Silvain Rideau
Bureau T5
silvain.rideau@ens.fr
www.eleves.ens.fr/home/srideau

TD de Logique 9 (Machines à registres)

28 novembre et 1 décembre 2014

Les exercices qui ne sont pas abordés en cours, seront (certainement) corrigés sur ma page personelle (voir ci-dessus).

Exercice 1 (Existence de racines) **:**

1. Soit n un entier. Montrer que l'ensemble $\{(a_0, ..., a_n) \in \mathbb{N}^{n+1}$: le polynôme $a_0 + a_1X + a_2X^2 + ... + a_nX^n$ admet une racine dans $\mathbb{Z}\}$ est primitif récursif.
2. Qu'en est-il si on remplace "racine dans $\mathbb{Z}$« par "racine dans $\mathbb{Q}$ ».

Exercice 2 (Machines à registres) **:**
On va considérer dans cet exercice un modèle de calcul un peu plus concret que les fonctions récursives. On appelle machine à registres une machine qui a une infinité de registres (indicés par $\mathbb{N}$) qui peuvent contenir des entiers et qui exécute un programme composé d'une liste de commandes suivantes :

- $\mathrm{Incr}(i)$ qui incrémente le registre i de 1 ;
- $\mathrm{Zero}(i)$ qui remet le registre i à 0 ;
- $\mathrm{Cp}(i, j)$ qui copie la valeur du registre j dans le registre i ;
- $\mathrm{Test}(i, j, k)$ qui, si le registre i et le registre j contiennent la même valeur, fait aller directement à la commande k (et sinon ne fait rien) ;
- Fin qui dit que le programme est fini.

Sauf en cas de Test un programme est exécuté linéairement. Formellement un programme est une fonction d'un entier $n = \{0, \ldots, n-1\}$ dans l'ensemble des commandes, i.e. une suite finie de commandes indiquées par des entiers $< n$. Un état d'une machine à registres est la donnée d'un entier (la ligne de commande à exécuter) et une fonction $\mathbb{N} \to \mathbb{N}$ à support fini (l'état des registres).
On a une fonction exec qui a un programme P et un état (k, R) associe :

- Si $P(k) = \mathrm{Incr}(i)$, l'état $(k+1, S)$ où $S(j) = R(j)$ si $j \neq i$ et $S(i) = R(i) + 1$;
- Si $P(k) = \mathrm{Zero}(i)$, l'état $(k+1, S)$ où $S(j) = R(j)$ si $j \neq i$ et $S(i) = 0$;
- Si $P(k) = \mathrm{Cp}(i, j)$, l'état $(k+1, S)$ où $S(l) = R(l)$ si $l \neq i$ et $S(i) = R(j)$;
- Si $P(k) = \mathrm{Test}(i, j, l)$ et $R(i) = R(j)$, l'état (l, R) ;
- Si $P(k) = \mathrm{Test}(i, j, l)$ et $R(i) \neq R(j)$, l'état $(k+1, R)$;
- Si $P(k) = \mathrm{Fin}$, l'état (k, R).

Un état (k, R) est dit final pour le programme P si $P(k) = \mathrm{Fin}$. On dit qu'un programme termine sur l'entrée $\overline{x}$ s'il existe n tel que la n-ième itération de $\mathrm{exec}(P, \cdot)$ sur l'état $(0, R)$, où $R(i) = x_i$ si $i < |x|$ et $R(i) = 0$ sinon, est un état final (k, S). On dit alors que le résultat est $S(0)$. On dit qu'une fonction (partielle) $\mathbb{N}^k \to \mathbb{N}$ est calculable par machine à registres s'il existe un programme P tel que pour tout $\overline{x}$ tel que $f(\overline{x})$ est défini, le programme P termine sur $\overline{x}$ avec comme résultat $f(\overline{x})$ et si $f(\overline{x})$ n'est pas définie, P ne termine pas sur $\overline{x}$.

1. Montrer que la fonction prédécesseur est calculable par machine à registres.
2. Montrer que la fonction somme est calculable par machine à registre.
3. Montrer que toute fonction récursive est calculable par machine à registres.

4. Montrer qu'il existe une fonction surjective C de $\mathbb{N}$ dans l'ensemble des programmes telle que le prédicat $\mathcal{A}_n(i, \overline{x}, t) :=$ « $C(i)$ termine sur l'entrée $\overline{x}$ en moins de t étapes » est primitif récursif, , où $|x| = n$. De même pour le prédicat $\mathcal{B}_n(i, \overline{x}, y, t) :=$ « $C(i)$ termine avec le résultat y sur l'entrée $\overline{x}$ en moins de t étapes ».

5. En déduire qu'il existe une fonction récursive qui énumère les fonctions $\mathbb{N}^k \to \mathbb{N}$ calculables par machine à registres.

