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

Corrigé du TD de Logique 9 (Machines à registres)

28 novembre et 1 décembre 2014

Attention, le Td et sa correction sont écrits en considérant qu'un schéma μ rend le premier élément qui vérifie un certain prédicat primitif récursif (et non pas le premier zéro d'une fonction primitive récursive comme il semblerait que ce soit le cas dans le cours).

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

1. Remarquons tout d'abord que si $\sum_{i=0}^{n} a_i x^i = 0$, on a en particulier $a_0 = x(\sum_{i=1}^{n} a_i x^{i-1})$ et donc x divise a_0, en particulier $|x| \leqslant a_0$, de plus une racine d'un tel polynôme est forcément négative. L'ensemble que l'on cherche est donc :
$$\exists x \leqslant a_0\, a_0 + a_2 x^2 + \cdots + a_{2\lfloor n/2 \rfloor} x^{2\lfloor n/2 \rfloor} = a_1 x + \cdots + a_{2\lfloor (n-1)/2 \rfloor + 1} x^{2\lfloor (n-1)/2 \rfloor + 1}$$
qui est bien récursif.

2. Comme précédemment notre polynôme a une racine, il existe une p/q telle que p est négatif, $p \leqslant a_0$ et $q \leqslant a_n$. L'ensemble que l'on cherche est donc :
$$\exists x \leqslant a_0 \exists y \leqslant a_n\, a_0 y^n + \cdots + a_{2\lfloor n/2 \rfloor} x^{2\lfloor n/2 \rfloor} y^{n-2\lfloor n/2 \rfloor} = a_1 x y^{n-1} + \cdots + a_{2\lfloor (n-1)/2 \rfloor + 1} x^{2\lfloor (n-1)/2 \rfloor + 1} y^{n-2\lfloor (n-1)/2 \rfloor - 1}$$

Exercice 3 (Fonctions universelle primitive récursive) **:**

1. Si la machine d'indice i calcule f et $T(i,x) \geqslant g(x)$ alors $f(x) = \mu y \leqslant \max(x) + g(x)[\exists t \leqslant g(x),\, \mathcal{B}_1(i,x,y,t)]$ qui est bien récursive primitive. La réciproque est plus compliquée et elle se fait par induction :
   - Les projection, la fonction successeur et la constante égale à zéro se calculent en temps constant (égal à 2) ; les programmes en question étant respectivement $\mathrm{Cp}(0,i); \mathrm{Fin}$, $\mathrm{Incr}(0); \mathrm{Fin}$ et $\mathrm{Zero}(0); \mathrm{Fin}$.
   - La composition de $f(\overline{y})$ (calculée par le programme $P = C(n)$)avec les $g_i(\overline{x})$ calculée avec le programme $Q_i = C(n_i)$ consiste à recopier les $overlinex$, calculer Q_i, recopier les $overlinex$ un peu plus loin, calculer Q_1 etc, et enfin copier chacun des résultats sur les premiers registres puis appliquer P. Le temps de calcul est donc égal à $\sum_i T(n_i, \overline{x}) + T(n, \overline{g}(\overline{x})) + C$ où $C \in \mathbb{N}$ qui est bien primitive récursive si $T(n,\cdot)$ et les $T(n_i,\cdot)$ le sont.
   - La fonction définie par récurrence par le schéma $f(\overline{x},0) = g(\overline{x})$ et $f(\overline{x},y+1) = h(\overline{x},y,f(\overline{x},y))$, où g est calculé par $Q = C(n)$ et h est calculée par $R = C(m)$ se calcule en calculant d'abord Q puis pour tout $i < y$ en calculant R sur $\overline{x}, i, f(\overline{x},i)$ (plus un nombre linéaire en y de copies, de remises à zéro etc). Le temps de calcule de f est donc de la forme $T(n,\overline{x}) + \sum_{i<y} T(m,\overline{x},y,f(\overline{x},y)) + yC$ qui est bien primitive récursive si $T(n,\cdot)$ et $T(m,\cdot)$ le sont.

