Cours de Lambda-calcul.*

Ch. Berline

CHAPITRE 3. Représentation des fonctions récursives.†

Table des matières

Le but de ce chapitre est de montrer que *toute fonction récursive partielle est représentable dans le λ-calcul*, en un sens que nous précisons plus bas.

Les préliminaires requis consistent en : a) les définitions de la normalisation de tête et de la résolubilité, b) savoir que l'ensemble des termes résolubles est clos par β-équivalence, que rs résoluble implique r résoluble, et enfin c) les Théorèmes de normalisation gauche et de tête.

1 Fonctions représentables par un λ-terme.

On commence par décider de représenter chaque entier n par l'entier de Church $\hat{n}$ correspondant. D'autres choix sont possibles (voir exercices et aussi [1]), qui ont d'autres avantages algorithmiques, mais aucun choix n'est optimal de ce point de vue [5].

Etant donnée une fonction partielle de N^k dans N ; nous considèrerons aussi f comme une fonction totale à valeur dans $N \cup \{\uparrow\}$, $\uparrow$ signifie donc *non défini* et son contraire sera noté $\downarrow$. Pour des raisons de commodités nous adopterons parfois la notation $\Lambda\vec{x}.\, f(\vec{x})$ pour f .

*DEA de Logique et Fondements de l'Informatique (2000-2002), UFR de Mathématiques, Université de Paris 7.

†Version du 21 Mars 1997 et Février 01, révisée 27 Février 2002.

Dfinition 1 *Soit f une fonction partielle de N^k dans N. Etant donné un terme t du λ-calcul nous dirons que t* représente *(resp.* représente fortement*) f si pour tous $(n_1, ..., n_k) \in N^k$ on a*

$$f(n_1, ..., n_k) = n \quad \Rightarrow \quad t\widehat{n_1}...\widehat{n_k} =_\beta \hat{n}$$
$$f(n_1, ..., n_k) \uparrow \quad \Rightarrow \quad t\widehat{n_1}...\widehat{n_k} \text{ est non normalisable (resp. non résoluble)}$$

Notons que les deux définitions coïncident dans le cas des fonctions totales.

Par ailleurs on peut reformuler ces deux définitions de plusieurs façons différentes, qui font plus ou moins ressortir leur caractère calculatoire ou algébrique. Le lemme suivant est en effet très facile à montrer.

Lemme 2 *On obtient des définitions équivalentes en remplaçant séparément ou simultanément dans cette définition :*
1. $\Rightarrow$ par $\Leftrightarrow$
2. $=_\beta$ par $\to_\beta$
3. $=_\beta$ par $\to_g$ et "non normalisable (resp. non résoluble)" *par* "non normalisable à gauche (resp. non normalisable de tête).

Preuve. 1. devrait être clair.

2. est une conséquence directe de Church-Rosser et du fait que les entiers de Church sont normaux.

3. suit directement des Théorèmes de Normalisation gauche et de tête.

La dernière formulation met en évidence qu'on peut associer à chaque t représentant une fonction f un algorithme qui permet de calculer la valeur de f sur chaque entier de Church, ou encore un programme directement exécutable en "call-by-name". En particulier toute fonction représentable est "calculable". Par ailleurs ces définitions se transposent aisément aux fonctions à arguments et/ou valeurs booléennes. Elles conviennent plus généralement pour toute fonction entre données finitaires (arbres, listes, etc.), dès que l'on a fixé une représentation de ces données dans le λ-calcul (ce que nous ferons par la suite).

Remarque 3 *La première définition permet de voir immédiatement que si t représente f alors :*
a) Tout terme β-équivalent à t représente aussi f,
b) si f définie au moins en un point alors t est résoluble,
c) si f est définie au moins en un point alors il existe des termes normaux de tête *qui la représentent.*

Exercice 4 *Représenter les connecteurs booléens "non", $\wedge$, $\vee$, $\Rightarrow$.*

Remarquer qu'on peut représenter chacun de ces connecteurs par une infinité de termes normaux distincts.

L'objet de ce chapitre est de montrer :

Thorme 5 *Toute fonction récursive partielle est fortement représentable par un terme (clos) du λ-calcul.*

La réciproque est intuitivement vraie (Thèse de Church) et peut se démontrer formellement : c'est le Théorème de Church-Kleene (cf. Appendice "Petit guide de récursivité", où la démonstration est esquissée, et le livre d'Odifreddi [7] pour une démonstration complète).

