Examen du cours « Logique et théorie du calcul » (2012-2013)

Emmanuel Beffara & Yves Lafont
Institut de Mathématiques de Luminy
Université d'Aix-Marseille

jeudi 13 décembre 2012

1 Fonctions primitives récursives et réécriture de mots

Au lieu de définir les *fonctions primitives récursives* comme des applications $f : \mathbb{N}^p \to \mathbb{N}$ (avec $p \in \mathbb{N}$), on va plutôt les définir comme des applications $f : \mathbb{N}^p \to \mathbb{N}^q$ (avec $p, q \in \mathbb{N}$). On obtient ainsi une définition plus naturelle, et plus proche de la *réécriture de mots* ou des *machines à piles*. Cette définition, due à Albert Burroni, est inspirée par la *théorie des catégories*.

On considère la plus petite classe $\mathcal{P}$ qui contient les applications suivantes :

- $\mathbf{id}_p : \mathbb{N}^p \to \mathbb{N}^p$ pour tout $p \in \mathbb{N}$, définie par $\mathbf{id}_p(x_1, \dots, x_p) = (x_1, \dots, x_p)$;
- $\mathbf{0}_p : \mathbb{N}^p \to \mathbb{N}^{p+1}$ pour tout $p \in \mathbb{N}$, définie par $\mathbf{0}_p(x_1, \dots, x_p) = (0, x_1, \dots, x_p)$;
- $\mathbf{S}_p : \mathbb{N}^{p+1} \to \mathbb{N}^{p+1}$ pour tout $p \in \mathbb{N}$, définie par $\mathbf{S}_p(x_0, x_1, \dots, x_p) = (x_0 + 1, x_1, \dots, x_p)$;

et qui est close par les deux opérations suivantes :

- la *composée* $h = g \circ f : \mathbb{N}^p \to \mathbb{N}^r$, pour $f : \mathbb{N}^p \to \mathbb{N}^q$ et $g : \mathbb{N}^q \to \mathbb{N}^r$, définie par :
$$h(x_1, \dots, x_p) = g(f(x_1, \dots, x_p));$$
- l'*itérée* $k = g * f : \mathbb{N}^{p+1} \to \mathbb{N}^q$, pour $f : \mathbb{N}^p \to \mathbb{N}^q$ et $g : \mathbb{N}^q \to \mathbb{N}^q$, définie par :
$$\begin{cases} k(0, x_1, \dots, x_p) = f(x_1, \dots, x_p), \\ k(x_0 + 1, x_1, \dots, x_p) = g(k(x_0, x_1, \dots, x_p)). \end{cases}$$

Par exemple, on a $g * f(3, x_1, \dots, x_p) = g(g(g(f(x_1, \dots, x_p))))$.

On note () l'unique élément de $\mathbb{N}^0$, et on identifie $\mathbb{N}^1$ avec $\mathbb{N}$.

1. Montrer que pour tout $n \in \mathbb{N}$, l'application $\underline{n} : \mathbb{N}^0 \to \mathbb{N}^1$, définie par $\underline{n}() = n$, est dans la classe $\mathcal{P}$.
2. Montrer que l'application $\sigma : \mathbb{N}^2 \to \mathbb{N}$, définie par $\sigma(x, y) = x + y$, est dans la classe $\mathcal{P}$.
3. Montrer que l'application $d : \mathbb{N} \to \mathbb{N}$, définie par $d(x) = 2x$, est dans la classe $\mathcal{P}$.
4. Montrer que l'application $e : \mathbb{N} \to \mathbb{N}$, définie par $e(x) = 2^x$, est dans la classe $\mathcal{P}$.
5. Montrer que l'application $\delta : \mathbb{N} \to \mathbb{N}^2$, définie par $\delta(x) = (x, x)$, est dans la classe $\mathcal{P}$.

On introduit l'alphabet $\Sigma = \{\mathbf{o}, \mathbf{s}\}$, et on code l'entier $n \in \mathbb{N}$ par le mot $[n] = \mathbf{s}^n\mathbf{o} \in \Sigma^*$.
Par exemple, on a $[3] = \mathbf{ssso}$.

Plus généralement, on code le p-uplet $(n_1, \dots, n_p) \in \mathbb{N}^p$ par le mot $[n_1, \dots, n_p] = [n_1] \cdots [n_p] \in \Sigma^*$.
Par exemple, on a $[2, 3] = \mathbf{ssossso}$.

6. Montrer que si $f : \mathbb{N}^p \to \mathbb{N}^q$ est dans la classe $\mathcal{P}$, alors on peut construire un alphabet fini $\Sigma' \supset \Sigma$, un ensemble fini $\mathcal{R} \subset \Sigma'^* \times \Sigma'^*$, et un mot $u \in \Sigma'^*$ tels que, pour tout $(x_1, \dots, x_p) \in \mathbb{N}^p$, on a :
$$u\,[x_1, \dots, x_p] \to_{\mathcal{R}}^* [y_1, \dots, y_q] \text{ où } (y_1, \dots, y_q) = f(x_1, \dots, x_p).$$
7. Expliciter Σ' et $\mathcal{R}$ dans chacun des cas particuliers suivants :
$$\sigma : \mathbb{N}^2 \to \mathbb{N}, \qquad d : \mathbb{N} \to \mathbb{N}, \qquad e : \mathbb{N} \to \mathbb{N}.$$
8. Montrer que les trois systèmes de réécriture ci-dessus sont *convergents*, c'est-à-dire qu'ils satisfont les propriétés de *terminaison* et de *confluence*.

2 Arithmétique de Peano et calculabilité

On rappelle qu'un énoncé est dit Δ_0 s'il utilise le langage de l'arithmétique (avec relation d'ordre) et si dans cet énoncé toute quantification universelle est de la forme $\forall x((x \leq t) \to A)$, notée $(\forall x \leq t)A$, et toute quantification existentielle est de la forme $\exists x((x \leq t) \wedge A)$, notée $(\exists x \leq t)A$. Un énoncé Σ_1 est un énoncé de a la même forme précédé d'une quantification existentielle $\exists t$ pas nécessairement bornée.

1. Montrer que les énoncés suivants peuvent s'exprimer avec des énoncés Δ_0 :
   - « q est le quotient de a par b dans la division euclidienne »
   - « q est le reste dans la division euclidienne de a par b »
   - « a est un carré »
   - « a est une puissance de 2 »
   - « a est une puissance de 4 »

On considère des machines de Turing à 1 ruban sur un alphabet $\{a, b\}$ en plus des deux symboles spéciaux de case vide et case initiale.

2. Montrer que l'on peut représenter une configuration d'une machine de cette nature par un triplet $(q, a_1 \dots a_\ell, b_1 \dots b_r)$ où les a_i sont les lettres à gauche (a_1 est celle qui est immédiatement à gauche et a_ℓ celle qui est en première position sur le ruban).

On considère les mots comme des entiers écrits en base 4 (puisqu'il y a 4 symboles) et on représente donc une configuration par trois entiers (q, L, R).

3. Montrer que, pour une macine de Turing donnée M, on peut écrire l'énoncé suivant dans Δ_0 :

   (q, L, R) n'est pas une configuration finale et (q', L', R') est la configuration suivante de M

4. En déduire que l' énoncé suivants peut être exprimé en arithmétique par un énoncé Δ_0 :

   « partant de la configuration (q, L, R), la machine M termine en au plus t étapes sa configuration finale est (q', L', R') »

5. Conclure qu'une fonction (éventuellement partielle) de $\mathbb{N}$ dans $\mathbb{N}$ est calculable par une machine de Turing si et seulement si elle est récursive.