   Il suit de cette induction que pour tout fonction f primitive récursive on trouve i tel que $C(i)$ calcule f et $T(i,\overline{x})$ est primitif récursif.

2. Par le sens facile de la question précédente et le fait que que les ξ_n sont primitive récursives, pour tout choix de i, a et n, la fonction $g(i,a,n,\cdot)$ est primitive récursive (elle est calculée en temps primitif récursif par une machine à registres). Réciproquement, si f est primitive récursive, par la question précédente, il existe i tel que $C(i)$ calcule f en temps primitif récursif. Comme le max de f et du temps de calcul de f par la machine i est primitif récursif, il existe n tel que ξ_n domine cette fonction, i.e. il existe a tel que $\max(a, \xi_n(\overline{x}))$ majore à la fois f et son temps de calcule par la machine i. La fonction $g(i,a,n,\cdot)$ est alors exactement f.

3. Supposons que φ soit primitive récursive alors il existe i_0 tel que φ_{i_0} calcule $\varphi(x,x) + 1$ et donc $\varphi_{i_0}(i_0) = \varphi(i_0,i_0) = \varphi(i_0,i_0) + 1$ ce qui est absurde.

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. Soit f bijective récursive. La réciproque de f est donnée par $f^{-1}(y) = \mu x[f(x) = y]$.

2. (C'est la même question que le sens facile de 4.1) En reprenant les notations de l 'exercice 3, $\mathcal{A}_1 = \{(i,x,t) :$ la machine i termine en un temps t sur $x\}$ est primitif récursif. Le graphe de T est donc lui aussi primitif récursif car c'est $\mathcal{A}_1(i,t,x)$. S'il existait g primitive récursive telle que $T(x) \leqslant g(x)$, $f(x)$ serait aussi borné par $g(x) + x$ car on ne peut augmenter la valeur des registres que de 1 par étape de calcul. On aurait alors $f(x) = \mu y \geqslant g(x) + x \exists t \geqslant g(x) \mathbb{1}_{\mathcal{B}_1}(i,x,y,t)$ qui serait récursive primitive.

3. Comme $T(x) = \mu t\, \mathbb{1}_{\Gamma_T}(x,t)$ où Γ_T est le graphe de T, primitif récursif, il s'en suit que T est récursive. On a ensuite $g_0(0) = T(0)$ et $g_0(y+1) = \max(g_0(y) - 2y, T(y+1))$. Elle est donc bien récursive (totale). Comme $g_0(x)$ majore $T(x)$, il s'en suit que g_0 ne peut être récursive par la question précédente.

   Le graphe de g_0 est donné par $\mathbb{1}_{\Gamma_{g_0}}(0,x) = x \dot{=} T(0)$ et $\mathbb{1}_{\Gamma_{g_0}}(y+1,x) = [\mathbb{1}_{\Gamma_{g_0}}(y, x-2) \wedge \exists t \leqslant x - 2(y+1) \mathbb{1}_{\Gamma_T}(y+1,t)] \vee [\mathbb{1}_{\Gamma_T}(y+1, x-2(y+1)) \wedge \exists t \leqslant x - 2 \mathbb{1}_{\Gamma_{g_0}}(y,t)]$ qui est bien primitif récursif (ce n'est pas un schéma de récurence primitive classique car il fait appel à tous les résultats de $g_0(y,t)$ pour $t < x$ mais on peut le coder en utilisant des suites dans le schéma de récurrence primitive classique).

   Enfin, g_0 est une fonction récursive (strictement) croissante totale. Comme montré dans l'exercice 1, son image est donc un ensemble récursif.

4. La fonction g_1 existe et elle est unique car deux fonctions strictement croissantes sont égales si et seulement si elles ont la même image.