Nous adopterons ici la *définition de la classe des fonctions récursives partielles* comme la plus petite classe de fonctions partielles à arguments et valeurs dans N, qui contient : la fonction constante $\Lambda x.0$, la fonction successeur s, l'addition, la multiplication, la fonction caractéristique $\chi_{\leq}$ de $\leq$, les projections, et qui est close par composition et par l'opérateur μ.

$\chi_{\leq}$ est définie par $\chi_{\leq}(m,n) = 1$ si $m \leq n$ et 0 sinon, et
les *projections* sont les fonctions $p_n^k : x_1, ..., x_n \mapsto x_k$.

Le *schéma de composition* est en fait un ensemble de fonctionnelles $C_{n,k}$ telles que $C_{n,k}$ associe à tout n-uple $(f_1, ..., f_n)$ de fonctions de N^k dans N et à toute fonction partielle g de N^n dans N, la fonction partielle $h = g \circ (f_1, ..., f_n)$ de N^k dans N définie par :

$$\begin{array}{ll} h(\vec{p}) = g(f_1(\vec{p}), ..., f_n(\vec{p})) & \text{si } f_1(\vec{p}) \ ... \ f_n(\vec{p}) \text{ sont tous définis} \\ h(\vec{p}) = \uparrow & \text{sinon} \end{array}$$

En particulier $\Lambda x.1 \circ \Lambda x. \uparrow = \Lambda x. \uparrow$ et non $\Lambda x.1$.

Le *schéma* μ est lui même un ensemble de fonctionnelles μ_k, $k \geq 0$, associant à toute fonction partielle f de N^{k+1} dans N, la fonction $g \equiv \mu n.(f(n) = 0)$ de N^k dans N définie par :

$$\begin{array}{ll} g(\vec{p}) = m & \text{si } f(\vec{p}, l) \text{ est défini et non nul pour tout } l < m \text{ et } f(\vec{p}, m) = 0 \\ g(\vec{p}) = \uparrow & \text{sinon} \end{array}$$

Remarque 6 *Les fonctions récursives de N^k dans N^n s'identifient aux n-uplets de fonctions récursives $f_1, ..., f_n$ de N^k dans N. Une telle fonction est donc représentable par un élément de Λ^n. Pour la représenter par un seul terme du λ-calcul il suffit d'utiliser un combinateur qui permet de coder les éléments de Λ^n par des éléments de Λ. Le plus naturel correspond à l'opérateur $< -, ..., - > =_{def} \lambda x_1 ... \lambda x_n . \lambda z. \, z x_1 ... x_n$. La ième projection correspondante, soit p_i^n, est donnée par $p_i^n =_{def} \lambda r. \, (r)\pi_i^n$, où $\pi_i^n =_{def} \lambda x_1 ... \lambda x_n . x_i$ (on a en effet $p_n^i < u_1, ..., u_n > \rightarrow_h u_i$, où $< u_1, ..., u_n >$ est une abbréviation pour $\lambda z. z u_1 ... u_n$).*

2 Représentation des fonctions simples.

Elle est bien sûr relative à la représentation des entiers par les entiers de Church. Rappelons qu'il y a une infinité de solutions possibles à chaque fois, qui correspondent à façons différentes de calculer (le graphe de) la fonction. Les solutions ci-dessous sont les plus simples possibles, et sont des termes normaux.

La fonction $\Lambda x.0$ est représentée par $\lambda\nu.\bar{0}$.
Les *projecteurs* par $\lambda\nu_1....\lambda\nu_n.\nu_k$.
Le *successeur* par le terme $succ \equiv \lambda\nu.\lambda\varphi.\lambda x.\nu\varphi(\varphi)x$,
c'est à dire $\lambda\nu.\lambda\varphi.\lambda x.((\nu)\varphi)(\varphi)x$,
mais $\lambda\nu.\lambda\varphi.\lambda x.\varphi\,(\nu\varphi)x$ convient tout aussi bien.
L'addition est représenté par le terme $add \equiv \lambda\mu.\lambda\nu.\lambda\varphi.\lambda x.\mu\varphi(\nu\varphi)x$.
La multiplication par $\lambda\mu.\lambda\nu.\lambda\varphi.\lambda x.((\mu)(\nu)\varphi)x$ ou $\lambda\mu.\lambda\nu.\lambda\varphi.(\mu)(\nu)\varphi$, qui sont deux termes η-équivalents.
La fonction caractéristique de $\leq$ par le terme de Maurey :
$M \equiv \lambda\mu.\lambda\nu.\,(((\mu)A)\lambda d.1)\,((\nu)A)\lambda d.0$, avec $A \equiv \lambda\varphi.\lambda\psi.\psi\varphi$.

