Master Info - 2014-2015
MIF15 Complexité et Calculabilité

Contrôle continu 2
éléments de corrigé par P. Brunet et L. Gonnord

Durée 1H
Tous documents papiers et électroniques interdits.
Le barême est donné à titre **indicatif**.

Préliminaire : rappels sur les fonctions primitives récursives

Une fonction $f : \mathbb{N}^n \to \mathbb{N}$ est *récursive primitive* si elle est :

— Une des fonctions renvoyant 0 : $\mathsf{zero_n} : (x_1 \dots x_n) \mapsto 0$

— $\mathsf{succ} : x \mapsto x + 1$ la fonction successeur (alors $n = 1$) ;

— $\mathsf{id}^i_n : (x_1, \dots, x_n) \mapsto x_i$ les fonctions de projection, pour $1 \leq i \leq n$;

et stable par les opérations de base :

— $\mathsf{Comp}_n(g, h_1, \dots, h_m) : (x_1, \dots, x_n) \mapsto g(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n))$ la composition

— $\mathsf{Rec}(g, h)$ la fonction définie par récurrence comme

$$\begin{cases} f(x_1, \dots, x_{n-1}, 0) = g(x_1, \dots, x_{n-1}), \\ f(x_1, \dots, x_{n-1}, m+1) = h(x_1, \dots, x_n, m, f(x_1, \dots, x_n, m)), \end{cases}$$

1 Prouvons des choses élémentaires

Question 1 (1 point)
Montrer que l'on pourrait se restreindre à uniquement la fonction constante 0 d'arité 0 dans la définition des primitives récursives, c'est-à-dire que l'on peut construire $\mathsf{zero_n}$.

Solution:
Fixons n. La fonction $\mathsf{zero_n}$ peut aussi se définir par : $\mathsf{Comp_n}(\mathsf{zero_0})$.

Question 2 (1 point)
Montrer que si $f : \mathbb{N}^2 \to \mathbb{N}$ est primitive récursive, alors $\mathsf{Swap(f)} = g$ définie par $g(x, y) = f(y, x)$ est aussi primitive récursive.

Solution:
En utilisant les projections, on a $x = \mathsf{id}^2_2(x, y)$ et $y = \mathsf{id}^2_2(x, y)$ donc,

$$\mathsf{Swap(f)}(x, y) = g(x, y) = f(\mathsf{id}^1_2(x, y), \mathsf{id}^2_2(x, y))$$

Autrement dit :

$$\mathsf{Swap(f)} = \mathsf{Comp}_2(f, \mathsf{id}^1_2, \mathsf{id}^2_2)$$

Question 3 (3 points)

Montrer directement à l'aide des définitions uniquement (zéro, id, succ, Comp et Rec), et éventuellement la fonction Swap, que les fonctions suivantes sont primitives récursives :

— $\mathsf{pred}(\mathsf{n})$ qui vaut 0 si $n = 0$ et $n - 1$ sinon.

— $\mathsf{vabs}(\mathsf{n})$ valeur absolue de la différence.

— $\mathsf{eq}(\mathsf{m}, \mathsf{n})$ qui vaut 1 si $m = n$ et 0 sinon.

Solution:

— La fonction prédécesseur est définie par :

  — $\mathsf{pred}(0) = 0$

  — $\mathsf{pred}(m + 1) = m$

  Ce qui montre directement que $\mathsf{pred} = \mathsf{Rec}(\mathsf{zero}_0, \mathsf{id}_2^1)$.

— La fonction "valeur absolue de la différence" est une fonction de $\mathbb{N}^2$ vers $\mathbb{N}$ qui peut être définie par :

  — $\mathsf{vabs}(m, n) = m - n$ si $m \geq n$

  — $\mathsf{vabs}(m, n) = n - m$ si $m \leq n$

  Pour l'instant cela ne correspond pas tout à fait à la définition, on va donc plutôt utiliser la somme de deux différences tronquées :

$$\mathsf{difftronc}(m, n) = m - n \text{ si } m \geq n, 0 \text{ sinon.}$$

  Cette fonction auxiliaire peut se calculer par récurrence :

  — $\mathsf{difftronc}(m, 0) = m$

  — $\mathsf{difftronc}(m, n + 1) = \mathsf{pred}(\mathsf{difftronc}(m, n))$

  donc $\mathsf{difftronc} = \mathsf{Rec}(\mathsf{id}_1^1, \mathsf{Comp}(\mathsf{pred}, \mathsf{id}_2^1))$ Ensuite,

$$\mathsf{vabs}(m, n) = \mathsf{difftronc}(m, n) + \mathsf{difftronc}(n, m)$$

  Il faudrait en toute rigueur montrer en plus que la fonction + est primitive récursive, et le lecteur est reporté à son cours. Ensuite, une composition et un swap plus tard, on obtient ce que l'on veut.