   Il suffit donc de montrer que g_1 est primitive récursive. On remarque d'abord que $g(x+1) \geqslant g(x) + 2$ et donc que $|\{i \in I_0 : i \leqslant 2n\}| \leqslant n+1$. Il s'en suit donc que $g_1(x) \leqslant 2x$. On pose donc $g_1(0) = \mu t \leqslant 1\, \neg \mathbb{1}_{I_0}(t)$ et $g_1(x+1) = \mu t \leqslant 2x+1\, t > g_1(x) \wedge \neg \mathbb{1}_{I_0}(t)$.

5. Comme les g_i sont strictement croissantes elles sont en particulier injectives. De plus, comme leurs images sont disjointes, h est aussi injective. Enfin comme leurs images recouvrent $\mathbb{N}$, h est surjective. La récursivité de h est une conséquence de la récursivité des g_i, mais h n'est pas primitive récursive sinon g_0 le serait, ce qui contredit la question 3.

   Enfin, montrons que h^{-1} est primitive récursive. En effet, si $x \in I_0$, $h^{-1}(x) = 2[\mu t \leqslant x\, \mathbb{1}_{\Gamma_{g_0}}(t,x)]$ et si $x \notin I_0$, $h^{-1}(x) = 2[\mu t \leqslant x\, \mathbb{1}_{\Gamma_{g_1}}(t,x)] + 1$, ce qui permet de conclure. Pour les plus dubitatifs (ou les plus attentifs) le fait que Γ_{g_1} soit primitif récursif est une conséquence immédiate du fait que g_1 est elle-même primitive récursive.

Exercice 5 (Complexité) **:**

1. Soit $M_0 \in \mathbb{N}$ un entier. L'ensemble des nombres de complexité plus petite que M_0 est fini. En effet, il y a un nombre fini de programmes avec au plus M_0 lignes et un nombre fini d'entiers plus petits que e^{M_0} et donc de logarithme plus petit que M_0. Il s'en suit qu'il existe $A \in \mathbb{N}$ tel que tout $n > A$ a une complexité plus grande que M_0.

2. Supposons que pour tout $N \in \mathbb{N}$, $f(N) \leqslant g(N)$. On va supposer que f est complète (sinon on peut s'en sortir aussi mais c'est un peu agaçant...). Pour tout $y \in \mathbb{N}$ on pose $h(y) = \mu x\, f(x) \geqslant y$. La fonction h est alors récursive et soit l le nombre de ligne d'un programme qui calcule h. Par définition de la complexité, $g(h(y)) \leqslant l + \log(y) \leqslant l + \log(f(h(y)))$. Or $f(h(y)) \leqslant g(N)$, d'où $f(h(y)) \leqslant l + \log(f(h(y)))$. Il s'en suit que $h(y)$ est borné, ce qui contredit le fait que f tends vers $+\infty$.

Exercice 6 (Le retour des structures finies) **:**
Remarquons tout d'abord que quitte à remplacer les constantes c par un prédicat unaire U_c et les fonctions n-aire f par un prédicat $N+1$-aire R_f, et à rajouter dans T les formules qui disent que U_c est vrai en exactement 1 point et que R_f est un graphe de fonction. On peut supposer que $\mathcal{L}$ ne contient que des relations[1]. En effet tout modèle de notre nouvelle théorie induit un modèle de l'ancienne et vice-versa. Quitte à prendre comme unique relation le produit des relations de notre langage, on peut supposer que le langage est composé d'une unique relation[2] R. Enfin toute $\mathcal{L}$-structure finie est isomorphe à une $\mathcal{L}$ structure de domaine $\{1,\dots,n\}$.

[1] Il n'est pas nécessaire de faire une telle réduction mais cela facilite les notations.
[2] Encore une fois, ce n'est pas nécessaire mais cela va nous simplifier la vie.

