Lambda-calcul et langages fonctionnels: exercices

Jean Goubault-Larrecq

1er juillet 2024

1 Alpha-renommage, beta-réductions

Exercice 1 Montrer que, formellement, $\lambda x.\lambda y.xy\ \alpha\ \lambda y.\lambda x.yx$ est faux, mais que $\lambda x.\lambda y.xy =_\alpha \lambda y.\lambda x.yx$ est vrai.

Exercice 2 Dessiner le graphe de réductions du λ-terme $(\lambda x.(\lambda y.xy)x)z$.

Exercice 3 Dessiner le graphe de réductions de $(\lambda x.(\lambda y.y)x)z$, de $(\lambda f, x.f(fx))(\lambda g, y.gy)$, de $(\lambda x.xx)(\lambda x.xx)$.

Exercice 4 Dessiner le graphe de réductions de :

$$(\lambda y.(\lambda x.yxx)(\lambda z.z(xy)))(\lambda z'.z').$$

Exercice 5 Montrer que pour tous λ-termes u, w, u', w', et pour toute variable z :

(a) si $w \to w'$ alors $u[z := w] \to^* u[z := w']$;

(b) si $u \to u'$ alors $u[z := w] \to u'[z := w]$;

(c) si $u \to^* u'$ et $w \to^* w'$ alors $u[z := w] \to^* u'[z := w']$.

Exercice 6 Montrer que si $u \Rightarrow u'$ et $w \Rightarrow w'$ alors $u[z := w] \Rightarrow u'[z := w']$, où $\Rightarrow$ est la relation de réduction parallèle.

2 Substitutions

Exercice 7 Montrer que si x et y sont deux variables distinctes, et si x n'est pas libre dans w, alors alors $u[x := v][y := w] = u[y := w][x := v[y := w]]$. L'égalité est à α-équivalence près.

On définition une opération de *substitution parallèle* comme suit. Une *substitution* est une application θ des variables vers les λ-termes telle que $\theta(x) = x$ pour toute variable x, sauf un nombre fini. Le *domaine* de θ est $\{x \mid \theta(x) \neq x\}$. On appellera *support* de θ l'ensemble des variables de son domaine, plus l'ensembles des variables qui apparaissent libres dans un quelconque des termes $\theta(x)$, lorsque x parcourt le domaine de θ.

Lorsque $x_1, \ldots, x_n$ sont des variables deux à deux distinctes, $[x_1 := t_1, \cdots, x_n := t_n]$ dénote la substitution qui à chaque x_i associe t_i, et à chaque variable $x \notin \{x_1, \cdots, x_n\}$ associe x. On peut toujours écrire une substitution sous cette forme ; son domaine est alors inclus dans $\{x_1, \cdots, x_n\}$, et son support est inclus dans $\{x_1, \cdots, x_n\}$ union l'ensemble des variables libres de t_1, …, de t_n. On pose :

$$x\theta \stackrel{\text{def}}{=} \theta(x)$$

$$(uv)\theta \stackrel{\text{def}}{=} (u\theta)(v\theta)$$

$$(\lambda x.u)\theta \stackrel{\text{def}}{=} \lambda x.(u\theta)$$

où dans la dernière clause, x n'est pas dans le support de θ. Ceci donne une définition de $u\theta$ par récurrence sur la taille de u, modulo α-renommage. On notera que, lorsque θ s'écrit $[x := v]$, $u\theta$ est la notion usuelle de substitution $u[x := v]$.

Exercice 8 Que valent $(xy)[x := yz, y := xx]$, $(xy)[x := yz][y := xx]$, $(xy)[y := xx][x := yz]$? Expliquer pourquoi on appelle parfois une substitution θ comme ci-dessus une substitution *parallèle*.

Exercice 9 La *composée* $\theta\theta'$ de deux substitutions θ et θ' est définie comme la fonction qui à toute variable x associe $\theta(x)\theta'$. Montrer que :

(a) $u(\theta\theta') = (u\theta)\theta'$ pour tout λ-terme u ;

(b) la composition est associative : $(\theta\theta')\theta'' = \theta(\theta'\theta'')$;

(c) quel est l'élément neutre de la composition ?

Les substitutions forment donc un *monoïde*.

Exercice 10 Montrer que :

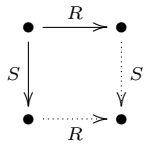
(a) si $u \to v$, alors $u\theta \to v\theta$ pour toute substitution θ ;

(b) si x n'est pas dans le support de θ, alors $u\theta[x := v\theta] = u[x := v]\theta = u(\theta + [x := v]$, où $\theta + [x := v]$ est la substitution qui à x associe $v\theta$ et à toute $y \neq x$ associe $\theta(y)$.

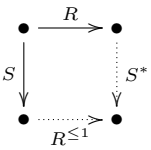
3 Confluence et le lemme de Hindley-Rosen

Une relation binaire R sur A est, formellement, juste un sous-ensemble de $A \times A$. On notera souvent $a \xrightarrow{R} b$ au lieu de $(a, b) \in R$.

Deux relations binaires R et S *commutent* si et seulement si :



autrement dit si, chaque fois que l'on a une situation représentée par les flèches pleines, il existe des flèches, notées en pointillé, qui complètent le diagramme. On dit que R *pseudo-commute* avec S si et seulement si :



(On note $R^{\leq n}$ pour au plus n étapes consécutives de R.) On notera bien que « R pseudo-commute avec S » n'est pas équivalent à « S pseudo-commute avec R ».

Exercice 11 Montrer que, si R et S sont confluentes et si R^* et S^* commutent, alors :

(a) $R^*; S^*$ est fortement confluente ;

(b) $R \cup S$ est confluente.

Ceci est le *lemme de Hindley-Rosen*.

Exercice 12 Montrer que si R pseudo-commute avec S, alors R^* et S^* commutent.

Exercice 13 Montrer que $\to_\eta$ est confluent. (Utiliser le lemme de Newman.) Montrer que $\to_\beta$ pseudo-commute avec $\to_\eta$, et en déduire que $\to_{\beta\eta}$ est confluent.

4 Un peu de confluence

Exercice 14 Montrer que $S \stackrel{\text{def}}{=} \lambda n, f, x.f(nfx)$ et $\lambda n, f, x.nf(fx)$ ne sont pas β-équivalents. (Ce sont deux termes codant la fonction successeur $n \mapsto n+1$.)

Exercice 15 Montrer que $\lambda m, n, f, x.mf(nfx)$, $\lambda m, n, f, x.nf(mfx)$, $\lambda m, n.mSn$ et $\lambda m, n.nSm$ sont deux à deux β-inéquivalents. (Ce sont autant de termes codant l'addition. S est défini à l'exercice précédent.)

Exercice 16 En considérant le terme $(\lambda x, y.x)u$, où $u \stackrel{\text{def}}{=} (\lambda z.z)z$, montrer que le λ-calcul avec réduction faible (pas sous les λ-abstractions) n'est pas confluent, et même pas localement confluent.

5 Le λ-calcul avec couples

On considère un λ-calcul étendu. On a trois constructions supplémentaires : $\pi_1 u$, $\pi_2 u$, et $\langle u, v\rangle$, où u et v sont des termes (étendus). On appelle les termes $\langle u, v\rangle$ les *couples*. On a aussi les règles de réécriture supplémentaires :

$$(\pi_1) \qquad \pi_1\langle u, v\rangle \to u$$
$$(\pi_2) \qquad \pi_2\langle u, v\rangle \to v$$

Elles s'appliquent sous n'importe quel contexte.

Exercice 17 Je considère un terme de la forme $\pi_1 u$. Vers quoi se réduit-il en une étape ? Distinguer plusieurs cas.

Exercice 18 En adaptant la technique des réductions parallèles, montrer que ce λ-calcul étendu est confluent.

6 Les paires surjectives

On considère le même λ-calcul étendu qu'à la section 5, mais avec la règle en plus :

$$(SP) \qquad \langle \pi_1 u, \pi_2 u\rangle \to u$$

Elle s'applique de nouveau sous n'importe quel contexte. (SP) signifie « surjective pairing » : grâce à cette règle, tout terme u est équivalent à un couple, via $u \leftrightarrow^* \langle \pi_1 u, \pi_2 u\rangle$.

Exercice 19 Montrer que le λ-calcul avec paires surjectives est localement confluent. Le but des exercices qui suivent est de montrer qu'il n'est *pas* confluent.

Exercice 20 On rappelle :

$\Theta \stackrel{\text{def}}{=} AA$ combinateur de point fixe de Turing

$A \stackrel{\text{def}}{=} \lambda g, h.h(ggh).$

On a $\Theta u \to^+ u(\Theta u)$ pour tout terme u. Fixons une variable z, et posons :

$$u_0 \stackrel{\text{def}}{=} \lambda x, y.\langle \pi_1(zy), \pi_2(z(xy))\rangle$$
$$u_1 \stackrel{\text{def}}{=} \Theta u_0$$
$$u_2 \stackrel{\text{def}}{=} \Theta u_1$$
$$u_3 \stackrel{\text{def}}{=} z(u_1 u_2).$$

Montrer que, pour tout terme t, on a :

$$u_1 t \to^+ \langle \pi_1(zt), \pi_2(z(u_1 t))\rangle. \tag{1}$$

Exercice 21 Montrer que $u_2 \to^+ u_3$ et $u_2 \to^+ u_1 u_3$.

Exercice 22 Montrer que les seules réductions $u_1 \to t_1 \to t_2 \to \cdots \to t_n \to \cdots$ partant de u_1 sont de la forme suivante :

(a) d'abord, les premiers termes t_i sont de la forme $(\lambda h.hw_1)u_0$, où les termes w_1 sont des réduits de Θh (en un nombre quelconque d'étapes) ;

(b) les suivants (s'il y en a) sont de la forme $u_0 w_2$, où w_2 est un réduit de $w_1[h := u_0]$, w_1 étant comme ci-dessus ;

(c) les suivants (toujours s'il y en a) sont de la forme $\lambda y.\langle \pi_1(zy), \pi_2(zw_3)\rangle$, où w_3 est un réduit de $w_2 y$, w_2 étant comme ci-dessus ;

(d) et finalement, éventuellement, le terme $\lambda y.zy$, qui est en forme normale, mais qu'on ne peut l'obtenir que si l'un des termes w_3 comme ci-dessus se réduit en y.