Notons qu'il est nettement moins facile de trouver un terme représentant le prédécesseur sur les entiers de Church (Exercice 13) et qu'il n'en existe aucun qui calcule le prédécesseur en un nombre constant d'étapes [5].

Exercice 7 *Vérifier que les termes ci-dessus représentent bien les fonctions indiquées, en suggérer éventuellement d'autres.*
Vérifier que le terme de Maurey permet de calculer l'inf de deux entiers m *et* n *en* $c \times \inf(n,m)$ *étapes, où* c *est une constante (on pourra commencer par réduire* $M\hat{m}\hat{n}$ *pour* $m \leq 2$*).*

Exercice 8 *Représenter l'exponentielle* $(m,n) \mapsto m^n$ *sur les entiers de Church.*

Exercice 9 *Soit* $\tilde{n} \equiv succ^n\hat{0}$ *(ainsi* $\widetilde{k+1} \equiv succ\ \tilde{k}$ *). Montrer que* $\tilde{k} \to_h \hat{k}$.

Exercice 10 *On se donne la représentation suivante de entiers :*
$\bar{0} \equiv \lambda x.x$, $\overline{n+1} \equiv \lambda\varphi_{n+1}.\,\varphi_{n+1}\bar{n}$.
1) Trouver un terme calculant le prédécesseur de tout entier non nul en un nombre constant d'étapes.
2) Trouver un terme $\varkappa$ *représentant la fonction caractéristique de* $\{0\}$ *à valeurs dans* $\{V,F\}$ *(on admettra que* $t = F$ *et* $t' = \lambda x.F$ *sont séparables, c'est à dire qu'il existe* $\bar{w}$ *tel que* $t\bar{w} =_\beta V$ *et* $t'\bar{w} =_\beta F$*).*

Exercice 11 *Même travail avec la représentation suivante :*
$\bar{0} \equiv \lambda x.x$, $\overline{n+1} \equiv \lambda\varphi_{n+1}.\,\varphi_{n+1}\bar{0}\bar{n}$.

Exercice 12 *Pour chacune des deux représentations ci-dessus*
1. *trouver un terme* U *tel que pour tout* $n \in N$ *on ait* $U\hat{n} \to_h \bar{n}$.
2. *Trouver un terme* U' *tel que pour tout* $n \in N$ *on ait* $U'\bar{n} \to_g \hat{n}$.
3. *Remarquer que les temps de calcul de* $U\hat{n}$ *et* $U'\bar{n}$ *sont linéaires en* n.

Exercice 13 Deux prédécesseurs sur les entiers de Church.
1. Représenter la fonction $F : N^2 \to N^2$ *qui est définie par* $F(m,n) = (m+1,m)$
2. Représenter la fonction $f : N \to N^2$ *qui est définie par* $f(0) = (0,0)$ *et* $f(n+1) = (n+1,n)$.
3. En déduire un terme représentant le prédécesseur sur les entiers de Church.
4. Vérifier qu'il calcule le prédécesseur de n *en un temps linéaire en* n.
5. Représenter la fonction $G : B \times N \to B \times N$ *qui est définie par*
$G(F,n) = (V,n)$ *et* $G(V,n) = (V,n+1)$
6. En déduire un autre prédécesseur sur les entiers de Church.

Nous allons maintenant représenter la définition par cas sur les entiers de Church. L'exercice suivant est important : il nous servira aussi, en particulier, pour représenter le schéma μ. Les trois premières assertions sont très faciles, la dernière demande un peu plus de réflexion.

Exercice 14 *Soit* $Z \equiv \lambda\nu.((\nu)\lambda d.F)V$, *alors*

(a) Z *représente la fonction caractéristique de* $\{0\}$ *(à valeurs booléennes),*

(b) $Z\hat{n}uv \to_h u$ *si* $n = 0$ *et* $Z\hat{n}uv \to_h v$ *si* $n \neq 0$,

(c) $Ztuv \to_h ((t)\,\lambda d.F)Vuv$, *donc* $Ztuv$ *est non résoluble si* t *n'est pas résoluble.*

(d) $Ztuv \to_h u$ *si* $t =_\beta \hat{0}$ *et* $Ztuv \to_h v$ *si* $t =_\beta \hat{n}$ *et* $n \neq 0$.