On code une $\mathcal{L}$-structure $\mathcal{M}$ (de domaine $\{1,\dots,n\}$) par

$$\alpha_2(|M|, u(R^{\mathcal{M}}));\ u(R^{\mathcal{M}}) = \prod_{\vec{i}\in R^{\mathcal{M}}} \rho_{\alpha_k(\vec{i})}$$

où R est un prédicat k-aire, ρ énumère les nombres premiers de façon coirssante et α_i est une bijection primitive récursive de $\mathbb{N}^i$ dans $\mathbb{N}$ croissante en toute les variables. On peut alors vérifier que le code d'une structure de cardinal n est borné par $b(n) = \alpha_2(n, \pi_{\alpha_k(n\dots n)}!)$ qui est récursive primitive et que l'ensemble des codes de structure I est primitif récurisif. On notera $\mathcal{M}_i$ la structure codée par $i \in I$.
De plus pour toute $\mathcal{L}$-formule $\varphi[\overline{x}]$, la fonction $f_\varphi(i,\vec{j})$ qui vaut 1 si $\mathcal{M}_i \vDash \varphi[\vec{j}]$ est primitive récursive. En effet, si φ est atomique, il suffit de vérifier que pour tout $\rho(\vec{j})$ divise $\pi_2^2(i)$. Il en découle immédiatement que c'est aussi vrai pour φ sans quantificateurs car les prédicats primitifs récursifs sont clos par combinaison booléenne et enfin pour φ quelconque car les prédicats primitifs récursifs sont clos par quantification bornée (ici par $\pi_1^2(i)$).
La fonction que l'on cherche est donc $\exists i \leqslant b(n) \bigwedge_{\varphi\in T} f_\varphi(i)$.

Exercice 7 (Ensembles des fonctions récursivement énumérables) **:**

1. C'est le même argument que dans le cas primitif récursif. Supposons que $\psi : \mathbb{N}^2 \to \mathbb{N}$ énumère les récursives totales. Comme ψ_i est récursive totale pour chaque i, $\lambda i\, \psi(i,i) + 1$ est récursive totale. Il existe donc i_0 telle que $\psi(i_0, x) = \psi(x,x) + 1$ et donc $\psi(i_0, i_0) = psi(i_0, i_0) + 1$, ce qui est absurde.

2. Soit ψ qui énumère les fonctions récursives. On définit alors $\theta(i,0) = \psi(i,0)$ et $\theta(i, x+1) = \max\{\theta(i,x) + 1, \psi(i, x+1)\}$. La fonction θ est bien récursive (totale) et les θ_i sont primitives récursives strictement croissante. De plus si ψ_i est strictement croissante, $\theta_i = \psi_i$. La fonction θ énumère donc bien toutes les fonctions primitives récursives strictement croissantes.

3. On définit $\zeta(i,0) = \psi(i,0)$ et $\zeta(i, x+1) = \psi(i, x+1)$ si $\psi(i, x+1) \notin \{\zeta(i,t) : t \leqslant x\}$ et sinon $\zeta(i, x+1) = \max\{\zeta(i,t) + 1 : t \leqslant x\}$. Cette fonction est récursive, chacune des ζ_i est primitive récursive injective et si ψ_i est primitive récursive injective, alors $\psi_i = \theta_i$.

4. Montrons d'abord le lemme suivant :

   Lemme 1 :

   Soit $\psi \in \mathcal{F}^2$ récursive telle que pour tout x, $A_x = \mathrm{Im}(\psi_x)$ est infini. Alors, il existe g récursive totale strictement croissante dont l'image n'est aucun des A_x.

   Démonstration. Soit g définie par $g(0) = 0$ et $g(x+1) = \psi(x, \mu t\, \psi(x,t) > g(x)) + 1$. Comme les A_x sont tous infinis, il est évident que g est récursive totale strictement croissante. De plus pour tout x, $y = \psi(x, \mu t\, \psi(x,t) > g(x)) \in A_x$ est tel que $g(x) < y < g(x+1)$ et donc $\mathrm{Im}(g)$ ne peut être égal à aucun des A_x. ∎

   On conclut alors immédiatement en remarquant que si $f \in \mathcal{F}^1$ est injective (a fortiori strictement croissante) son image est infinie. Par le lemme l'existence d'un énumération des fonctions récursives totales injection (respectivement strictement croissante) implique l'existence d'une fonction récursive totale strictement croissante (et donc injective) qui ne peut être énumérée.