TD de calculabilité, fonctions récursives primitives

Fonctions récursives primitives et récursives partielles

Exercice 1.
Montrer que les fonctions suivantes sont récursives primites.

1. *la fonction* Z $\begin{cases} \mathbb{N} \to \mathbb{N} \\ x \mapsto 0 \end{cases}$;
2. *les fonctions constantes égales à* 3 *d'arité* 0, 1 *et* 2 ;
3. *la fonction addition :* $(x, y) \mapsto x + y$;
4. *la fonction multiplication :* $(x, y) \mapsto x \times y$;
5. *la fonction exposant :* $(x, y) \mapsto x^y = x \uparrow y$;
6. *la fonction prédecesseur* $\text{Pred} : x \mapsto \begin{cases} x - 1 \ si\ x > 0 \\ 0 \ sinon ; \end{cases}$
7. *la fonction différence tronquée* $\text{Diff} : (x, y) \mapsto x \dot{-} y = \begin{cases} x - y \ si\ x \geq y \\ 0 sinon ; \end{cases}$

Exercice 2.
Montrer que le prédicat $x > y$ *est récursif primitif.*

Exercice 3.
Montrer que les constructeurs suivants sont récursifs primitifs (c'est à dire que s'ils sont utilisés sur des fonctions récursives primitives du bon type alors les fonctions qu'ils construisent sont bien récursives primitives).

1. *Pour* $k \in \mathbb{N}$, *si* P *est un prédicat d'arité* k, f *et* g *sont des fonctions d'arité* k, *alors* $\text{Si}(P, f, g)$ *est la fonction* h *d'arité* k *telle que lorsque* P *est vrai* $h = f$ *et lorsque* P *est faux* $h = g$.
2. *Pour* $k \in \mathbb{N}$, *si* f *est une fonction d'arité* $k + 1$ *alors* $\text{Somme}(f)$ *est la fonction* h *d'arité* $k + 1$ *telle que*
$$h(y, x_1, \ldots, x_k) = \sum_{i=0}^{y} f(i, x_1, \ldots, x_k)$$
3. *Pour* $k \in \mathbb{N}$, *si* f *est une fonction d'arité* $k + 1$ *alors* $\text{Produit}(f)$ *est la fonction* h *d'arité* $k + 1$ *telle que*
$$h(y, x_1, \ldots, x_k) = \prod_{i=0}^{y} f(i, x_1, \ldots, x_k)$$

Exercice 4.
Montrer que les constructeurs suivants sont récursifs primitifs :

1. *Minimisation bornée, version TD. Pour* $k \in \mathbb{N}$, *si* P *est un prédicat d'arité* $k+1$ *alors* $\mu_{\leq}(f)$ *est la fonction* h *d'arité* $k + 1$ *telle que*
$$h(y, x_1, \ldots, x_k) = \begin{cases} \text{Le plus petit } i \leq y \text{ tel que } P(i, x_1, \ldots, x_k) \text{ s'il en existe un} \\ y + 1 \text{ sinon.} \end{cases}$$
On peut faire l'analogie avec le code C suivant :

```
unsigned h(unsigned y, unsigned x1, ..., unsigned xk)
{
  int i;
  for (i = 0; i <= y; i++) {
    if (P(i, x1, ..., xk)) break;
  }
  return i;
}
```

2. *Minimisation bornée, version cours. Sous les mêmes hypothèses $\mu^t(h)$ est la fonction d'arité k (et non $k+1$) telle que*

$$h(x_1, \ldots, x_k) = \begin{cases} \textit{Le plus petit } i \leq t \textit{ tel que } P(i, x_1, \ldots, x_k) \textit{ s'il en existe un} \\ 0 \textit{ sinon.} \end{cases}$$

On peut faire l'analogie avec le code C suivant :

```
unsigned h(unsigned x1, ..., unsigned xk)
{
  int i;
  for (i = 0; i <= t; i++) { /* t constante prédéfinie */
    if (P(i, x1, ..., xk)) return i;
  }
  return 0;
}
```