Exercice 15 Définition par cas. *étant données 3 fonctions récursives partielles* f, g, h *représentées par* u, v, w *respectivement, représenter la fonction* k *définie par :* $k(\vec{x}) \equiv f(\vec{x})$ *si* $h(\vec{x}) = 0$, $k(\vec{x}) \equiv g(\vec{x})$ *si* $h(\vec{x}) \downarrow$ *et* $h(\vec{x}) \neq 0$, *et* $k(\vec{x}) \uparrow$ *sinon. En déduire :*

Exercice 16 *La représentation du* schéma de récursion primitive *qui permet de définir* f *à partir de* g *et* h *totales et d'arités convenables par :* $f(\vec{x}, 0) = g(\vec{x})$ *et* $f(\vec{x}, y + 1) = h(\vec{x}, y, f(\vec{x}, y))$, *en supposant connu un terme représentant le prédecesseur.*

3 La composition.

Nous ne traiterons que le cas des fonctions unaires : $h \equiv g \circ f$ avec f et g récursives partielles de N dans N, en effet les difficultés sont apparentes dès ce cas et leur traitement plus lisible ; pour le cas général nous renvoyons à [2].

Nous supposons f et g représentées respectivement par les termes u et v et cherchons un représentant pour h . Le candidat naturel, à savoir $v \circ u \equiv \lambda x.v(u)x$ ne convient que pour les fonctions totales : il est en effet facile de vérifier que si $g \equiv \Lambda x.1$ et $f \equiv \Lambda x.\uparrow$, alors $v \circ u$ représentera $\Lambda x.1$ et non $\Lambda x.\uparrow$ comme voulu. Ceci est lié au fait que v n'est pas obligée, dans ce cas, de calculer t pour donner la valeur de vt. Nous allons donc commencer par remplacer v par un autre terme w représentant également g, mais de façon plus serrée. En fait ce remplacement peut se faire uniformément en v (et u!) : nous allons en effet exhiber un terme T tel que Tv convienne pour tout v. L'opérateur adéquat de composition unaire sur les fonctions partielles récursives sera alors représenté par l'opérateur $\circ' =_{def} \lambda\phi\lambda\psi.\,(T)\phi \circ \psi =_{def} \lambda\phi\lambda\psi\lambda x.\,((T)\phi)(\psi)x$.

S l'on travaille avec les entiers de Church $\hat{k}$, il suffit de prendre :

$$T =_{def} \lambda\phi\lambda\nu.\,((\nu)\lambda\sigma.\,\sigma \circ succ)\,\phi\,\hat{0}$$

où *succ* est un terme représentant le successeur sur ces entiers.

Il est en effet facile de vérifier que :

$$Tv\hat{k} \to_h ((\hat{k})\,\lambda\sigma.\,\sigma \circ succ)v\hat{0} \to_h v(succ^k\hat{0}) \to_\beta v\hat{k}$$

Puisque $Tv\hat{k} =_\beta v\hat{k}$ pour tout $k \in N$ on a que $w =_{def} Tv$ représente encore g : si $g(k) \downarrow$ c'est clair, et si $g(k) \uparrow$ on utilise la remarque que tout terme β-équivalent à un terme non résoluble est lui-même non résoluble. Mais on a gagné quelque chose de plus : si t n'est pas résoluble alors wt ne l'est pas non plus (puisque $Tvt \to_h ((t)\,\lambda\sigma.\,\sigma \circ succ)v\hat{0}$ et que rs résoluble implique r résoluble).

Remarque 17 *Le terme T est un* opérateur de mise en mémoire, *au sens de [3]. L'exercice suivant montre qu'il permet* en partie *de simuler l'appel par valeur (où on calcule l'argument avant de le donner à une abstraction) dans la réduction gauche. En ce sens qu'il permet de réduire le coût de la duplication éventuelle du calcul de l'argument dans la réduction gauche.* [1]

Exercice 18 * *Montrer que si $\phi \in V$ et $t =_\beta \hat{n}$ alors $T\phi t \to_h \phi\tilde{n}$*
(rappel : $\tilde{n} =_{def} succ^k\bar{0}$).