Exercice 23 Supposons que le λ-calcul avec paires surjectives soit confluent.

(a) En utilisant (1), montrer que le cas (d) de l'**Exercice 22** ne peut pas se produire. Indication : si le cas (d) se produit, alors $u_1 y \to^* y$, en déduire que $u_1 y \to^+ \langle \pi_1(zy), \pi_2(zy)\rangle \to zy$ et conclure.

(b) En utilisant l'**Exercice 22**, en déduire que $u_1 u_3$ ne peut se réduire en un terme de la forme zw (pour un certain terme w) que si u_1 se réduit en un terme $\lambda y.\langle \pi_1(zy), \pi_2(zw_3)\rangle$ (où w_3 est comme à l'**Exercice 22** (c)), et u_3 et $w_3[y := u_3]$ se réduisent tous les deux à w.

(c) En déduire que si u_3 et $u_1 u_3$ ont un réduit commun de la forme zw, pour un certain terme w, alors w est aussi un réduit commun de u_3 et de $u_1 u_3$.

(d) En déduire une contradiction.

Le λ-calcul avec paires surjectives n'est donc *pas* confluent. (Il est localement confluent.) C'est un résultat de Jan Willem Klop (1982). La démonstration ci-dessus est due à Pierre-Louis Curien et Thérèse Hardin (1990).

7 Le $\lambda\delta$-calcul et le $\lambda\delta_C$-calcul

On enrichit le λ-calcul avec une constante δ, et les règles :

$$\delta uv \to \mathbf{V} \qquad \text{si } u \text{ et } v \text{ sont des termes canoniques identiques} \tag{2}$$

$$\delta uv \to \mathbf{F} \qquad \text{si } u \text{ et } v \text{ sont des termes canoniques distincts.} \tag{3}$$

où « identique » est à prendre à α-renommage près, et $\mathbf{V} \stackrel{\text{def}}{=} \lambda x, y.x$ et $\mathbf{F} \stackrel{\text{def}}{=} \lambda x, y.y$ sont les booléens de Church usuels. On notera aussi $\mathbf{I} \stackrel{\text{def}}{=} \lambda x.x$. Un terme de ce nouveau langage est dit *canonique* si et seulement s'il est β-normal, et ne contient pas de sous-terme de la forme δst. (Il peut contenir des sous-termes de la forme δs, ou bien δ tout seul.)

On parlera de $\beta\delta$-réduction pour la notion de réduction définie par β et les deux règles ci-dessus. Il est tacite qu'elle passe au contexte. La $\beta\delta$-convertibilité est la plus petite relation d'équivalence contenant la $\beta\delta$-réduction.

Exercice 24 En considérant le terme $d_0 \stackrel{\text{def}}{=} (\lambda x, y.\delta xy)\mathbf{II}$,

(a) montrer que $\mathbf{V}$ et $\mathbf{F}$ sont $\beta\delta$-convertibles ;

(b) en déduire que le $\lambda\delta$-calcul a une théorie triviale : *tous* les $\lambda\delta$-termes sont $\beta\delta$-convertibles.

Le $\lambda\delta_C$-calcul est défini comme le $\lambda\delta$-calcul, à ceci près que les règles (2) et (3) ne s'appliquent que si u et v sont non seulement canoniques mais *clos* (sans variable libre).

Exercice 25 En utilisant le lemme de Hindley-Rosen, montrer que le $\lambda\delta_C$-calcul est confluent. En déduire que le $\lambda\delta_C$-calcul a une théorie non triviale : montrer par exemple que deux variables distinctes ne sont jamais $\beta\delta_C$-convertibles.

8 L'arithmétique des entiers de Church

Exercice 26 On rappelle le codage du prédécesseur donné en cours :

$$\ulcorner P \urcorner \stackrel{\text{def}}{=} \lambda z.\ulcorner p_1 \urcorner(z(\lambda c.\langle \ulcorner S \urcorner(\ulcorner p_1 \urcorner c), \ulcorner p_1 \urcorner c\rangle)\langle \ulcorner 0 \urcorner, \ulcorner 0 \urcorner\rangle.$$

On rappelle qu'on a les réductions $\ulcorner S \urcorner\ulcorner n \urcorner \to^* \ulcorner n+1 \urcorner$ pour tout $n \in \mathbb{N}$, $\ulcorner p_1 \urcorner\langle u, v\rangle \to^* u$ et $\ulcorner p_2 \urcorner\langle u, v\rangle \to^* v$. Montrer que $\ulcorner P \urcorner\ulcorner 0 \urcorner \to^* \ulcorner 0 \urcorner$ et que $\ulcorner P \urcorner\ulcorner n+1 \urcorner \to^* \ulcorner n \urcorner$ pour tout $n \in \mathbb{N}$.

Exercice 27 On rappelle le codage de la récurrence primitive donnée en cours (où j'ai cependant remplacé Y par Θ) :

$$\begin{aligned}\ulcorner R_{f,g} \urcorner \stackrel{\text{def}}{=} \Theta(\lambda R, z, z_1, \cdots, z_k.&\\ &\ulcorner \text{if} \urcorner(\ulcorner = 0 \urcorner z)\\ &\quad(\ulcorner f \urcorner z_1 \cdots z_k)\\ &\quad(\ulcorner g \urcorner(\ulcorner P \urcorner z)(R(\ulcorner P \urcorner z)z_1 \cdots z_k)z_1 \cdots z_k),\end{aligned}$$

où $f\colon \mathbb{N}^k \to \mathbb{N}$ et $g\colon \mathbb{N}^{k+2} \to \mathbb{N}$ sont dans **Rec**, $\ulcorner = 0 \urcorner$ code correctement le teste à zéro, au sens où $\ulcorner = 0 \urcorner\ulcorner 0 \urcorner \to^* \mathbf{V}$ et $\ulcorner = 0 \urcorner\ulcorner n+1 \urcorner \to^* \mathbf{F}$ pour tout $n \in \mathbb{N}$; et où $\ulcorner \text{if} \urcorner\mathbf{V}te \to^* t$, $\ulcorner \text{if} \urcorner\mathbf{F}te \to^* e$. On suppose aussi que $\ulcorner f \urcorner$ code correctement f, au sens où $\ulcorner f \urcorner\ulcorner n_1 \urcorner \cdots \ulcorner n_k \urcorner \to^* \ulcorner f(n_1, \cdots, n_k) \urcorner$ si $f(n_1, \cdots, n_k)$ est définie, et de même que $\ulcorner g \urcorner$ code correctement g. Montrer que $\ulcorner R_{f,g} \urcorner$ code correctement $R_{f,g}$, autrement dit montrer que l'on a :

$$\begin{aligned}\ulcorner R_{f,g} \urcorner\ulcorner 0 \urcorner\ulcorner m \urcorner_1 \cdots \ulcorner m \urcorner_k &\to^* \ulcorner f(m_1, \cdots, m_k) \urcorner\\ \ulcorner R_{f,g} \urcorner\ulcorner n+1 \urcorner\ulcorner m \urcorner_1 \cdots \ulcorner m \urcorner_k &\to^* \ulcorner g(n, R_{f,g}(n, m_1, \cdots, m_k), m_1, \cdots, m_k) \urcorner\end{aligned}$$

pour tous entiers naturels n, m_1, …, m_k, à condition que les expressions $f(m_1, \cdots, m_k)$, resp. $g(n, R_{f,g}(n, m_1, \cdots, m_k), m_1, \cdots, m_k)$, soient définies.

Exercice 28 On rappelle le codage de la minimisation donné en cours (avec Θ à la place de Y) :

$$\begin{aligned}\ulcorner \mu f \urcorner \stackrel{\text{def}}{=} \lambda z_1, \cdots, z_k.&\\ &\Theta(\lambda search, z.\\ &\qquad \ulcorner if \urcorner(\ulcorner = 0 \urcorner(\ulcorner f \urcorner z z_1 \cdots z_k))\\ &\qquad\quad z\\ &\qquad\quad search(\ulcorner S \urcorner z)),\end{aligned}$$

On suppose aussi que $\ulcorner f \urcorner$ code correctement f, au sens où $\ulcorner f \urcorner\ulcorner n \urcorner\ulcorner n_1 \urcorner \cdots \ulcorner n_k \urcorner \to^* \ulcorner f(n, n_1, \cdots, n_k) \urcorner$ si $f(n_1, \cdots, n_k)$ est définie. Montrer qu'alors $\ulcorner \mu f \urcorner$ code

correctement la minimisation μf, au sens où si :

$$\begin{aligned} \ulcorner f \urcorner \ulcorner 0 \urcorner u_1 \cdots u_k &\to^* \ulcorner m_0 \urcorner && (m_0 \neq 0) \\ &\vdots \\ \ulcorner f \urcorner \ulcorner n-1 \urcorner u_1 \cdots u_k &\to^* \ulcorner m_{n-1} \urcorner && (m_{n-1} \neq 0) \\ \ulcorner f \urcorner \ulcorner n \urcorner u_1 \cdots u_k &\to^* \ulcorner 0 \urcorner, \end{aligned}$$

alors :

$$\ulcorner \mu f \urcorner u_1 \cdots u_k \to^* \ulcorner n \urcorner.$$

On opérera une récurrence sur n.

Exercice 29 Montrer que :

(a) $\ulcorner \times \urcorner \stackrel{\text{def}}{=} \lambda m, n, f.m(nf)$ code correctement la multiplication entière ;

(b) proposer un autre λ-terme codant correctement la multiplication entière, mais qui n'est pas β-équivalent à $\ulcorner \times \urcorner$;

(c) le λ-terme $\lambda m, n.mn$ code correctement une opération arithmétique binaire bien connue (en tous cas lorsque le couple d'entiers en argument est différent de $(0, 0)$) : laquelle ?

