Modèles de Calcul
Examen
L3 Informatique

03/05/2012

Durée : 3h00 sans document.

1 Machines de Turing : calculabilité

Dans cet exercice on considère que les prédicats (fonctions dont le résultat est un booléen) sont codés de la manière suivante: si A est un prédicat alors $A(x) = 1$ si $A(x)$ est vrai et 0 sinon.

On suppose pour les trois premières questions que le prédicat A se réduit à B, c'est à dire qu'il existe une fonction f totale et calculable telle que $A(x) = 1$ si et seulement si $B(f(x)) = 1$. On note E et F les ensembles suivants :

$$E = \{x \mid A(x) = 1\} \quad F = \{x \mid B(x) = 1\}$$

autrement dit les prédicats A, B sont les fonctions caractéristiques des ensembles E, F.

Q.1 Montrez que si F est décidable alors E l'est également.

Q.2 Montrez que si F est récursivement énumérable alors E l'est également.

Q.3 Expliquez comment formuler les résultats précédents lorsqu'on veut prouver qu'un ensemble est indécidable, ou non récursivement énumérable. (La question est volontairement vague, la réponse est très courte et vous servira pour résoudre les trois dernières questions de cet exercice).

On considère la procédure suivante :

```
int gq(int p (int),int x, int n)
{
  int y = p(x);
  return n;
}
```

On note $q = f(p, x)$ la fonction définie par $q(n) = gq(p, x, n)$.

Q.4 Montrez que f réduit le problème de la *terminaison* du calcul de $p(x)$ à celui de savoir si q calcule l'identité.

Q.5 En déduire que l'ensemble des procédures qui *ne calculent pas l'identité* n'est pas récursivement énumérable.

On remplace la procédure gq par une procédure gr, où la variable t représente le temps ; On suppose l'existence d'un prédicat h total et calculable tel que $h(p, x, t) = 1$ si et seulement si $p(x)$ termine en un temps plus petit que t.

```
int gr(int p (int), int x, int t)
{
  if (h(p,x,t)) return p(x);
     else return t;
}
```

On note $r = g(p, x)$ la fonction définie par $r(t) = gr(p, x, t)$.

Q.6 Montrer que g réduit le problème de la terminaison du calcul de $p(x)$ à celui de savoir si r *ne calcule pas* l'identité.

Q.7 En déduire que l'ensemble des procédures qui calculent l'identité n'est pas récursivement énumérable.

2 Machine de Turing : Programmation

Pour chaque machine de Turing construite il faut donner une explication de son principe en plus de sa table de transition. Une machine comportant la moindre faute sans explications ne sera pas comptabilisée.

Vous pouvez introduire autant de symboles que vous voulez pour l'alphabet de bande, cette dernière sera considérée comme infinie dans les deux sens.

Q.8 Construire une machine de Turing qui reconnaisse le langage $L = \{ww \mid w \in \{0,1\}^*\}$.

Q.9 Construire une machine de Turing qui prend en entrée deux entiers écrits en binaires et s'arrête avec sur le ruband la somme en binaire de ces deux entiers. Par exemple si le ruband d'entré est $1011B101$ avec la machine qui pointe sur le premier 1 alors en sortie le ruban doit uniquement contenir le mot 10000.

3 λ-calcul : Programmation

Pour programmer il vous est conseillé d'introduire des macros pour les fonctions intermédiaires et de les utiliser ensuite pour éviter d'avoir un programme illisible.

Rappels

Les booléens sont définis par $\texttt{true} \equiv \lambda xy.x$ et $\texttt{false} \equiv \lambda xy.y$. Le if est codé par $\texttt{if} \equiv \lambda z.(z\ M\ N)$
Les couples sont égalements encodés par $[M, N] \equiv \lambda z.(z\ M\ N)$.
Les entiers peuvent être codés par :

Church	Barendregt
$\lceil 0 \rceil \equiv \lambda fx.x$	$\lfloor 0 \rfloor \equiv \lambda x.x$
$\lceil n \rceil \equiv \lambda fx.(\overbrace{f\ \ldots (f}^{n}\ x))$	$\lfloor n \rfloor \equiv [\texttt{false}, \lfloor n-1 \rfloor]$
$0_c? \equiv \lambda n.(n\ \lambda x.\texttt{false}\ \texttt{true})$	$0_b? \equiv \lambda x.(x\ \texttt{true})$

On considère que les fonctions de base suivantes **Add, Mult, Div, Sub** codants respectivement l'addition, la multiplication, la division et la soustraction ainsi que l'égalité, notée **Eg** sur les entiers sont données.