— Ensuite, l'égalité $\mathsf{eq}(m, n)$ ssi $\mathsf{vabs}(m, n) = 0$. Il reste à définir un prédicat d'égalité à 0, laissé au lecteur.

2 Moins élémentaire

À partir de cette étape, on pourra écrire des définitions de fonctions primitives récursives par cas (si les cas sont des tests primitifs récursifs) et en utilisant des récursions qui font strictement décroître un certain paramètre (en montrant que la récursion termine). Vous n'avez en revanche pas le droit d'utiliser le théorème de récursion bornée [1] vu en TD.

Question 4 (3 points)

On ne demande pas de montrer que mod et pow sont primitives. Montrer que la fonction

1. Rappel : pour une fonction μ-récursive f, s'il existe une fonction primitive récursive b telle que $\forall n, \mathsf{f}(\mathsf{n}) \leq \mathsf{b}(\mathsf{n})$, alors f est primitive récursive

$\mathsf{sqr}(\mathsf{m}, \mathsf{n})$ racine n-ième de m, est primitive récursive. On commencera par définir la racine carrée comme fonction entière, on généralisera à la racine n-ième, avant toute tentative de résolution de l'exercice.

Solution:
La racine n-ième de m ($\mathsf{sqr}(m, n)$) est le plus grand entier k tel que $k^n \leq m$. Elle peut être définie comme cela :

- $\mathsf{sqr}(0, n) = 0$
- $\mathsf{sqr}(m + 1, n) = \mathsf{sqr}(m, n) + quelquechose$

Pour savoir comment exprimer ce quelquechose, on va regarder la racine carrée de m (donc $n = 2$) :

- La racine carrée entière de 5 est 2, la racine carrée entière de 4 est 2 aussi. Dans cet exemple, le quelquechose vaut 0.
- La racine carrée de 3 est égale à 1. On a donc ici $\mathsf{sqr}(m + 1, 2) = \mathsf{sqr}(m, 2) + 1$.

En regardant bien, on doit ajouter 1 si en élevant la quantité obtenue par récursion $\mathsf{sqr}(m, 2)$ au carré, cette expression est égale à $m + 1$. En d'autres termes :

- $\mathsf{sqr}(0, 2) = 0$
- $\mathsf{sqr}(m + 1, 2) = \mathsf{sqr}(m, 2) + \mathsf{eq}(\mathsf{pow}(1 + \mathsf{sqr}(m, 2), 2), m + 1)$

Dans le cas général on peut extrapoler un peu et on trouve :

- $\mathsf{sqr}(0, n) = 0$
- $\mathsf{sqr}(m + 1, n) = \mathsf{sqr}(m, n) + \mathsf{eq}(\mathsf{pow}(1 + \mathsf{sqr}(m, n), n), m + 1)$

avec eq primitive récursive. Cette expression permet de conclure. On peut aussi écrire une définition par cas au lieu d'utiliser le prédicat eq.

Rédaction Alternative
On peut construire $\mathsf{sqrt}(m, n) = h(m, n, m)$ avec :

$$\begin{aligned} h(m, n, 0) =& \mathsf{zero}_0 \\ h(m, n, p + 1) =& \text{ if } \mathsf{pow}(p, n) < n \\ & \text{ then } \mathsf{succ}(p) \\ & \text{ else } h(m, n, p) \end{aligned}$$

Question 5 (2 points)
(Indépendante de la précédente) Montrer que si $g(x, y)$ est primitive récursive alors $S_g(z, y) = \sum_{x=0}^{z} g(x, y)$ l'est.

Solution:
C'est un cas typique de programmation récursive :

- si $z = 0$ le résultat est $g(0, y)$.
- pour $z + 1$ il faut ajouter $g(z + 1, y)$ au résultat en z.