(d) montrer que le terme identité $\lambda m.m$ code la même opération binaire ;

(e) proposer un λ-terme $\ulcorner \dot{-} \urcorner$ codant correctement l'opération de soustraction tronquée $m, n \mapsto m \dot{-} n$, où $m \dot{-} n$ vaut $m - n$ si $m \geq n$, et 0 sinon ;

(f) proposer un λ-terme $\ulcorner = 0 \urcorner$ codant correctement l'opération de test à zéro, c'est-à-dire que $\ulcorner = 0 \urcorner \ulcorner m \urcorner =_\beta \mathbf{V}$ si $n = 0$, $\mathbf{F}$ sinon ($\mathbf{V} \stackrel{\text{def}}{=} \lambda x, y.x$ et $\mathbf{F} \stackrel{\text{def}}{=} \lambda x, y.y$ sont les booléens de Church usuels) ;

(g) en déduire des λ-termes $\ulcorner \leq \urcorner$ et $\ulcorner = \urcorner$ codant correctement le test binaire « inférieur ou égal à », et le test d'égalité entre entiers ;

(h) définir $\ulcorner \wedge \urcorner$, un λ-terme codant correctement le et logique, au sens où $\ulcorner \wedge \urcorner \mathbf{VV} =_\beta \mathbf{V}$, $\ulcorner \wedge \urcorner \mathbf{VF} =_\beta \mathbf{F}$, $\ulcorner \wedge \urcorner \mathbf{FV} =_\beta \mathbf{F}$, $\ulcorner \wedge \urcorner \mathbf{FF} =_\beta \mathbf{F}$;

(i) faire pareil avec le ou logique ($\ulcorner \vee \urcorner$), et la négation ($\ulcorner \neg \urcorner$) ;

(j) étant donnés deux λ-termes u et v et une variable x non libre ni dans u ni dans v, on pose :

$$G_{u,v} \stackrel{\text{def}}{=} \lambda n, m.nF(\lambda x.u)(mF(\lambda x.v))$$
$$F \stackrel{\text{def}}{=} \lambda f, g.gf.$$

Montrer que $G_{\mathbf{V},\mathbf{F}}$ (le *terme de Maurey*) code correctement le test « inférieur ou égal à » sur les entiers, et que $G_{\mathbf{F},\mathbf{V}}$ code correctement le test « strictement supérieur à » sur les entiers ; quel avantage ce terme a-t-il comparé au terme $\ulcorner \leq \urcorner$ de (g) ?

9 Les listes

Exercice 30 On propose le codage suivant des listes à la OCaml. L'idée est qu'une liste $[u_1; \cdots; u_n]$ sera représentée par un terme de la forme $\lambda cons, nil.cons\ u_1 (cons\ u_2 \cdots (cons\ u_n\ nil))$.

$$\ulcorner nil \urcorner \stackrel{\text{def}}{=} \lambda cons, nil.nil$$
$$\ulcorner :: \urcorner \stackrel{\text{def}}{=} \lambda x, \ell, cons, nil.cons\ x\ (\ell\ cons\ nil)$$
$$\ulcorner hd \urcorner \stackrel{\text{def}}{=} \lambda \ell.\ell(\lambda x, r.x)A,$$

où A est un λ-terme fixé mais arbitraire. Montrer que :

$$\ulcorner hd \urcorner (\ulcorner :: \urcorner\ u\ r) \to^* u.$$

Exercice 31 On utilise la notation :

$$[v_1; \cdots; v_k] \stackrel{\text{def}}{=} \lambda cons, nil.cons\ v_1 (cons\ v_2 \cdots (cons\ v_k\ nil)),$$

pour tous termes v_1, ..., v_k, conformément à l'intuition donnée en début de section.

(a) Montrer que :

$$\ulcorner map \urcorner\ u\ [v_1; \cdots, v_k] \to^* [uv_1; \cdots; uv_k]$$

pour tous termes u, v_1, ..., v_k, où :

$$\ulcorner map \urcorner \stackrel{\text{def}}{=} \lambda f, \ell, cons, nil.\ell(\lambda x, r.cons(fx)r)nil.$$

(b) En généralisant, on pose :

$$\ulcorner fold \urcorner \stackrel{\text{def}}{=} \lambda mul, one, u, \ell \cdot \ell(\lambda x, r \cdot mul(ux)r)one$$

Vérifier que :

$$\ulcorner fold \urcorner\ mul\ one\ u\ \ulcorner nil \urcorner =_\beta one \tag{4}$$

$$\ulcorner fold \urcorner\ mul\ one\ u\ (\ulcorner :: \urcorner\ v\ w) =_\beta mul(uv)(\ulcorner fold \urcorner\ mul\ one\ u\ w) \tag{5}$$

pour tous λ-termes mul, one, u, v, w, et en déduire que :

$$\ulcorner fold \urcorner\ mul\ one\ \ u\ [v_1; \cdots; v_k] =_\beta mul(uv_1)(mul(uv_2) \cdots (mul(uv_k)one) \cdots)$$

pour tous termes mul, one, u, v_1, ..., v_k ($k \in \mathbb{N}$).

Exercice 32 En s'inspirant du codage du prédécesseur sur les entiers de Church, définir un λ-terme $\ulcorner tl \urcorner$ tel que pour tous λ-termes u, v_1, ..., v_k ($k \in \mathbb{N}$), on ait $\ulcorner tl \urcorner (\ulcorner :: \urcorner\ u\ [v_1; \cdots; v_k]) =_\beta [v_1; \cdots; v_k]$. On aura besoin des couples et il est recommandé d'utiliser $\ulcorner fold \urcorner$ appliqué à des termes Mul, One, Id (plus un dernier) que l'on définira.

10 Le λ-calcul en appel par valeur

La notion a été introduite par Gordon Plotkin en 1975. On définit les *valeurs* V :

$$\begin{array}{rl} V ::= x & \text{variables} \\ \mid \lambda x.v & \text{abstractions,} \end{array}$$

où v est un λ-terme arbitraire, pas nécessairement une valeur. La règle de réduction en *appel par valeur*, ou $\beta_{\mathtt{v}}$*-réduction*, est :

$$(\beta_{\mathtt{v}}) \qquad (\lambda x.u)V \to u[x := V],$$

où V est restreint à être une valeur. Cette règle s'applique sous n'importe quel contexte. On écrira $u \to_{\mathtt{v}} v$ pour dire que u se réécrit (en une étape) en v par cette règle, s'il y a besoin.

Exercice 33 Refaire l'**Exercice 2**, l'**Exercice 3**, et l'**Exercice 4**, avec la $\beta_{\mathtt{V}}$-réduction au lieu de la β-réduction. Comparer.

Exercice 34 Montrer que si $u \to_{\mathtt{V}}^* v$ et si v est β-normal (i.e., normal pour la β-réduction), alors v est l'unique forme normale de u en λ-calcul ordinaire. Exhiber un terme u qui a une forme normale $u\downarrow$ (pour la β-réduction), mais tel que $u \not\to_{\mathtt{V}}^* u\downarrow$; la $\beta_{\mathtt{V}}$-réduction est donc incomplète.

Exercice 35 Exhiber un terme normalisable (pour la β-réduction) mais qui n'a aucune forme normale pour la $\beta_{\mathtt{V}}$-réduction. La $\beta_{\mathtt{V}}$-réduction n'est donc pas standardisante.

Exercice 36 En adaptant la technique des réductions parallèles, montrer que la relation de $\beta_{\mathtt{V}}$-réduction est confluente.

Exercice 37 On définit la traduction suivante du λ-calcul vers lui-même, dite *traduction par passage à la continuation* (pour l'appel par valeur) :

$$cps^\circ(x) \stackrel{\text{def}}{=} x \qquad cps(V,\kappa) \stackrel{\text{def}}{=} \kappa(cps^\circ(V))$$
$$cps^\circ(\lambda x.u) \stackrel{\text{def}}{=} \lambda x,k.cps(u,k) \qquad cps(uv,\kappa) \stackrel{\text{def}}{=} cps(u,\lambda x.cps(v,\lambda y.xy\kappa))$$

où u, v sont des λ-termes, V désigne une valeur, κ est une valeur (la *continuation*), où k est une variable fraîche dans la définition de $cps^\circ(\lambda x.u)$, et où x et y sont des variables fraîches distinctes dans la définition de $cps(uv,\kappa)$. On gardera la même convention dans la suite. Montrer que :

(a) les variables libres de $cps^\circ(V)$ sont exactement celles de V, et les variables libres de $cps(u,\kappa)$ sont celles de u plus celles de κ ;

(b) pour toute valeur V_0, $V_0[z := V]$ est une valeur ;

(c) si k n'est pas libre dans u, alors $cps(u,\kappa)[k := \kappa'] = cps(u,\kappa[k := \kappa'])$;

(d) $cps(u,\kappa)[z := cps^\circ(V)] = cps(u[z := V])(\kappa[z := cps^\circ(V)])$ et $cps^\circ(V_0)[z := cps^\circ(V)] = cps^\circ(V_0[z := V])$ (pour toute valeur V_0) ;

(e) $cps((\lambda x.u)V,\kappa) \to^* cps(u[x := V],\kappa)$ (et ce, même par des réductions par valeur, et qui sont aussi de tête, et faibles) ;

(f) si $\kappa \to \kappa'$, alors $cps(u,\kappa) \to cps(u,\kappa')$ (et de même avec $\to_{\mathtt{V}}$ à la place de $\to$) ;

(g) si $u \to_{\mathtt{V}}^* v$, alors $cps(u,\kappa) \to^* cps(v,\kappa)$ (et ce, même par des réductions par valeur).

Donc la traduction par passage à la continuation simule correctement l'appel par valeur.

L'un des intérêts de la traduction par passage à la continuation vient du fait que les continuations κ ne sont jamais utilisées que sous la forme $\kappa(cps^\circ(V))$: le résultat final du calcul est le résultat retourné par κ, et l'on peut donc optimiser les appels à κ par un goto pointant vers le code de κ, sans rien empiler sur la pile. Un autre intérêt est que cette traduction s'étend à des λ-calculs enrichis de constructions permettant notamment le lancement et la récupération d'exception, et plus généralement la primitive `call/cc` de Scheme et de Standard ML of New Jersey.

11 Le théorème des développements finis

On se fixe un ensemble Σ de cardinal au moins 2. Une *étoile*, souvent notée $*$, est un ensemble non vide fini d'éléments de Σ.

On définit les λ^*-*termes* comme suit :

$$\begin{array}{rr} s,t,u,v,\cdots ::= x,y,z,\cdots & \text{variables} \\ \mid uv & \text{applications} \\ \mid \lambda x.u & \lambda\text{-abstractions} \\ \mid (\lambda^* x.u)v & (* \in \Sigma) \end{array}$$

La variable x est liée dans $\lambda x.u$ et dans $(\lambda^* x.u)v$. Sa portée est u dans les deux cas. On raisonne modulo (la version adéquate du) α-renommage.