En déduire que $t =_\beta \hat{n}$ implique $Tvt \to_h v\tilde{n}$ pour tout terme $v \in \Lambda$.

Aide. *On a vu dans cette section que le cas $t \equiv \hat{n}$, était facile. Dans le cas général il faut utiliser d'une part la substitutivité de $\to_h$ et d'autre part, répétitivement, le Lemme*[2] *qui décrit les rapports entre l'écriture syntaxique de deux termes normaux de tête équivalents. On commencera par montrer que si $t =_\beta \hat{n}$ alors t a une réduction de tête de la forme : $t \to_h \lambda f.t_1 \to_h \lambda f.\lambda x.t_2 \to_h$ ... avec $t_1 =_\beta \lambda x.f^n x$ et $t_2 =_\beta f^n x$, la suite dépendant de la valeur de n.*

Exercice 19 *Montrer que plus généralement il existe pour tous n, k un terme du λ-calcul qui représente (en un sens intuitif) l'opérateur de composition $C_{n,k}$ d'une fonction représentable de arité n avec n fonctions représentables de arité k.*

Commentaires. Le fait que les opérateurs de composition soient eux-même représentables dans le λ-calcul est profondément de même nature que la conséquence du Théorème SMN que vous avez vue dans le cours fondamental ([4, p.49]) et qui affirme l'existence de fonctions (primitives) récursives calculant l'indice d'une fonction récursive partielle composée en fonction des indices de ses composantes (indices dans l'énumération des fonctions récursives partielles par une fonction récursive partielle universelle). D'ailleurs, en supposant fixée une arithmétisation du λ-calcul (où l'on code les termes du λ-calcul par des entiers en respectant leur structure), la fonction qui calcule le code de $Tv \circ u$ en fonction des codes de u et v, est clairement récursive primitive, et il en est de même pour les compositions (n, k)-aires. Mais l'intérêt du λ-calcul c'est justement qu'il permet de mathématiser le phénomène "naturel" qui est derrière, en restant "près du sens". Pour commencer il n'est nul besoin de codages arbitraires, si on excepte le choix de la représentation des entiers au départ, qui de toute manière se fait aussi en gardant du sens.

[1] Ces opérateurs interviennent aussi quand on étend la Correspondance de Curry-Howard à la logique classique et qu'on l'étudie via le réalisabilité [3]. Mais ceci est une autre histoire (typée !).

[2] *Lemme.* Deux *fnt* t et t' sont β-équivalentes ssi il existe $\bar{x}, y, \bar{t}, \bar{t}'$ tels que $t = \lambda\bar{x}.y\bar{t}$, $t' = \lambda\bar{x}.y\bar{t}'$, $l(t) = l(t') \geqslant 0$ et $\forall i \leq l(\bar{t})\ t_i =_\beta t'_i$.

4 Le schéma μ.

Nous faisons la démonstration dans le cas où f est binaire. On cherche à représenter $g : N \to N$ définie par

$$\begin{array}{ll} g(p) = m & \text{si pour tous } n < m \text{ on a } f(p,n)\downarrow \text{ et } f(p,n) \neq 0, \text{ et si } f(p,m) = 0, \\ g(p)\uparrow & \text{sinon} \end{array}$$

Je dis qu'il suffit de trouver un terme t clos tel que :
(a) si $f(p,n)\uparrow$, alors $t\,p\,\tilde{n}$ est non résoluble,
(b) si $f(p,n) = 0$, alors $t\,p\,\tilde{n} =_\beta \tilde{n}$,
(c) sinon $t\,p\,\tilde{n} \to_h t\,p\,\widetilde{n+1}$.
où $\tilde{n} \equiv succ^n \hat{0}$, et qu'alors $w \equiv \lambda\nu.\, t\;\nu\,\tilde{0}$ représentera g.

