Calculabilité

UJF — MINF

Corrigé de l'examen du 23 avril 2003

1 Analyse d'une fonction

– On formalise la définition:

$g(x) = \mu z < (x+2)^2.\Big(EstPremier(z) \wedge \neg\exists y < (x+2)^2\big(EstPremier(y) \wedge |x-y| < |x-z|\big)\Big)$

On a obtenu g à partir des fonctions et prédicats RP vus en TD en utilisant les opérations qui préservent la récursivité primitive. Donc g est **récursive primitive**.

– g **est récursive partielle**, parce que chaque fonction récursive primitive est récursive partielle.

– g **est calculable par une machine de Turing**, parce que par théorème d'équivalence chaque fonction récursive partielle est TM-calculable.

2 Analyse de décidabilité

On considère le prédicat $P(x) \equiv (\varphi_x(x^2) = x^6)$

1. P en français:
   La machine de Turing de no de Gödel x pour l'entrée x^2 s'arrête avec le résultat x^6
2. On prouve que P est indécidable en montrant que $K \leq_m P$.
   Soit g définie comme suit:
   $$g(x,y) = \begin{cases} y^3 & \text{si} \quad \varphi_x(x)\downarrow \\ \uparrow & \text{si} \quad \varphi_x(x)\uparrow \end{cases}$$
   Mais $g(x,y) = U(x,x)\cdot 0 + y^3$, donc g est récursive partielle et elle a un index c t.q. $g(x,y) = \varphi_c(x,y)$. Par théoreme s-m-n on a : $\varphi_c(x,y) = \varphi_{s(c,x)}(y) = \varphi_{h(x)}(y)$ (avec h réc. totale). On prouve maintenant que $K(x) \Leftrightarrow P(h(x))$. Effectivement:
   $$\begin{aligned} K(x) \quad &\Longleftrightarrow \varphi_x(x)\downarrow \\ &\Longrightarrow \forall y(g(x,y) = \varphi_{h(x)}(y) = y^3) \\ &\Longrightarrow \varphi_{h(x)}(h(x)^2) = h(x)^6 \\ &\Longrightarrow P(h(x)) \end{aligned}$$
   $$\begin{aligned} \neg K(x) \quad &\Longleftrightarrow \varphi_x(x)\uparrow \\ &\Longrightarrow \forall y\ g(x,y) = \varphi_{h(x)}(y)\uparrow \\ &\Longrightarrow \varphi_{h(x)}(h(x)^2)\uparrow \\ &\Longrightarrow \neg P(h(x)) \end{aligned}$$
   On voit que $K \leq_m P$ par h, et donc P est **indécidable**.
3. On peut représenter P comme suit:
   $$P(x) = \text{Résultat}(x,x^2,x^6),$$
   (où le prédicat r.e. Résultat$(x,y,z) \equiv (\varphi_x(y) = z)$ était défini en cours). On en déduit que P est **r.e.**
4. Par théorème de Post: P décidable $\Leftrightarrow$ P r.e. et $\overline{P}$ r.e. On vient de prouver que P n'est pas décidable et que P est r.e. On en déduit que $\overline{P}$ **n'est pas r.e.**

3 Machines multiplicatives

1. Une machine multiplicative qui à partir de chaque R initial de la forme $R = 2^n$ s'arrête avec $R = 5 \cdot 7^n$.

   init: **if** $R \bmod 2 = 1$ **goto** a **else** f
   a: $R := R/2$ **goto** b
   b: $R := R * 7$ **goto** init
   f: $R := R * 5$ **goto** s
   s: Stop

2. Pour prouver que $ArretMult$ est indécidable on va simuler la machine à 2 compteurs.
   Pour chaque machine C à 2 compteurs on construira une machine multiplicative $M(C)$ qui la simule.
   L'ensemble d'états de $M(C)$ sera le même que dans C, l'état initial aussi. On représentera une configuration (q,x,y) de C par la configuration $(q,2^x3^y)$ de $M(C)$.
   Pour chaque instruction de C la machine multiplicative $M(C)$ aura une instruction équivalente:

Machine à compteurs	Machine multiplicative
q: $x++$; **goto** p	q: $R := R * 2$; **goto** p
q: $x--$; **goto** p	q: $R := R/2$; **goto** p
q: **if** $x = 0$ **then goto** p **else goto** r	q: **if** $R \bmod 2 = 0$ **then goto** r **else goto** p
q: $y++$; **goto** p	q: $R := R * 3$; **goto** p
q: $y--$; **goto** p	q: $R := R/3$; **goto** p
q: **if** $y = 0$ **then goto** p **else goto** r	q: **if** $R \bmod 3 = 0$ **then goto** r **else goto** p
q: **stop**	q: **stop**

   Il est évident qu'à chaque transition (et à chaque calcul) de C correspond une transition (un calcul) de $M(C)$ et que les deux machines s'arrêtent au même temps.
   Par conséquent $Arret2C(C,x,y) \Leftrightarrow ArretMult(M(C),2^x3^y)$. On a donc effectué la réduction $Arret2C \leq_m ArretMult$. On sait que Arret2c est indécidable et on en déduit que ArretMult est aussi **indécidable**.
3. On va prouver qu'il n'existe pas de machine multiplicative qui à partir de chaque R initial s'arrête avec $R = 1$.
   L'idée informelle est qu'une telle machine devrait savoir diviser par tous les nombres premiers, ce qui est impossible pour une machine avec un nombre fini de constantes. Essayons de rendre formel ce raisonnement.
   Supposons qu'une telle machine existe. Elle utilise un nombre fini de constantes. Soit N la constante maximale utilisée, et p un nombre premier supérieur à N.
   Supposons que la machine démarre avec $R = p$. Par construction, la machine ne divise jamais par p (ni par un nombre multiple de p). Donc après chaque instruction le contenu de régistre R reste un multiple de p. Comme 1 n'est pas un multiple de p, on a prouvé que jamais pour ce calcul R ne contient 1. Contradiction.
   On a prouvé qu'une machine spécifiée dans l'énoncé **n'existe pas**.

4 Enumération monotone

– Soit A décidable.
  S'il est **fini**, soit $\{a_0,a_1,\ldots,a_n\}$ la liste de tous ses éléments dans l'ordre croissant. Alors on peut prendre g définie comme suit:

$$\begin{cases} g(0) = a_0 \\ g(1) = a_1 \\ \cdots \quad \cdots \quad \cdots \\ g(n) = a_n \\ g(x) = a_n \text{ pour } x > n \end{cases}$$

Cette fonction est croissante et $A = Im(g)$ par construction, sa définition par cas montre qu'elle est même récursive primitive.
Supposons maintenant que A est infini. Definissons g par

$$\begin{cases} g(0) & = \mu x.A(x) \\ g(n+1) & = \mu x.A(x) \wedge x > f(n) \end{cases}$$

g est une énumeration monotone (récursive totale) de A.

– On suppose que A est énumerable monotonement (et infini) et on prouve que A est décidable (si A est fini, il est trivialement décidable).
Par hypothèse il existe une fonction g récursive totale et croissante t.q. $A = Im(g)$. On remarque que comme A est infini, la fonction g est non-bornée.
L'idée informelle de l'algorithme qui décide si $x \in A$ est la suivante. On utilise g pour générer en ordre croissant les éléments de A. Il y a deux cas de figures possibles:
  – *à un certain moment x est généré. Dans ce cas on renvoie "Oui".*
  – *à un certain moment un entier supérieur à x est généré (sans que x aie été généré avant). Dans ce cas, comme g est monotone, on peut être sur que x ne sera jamais généré. Donc on renvoie "Non".*

Essayons de rendre formel cet algorithme de décision.
Soit $f(x) = \mu i.g(i) \geq x$. Cette fonction (qui correspond à l'endroit où la séquence croissante définie par g atteint ou dépasse la valeur x) est récursive partielle par définition, et définie partout comme g est non-bornée. Donc f est récursive totale.
On remarque maintenant que pour chaque x on a $A(x)$ si et seulement si $g(i) = x$ pour certain i, et cela si et seulement si $g(i) = x$ pour $i = f(x)$.
Finalement on a $A(x) \Leftrightarrow g(f(x)) = x$ et

$$\chi_A(x) = \chi_=(g(f(x)),x)$$

La fonction caractéristique de A est donc récursive totale et A est décidable. cqfd