Attention ! Un terme de la forme $(\lambda^* x.u)v$ n'est *pas* une application, malgré les apparences. On pourra y penser comme l'équivalent de la notation OCaml `let` $x = v$ `in` u (qui n'est pas une application, de façon plus claire). La notation $(\lambda^* x.u)v$ a des avantages, que l'on découvrira plus bas.

Attention aussi au fait que $\lambda^* x.u$, isolément, n'est *pas* une construction du langage.

L'unique règle de réduction du λ^*-calcul est :

$$(\beta^*) \qquad (\lambda^* x.u)v \to u[x := v].$$

Elle s'applique sous n'importe quel contexte. On notera que $(\lambda x.u)v$ n'est *pas* un rédex en λ^*-calcul.

Exercice 38 On note SN l'ensemble des termes fortement normalisables en λ^*-calcul, et $\overline{SN}$ l'ensemble des substitutions θ telles que $\theta(x) \in SN$ pour toute variable x. (Voir la section 2 pour la notion de substitution ; on pourra utiliser les résultats de cette section, qui restent vrais en λ^*-calcul, sans les redémontrer.) Montrer par récurrence sur la taille de u que si $u \in SN$, alors pour toute $\theta \in \overline{SN}$, $u\theta \in SN$. On pourra raisonner par l'absurde dans chaque cas, et analyser la forme des réécritures infinies supposées partant de $u\theta$; l'**Exercice 10** devrait montrer son utilité.

Exercice 39 Montrer que (β^*) est localement confluente. Quel théorème du cours garantit alors qu'elle est confluente ? On notera dans la suite $u\downarrow$ l'unique forme β^*-normale de tout λ^*-terme u.

Nous venons de démontrer le *théorème des développements finis* : le λ^*-calcul est fortement normalisant et confluent. J'expliquerai le nom plus bas.

Pour $a \in \Sigma$, disons qu'un λ^*-terme u est une *a-décoration* d'un λ^*-terme v si et seulement si on peut obtenir v à partir de u en effaçant un certain nombre d'occurrences de a dans u. (Donc, effacer a de $(\lambda^{\{a,b\}}x.u)v$ revient à remplacer l'exposant $\{a,b\}$ par $\{b\}$, notamment. Effacer a de $(\lambda^{\{a\}}x.u)v$ est par définition $(\lambda x.u)v$; on notera que ce dernier terme *est* une application.)

Notons que tout λ-terme peut être vu comme un λ^*-terme, sans aucune étoile. Réciproquement, un λ^*-terme sans étoile est un λ-terme.

Exercice 40 Soit $a \in \Sigma$. Quelles sont les a-décorations des λ-termes de l'**Exercice 2**, de l'**Exercice 3**, de l'**Exercice 4** ?

Exercice 41 On note $\to_1^a$ la relation entre λ-termes définie par $u \to_1^a v$ si et seulement s'il existe une a-décoration u^a de u dont la β^*-forme normale $(u^a)\downarrow$ soit exactement v. Montrer que :

(a) pour tous $a, b \in \Sigma$, pour tous λ-termes u et v, $u \to_1^a v$ si et seulement si $u \to_1^b v$; on notera dans la suite $\to_1$ pour la relation $\to_1^a$, vu qu'elle est indépendante de a ;

(b) pour tous λ^*-termes u et v tels que $u \to^* v$ en λ^*-calcul, toute a-décoration u' de u se réduit en au moins une a-décoration v' de v ; de plus, si v n'a aucune étoile contenant $b \in \Sigma$, alors v' non plus ;

(c) en déduire que la relation $\to_1$ est fortement confluente ;

(d) montrer que $\to \subseteq \to_1 \subseteq \to^*$, et en déduire une nouvelle démonstration de la confluence du λ-calcul.

Exercice 42 Bonus. Montrer que $\to_1$ et la relation de réduction parallèle $\Rightarrow$ du cours sont en fait la même relation.

Note historique : un *développement* d'un λ-terme u est une suite de réductions partant de u, où seuls les rédex *déjà existants* dans u (et leurs résiduels, aussi appelés radicaux, c'est-à-dire leurs recopies au cours de la réduction) sont contractés. Formellement, on obtient les développements de u en décorant d'abord u par un $a \in \Sigma$; il n'est possible de rajouter a que sur les λ en tête de rédexes déjà existants dans u. Toute réécriture partant de la a-décoration u' ainsi obtenue de u en λ^*-calcul mène à une réécriture partant de u en λ-calcul à condition d'effacer tous les a en exposant restants dans les termes de la réduction. Par définition, ces réécritures assez particulières sont les *développements* de u. Le théorème des développements finis dit que la relation de réécriture ainsi restreinte est fortement normalisante et confluente.

12 Réductions de tête

Exercice 43 Montrer que :

(a) si $u \to_{\mathrm{t}} u'$ alors $\lambda x.u \to_{\mathrm{t}} \lambda x.u'$;

(b) donner un contre-exemple montrant que $u \to_{\mathrm{t}} u'$ n'implique pas $uw \to_{\mathrm{t}} u'w$;

(c) montrer que si $\lambda x.u \to_{\mathrm{t}} v$, alors v est une λ-abstraction, et si l'on écrit, à α-renommage près, v sous la forme $\lambda x.u'$, alors $u \to_{\mathrm{t}} u'$; c'est une forme de réciproque de (a) ;

(d) montrer que si $u \to_{\mathrm{t}} u'$, alors $u[x := v] \to_{\mathrm{t}} u'[x := v]$;

Exercice 44 Montrer que tout β-réduit d'une forme normale de tête est normal de tête.

13 Réductions de tête faibles

Une *forme de tête faible* est une forme de tête qui n'est pas une λ-abstraction, autrement dit un terme de la forme $hu_1 \cdots u_m$, où soit h est une variable, soit h est une λ-abstraction et $m \geq 1$.

La relation de *réduction de tête faible* $\to_{\mathrm{tf}}$ est définie par $u \to_{\mathrm{tf}} u'$ si et seulement si u se réduit en u' par une réduction de tête, qui est en même temps faible (ne réduit pas sous un λ). Autrement dit, $u \to_{\mathrm{tf}} u'$ si et seulement si on peut écrire u sous la forme $(\lambda x.s)u_1 \cdots u_m$ avec $m \geq 1$, et $u' = s[x := u_1]u_2 \cdots u_m$.

La relation $\to_{\mathrm{tf}}$ n'est *pas* compatible aux contextes.

Exercice 45 Montrer que :

(a) si $u \to_{\mathrm{tf}} u'$ alors $uw \to_{\mathrm{tf}} u'w$ pour tout λ-terme w ;

(b) si $u \to_{\mathrm{t}}^* t$, et si t est une forme de tête faible $hu_1 \cdots u_m$, alors $u \to_{\mathrm{tf}}^* t$;

(c) donner un contre-exemple montrant que $u \to_{\mathrm{tf}} u'$ n'implique pas $\lambda x.u \to_{\mathrm{tf}} \lambda x.u'$.

14 Le théorème de standardisation

Nous donnons la démonstration de standardisation du λ-calcul due à René David.

On rappelle la définition de la relation $\Rightarrow_{\mathrm{s}}$: on définit $u \Rightarrow_{\mathrm{s}} v$ par récurrence sur la taille de v (pas de u !), si et seulement si :

$$\begin{array}{lccccc} u \xrightarrow[\mathrm{t}]{*} \lambda x_1, \cdots, x_n. & u_0 & u_1 & \cdots & u_m & \\ & \Vert_{\mathrm{s}} & \Vert_{\mathrm{s}} & & \Vert_{\mathrm{s}} & \\ \lambda x_1, \cdots, x_n. & v_0 & v_1 & \cdots & v_m & \overset{\text{tête}}{=\!=} v \end{array}$$

Par convention :

— tous les termes non définis sont quantifiés existentiellement ;

— $u_0 \Vert\!\Rightarrow_{\mathrm{s}} v_0$ signifie « v_0 est une variable x et alors $u_0 = x$, ou bien v_0 est de la forme $\lambda x.v'$, u_0 est de la forme $\lambda x.u'$, et $u' \Rightarrow_{\mathrm{s}} v$ » ;

— et $\lambda x_1, \cdots, x_n.v_0v_1 \cdots v_m \overset{\text{tête}}{=\!=} v$ sur la ligne du bas signifie que la forme de tête de v est exactement $\lambda x_1, \cdots, x_n.v_0v_1 \cdots v_m$ (à α-renommage près).

On notera bien qu'on ne demande pas que $\lambda x_1, \cdots, x_n.u_0u_1 \cdots u_m$ soit en forme normale de tête. C'est cependant une forme de tête, comme conséquence de $u_0 \Vert\!\Rightarrow_{\mathrm{s}} v_0$.

Exercice 46 Vérifier que la définition de $\Rightarrow_{\mathrm{s}}$ est bien formée, par récurrence sur la taille de v, autrement dit que $u \Rightarrow_{\mathrm{s}} v$ est bien définie en termes de relations $s \Rightarrow_{\mathrm{s}} t$ n'impliquant que des termes t strictement plus petits que v.

Exercice 47 Démontrer :

(a) $\Rightarrow_{\mathrm{s}}$ est réflexive : pour tout u, $u \Rightarrow_{\mathrm{s}} u$;

(b) si $u \rightarrow_{\mathrm{t}} v$ alors $u \Rightarrow_{\mathrm{s}} v$ (utiliser (a)) ;

(c) si $u \rightarrow_{\mathrm{t}} v \Rightarrow_{\mathrm{s}} w$, alors $u \Rightarrow_{\mathrm{s}} w$;

(d) si $\lambda x.u' \Rightarrow_{\mathrm{s}} v$, alors v s'écrit $\lambda x.v'$ et $v \Rightarrow_{\mathrm{s}} v'$;

(e) si $u \Rightarrow_{\mathrm{s}} v$ et $u' \Rightarrow_{\mathrm{s}} v'$, alors $uu' \Rightarrow_{\mathrm{s}} vv'$ (vous aurez besoin de l'**Exercice 45**) ;

(f) si $u \Rightarrow_{\mathrm{s}} v$ alors $\lambda x.u \Rightarrow_{\mathrm{s}} \lambda x.v$ (utiliser l'**Exercice 43**a et (e)) ;

(g) de (e) et (f), déduire que $\Rightarrow_{\mathrm{s}}$ est compatible aux contextes ;

(h) toujours en utilisant (e) et (f), montrer que si $u_i \Rightarrow_{\mathrm{s}} w_i$ pour tout i, $0 \leq i \leq m$, alors $\lambda x_1, \cdots, x_n.u_0u_1 \cdots u_m \Rightarrow_{\mathrm{s}} \lambda x_1, \cdots, x_n.v_0v_1 \cdots v_m$;

(i) montrer que si $u' \Rightarrow_{\mathrm{s}} w'$ et $u_1 \Rightarrow_{\mathrm{s}} w_1$, alors $u'[x := u_1] \Rightarrow_{\mathrm{s}} w'[x := w_1]$ (vous aurez besoin de (a), (f), (h), et (c))

(j) montrer le *lemme de tri* : si $u \Rightarrow_{\mathrm{s}} w \rightarrow v$, alors $u \Rightarrow_{\mathrm{s}} v$ (vous aurez besoin de (i) dans le cas crucial) ;

(k) en déduire que si $u \rightarrow^* v$, alors $u \Rightarrow_{\mathrm{s}} v$ (utiliser (a) et le lemme de tri (j)) ;

(l) conclure : $\Rightarrow_{\mathrm{s}}$ coïncide avec la relation $\rightarrow^*$.