Minimisation généralisée. Le constructeur de suivant n'est pas récursif primitif. Pour $k \in \mathbb{N}$, si P est un prédicat d'arité $k+1$ alors $\mu(P)$ est la fonction h d'arité k telle que

$$h(x_1, \ldots, x_k) = \begin{cases} \text{Le plus petit } i \text{ tel que } P(i, x_1, \ldots, x_k) \text{ s'il en existe un} \\ \uparrow \text{ (la divergence) sinon.} \end{cases}$$

On ajoute la minimisation générale aux constructeurs des primitives récursives, ceci forme la classe des fonctions récursives (partielles).

Exercice 5.
Démontrer que les fonctions suivantes sont récursives partielles :

1. *la division entière $\lfloor x/y \rfloor$ (question subsidiaire : en quels points cette fonction n'est pas définie ? Pouvez vous l'étendre en une fonction récursive primitive ?)*
2. *la racine carré entière $\lfloor \sqrt{x} \rfloor$.*
3. *Le prédicat de disibilité $x|y$ (x divise y) est-il récursif partiel ? Récursif primitif ?*

Corrigé

Correction de l'exercice 1.

1. On imagine bien que l'on a besoin d'utiliser la fonction constante Zero égale à 0 d'arité 0. Mais la réponse reste difficile. Cela tient au fait que les projections ne servent à rien pour passer de l'arité 1 à l'arité 0. On est obligé d'utiliser RP qui permet en particulier de mélanger une fonction d'arité 0 avec une fonction d'arité 2 pour obtenir une fonction d'arité 1.
   On pose $\mathrm{Z} = \mathrm{RP}(f, g)$. On résout alors en f (d'arité 0) et g (d'arité 2), le système obtenu à partir de la définition de RP :
   $$\begin{cases} \mathrm{Z}(0) = f() \\ \mathrm{Z}(n+1) = g(n, \mathrm{Z}(n)) \end{cases}$$
   Comme Z doit être partout nulle, on en déduit que $f = \mathrm{Zero}$ et $g = p_2^2$ conviennent. Ainsi : $\mathrm{Z} = \mathrm{RP}(\mathrm{Zero}, p_2^2)$;
2. Une fois la fonction Z (d'arité 1) définie, ça va tout seul. La fonction trois d'arité 1 est simplement $\mathrm{Comp}(\mathrm{Succ}, \mathrm{Comp}(\mathrm{Succ}, \mathrm{Comp}(\mathrm{Succ}, \mathrm{Z})))$. À l'arité 0 il suffit d'utiliser Zero au lieu de Z dans l'expression précédente. Pour passer à l'arité 2 il suffit de précomposer avec une projection : $\mathrm{Comp}(\mathrm{Succ}, \mathrm{Comp}(\mathrm{Succ}, \mathrm{Comp}(\mathrm{Succ}, \mathrm{Comp}(\mathrm{Z}, p_1^2))))$.
3. L'addition se définit récursivement en utilisant le successeur Succ. Notons a la fonction telle que $a(x, y) = x + y$. On pose $a = \mathrm{RP}(f, g)$ où f d'arité 1 et g d'arité 3 sont à trouver. On obtient par définition de RP le système :
   $$\begin{cases} a(0, y) = f(y) \\ a(n+1, y) = g(n, y, a(n, y)) \end{cases}$$
   comme $a(0, x)$ doit être égal à $0 + y = y$ on prend $f = p_1^1$ (l'identité). Comme $a(n+1, y)$ doit être égal à $n + 1 + y$ et que $a(n, y)$ devrait être égal à $n + y$ il suffit de prendre g telle que $g(x, y, z) = \mathrm{Succ}(z)$. On pose donc $g = \mathrm{Comp}(\mathrm{Succ}, p_3^3)$. Finalement, en posant $a = \mathrm{RP}(p_1^1, \mathrm{Comp}(\mathrm{Succ}, p_3^3))$ on peut montrer aisément (par ce qui précède), par récurrence sur y, que $a(x, y) = x + y$.
4. Pour la mulitiplication on raisonne comme précedemment et on trouve que la multiplication peut être définie par la fonction $m = \mathrm{RP}(\mathrm{Z}, \mathrm{Comp}(a, p_2^3, p_3^3))$.
5. De même pour l'exponentiation on trouve $e = \mathrm{RP}(\mathrm{Comp}(\mathrm{Succ}, \mathrm{Z}), \mathrm{Comp}(m, p_2^3, p_3^3))$.
6. Pour la fonction prédécesseur on distingue deux cas, selon si x est nul ou non. On applique donc une récursion $\mathrm{Pred} = \mathrm{RP}(f, g)$ avec f d'arité 0 et g d'arité 2. On résout :
   $$\begin{cases} \mathrm{Pred}(0) = f() = 0 \\ \mathrm{Pred}(\mathrm{Succ}(x)) = g(x, \mathrm{Pred}(x)) = x \end{cases}$$
   Il vient alors que f doit être la fonction Zero et que choisir $g = p_1^2$ convient. Donc $\mathrm{Pred} = \mathrm{RP}(\mathrm{Zero}, p_1^2)$.
7. Il est plus facile d'écrire la différence tronquée en intervertissant les arguments $x \dot{-} y = \mathrm{Diff}_2(y, x)$ pour faire la récursion sur y (le retour à Diff est un jeu de compostion/projections). On pose $\mathrm{Diff}_2 = \mathrm{RP}(f, g)$ et on résout en f (arité 1) et g (arité 3) :
   $$\begin{cases} \mathrm{Diff}_2(0, x) = f(x) \\ \mathrm{Diff}_2(n+1, x) = g(n, x, \mathrm{Diff}_2(n, x)) \end{cases}$$
   Il faut $f(x) = x$ donc $f = p_1^1$. Par ailleurs, on vérifie que $x \dot{-} (n+1) = \mathrm{Pred}(x \dot{-} n)$ ce qui suggère de prendre $g(i, j, k) = \mathrm{Pred}(k)$ et ainsi $g = \mathrm{Comp}(\mathrm{Pred}, p_3^3)$ convient.