En effet $wp \to_h t\,p\,\tilde{0}$ et :

1. si $f(p,0)\uparrow$; alors, d'une part wp n'est pas résoluble car $tp\tilde{0}$ ne l'est pas, et de l'autre $g(p)\uparrow$, c'est bien ce qu'on voulait.
2. si $f(p,0) = 0$; alors, d'une part $wp =_\beta \tilde{0}$, et de l'autre $g(p) = 0$, parfait.
3. si $f(p,0)\downarrow$ et $\neq 0$; alors $wp \to_h t\,p\,\tilde{1}$; que vaut $g(p)$? tout ce qu'on sait c'est qu'il ne vaut pas zéro et qu'il faut maintenant considérer $f(p,1)$.
   (a) si $f(p,1)\uparrow$; alors, d'une part wp n'est pas résoluble car $tp\tilde{1}$ ne l'est pas, et de l'autre $g(p)\uparrow$, parfait.
   (b) si $f(p,1) = 0$; alors, d'une part $wp =_\beta \tilde{1}$, et de l'autre $g(p) = 1$, parfait.
   (c) si $f(p,1)\downarrow$ et $\neq 0$; alors $wp \to_h t\,p\,\tilde{2}$; que vaut $g(p)$? tout ce qu'on sait c'est qu'il ne vaut ni zéro ni un, et qu'il faut maintenant considérer $f(p,\tilde{2})$.
   etc.

Finalement, si $g(p) = m$ on aura $wp \to_h t\,p\,\tilde{m} =_\beta \tilde{m} =_\beta \hat{n}$, et si $g(p)\uparrow$ c'est qu'il se passe l'un de deux phénomènes suivants :

(i) $f(p,n)$ est défini non nul pour tout n, dans ce cas la réduction de tête de wp est infinie et passe par tous les $tp\tilde{n}$, et wp n'est pas résoluble;

(ii) On rencontre un n tel que $f(p,n)\uparrow$ avant de rencontrer un m tel que $f(p,m) = 0$; la première fois que ça arrive on tombe dans un cas (a) et on conclue que wp n'est pas résoluble.

En conséquence de quoi w représentera bien g. Il suffit maintenant de trouver un terme t satisfaisant les propriétés requises.

Une petite remarque auparavant : nous devons travailler avec $\tilde{n}$ au lieu de $\hat{n}$ pour une raison technique qui apparaîtra bientôt, et qui est liée au fait que dans une récurrence on est amené à calculer en premier l'argument sur lequel on fait la récurrence. En revanche la représentation de l'entier p, qui joue ici seulement le rôle de paramètre, n'a aucune importance dans cette démonstration, et c'est pour cela que nous ne nous sommes pas donnée la peine de préciser quel chapeau il portait.

En fait il suffit (cf. Exercice 14.d) de trouver un t tel que :

$$t\,p\tilde{n} \to_h (\,Z\,(up)\tilde{n}\,)\ \tilde{n}\ (tp)\,\widetilde{n+1} \tag{1}$$

(Z a trois arguments, qui sont respectivement : $(up)\tilde{n}$, $\tilde{n}$ et $(tp)\,\widetilde{n+1}$)
et pour ceci il suffit de trouver t tel que :

$$t\ \to_h \lambda x.\lambda y.\,(\,Z\,(ux)y\,)\,y\ (tx)(succ)y$$

ou encore, tel que :

$$t \to_h Mt \tag{2}$$

avec :

$$M \equiv \lambda\tau.\,\lambda x.\lambda y.\,(\,Z\,(ux)y\,)\,y\ (\tau x)(succ)y$$

Or, pour satis faire (2) il suffit de prendre $t \equiv \Theta M$, où Θ est le *combinateur de point fixe de Türing,* qui est défini par : $\Theta \equiv AA$ où $A \equiv \lambda a.\lambda f.\,(f)((a)a)f$.

Remarque 20 *Ici encore il est facile de trouver un terme qui représente l'opérateur* μ_1 *lui même, et il en est évidemment de même pour tous les* μ_k .

Remarque 21 *C'est pour être en mesure de résoudre (1) que nous avons besoin de quitter les entiers de Church pour les* $\tilde{n}$, *en effet, si il est vrai que* $succ\,\hat{n} \to_h \widehat{n+1}$, *il est faux que ces termes soient vraiment égaux, alors que c'est vrai pour les entiers tildés, qui nous permettent donc de satisfaire à l'exigence que nous avons sur le fait que la réduction doit être de tête.*

Remarque 22 *De façon générale on appelle* combinateur de point fixe *tout terme* Z *tel que pour tout* $u \in \Lambda$ *on ait* $u =_\beta u(Z)u$, *mais le fait de prendre ici* Θ *est important pour respecter* $\to_h$.

Exercice 23 Programmation en λ-calcul pur du "stream" des entiers.