Exercice 48 Montrer que :

(a) si $u \Rightarrow_{\mathrm{s}} v$ et v est en forme normale, alors $u \rightarrow^* v$ par une réduction gauche (utiliser l'**Exercice 47**) ;

(b) en déduire le théorème de *standardisation* : pour tout λ-terme u, u est normalisable si et seulement si sa réduction gauche termine.

15 Résolubilité et formes normales de tête

Si $\vec{u}$ est une suite finie de termes u_1, u_2, …, u_n, on notera $t\vec{u}$ pour $tu_1u_2\cdots u_n$. (On peut avoir $n = 0$.)

Un terme *clos* est un terme sans variable libre.

Un terme clos t est *résoluble* si et seulement s'il existe une suite finie $\vec{u}$ de λ-termes telle que $t\vec{u} =_\beta \mathbf{I}$, où $\mathbf{I}$ est par définition le terme $\lambda x.x$.

Un terme t, non nécessairement clos, est résoluble si et seulement si le terme $\lambda x_1, \cdots, x_m.t$, où x_1, …, x_m sont les variables libres de t, est résoluble. (Le terme en question est clos, et l'on applique donc la définition précédente de la résolubilité.) On appellera $\lambda x_1, \cdots, x_m.t$ la *fermeture* de t, même si elle n'est unique qu'à permutation des variables x_1, …, x_m près.

On dira qu'un λ-terme u *a une forme normale de tête* si et seulement si sa réduction de tête termine.

Exercice 49 Montrer que :

(a) si $u \to^* w$ et si w est en forme normale de tête, alors u a une forme normale de tête (utiliser la standardisation) ;

(b) si $u =_\beta v$ et si v est en forme normale de tête, alors u a une forme normale de tête ;

(c) u a une forme normale de tête si et seulement s'il existe un λ-terme v en forme normale de tête tel que $u =_\beta v$ (utiliser (b) et l'**Exercice 44**) ;

(d) une λ-abstraction $\lambda x.u$ a une forme normale de tête si et seulement si u a une forme normale de tête (utiliser (c) et l'**Exercice 43**c) ;

(e) si $u[x := v]$ a une forme normale de tête, alors u aussi (utiliser l'**Exercice 43**d) ;

(f) si uv a une forme normale de tête, alors u aussi (faire deux cas, et dans l'un des deux, utiliser (e) et (d)) ;

(g) en déduire que tout terme résoluble a une forme normale de tête.

Exercice 50 Montrer que :

(a) pour tout terme clos t en forme normale de tête, pour tout λ-terme v, il existe une liste finie $\vec{u}$ de λ-termes telle que $t\vec{u} \to_{\mathrm{t}}^* v$;

(b) en déduire (ainsi que de l'**Exercice 49**) que les conditions suivantes sont équivalentes pour un λ-terme t, pas nécessairement clos ni en forme normale de tête :

i. t a une forme normale de tête (à savoir, la réduction de tête partant de t termine) ;

ii. t est β-équivalent à une forme normale de tête ;

iii. t est résoluble ;

iv. en notant x_1, …, x_k les variables libres de t, pour tout λ-terme v, il existe une liste finie $u_1, \cdots, u_n$ de λ-termes, avec $n \geq k$, telle que $t[x_1 := u_1]\cdots[x_k := u_k]u_{k+1}\cdots u_n\vec{u} =_\beta v$.

(c) Vous avez maintenant tous les outils pour répondre à la requête suivante : exhiber un terme non résoluble.

Ces résultats sont dus à Christopher P. Wadsworth (1971).

16 Étiquettes de Hyland-Wadsworth

Un *monoïde à soulignement* est un quadruplet $M = (|M|, 1, ., \_)$, où $(|M|, 1, .)$ est un monoïde, et $\_$ est une fonction de $|M|$ dans $|M|$, qui à α associe un élément noté

$\underline{\alpha}$. Autrement dit, $|M|$ est un ensemble, appelé le *support* de M, l'élément *neutre* 1 est un élément de M ; pour tous $\alpha, \beta \in |M|$, on peut former le *produit* $\alpha.\beta \in M$ (on notera aussi $\alpha\beta$ au lieu de $\alpha.\beta$), de sorte que $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ (donnant ainsi un sens à la notation $\alpha\beta\gamma$), et $1\alpha = \alpha 1 = \alpha$; enfin pour tout $\alpha \in |M|$, le *souligné* $\underline{\alpha}$ est un élément de $|M|$. Le souligné n'a aucune propriété spéciale a priori.

Pour toute partie P d'un monoïde à soulignement M, on définit le $\lambda^{P,M}$-calcul, comme suit.

Les $\lambda^{P,M}$-termes sont obtenus en décorant chaque λ-terme par un élément de M, son *étiquette*. Autrement dit, la syntaxe du $\lambda^{P,M}$-calcul est donnée par :

$$\begin{array}{rcll} s,t,u,v,\ldots & ::= & M^\alpha & \text{terme étiqueté} \\ M,N,P,\ldots & ::= & x & \text{variable} \\ & | & uv & \text{application (de termes } \textit{étiquetés}) \\ & | & \lambda x.u & \text{abstraction (le corps } u \text{ est } \textit{étiqueté}) \end{array}$$

L'α-renommage est comme dans le λ-calcul, mais laisse les étiquettes invariantes. Par exemple, $(\lambda x.x^\alpha)^\beta$ est alpha-équivalente à $(\lambda y.y^\alpha)^\beta$, mais pas à $(\lambda y.y^\gamma)^\beta$ si $\alpha \neq \gamma$.

On peut *appliquer une étiquette* β à un terme étiqueté u pour obtenir un terme étiqueté noté $(u)^\beta$ (attention : les parenthèses font partie de la notation ; u^β n'a pas de sens ; seul M^β a un sens), défini par : tout terme étiqueté s'écrit de façon unique $u = M^\alpha$ pour un certain terme non étiqueté M, alors $(u)^\beta \stackrel{\text{def}}{=} M^{\alpha\beta}$.

On définit la substitution $u[x := v]$ (modulo alpha-renommage) comme suit. Notons que ceci utilise l'application des étiquettes, que nous venons de définir.

$$\begin{array}{rcll} x^\alpha[x := v] & = & (v)^\alpha & \\ y^\alpha[x := v] & = & y^\alpha & (y \text{ variable autre que } x) \\ (u_1u_2)^\alpha[x := v] & = & ((u_1[x := v])(u_2[x := v]))^\alpha & \\ (\lambda z.u)^\alpha[x := v] & = & (\lambda z.(u[x := v]))^\alpha & (z \text{ non libre dans } v \text{ et différente de } x) \end{array}$$

Finalement, on définit la (β^P)-réduction par la règle :

$$(\beta^P) \qquad ((\lambda x.u)^\alpha v)^\beta \to (u[x := (v)^{\underline{\alpha}}])^{\alpha\beta} \qquad (\text{si } \alpha \in P)$$

Donc on interdit la (β^P)-réduction si α n'est pas dans P ; et on effectue deux applications d'étiquettes, une à v avec l'étiquette α soulignée, et une au contracté tout entier. Une façon un peu plus simple de décrire ceci, quoique pas entièrement correcte, est $(\lambda x.u)^\alpha v \to (u[x := (v)^{\underline{\alpha}}])^\alpha$.

Par exemple, si P est suffisamment grand, on aura :

$$\begin{array}{rcl} ((\lambda x.(x^1x^2)^3)^4(\lambda x.(x^5x^6)^7)^8)^9 & \to & ((\lambda x.(x^5x^6)^7)^{8\underline{4}1}(\lambda x.(x^5x^6)^7)^{8\underline{4}2})^{349} \\ & \to & ((\lambda x.(x^5x^6)^7)^{8\underline{4}2\underline{8}\underline{\underline{4}}15}(\lambda x.(x^5x^6)^7)^{8\underline{4}2\underline{8}\underline{\underline{4}}16})^{78\underline{4}1349} \\ & \to & \ldots \end{array}$$

Dans le reste de cette section, nous utiliserons le monoïde à soulignement de *Hyland-Wadsworth*, défini comme suit :

— $|M| \stackrel{\text{def}}{=} \mathbb{N} \cup \{\infty\}$;
— l'élément neutre est ∞ ;
— le produit $\alpha\beta$ est défini comme $\min(\alpha, \beta)$;
— $\underline{\infty} \stackrel{\text{def}}{=} \infty$, $\underline{\alpha} \stackrel{\text{def}}{=} \alpha - 1$ si $\alpha \notin \{0, \infty\}$, $\underline{0} \stackrel{\text{def}}{=} 0$.

De plus, $P \stackrel{\text{def}}{=} \mathbb{N} \setminus \{0\}$. La (β^P)-réduction devient donc :

$$(\beta^P) \qquad ((\lambda x.u)^\alpha v)^\beta \to (u[x := (v)^{\alpha-1}])^{\min(\alpha,\beta)} \qquad (\text{si } \alpha \neq 0, \infty)$$

Dans cette section, on notera SN l'ensemble des termes étiquetés fortement normalisables. Posons aussi $\ell(u)$ l'*étiquette* de u, c'est-à-dire $\ell(M^\alpha) \stackrel{\text{def}}{=} \alpha$.