Correction de l'exercice 2.

On peut, par exemple, calculer $(x \dot{-} y) \dot{-} (x \dot{-} \operatorname{Succ} y)$. D'autres solutions sont bien entendu possibles.

Correction de l'exercice 3.

1. Soit $k \in \mathbb{N}$, P un prédicat, et f et g deux fonctions, tous récursif primitifs et d'arité k. Montrons que $h = \mathrm{Si}(P, f, g)$ est récursive primitive.
   L'expression $P(\overrightarrow{x}) \times f(\overrightarrow{x}) + (1 \dot{-} P(\overrightarrow{x})) \times g(\overrightarrow{x})$ vaut $f(\overrightarrow{x})$ si P est vrai et $g(\overrightarrow{x})$ sinon. Elle est donc égale à $h(\overrightarrow{x})$. La fonction h peut donc s'écrire comme composition de fonctions récursives primitives, elle est donc elle-même récursiove primitive. Formellement (c'est optionnel) :
   $$h = \mathrm{Comp}(\mathrm{Plus}, \mathrm{Comp}(\mathrm{Fois}, P, f), \mathrm{Comp}(\mathrm{Fois}, \mathrm{Comp}(\mathrm{Diff}, \mathrm{Un}_k, g)))$$
   où Un_k est la fonction constante égale à 1 d'arité k.
2. Soient f une fonction d'arité $k+1$ et h la fonction $\mathrm{Somme}(f)$. Montrons que h est récursive primitive.
   On cherche f_0 et g récursives primitives telles que $h = \mathrm{RP}(f_0, g)$. Il faut et suffit que $f(\overrightarrow{x}) = h(0, \overrightarrow{x}) = F(0, \overrightarrow{x})$ et que $g(y, \overrightarrow{(}x), h(y, \overrightarrow{x})) = h(y, \overrightarrow{(}x)) + F(\mathrm{Succ}(y), \overrightarrow{(}x))$.
   Il suffit de prendre
   $$f_0 = \mathrm{Comp}(F, \mathrm{Zero}_k, p_1^k, \ldots, p_k^k)$$
   et
   $$g = \mathrm{Comp}(\mathrm{Plus}, p_{k+2}^{k+2}, \mathrm{Comp}(F, \mathrm{Comp}(\mathrm{Succ}, p_1^{k+2}), p_2^{k+2}, \ldots, p_{k+1}^{k+2})).$$
3. Le produit se traite de la même manière sans difficulté.