Ainsi :

$$\begin{cases} S_g(0, y) = g(0, y) \\ S_g(z + 1, y) = g(z + 1, y) + S_g(z, y) \end{cases}$$

Clairement, il s'agit d'une récurrence (dans l'ordre inverse de la définition). On définit une fonction auxiliaire avec les paramètres inversés : $S'_g = \mathsf{Swap}(S_g)$, c'est à dire qu'on a

$S'_g(y,0) = g(0,y)$ et $S'_g(y,z+1) = g(z+1,y) + S'_g(y,z)$. On va donc fixer $S'_g = \mathsf{Rec}(gg, hh)$ avec :

— $gg : y \mapsto g(0,y)$.

— $hh : (y,z,t) \mapsto g(z+1,y) + t$.

On a donc $gg = \mathsf{Comp}_1(g, \mathsf{zero}_1, \mathsf{id}^1_1)$, pour hh il faut encore bosser un peu $hh = \mathsf{Comp}_3(+, uu, \mathsf{id}^3_3)$ avec $uu : (y,z,t) \mapsto g(z+1,y)$, ie $uu = \mathsf{Comp}_3(g, tt, \mathsf{id}^1_3)$, et finalement $tt : (y,z,t) \mapsto z+1$, c'est-à-dire
$tt = \mathsf{Comp}_3(\mathsf{succ}, \mathsf{id}^2_3)$. Pour résumer, on obtient :

$$hh = \mathsf{Comp}_3(+, \mathsf{Comp}_3(g, \mathsf{Comp}_3(\mathsf{succ}, \mathsf{id}^2_3), \mathsf{id}^1_3), \mathsf{id}^3_3),$$

du coup, comme $S_g = \mathsf{Swap}(S'_g)$:

$$S_g = \mathsf{Swap}(\mathsf{Rec}(\mathsf{Comp}_1(g, \mathsf{zero}_1, \mathsf{id}^1_1), \mathsf{Comp}_3(+, \mathsf{Comp}_3(g, \mathsf{Comp}_3(\mathsf{succ}, \mathsf{id}^2_3), \mathsf{id}^1_3), \mathsf{id}^3_3)))$$

Question 6 (3 points)

Montrer que si $R(x,y)$ est un prédicat primitif récursif, alors le prédicat $P(z,y) = \exists x \leq z R(x,y)$ l'est aussi. *On construira judicieusement un prédicat g pour la question précédente, à partir de R.*

Solution:

Transformer une somme en un quantificateur existentiel n'est pas difficile si on se souvient que le "+" booléen se comporte comme un "ou". Dans la question précédente, si on remplace g par R, la fonction S_R compte le nombre de x plus petits que z qui satisfont R. Il suffit donc de vérifier ensuite que S_R est non nul.

Soit donc le prédicat $\mathsf{NonNul}(x)$ qui est vrai si $x \neq 0$. La preuve que ce prédicat est primitif récursif est laissé au lecteur. Le prédicat P est donc la composition de NonNul avec le prédicat S_R de la question précédente.

Rédaction Alternative

On s'arrête dans la récursion dès qu'on a trouvé un z satisfaisant $R(z,y)$) :

$$\begin{cases} P(0,y) = R(0,y) \\ P(z+1,y) = R(z+1,y) \text{ ou } P(z,y) \end{cases}$$

Question 7 (4 points)

En utilisant la question précédente, montrer que prédicat "x est un nombre premier" est récursif primitif. *Indication : on construira en fait le prédicat* Composite *qui dit si x possède un diviseur.*

Solution:

Voici les étapes :

— Divisibilité : $\mathsf{div}(x,z) = \exists y \leq x, yz = x$.

— Nombre composite : $\mathsf{Composite}(x) = \exists y \leq x, y > 1 \wedge y < x \wedge \mathsf{div}(x,y)$.

— Nombre premier : $\mathsf{Prime}(x) = \neg\mathsf{Composite}(x)$.

3 Une grammaire pour finir

Question 8 (3 points)

Soit $\Sigma = \{a, b, c\}$. Écrire une grammaire (générale) qui génère les expressions "à la Lisp" :

- Chaque élément de Σ est une expression
- si $e_1, e_2, \ldots e_k (k \geq 2)$ sont des expressions, alors $(+e_1, e_2, \ldots e_k)$ et $(\times e_1, e_2, \ldots e_k)$ sont des expressions.

Donner un exemple de dérivation.

Solution:

Une grammaire hors contexte suffit :

```
E -> a | b | c | (+ E E LE) | (* E E LE)
LE -> eps | E LE
```

L'idée est que chaque opération est "au moins binaire", donc on force cela dans la grammaire.