Exercice 51 Montrer que :

(a) $u \to v$ implique $\ell(u) \geq \ell(v)$;

(b) On définit une relation binaire $\triangleright$ et simultanément une notion de terme étiqueté *calculable*, comme suit. (Cette notion de calculabilité n'a pas grand chose à voir avec la notion usuelle de calculabilité.)

— $s \triangleright t$ si et seulement si $s \to t$, ou bien si s est de la forme $(\lambda x.u)^\alpha$, et $t = (u[x := (v)^{\alpha'}])^{\alpha''}$ pour certains $\alpha' < \alpha$, $\alpha'' \leq \alpha$, et pour un certain terme étiqueté v tel que $(v)^{\alpha'}$ est calculable ;

— s est *calculable* si et seulement s'il n'existe pas de chaîne infinie $s = s_0 \triangleright s_1 \triangleright \cdots \triangleright s_k \triangleright \cdots$.

Montrer que ceci est une définition correcte par récurrence sur $\ell(s)$.

(c) Montrer que tout terme calculable est fortement normalisable.

(d) Montrer que si $s \to t$ et s est calculable, alors t est calculable.

(e) Disons qu'un terme étiqueté est *neutre* si et seulement s'il n'est pas de la forme $(\lambda x.u)^\alpha$ pour aucune étiquette α. Montrer que si s est un terme étiqueté neutre dont tous les réduits en une étape (pour $\to$) sont calculables, alors s lui-même est calculable.

(f) En déduire que toute variable étiquetée x^α est calculable.

(g) Montrer que, si $(s)^\gamma \triangleright t$, alors il existe un terme étiqueté t' tel que $s \triangleright t'$ et $(t')^\gamma = t$.

(h) En déduire que si s est calculable, alors $(s)^\gamma$ est calculable pour tout $\gamma \in \mathbb{N}$.

(i) En utilisant (e), montrer que si s et t sont calculables, alors $(st)^\gamma$ est calculable pour tout $\gamma \in \mathbb{N} \cup \{\infty\}$. Vous aurez besoin d'effectuer une récurrence sur la somme $\nu(s)+\nu(t)$ des longueurs maximales de réduction de s et de t (utiliser (c) et le lemme de Kőnig pour donner un sens à ces quantités).

(j) Montrer que si $u[x := v]$ est calculable pour tout terme étiqueté calculable v, alors $(\lambda x.u)^\alpha$ est calculable pour tout $\alpha \in \mathbb{N} \cup \{\infty\}$. Vous aurez besoin d'utiliser (d) et (h).

(k) En déduire que tout terme à étiquettes de Hyland-Wadsworth est calculable, donc fortement normalisable.

Cette preuve de terminaison est due à Roel de Vrijer (2007).

Exercice 52 Proposer une traduction simple du λ^*-calcul vers le λ-calcul à étiquettes de Hyland-Wadsworth, et l'utiliser pour donner une démonstration simple de la terminaison du λ^*-calcul, comme corollaire de l'**Exercice 51**k. (On n'utilisera donc pas les résultats de la section 11.)

17 Étiquettes de Lévy

Soit A un ensemble non vide. On définit le monoïde à soulignement *libre* sur A, noté $L(A)$, comme suit :

— $|L(A)|$ est l'ensemble des *étiquettes de Lévy*, définies comme les mots $\alpha, \beta, \cdots ::= c_1 c_2 \cdots c_n$ où $n \in \mathbb{N}$ et chaque lettre c_i est soit dans A, soit un objet de la forme $\underline{\beta}_i$, où β_i est elle-même (récursivement) dans $L(A)$. Le souligné ici doit être compris comme un simple symbole. Autrement dit, une étiquette de Lévy est une forêt : la notation $c_1 c_2 \cdots c_n$ est une forêt à n racines, et c_i est une feuille si dans A, ou la racine d'un arbre dont les fils sont listés dans la forêt β_i, si de la forme $\underline{\beta}_i$.

— L'élément neutre est la forêt vide ϵ (le mot vide).

— Le produit est obtenu par concaténation.

— Le souligné est déjà défini.

La *hauteur* $h(\alpha)$ d'une étiquette de Lévy est la profondeur de la forêt α :

$$\begin{aligned} h(c_1 c_2 \cdots c_n) &\stackrel{\text{def}}{=} \max(h_1(c_1), h_1(c_2), \cdots, h_1(c_n)) \\ h_1(a) &\stackrel{\text{def}}{=} 0 \\ h_1(\underline{\beta}) &\stackrel{\text{def}}{=} h(\beta) + 1. \end{aligned}$$

Par convention, le max d'une liste vide vaut 0.

En instanciant la notion déjà introduite à la section 16, avec $M \stackrel{\text{def}}{=} L(A)$ et un sous-ensemble P que nous ferons varier, la réduction du calcul à étiquettes de Lévy est :

$$(\beta^P) \qquad ((\lambda x.u)^\alpha v)^\beta \to (u[x := (v)^{\underline{\alpha}}])^{\alpha\beta} \qquad (\text{si } \alpha \in P)$$

Exercice 53 Considérons un ensemble P de *hauteur bornée* : autrement dit, il existe un entier naturel n tel que pour toute étiquette α dans P, $h(\alpha) < n$. Pour tout λ-terme u à étiquettes de Lévy dans P, notons u^* le λ-terme à étiquettes de Wadsworth obtenu en remplaçant chaque étiquette de Lévy α dans u par $n - h(\alpha)$. Montrer les propriétés suivantes, où s, t, u, v, w sont des λ-termes à étiquettes de Lévy quelconques :

(a) $((w)^\alpha)^* = (w^*)^{n-h(\alpha)}$;

(b) $(u[x := w])^* = u^*[x := w^*]$;

(c) si $s \to t$ alors $s^* \to t^*$;

(d) en déduire que tout λ-terme à étiquettes de Lévy prises dans un ensemble P de hauteur bornée est fortement normalisable.

Exercice 54 Un *relèvement* d'un λ-terme t est un λ-terme à étiquettes de Lévy u tel que t s'obtient en effaçant toutes les étiquettes de u. (On dira aussi que t est l'*effacement* de u ; on pourra le noter $u = E(t)$.) Montrer que :

(a) pour tous λ-termes s, t_1, t_2 tels que $s \to^* t_1$ et $s \to^* t_2$, il existe des relèvements u, v_1, et v_2 de s, t_1 et t_2 respectivement tels que $u \to^* v_1$ et $u \to^* v_2$ en λ-calcul à étiquettes de Lévy ;

(b) pour tout ensemble P d'étiquettes de Lévy, montrer que la relation (β^P) est localement confluente ;

(c) en utilisant l'**Exercice 53**, en déduire que si P est à hauteur bornée, alors (β^P) est confluent ;

(d) en utilisant (a), en déduire une nouvelle preuve que le λ-calcul est confluent. (Cette preuve ne devra pas utiliser la confluence du λ-calcul, bien entendu, sinon cette preuve serait circulaire. La preuve est de surcroît directe : il n'y a pas à inventer de relation $\Rightarrow$ telle que $\to \subseteq \Rightarrow \subseteq \to^*$.)

Ces arguments sont dûs à Jean-Jacques Lévy, dans sa thèse d'état (1978). On peut aussi utiliser les étiquetages de Lévy pour donner une (autre) preuve de standardisation du λ-calcul.

Une autre application des étiquetages de Lévy est donnée à la section suivante.

18 Réductions quasi-internes

Une réduction $s \to t$ en λ-calcul à étiquettes de Lévy est *interne* si et seulement si le rédex contracté dans s n'a aucun sous-terme strict qui soit un rédex.

Par exemple, $((\lambda y.(\lambda x.x^{\zeta})^{\alpha}y^{\eta})^{\beta}z^{\theta})^{\gamma} \to ((\lambda y.y^{\eta\underline{\alpha}\zeta\alpha})^{\beta}z^{\theta})^{\gamma}$ est une réduction interne si $\alpha \in P$ (et n'est pas une réduction sinon). La réduction $((\lambda y.(\lambda x.x^{\zeta})^{\alpha}y^{\eta})^{\beta}z^{\theta})^{\gamma} \to ((\lambda x.x^{\zeta})^{\alpha}z^{\theta\underline{\beta}\eta})^{\beta\gamma}$ est une réduction interne si $\beta \in P$ et $\alpha \notin P$; si $\alpha, \beta \in P$ c'est une réduction qui n'est pas interne.

On notera $s \to_{\text{int}} t$ si s se réduit par une réduction en une étape qui est interne.

On définit une notion de réduction *quasi-interne* en λ-calcul comme suit. (La notion, appelée "inside-out reduction", a été introduite par Welch en 1975.) L'idée est la suivante. Lorsque l'on réduit s en t par β-réduction, les rédex non contractés dans s se retrouvent copiés un certain nombre de fois dans t (possiblement 0). Les rédex de t ainsi obtenus sont appelés les *résidus* de rédex de s. En général, t contient d'autres rédex, les rédex *créés* par la contraction. Par exemple, dans $(\lambda x.xx)(\lambda y.yy)$, l'unique rédex dans le contracté $(\lambda y.yy)(\lambda y.yy)$ est créé. Une réduction $s = s_0 \to s_1 \to \ldots \to s_n = t$ est intuitivement quasi-interne si et seulement si aucun rédex contracté dans cette réduction n'est résidu d'un rédex qui apparaît comme sous-terme d'un rédex contracté avant lui. Autrement dit, si on décide de contracter un rédex r, qui contient un rédex r' comme sous-terme (strict), on ne pourra plus jamais contracter (aucun résidu de) r' à l'avenir. On notera alors $s \Longrightarrow_{\text{io}} t$.

Plutôt que de formaliser la notion, on admettra que l'effacement d'une réduction interne en λ-calcul à étiquettes de Lévy est une réduction quasi-interne en λ-calcul. Autrement dit, on admettra que si $s = s_0 \to_{\text{int}} s_1 \to_{\text{int}} \cdots \to_{\text{int}} s_n = t$ par réductions internes en λ-calcul à étiquettes de Lévy, alors $E(s) \Longrightarrow_{\text{io}} E(t)$.

Exercice 55 Montrer que :

(a) les réductions quasi-internes sont *cofinales* en λ-calcul ; ceci signifie que si $s \to^* t$ en λ-calcul, alors il existe un λ-terme u tel que $s \Longrightarrow_{\text{io}} u$ et $t \to^* u$;

(b) en déduire que toute stratégie quasi-interne en λ-calcul est standardisante : si s a une forme normale t, alors $s \Longrightarrow_{\text{io}} t$.

