit est récursive primitive :

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 1 \\ f(n+2) &= f(n+1) + f(n) \end{aligned}$$

Exercice 6
On rappelle la définition de la fonction d'Ackermann :

$$\begin{aligned} A(0,m) &= m+1 \\ A(n+1,0) &= A(n,1) \\ A(n+1,m+1) &= A(n,A(n+1,m)) \end{aligned}$$

1. Montrer que le graphe de la fonction d'Ackermann est récursif primitif.
2. Montrer que l'on peut implémenter Ackermann dans le langage des fonctions récursives primitives étendu avec la boucle `while` (avec condition et corps primitifs récursifs) ou le schéma de minimisation non bornée (remplacer $m \leq \psi(\vec{n})$ par $m \in \mathbb{N}$ dans l'exercice 2).

Gros bonus : Coder Ackermann avec le schéma de récursion primitive dans un cadre fonctionnel, *i.e.*, on peut retourner une fonction et pas seulement un entier. Indice : preuve = programme, récursion primitive = récurrence, comment prouveriez vous en arithmétique que la fonction Ackermann est totale ?