TD Fonctions récursives

Olivier Raynaud (raynaud@isima.fr)

FIGURE 1 – L'escalier de Penrose Film Inception https ://www.youtube.com/watch ?v=n2funNjIXhY

Question 1. *Montrer que les fonctions suivantes sont récursives primitives.*

1. *La fonction prédécesseur définie par* $pred(n) = max(0, n-1)$;
2. *La fonction différence définie par* $sous(n,m) = n - m$ *si* $n \geq m$ *et* 0 *sinon ;*
3. *Les fonctions* $min(n,m)$ *et* $max(n,m)$;
4. *La fonction égalité à* 0 *définie par* $egal_0?(n) = 1$ *si* $n = 0$ *et* 0 *sinon ;*
5. *La fonction égalité définie par* $egalite?(m,n) = 1$ *si* $m = n$ *et* 0 *sinon ;*
6. *La fonction* $pos?()$ *où* $pos?(n)$ *vaut* 1 *si* $n > 0$ *et* 0 *sinon ;*
7. *La fonction* $sup?()$ *où* $sup?(n,m)$ *vaut* 1 *si* $n > m$ *et* 0 *sinon ;*
8. *La fonction* $prod()$ *où* $prod(n,m)$ *est le produit de* n *par* m.
9. *La fonction* $div()$ *où* $div(m,n)$ *est le quotient de la division euclidienne de* m *par* n ;
10. *La fonction* $mod()$ *où* $mod(m,n)$ *est le reste de la division euclidienne de* m *par* n ;
11. *La fonction puissance définie par* $puis(n,m) = m^n$;
12. *La fonction* $rac(n,m)$ *où* $rac(n,m)$ *est le plus grand entier* k *tel que* k^m *est égale à* n.
13. *La fonction* $log()$ *où* $log(n,m)$ *est le logarithme en base* m *de* n, *c'est à dire le plus petit entier* k *tel que* $m^i \leq k$;
14. *La fonction* $premier?()$ *où* $premier?(p) = 1$ *si* p *est premier et* 0 *sinon ;*
15. *La fonction* $val()$ *où* $val(n,m)$ *est le plus grand entier* k *tel que* m^k *divise* n.

Exercice 1 (Prédicats)**.**
On dit que $A \subseteq \mathbb{N}$ est un ensemble récursif si sa fonction caractéristique est récursive, c'est à dire si la fonction $\chi_A : \mathbb{N} \to \mathbb{N}$ est récursive où χ_A est définie par $\chi_A(n) = 1$ si $n \in A$, et 0 sinon. On dit qu'un prédicat logique P d'arité p est un prédicat récursif si l'ensemble des $p-$uplets vérifiant P est récursif (on identifie alors P à ce sous-ensemble de $\mathbb{N}^p$).

Question 1. Clôture par les opérateurs booléens
Soient Aet B deux ensembles récursifs, montrer que les ensembles $A \cap B$, $A \cup B$ et A^C sont récursifs.

Question 2. Clôture par le schéma de définition par cas.
Montrer que si les ensembles disjoints $A_1, ..., A_k$ sont récursifs et que si les fonctions $f_1, ..., f_k$ et f sont récursives, alors la fonction g définie par :

$$g(x) = \begin{cases} f_1(x) \text{ si } x \in A_1 \\ ... \\ f_k(x) \text{ si } x \in A_k \\ f(x) \text{ sinon} \end{cases}$$

est récursive.

Question 3. Clôture par quantification bornée.

1. *Montrer que si $f : \mathbb{N} \Longrightarrow \mathbb{N}$ est récursive, alors la somme bornée $S_f(n) = \sum_{i=0}^{n} f(i)$ et le produit borné $\prod_{i=0}^{n} f(i)$ sont récursifs.*
2. *En déduire que si le prédicat unaire $P \subseteq \mathbb{N}$ est récursif, alors les prédicats définis par $E(n) := (\exists i \leq n)P(i)$ et $F(n) := (\forall i \leq n)P(i)$ sont récursifs.*

Question 4. Clôture par minimisation bornée.
On définit la fonction g par schéma de minimisation bornée par :

$$g(x) = \begin{cases} min\{y \leq x \mid f(x,y) = 0\} \text{ si il existe un tel } y \\ 0 \text{ sinon} \end{cases}$$

Montrer que l'ensemble des fonctions récursives primitives est clos par minimisation bornée.