La réduction gauche n'est donc pas l'unique statégie standardisante.

19 Implémentations du λ-calcul

Exercice 56 En cours, on a écrit deux interprètes du λ-calcul en Caml (`norm` et `norm2`). Compléter ces interprètes en écrivant la fonction de substitution manquante `subst`. Faites bien attention à gérer le α-renommage.

Exercice 57 On considère la machine de Krivine vue en cours :

$$\begin{array}{ll} \text{(explore)} & uv, args \rightsquigarrow u, v :: args \\ (\beta) & \lambda x.u, v :: rest \rightsquigarrow u[x := v], rest \end{array}$$

Il est entendu que $\rightsquigarrow$ ne passe pas aux contextes (quoi que ceci veuille dire ici). Montrer :

(a) le théorème de *correction* : si $u, [v_1; \cdots; v_n] \rightsquigarrow^* u', [v'_1; \cdots; v'_m]$, alors $uv_1 \cdots v_n \to^*_{\text{tf}} u'v'_1 \cdots v'_m$;

(b) le théorème de *progrès* : les configurations stoppées sont celles de la forme $x, [v_1; \cdots, v_n]$ ou $\lambda x.u, []$, et correspondent donc respectivement aux formes de tête faibles $xv_1 \cdots v_n$ et $\lambda x.u$.

Exercice 58 On définit une autre machine de Krivine comme suit. Ses configurations sont maintenant de la forme $u, args/vars$ où $args$ est une liste de λ-termes (comme avant), et $vars$ est une liste de variables. Les nouvelles configurations, de la

forme $u, [v_1; \cdots; v_m]/[x_n; \cdots, x_1]$, représentent le terme $\lambda x_1, \cdots, x_n.uv_1 \cdots v_m$. La machine elle-même est décrite par les règles :

$$\begin{aligned}
\text{(explore-lam)} \qquad & \lambda x.u, []/vars \rightsquigarrow u, []/x :: vars \\
\text{(explore-app)} \qquad & uv, args/vars \rightsquigarrow u, v :: args/vars \\
(\beta) \qquad & \lambda x.u, v :: rest/vars \rightsquigarrow u[x := v], rest/vars
\end{aligned}$$

Pour évaluer un λ-terme u en forme normale de tête, on démarre la machine dans la configuration $u, []/[]$. Démontrer les théorèmes de correction et de progrès associés. La notion de réduction implémentée en λ-calcul est la réduction de tête $\to_{\mathrm{t}}$, pas la réduction de tête faible.

Exercice 59 On souhaite implémenter la substitution de façon efficace, en le faisant de façon paresseuse. Pour ceci, on considère des termes étendus par une construction $u\langle x := v\rangle$, qui représente la donnée d'un terme u et d'une substitution à effectuer de x par v. Ceci se formalise par un nouveau calcul, le $\lambda\mathbf{x}$-calcul, dont la grammaire des termes est :

$$\begin{aligned}
M, N, \cdots ::= {} & x, y, z, \cdots && \text{variables} \\
\mid {} & MN && \text{applications} \\
\mid {} & \lambda x.M && \lambda\text{-abstractions} \\
\mid {} & M\langle x := N\rangle && \text{substitution explicite.}
\end{aligned}$$

(La construction $M\langle x := N\rangle$ correspond au `let` $x = N$ `in` M d'OCaml.) Il n'y a *pas* d'équivalence à α-renommage près pour ces termes. L'ensemble $\mathrm{fv}(M)$ des variables libres d'un $\lambda\mathbf{x}$-terme M est défini par :

$$\begin{aligned}
\mathrm{fv}(x) &\stackrel{\text{def}}{=} \{x\} & \mathrm{fv}(MN) &\stackrel{\text{def}}{=} \mathrm{fv}(M) \cup \mathrm{fv}(N) \\
\mathrm{fv}(\lambda x.M) &\stackrel{\text{def}}{=} \mathrm{fv}(M) \setminus \{x\} & \mathrm{fv}(M\langle x := N\rangle) &\stackrel{\text{def}}{=} (\mathrm{fv}(M) \setminus \{x\}) \cup \mathrm{fv}(N).
\end{aligned}$$

On se donne une machine de Krivine dont les configurations sont de la forme $M, env, args$, où :

- M est un $\lambda\mathbf{x}$-terme ;
- env est une liste $[\langle x_1 := N_1\rangle; \cdots; \langle x_k := N_k\rangle]$ de substitutions explicites $(k \in \mathbb{N})$;
- $args$ est une liste de $\lambda\mathbf{x}$-termes.

Lorsque $env = [\langle x_1 := N_1\rangle; \cdots; \langle x_k := N_k\rangle]$, on notera $M\ env$ le $\lambda\mathbf{x}$-terme $M\langle x_1 := N_1\rangle \cdots \langle x_k := N_k\rangle$. Les règles sont les suivantes :

$$\begin{aligned}
\text{(explore-app)} \qquad & MN, env, args \rightsquigarrow M, env, (N\ env) :: args \\
\text{(explore-sub)} \qquad & M\langle x := N\rangle, env, args \rightsquigarrow M, \langle x := N\rangle :: env, args \\
(\beta) \qquad & \lambda x.M, env, N :: args \rightsquigarrow M, \langle x := N\rangle :: env, args \\
\text{(subst-var-drop)} \qquad & y, \langle x := N\rangle :: env, args \rightsquigarrow y, env, args \qquad (\text{si } x \neq y) \\
\text{(subst-var)} \qquad & x, \langle x := N\rangle :: env, args \rightsquigarrow N, env, args
\end{aligned}$$

Chaque configuration $M, env, args$ représente un λ-terme $[\![M, env, args]\!]$, défini en deux temps. Dans un premier temps, on définit le λ-terme $[\![M]\!]$ que

représente chaque $\lambda\mathbf{x}$-terme M :

$$\begin{aligned}
[\![x]\!] &\stackrel{\text{def}}{=} x \\
[\![MN]\!] &\stackrel{\text{def}}{=} [\![M]\!]\,[\![N]\!] \\
[\![\lambda x.M]\!] &\stackrel{\text{def}}{=} \lambda x.\,[\![M]\!] \\
[\![M\langle x := N\rangle]\!] &\stackrel{\text{def}}{=} [\![M]\!]\,[x := [\![N]\!]]
\end{aligned}$$

Dans un deuxième temps, on pose :

$$[\![M, env, args]\!] \stackrel{\text{def}}{=} [\![M]\!]\,[x_1 := [\![N_1]\!]] \cdots [x_k := [\![N_k]\!]]\,[\![P_1]\!] \cdots [\![P_m]\!]$$

lorsque $env = [\langle x_1 := N_1\rangle; \cdots; \langle x_k := N_k\rangle]$ et $args = [P_1; \cdots; P_m]$.

(a) Montrer que cette machine n'est pas correcte (au moins sans restriction supplémentaire) : exhiber un λ-terme (un $\lambda\mathbf{x}$-terme sans substitution explicite) M et une configuration $N, env, args$ tels que $M, [], [] \rightsquigarrow^+ N, env, arg$, mais $[\![M]\!] \neq_\beta [\![N, env, args]\!]$. À titre d'indication, le problème est un problème de capture de variables dans la règle (β).

Le *domaine* $\mathrm{dom}(env)$ d'un environnement $env \stackrel{\text{def}}{=} [\langle x_1 := N_1\rangle; \cdots; \langle x_k := N_k\rangle]$ est l'ensemble $\{x_1, \cdots, x_k\}$. L'ensemble des variables *libres* $\mathrm{fv}(env)$ d'un tel environnement est $\bigcup_{i=1}^{k}(\mathrm{fv}(N_i) \setminus \{x_{i+1}, \cdots, x_k\})$. L'ensemble des variables libres $\mathrm{fv}(args)$ d'une liste $args \stackrel{\text{def}}{=} [P_1; \cdots; P_m]$ est $\bigcup_{j=1}^{m} \mathrm{fv}(P_j)$. L'ensemble des variables libres $\mathrm{fv}(M, env, args)$ d'une configuration $M, env, args$ est $(\mathrm{fv}(M) \setminus \mathrm{dom}(env)) \cup \mathrm{fv}(env) \cup \mathrm{fv}(args)$. Une configuration est *close* si et seulement son ensemble de variables libres est vide.

(b) Vérifier que si $M, env, args \rightsquigarrow M', env', args'$, alors $\mathrm{fv}(M', env', args') \subseteq \mathrm{fv}(M, env, args)$. Vous aurez besoin de démontrer que $\mathrm{fv}(N\ env) = (\mathrm{fv}(N) \setminus \mathrm{dom}(env)) \cup \mathrm{fv}(env)$. En déduire que la relation $\rightsquigarrow$ se restreint aux configurations closes, autrement que si $M, env, args \rightsquigarrow M', env', args'$ et si $M, env, args$ est close, alors $M', env', args'$ aussi.

(c) Montrer que $\mathrm{fv}([\![M, env, args]\!]) \subseteq \mathrm{fv}(M, env, args)$; montrer par un contre-exemple que l'inclusion peut être stricte.

(d) Montrer que la machine de Krivine ci-dessus est correcte pour les réductions de tête faibles *closes* : si $M, env, args \rightsquigarrow M', env', args'$ et si $M, env, args$ est close, alors $[\![M, env, args]\!] \to_{\text{tf}}^* [\![M', env', args']\!]$. On rappelle que, contrairement aux $\lambda\mathbf{x}$-termes, les λ-termes sont interprétés modulo α-renommage.

(e) Montrer que les configurations closes stoppées sont celles de la forme $\lambda x.M, env, []$, et en déduire un théorème de progrès. Cette machine implémente donc exactement les réductions de tête faible, sans effectuer explicitement aucune opération de substitution ni d'α-renommage.

20 La fin du théorème de Kleene

Cette section fortement influencée par le cours de Jean Gallier à l'université de Pennsylvanie (`https://www.seas.upenn.edu/~cis5110/notes/cis511-c-sl5.pdf`.