Correction de l'exercice 4.

Pour démontrer que $h = \mu_{\leq}(P)$ est récursive primitive on cherche f et g récursives primitives telles que $h = \mathrm{RP}(f, g)$. Il faut et il suffit que

$$f(\overrightarrow{x}) = h(0, \overrightarrow{x}) = \begin{cases} 0 \text{ si } P(0, \overrightarrow{x}) \\ 1 \text{ sinon} \end{cases}$$
$$g(y, \overrightarrow{x}, h(y, \overrightarrow{x})) = h(\mathrm{Succ}(y), \overrightarrow{x}).$$

Pour f on prend donc $\neg P$. Considérons maintenant g. Dans l'expression précédente, il suffit de faire deux cas. Si $y \geq h(y, \overrightarrow{x})$ alors l'expression doit valoir $h(y, \overrightarrow{x})$ qui est dans ce cas le plus petit i tel que $P(i, \overrightarrow{x})$. Sinon, $h(y, \overrightarrow{x}) = y + 1$ ce qui signifie que, pour tous les $i \leq y$, $\neg P(i, \overrightarrow{x})$. Dans ce second cas il faut évaluer $P(y+1, \overrightarrow{x})$ et si c'est vrai rendre $y+1$, sinon rendre $y+2$. Le raisonnement par cas et les petits calculs nécessaires sont récursif primitif. On en déduit qu'un g convenable peut s'écrire comme une fonction primtive (l'expression formelle est technique et sans grand intérêt).

Une solution alternative (plus simple) est donnée sur la page de wikipedia concernant les fonctions récursives primitives. Pour la fonction μ^t la démonstration a été vue en cours et est similaire à ce qu'on vient d'écrire.

Minimisation généralisée. Le constructeur de suivant n'est pas récursif primitif. Pour $k \in \mathbb{N}$, si P est un prédicat d'arité $k+1$ alors $\mu(P)$ est la fonction h d'arité k telle que

$$h(x_1, \ldots, x_k) = \begin{cases} \text{Le plus petit } i \text{ tel que } P(i, x_1, \ldots, x_k) \text{ s'il en existe un} \\ \uparrow \text{ (la divergence) sinon.} \end{cases}$$

On ajoute la minimisation générale aux constructeurs des primitives récursives, ceci forme la classe des fonctions récursives (partielles).

Correction de l'exercice 5.

1. Il suffit de trouver le plus grand i tel que $y \times i \leq x$, ou de manière équivalente le plus petit i tel que $y \times (i+1) > x$. On aura alors $i = \lfloor x/y \rfloor$.
   On pose donc $f(x,y) = \mu P(i,x,y)$ où P est le prédicat $y \times (i+1) > x$ (il est facile de voir que ce prédicat est récursif primitif). La division entière n'est pas définie pour $y = 0$. (Si on pose $f(x,0) = 0$, il est possible d'écrire f comme une fonction récursive primitive.)
2. Il suffit de trouver le plus grand i tel que $i^2 \leq x$ ou de manière équivalent le plus petit i tel que $(i+1)^2 > x$. On peut donc poser $f = \mu P$ où P est le prédicat récursif primitif $(i+1)^2 > x$. On peut aussi rendre f cette fonction récursive primitive en prenant $f = \mathrm{Comp}(\mu_{\leq} P, p_1^1, p_1^1)$.
3. Non corrigé.