Question 5. *Montrer que tout ensemble fini A est récursif. Montrer que pour k fixé, le prédicat P_k défini par 'être multiple de k' est récursif.*

Question 6. *Montrer que la fonction p qui associe à n le $(n+1)$-unième nombre premier est récursive primitive. En déduire que la fonction G_k de codage des suite de longueur k définie par : $G_k(n_1, n_2, ..., n_k) = p_1^{n_1+1}.p_2^{n_2+1}. \ ... \ .p_k^{n_k+1}$ est récursive primitive.*

Exercice 2 (Racines de polynôme)**.** *Soit g_i la fonction partielle de $\mathbb{N}^{2k}$ dans $\mathbb{N}$ définie par :*
Soit $0 \leq b_1 \leq b_2 \leq ... \leq b_l$ les racines entières positives du polynôme $P(X) = \sum_{j=1}^{k} (-1)^{\epsilon_j} a_j X^j$.
Alors $g_i(a_1, \epsilon_1, a_2, \epsilon_2, ..., a_k, \epsilon_k)$ est définie si et seulement si $\forall j$, $\epsilon \in \{0, 1\}$ et $i \leq l$, et vaut dans ce cas b_i.
En d'autres termes, les entrées de g_i correspondent aux coefficients entiers d'un polynôme de degré au plus k (+ une façon de coder les entiers négatifs) et g_i est la plus petite racine entière positive du polynôme si elle existe. Montrer que g_i est récursive.

Exercice 3. *Numérotation de Godël*

Considérons un codage G de l'ensemble des fonctions μ-récursives REC dans $\mathbb{N}$. Pour ce codage chacune des fonctions $cst()$ et $succ()$ est codée par un couple dont la première valeur est le type de la fonction et la seconde son arité. Les fonctions de projections sont codées par un triplet composé du type et des indices de projection.

- *Fonction nulle k-aire : $G(cst()) = (1, k)$;*
- *Fonction successeur unaire : $G(succ()) = (2, 1)$; La fonction succ est d'aritÈ 1.*
- *Fonction de projection $k - aire$: $G(\pi_i^k) = (3, k, i)$.*

Les opérateurs de composition, de récursion et de minimisation qui permettent de construire de nouvelles fonctions à partir des premières sont codées de la façon suivante :

- *Composition de g k-aire avec $h_1, h_2, ..., h_k$*
  $G(Comp, g, h_1, ..., h_k) = (4, k, G(g), G(h-1), ..., G(h_k))$
- *RÈcursion primitive $(k+1)$-aire de g k-aire avec h $(k+2)$-aire*
  $G(RecursionPrimitive, g, h) = (5, k+1, G(g), G(h))$
- *Minimisation k-aire sur f $(k+1)$-aire*
  $G(Minimisation, f) = (6, k, G(f))$

Considérons maintenant la fonction $\mathcal{G}$ qui associe un entier naturel unique à chaque uplet. Cette fonction agit sur les couples et triplet associés aux fonctions de base de la manière suivante :

$\mathcal{G} : (u_1, u_2) \to mult(exp(premier(1), u_1{+}1), exp(premier(2), u_2{+}1)) \to p_1^{u_1+1}.p_2^{u_2+1} \to 2^{u_1+1}.3^{u_2+1}.$

Nous pouvons illustrer la définition de la fonction $\mathcal{G}$ pour les opérateurs de composition, de récursion et de minimisation à partir de l'exemple suivant :
pour la récursion primitive $(5, k+1, G(g), G(h))$:

$\mathcal{G} : (4, k+1, G(g), G(h)) \to p_1^{4+1}.p_2^{(k+1)+1}.p_3^{\mathcal{G}(G(g))+1}.p_4^{\mathcal{G}(G(h))+1} \to 2^5.3^{k+2}.5^{\mathcal{G}(G(g))+1}.7^{\mathcal{G}(G(h))+1}.$

Théoréme 1 (Numérotation de Godël)**.** *La fonction $\mathcal{G}$, de l'ensemble des fonctions $\mu-$ récursives dans $\mathbb{N}$, est une bijection primitive récursive.*

Question 1. *Calculer les nombres de Godël des fonctions données à la question de l'exercice précédent.*

Question 2. *Démontrer le Théorème 1.*

Exercice 4 (Argument diagonal)**.** *Existe-t-il des fonctions non récursives primitives ?*
Les fonctions récursives primitives sont telles que nous pouvons prédire à l'avance la durée des calculs. Dans cet exercice nous allons montrer qu'il existe des fonctions non récursives primitives.

Définition 1. *Nous appelons langage Bucle le Langage de Description d'Algorithme (L.D.A.) duquel nous avons supprimé la boucle 'TantQue'.*

Considérons l'ensemble $\mathcal{L}_B$ de tous les programmes, cet ensemble est infini. Soit le sous ensemble ${\mathcal{L}'}_B$ de tous les programmes sans appel, écrits en Bucle, calculant des fonctions à un seul argument.

Question 1. *Dire si les ensembles $\mathcal{L}_B$ et ${\mathcal{L}'}_B$ sont dénombrables ou non. Proposer une méthode de construction d'un annuaire de ${\mathcal{L}'}_B$. Ecrivez le premier programme de votre annuaire.*

Nous disposons maintenant d'un annuaire des programmes en Bucle. Dans la suite nous appellerons $programme\{nx(n)\}$ le $x$ième programme de notre annuaire avec l'entrée entière n.

Question 2. *En utilisant l'argument de Cantor montrer qu'il existe une fonction qui n'appartient pas à votre annuaire bien que se soit une fonction à 1 variable parfaitement définie et calculable. En utilisant un autre argument qu'est ce qui assure, dans la définition donnée de la fonction $diagonale(n)$, que cette fonction n'est pas récursive primitive ?*

Exercice 5. *Le langage Mucle*

Définition 2 (Le langage Mucle)**.**
Le langage Mucle est identique au Bucle, à un point prés : des boucles avec et sans plafond y sont admises. Ce nouveau type de boucle s'appellera une **boucle** μ*.*

Les programmes en Mucle ont l'inconvénient majeur de ne pas toujours se terminer. Cela nous amène a créer deux classes de programmes :

1. *les programmes en Mucle qui se terminent toujours appelés les programmes terminateurs*
2. *les programmes en Mucle pour lesquels il existe au moins une entrée qui fait qu'ils ne s'arrêtent pas, ce sont les non terminateurs.*

Dans cet exercice nous nous intéressons à l'étude d'une procédure qui permettra de dire à quelle classe appartient un programme en Mucle donné :

terminateur(p) *:* **Oui** *si* p *s'arrête toujours,* **Non** *dans le cas contraire ;*

Question 1. *Quelle serait la conséquence de l'existence la procédure terminareur*$()$ *? sur l'étude des conjectures comme celle de la La Tortue ou celle de Syracuse ?*

Question 2. *Soit* ${\mathcal{L}'}_M$ *l'annuaire de tous les programmes écrits en Mucle calculant des fonctions à un seul argument entier. Dire pourquoi une fonction diagonale*(n) *définie sur* $\mathcal{L}_M$ *serait mal définie. Proposer alors un nouveau filtre à votre annuaire.*

Question 3. *Construisez une fonction à une variable, parfaitement définie qui ne soit pas dans votre annuaire. Conclure.*

Question 4. *Existe-t-il des langages plus puissant que le langage Mucle ?*