Le but de cet exercice est de trouver un terme t *du* λ*-calcul pur dont la réduction gauche"produise" la liste infinie des entiers, dans l'ordre.*
On rajoute au λ*-calcul non typé deux constantes de termes* $\underline{0}$ *et* s.
On pose : $\underline{n} \equiv s^n\underline{0}$ *et* $cons \equiv \lambda x\lambda L\lambda g.\,(g)x(L)g$. *On pourra dénoter* $(cons)xL$ *par* $x * L$.
Trouver un terme clos F *tel que*

$$\forall n\ \ (F)\underline{n} \to_g (cons)\underline{n}(F)(s)\underline{n}$$

Montrer que $t = F\underline{0}$ *répond à la question, et la préciser (que signifie "produit" ?)*
Montrer que t *est résoluble et non normalisable ; quelle est sa variable de tête ? le nombre d'arguments de cette variable ?*
Dessiner l'arbre de Böhm de t.
Remarque *: on peut aussi bien supprimer le* g *de droite dans la définition de* $cons$, *ça ne change pas le problème mais complique un peu la représentation du stream et l'arbre de Böhm.*

Quelques solutions possibles [A vérifier].

Solution 24 *Exercice 8.* $\lambda\mu\lambda\nu.\nu\mu$.

Solution 25 *Exercice 10. 1)* $\lambda\nu.\nu I$; *2)* $\varkappa' = \lambda\nu\,\nu(\lambda x.F)$ *pour une fonction à valeurs dans* $\{F, \lambda x.F\}$; *on en déduit* $\varkappa$ *en utilisant la séparabilité de* F.*et* $\lambda x.F$.

Solution 26 *Exercice 11. 1)* $\lambda\nu.\nu I$; *2)* $\varkappa' = \lambda\nu.\nu V$ *pour une fonction à valeurs dans* $\{I, V\}$ *; on en déduit* $\varkappa$ *en utilisant la séparabilité de* I, V.

Solution 27 *Exercice 12. 1)* $U = \lambda\nu.(\nu\,\lambda x\lambda\varphi.\varphi x)\bar{0}$, *en ce qui concerne la première représentation. 2) Il suffit de trouver* U' *tel que* $U' =_\beta \lambda\nu.\,(\varkappa\nu)\hat{0}\,(suc)(U')(pred')\nu$, *où* suc *est un successeur sur les entiers de Church et* $pred'$ *un prédecesseur sur les autres entiers*

Solution 28 *Exercice 13. Il y a une ambigüité dans l'énoncé (sur les couples), en fait on va tous les coder par un terme de* λ.

1) $u_F = \lambda z. < succ\,(p_2^1)z,\, p_2^1 z >$

2) $u_f = \lambda\nu.\nu\, u_F < 0, 0 >$

3) $pred = \lambda\nu.p_2^2(\nu u_F) < 0, 0 >$ *ou* $\lambda\nu.p_2^2\,(u_f)\nu$

5) $u_G = \lambda z.\, p_2^1 z < V, succ(p_2^2)z > < V, p_2^2 z >$

6) $pred' = \lambda\nu.\, p_2^2\,(\nu u_G)\ < F, 0 >$ *(ici,* F *c'est* $Faux$*!).*

Références

[1] H. Barendregt, The lambda-calculus : its syntax and its semantics, Studies in Logic n°, 1984

[2] J.L. Krivine, Lambda-calcul, types et modèles, ed. Masson, 1990, version anglaise Ellis-Horwood, 1993.

[3] J.L. Krivine, [choisir référence]

[4] R. Cori et D. Lascar, Logique Mathématique II, ed. Masson, Paris, 1993.

[5] M. Parigot et P. Rozière, Constant time reductions in λ-calculus, in Mathematical Foundations of Computer Science, LNCS 711, p.608-617, 1993.

Complément de lecture pour les curieux des fonctions récursives :

[6] S.C. Kleene, Origins of recursive function theory, Ann. History Comput. 3 (1) 52-67.

[7] G. Odifreddi, Classical Recursion Theory, Studies in Mathematical Logic, vol.125, North-Holland, Amsterdam, 1989.

[8] R. Soare, Computability and recursion, Bull. Symb. Logic, vol.2, n°3, Sept.96, p.284-321 *(discussion sur les termes de "fonctions récursives" et "fonctions calculables", plus histoire des diverses notions).*