De plus on rappelle que le combinateur de point fixe Y est défini par $Y \stackrel{\text{def}}{=} \lambda f.(\lambda x.(f\ (x\ x))\ \lambda x.(f\ (x\ x)))$ et qu'il a la propriété suivante : $(Y\ M) \rightarrow_\beta^* (M\ (Y\ M))$.

On considère également les fonctions sur les listes dont on rappelle le codage en λ-calcul : une liste $[1;2;3;4]$ est codée par le terme $(cons\ \lceil 1\rceil\ (cons\ \lceil 2\rceil\ (cons\ \lceil 3\rceil\ (cons\ \lceil 4\rceil\ Nil))))$ où

$$\begin{array}{lcl} cons & \stackrel{\text{def}}{=} & \lambda elc.(c\ e\ l) \\ Nil & \stackrel{\text{def}}{=} & \lambda l.\texttt{true} \\ head & \stackrel{\text{def}}{=} & \lambda l.(l\ \texttt{true}) \\ tail & \stackrel{\text{def}}{=} & \lambda l.(l\ \texttt{false}) \\ Null? & \stackrel{\text{def}}{=} & \lambda l.(l\ \lambda ht.\texttt{false}) \end{array}$$

Q.10 Ecrire un prédicat P qu'il faut appliquer à une liste d'entiers. Si, et uniquement si, la liste l est telle qu'il existe une liste l' et l est la concaténation de l' avec l' alors $(P\ l)$ se réduit vers `true` et sinon vers `false` (il s'agit de la version λ-calcul de la question **Q.8**).

Q.11 Ecrire un combinateur d'addition binaire Adb qui prend en entrée deux listes d'entiers de Church représentants des nombres écrits en binaire et qui se réduit vers leur somme écrite en binaire (il s'agit de la version λ-calcul de la question **Q.9**). On doit donc avoir le λ-terme suivant :

$$(Adb\ (cons\ \lceil 1\rceil\ (cons\ \lceil 0\rceil\ (cons\ \lceil 1\rceil(cons\ \lceil 1\rceil\ \ Nil))))\ (cons\ \lceil 1\rceil\ (cons\ \lceil 0\rceil\ (cons\ \lceil 1\rceil\ Nil))))$$

qui se réduit sur

$$(cons\ \lceil 1\rceil\ (cons\ \lceil 0\rceil\ (cons\ \lceil 0\rceil\ (cons\ \lceil 0\rceil\ (cons\ \lceil 0\rceil\ Nil)))))$$

4 λ-calcul : Problème

On définit la réduction parallèle, notée $\triangleright$ sur les λ-termes par :

- $u \triangleright u$ pour tout u
- Si $u \triangleright u'$ et $v \triangleright v'$ alors $(u\ v) \triangleright (u'\ v')$
- Si $u \triangleright u'$ alors $\lambda x.u \triangleright \lambda x.u'$
- Si $u \triangleright u'$ et $v \triangleright v'$ alors $(\lambda x.u\ v) \triangleright u'[x := v']$.

Q.12 Montrez que si $u \triangleright u'$ et $v \triangleright v'$ alors $u[x := v] \triangleright u'[x := v']$.

Q.13 Montrez que si $u \triangleright v$ et $u \triangleright w$ alors il existe t tel que $v \triangleright^* t$ et que l'on ait soit $w \triangleright t$ soit $w \equiv t$.

Q.14 Montrez que $\rightarrow_\beta \subset \triangleright \subset \rightarrow_\beta^*$ (montrez qu'il s'agit d'inclusion stricte).

Q.15 En conclure que $\rightarrow_\beta$ est confluente (c'est à dire que pour tout $u \rightarrow_\beta v$ et $u \rightarrow_\beta w$ il existe un terme t tel que $v \rightarrow_\beta^* t$ et $w \rightarrow_\beta^*$).