6. En déduire aussi qu'une fonction est calculable par machine à registres si et seulement si elle est récursive.

7. En déduire l'existence d'une fonction récursive universelle.

Exercice 3 (Fonction universelle primitive récursive) **:**
On note T la fonction récursive partielle qui à i et x associe le temps de calcul du programme $C(i)$ sur l'entrée x.

1. Soit $f : \mathbb{N} \to \mathbb{N}$, montrer que f est primitive récursive si et seulement s'il existe un entier i et une fonction $g : \mathbb{N} \to \mathbb{N}$ primitive récursive telle que la machine d'indice i calcule f et $T(i, x) \leqslant g(x)$.

2. En déduire qu'il existe une fonction récursive $\varphi : \mathbb{N}^2 \to \mathbb{N}$ telle que

$$\{\varphi_i : i \in \mathbb{N}\} = \{f : \mathbb{N} \to \mathbb{N} : f \text{ primitive récursive}\}$$

   [Indication : considérer $g(i, a, n, x) = \mu y \leqslant \max(a, \xi_n(x))[\exists t \leqslant \max(a, \xi_n(x))(i, x, y, t) \in \mathcal{B}_1]$ où ξ est la fonction d'Ackermann.]

3. Montrer que cette fonction φ ne peut pas être primitive récursive.
   [Indication : Comme on pourrait s'en douter c'est un argument diagonal, regardez $x \mapsto \varphi(x, x) + 1$.]

Exercice 4 (Une bijection primitive récursive d'inverse non primitif récursif, ou comment refaire la fin du Td précédent de façon moins ad-hoc) **:**

1. (Rappel du Td précédent) Montrer que l'ensemble des bijections récursives de $\mathbb{N}$ dans $\mathbb{N}$ forme un sous-groupe du groupe des permutations de $\mathbb{N}$.

2. Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction récursive (totale), mais non récursive primitive, et soit i l'indice d'un programme P qui calcule f. On considère la fonction T qui à x associe le temps de calcul de $f(x)$ par P, i.e. $T(x) = \mu t[(i, t, x) \in \mathcal{A}]$. Montrer que le graphe de T est primitif récursif. En déduire qu'il n'y a pas de fonction récursive primitive $g : \mathbb{N} \to \mathbb{N}$ telle que $g(x) \geqslant T(x)$ pour tout $x \in \mathbb{N}$.

3. On pose $g_0(x) = \sup\{T(y); y \leqslant x\} + 2x$. Montrer que g_0 est récursive, mais non primitive récursive, et que le graphe G_0 de g_0 ainsi que l'image I_0 de g_0 sont des ensembles primitifs récursifs.

4. Montrer qu'il existe une (unique) fonction strictement croissante g_1 récursive primitive dont l'image soit égale à $\mathbb{N} \setminus I_0$.

5. Montrer que la fonction $h : \mathbb{N} \to \mathbb{N}$ donnée par $h(2x) = g_0(x)$, $h(2x+1) = g_1(x)$ est bijective, récursive, mais non récursive primitive. Montrer que son inverse h^{-1} est récursive primitive.

Exercice 5 (Complexité) **:**
Étant donnée un programme P et une entrée n, on définit la complexité de (P, n) comme étant le nombre de lignes de P plus $\log(n)$. La complexité d'un entier N est la complexité minimale d'un couple (P, n) tel que le programme P termine sur l'entrée n et renvoie N en sortie.

1. Soit g la fonction qui à N associe sa complexité. Montrer que g tend vers $+\infty$.

2. Soit f une fonction récursive tendant vers $+\infty$. Montrer qu'il existe N tel que $f(N) > g(N)$. En particulier, g n'est pas récursive.

Exercice 6 (Le retour des structures finies) **:**
Soit $\mathcal{L}$ un langage fini et T une $\mathcal{L}$-théorie finie, montrer que l'ensemble :

$$\{n \in \mathbb{N} : \exists M \vDash T, |M| = n\}$$

est primitif récursif.

Exercice 7 (Ensembles de fonctions récursivement énumérables) **:**
Soit $\mathfrak{F}$ un sous-ensemble des fonctions $\mathbb{N} \to \mathbb{N}$, on dit que $\mathfrak{F}$ est récursivement énumérable si il existe $\psi : \mathbb{N}^2 \to \mathbb{N}$ récursive telle que

$$\{\psi_i : i \in \mathbb{N}\} = \mathfrak{F}.$$

On a montré dans l'exercice 3 que l'ensemble des fonctions primitives récursives à une variable est récursivement énuméré.

1. Montrer que l'ensemble des fonctions récursives totales à une variable n'est pas récursivement énumérable.
2. Montrer que l'ensemble des fonctions récursives primitives strictement croissantes (en une variable) est récursivement énumérable.
3. Montrer que l'ensemble des fonctions récursives primitives injectives (en une variable) est récursivement énumérable.
4. Montrer que l'ensemble des fonctions récursives totales injectives (respectivement strictement croissantes) en une variable n'est pas récursivement énumérable.