On reprend les notations de la section 8. Pour toute fonction récursive (partielle) $f\colon \mathbb{N}^k \to \mathbb{N}$, et pour tout λ-terme $\ulcorner f \urcorner$, on dira que $\ulcorner f \urcorner$ *code fidèlement* f si et seulement si :

(A) il code correctement f, autrement dit pour tous entiers naturels $n_1, \ldots, n_k$ tels que $f(n_1, \cdots, n_k)$ est définie, $\ulcorner f \urcorner \ulcorner n_1 \urcorner \cdots \ulcorner n_k \urcorner \to^* \ulcorner f(n_1, \cdots, n_k) \urcorner$;

(B) et pour tous entiers naturels $n_1, \ldots, n_k$ tels que $f(n_1, \cdots, n_k)$ n'est pas définie, la réduction de tête de $\ulcorner f \urcorner \ulcorner n_1 \urcorner \cdots \ulcorner n_k \urcorner$ ne termine pas.

On dit que $\ulcorner f \urcorner$ *code* f si et seulement si la condition (A) est vérifiée, pas nécessairement (B). Si $\ulcorner f \urcorner$ code fidèlement f, alors en particulier $\ulcorner f \urcorner$ code f. La réciproque est vraie si f est une fonction totale, puisqu'alors (B) est trivialement vraie.

Exercice 60 Montrer que :

(a) le λ-terme $\lambda z_1, \cdots, z_k.\ulcorner n \urcorner$ code fidèlement la fonction constante à k paramètres retournant toujours $n \in \mathbb{N}$;

(b) $\ulcorner S \urcorner \stackrel{\text{def}}{=} \lambda z, f, x.f(zfx)$ code fidèlement la fonction successeur ;

(c) $\ulcorner \pi \urcorner_i^k \stackrel{\text{def}}{=} \lambda z_1, \cdots, z_k.z_i$ code fidèlement la projection des k-uplets sur leur $i$ième composante ;

Exercice 61 On reprend les notations de l'**Exercice 28**. Soit f une fonction récursive totale de $\mathbb{N}^{k+1}$ dans $\mathbb{N}$, codée (nécessairement fidèlement) par un λ-terme $\ulcorner f \urcorner$. Montrer que $\ulcorner \mu f \urcorner$ code μf fidèlement.

Exercice 62 On sait que si $\ulcorner f \urcorner$ code f et chaque $\ulcorner g_i \urcorner$ code g_i, alors $\ulcorner f \circ \langle g_1, \cdots, g_k \rangle \urcorner \stackrel{\text{def}}{=} \lambda z_1, \cdots, z_\ell.\ulcorner f \urcorner (\ulcorner g_1 \urcorner z_1 \cdots z_\ell) \cdots (\ulcorner g_k \urcorner z_1 \cdots z_\ell)$ code $f \circ \langle g_1, \cdots, g_k \rangle$ (on supposera f d'arité k et chaque g_i d'arité ℓ).

(a) Montrer à l'aide d'un contre-exemple que $\ulcorner f \circ \langle g_1, \cdots, g_k \rangle \urcorner$ ne code pas nécessairement fidèlement $f \circ \langle g_1, \cdots, g_k \rangle$, même si $\ulcorner f \urcorner$ code fidèlement f et chaque $\ulcorner g_i \urcorner$ code fidèlement g_i.

(b) On pose $\mathbf{K} \stackrel{\text{def}}{=} \lambda x, y.x$, $\mathbf{I} \stackrel{\text{def}}{=} \lambda x.x$. Montrer que $\ulcorner n \urcorner (\mathbf{KI})\mathbf{I} \to^* \mathbf{I}$ pour tout entier n, et que si u la réduction de tête ne termine pas, alors celle de $u(\mathbf{KI})\mathbf{I}$ non plus (utiliser l'**Exercice 49** (f)). En déduire que si $\ulcorner f \urcorner$ code fidèlement f et chaque $\ulcorner g_i \urcorner$ code fidèlement g_i, alors :

$$\begin{aligned}\ulcorner f \circ \langle g_1, \cdots, g_k \rangle \urcorner' \stackrel{\text{def}}{=} \lambda z_1, \cdots, z_\ell.\ & \\ & \ulcorner g_1 \urcorner z_1 \cdots z_\ell (\mathbf{KI})\mathbf{I})(\\ & \ulcorner g_2 \urcorner z_1 \cdots z_\ell (\mathbf{KI})\mathbf{I})(\\ & \cdots \\ & (\ulcorner g_k \urcorner z_1 \cdots z_\ell (\mathbf{KI})\mathbf{I}(\\ & \quad \ulcorner f \urcorner (\ulcorner g_1 \urcorner z_1 \cdots z_\ell) \cdots (\ulcorner g_k \urcorner z_1 \cdots z_\ell)) \cdots)\end{aligned}$$

code fidèlement $f \circ \langle g_1, \cdots, g_k \rangle$.

(c) En s'inspirant de (b), donner un λ-terme $\ulcorner R_{f,g} \urcorner'$ qui code fidèlement $R_{f,g}$, en fonction d'un λ-terme $\ulcorner f \urcorner$ codant fidèlement f et d'un λ-terme codant fidèlement g. (Voir l'**Exercice 27** pour le codage usuel $\ulcorner R_{f,g} \urcorner$ de $R_{f,g}$.)

(d) En déduire que pour toute fonction récursive f, il existe un λ-terme $\ulcorner f \urcorner$ qui la code fidèlement.

Une autre démonstration, plus légère du côté du λ-calcul et plus lourde du côté de la théorie de la calculabilité, est donnée ci-dessous. Le théorème de *forme normale de Kleene* énonce que toute fonction partielle récursive $f \colon \mathbb{N}^m \to \mathbb{N}$ peut s'écrire comme $g \circ \mu h$, où g est une fonction primitive récursive de $\mathbb{N}$ dans $\mathbb{N}$ et h est une fonction primitive récursive de $\mathbb{N}^{m+1}$ dans $\mathbb{N}$.

Exercice 63 Montrer que si $\ulcorner g \urcorner$ code (fidèlement) une fonction récursive totale g et si $\ulcorner h \urcorner$ code (fidèlement) une fonction récursive totale, alors il existe un λ-terme qui code fidèlement $g \circ \mu h$. (Distinguer les cas où g est une fonction constante et où g n'est pas constante.) En utilisant le théorème de forme normale de Kleene, en conclure que pour toute fonction récursive f, il existe un λ-terme $\ulcorner f \urcorner$ qui la code fidèlement.

21 Un codage direct des machines de Turing vers le λ-calcul

On rappelle que l'on peut coder les listes de λ-termes en λ-calcul (section 9). Soit Σ un alphabet fini non vide.

On numérotera les éléments de Σ de 1 à n, $n \geq 1$. Nous coderons la lettre numéro i par l'entier de Church $\ulcorner i \urcorner$. On peut alors coder n'importe quel mot sur Σ par la liste de ses lettres.

De même, nous supposerons que les états q d'une machine de Turing M sont numérotés de 1 à un entier N, et $\ulcorner q \urcorner$ est l'entier de Church correspondant.

Une configuration (ℓ, q, r) d'une machine de Turing M, où ℓ est le mot à gauche de la tête, q est l'état de contrôle, et r est le mot à droite de la tête, peut se coder comme un triplet $\langle \ulcorner q \urcorner, \langle \ulcorner \ell^{op} \urcorner, \ulcorner r \urcorner \rangle \rangle$, où ℓ^{op} est le mot ℓ écrit à l'envers ; l'intérêt est que la première lettre de ℓ^{op} (si ℓ est non vide) est la lettre immédiatement à gauche de la tête sur la bande de M.

On notera $\ulcorner \ell, q, r \urcorner$ comme abréviation de $\langle \ulcorner q \urcorner, \langle \ulcorner \ell \urcorner^{op}, \ulcorner r \urcorner \rangle \rangle$.

Exercice 64 Écrire un λ-terme $\mathtt{step}_M$ tel que pour toutes configurations (ℓ, q, r) et (ℓ', q', r'), où (ℓ', q', r') est obtenue en simulant M pendant une étape à partir de (ℓ, q, r), alors :

$$\mathtt{step}_M \ulcorner \ell, q, r \urcorner =_\beta \ulcorner \ell', q', r' \urcorner.$$

Exercice 65 On suppose que la machine M a un seul état d'arrêt, numéroté 2, l'état initial étant le numéro 1. On supposera aussi que les états de contrôle sont numérotés de 1 à N, et on réserve les numéros $N+1, \ldots,$ à l'alphabet de bande. Écrire un λ-terme $\mathtt{stop}_M$ tel que pour toute configuration (ℓ, q, r),

$$\begin{aligned} \mathtt{stop}_M \ulcorner \ell, q, r \urcorner &=_\beta \mathbf{V} && \text{si } q = 2 \\ \mathtt{stop}_M \ulcorner \ell, q, r \urcorner &=_\beta \mathbf{F} && \text{sinon.} \end{aligned}$$

Exercice 66 En utilisant les questions précédentes, écrire un λ-terme $\mathtt{simul}_M$ tel que :

$\mathtt{simul}_M \ulcorner \ell, q, r \urcorner =_\beta \mathbf{V}$ si M termine à partir de la configuration (ℓ, q, r)

$\mathtt{simul}_M \ulcorner \ell, q, r \urcorner$ n'a pas de forme normale de tête sinon.

On utilisera la standardisation pour justifier cette dernière affirmation.

Exercice 67 En déduire une démonstration directe que le langage des λ-termes u tels que $u =_\beta \mathbf{V}$ est indécidable. Pourquoi est-il corécursivement énumérable ? (Encore une fois, penser à la standardisation.)

Exercice 68 Un corollaire immédiat de la question précédente est que la β-équivalence est indécidable, autrement dit que le langage des couples (u, v) tels que $u =_\beta v$ est indécidable. Montrer que ce langage est corécursivement énumérable.

22 Le théorème de Scott-Curry

On se fixe une énumération calculable $n \mapsto t_n$ des λ-termes. (On code donc les λ-termes sous forme de mots, mais nous laisserons implicite ce codage.) On note $t \mapsto [\![t]\!]$ la fonction inverse.

Exercice 69 Justifier que la fonction qui à tout $n \in \mathbb{N}$ associe $\ulcorner[\![t_n \ulcorner n \urcorner]\!]\urcorner$ est calculable. On ne donnera pas la machine de Turing explicitement, et on se contentera d'une description à haut niveau d'un algorithme.

Exercice 70 En déduire :

(a) qu'il existe un λ-terme δ tel que $\delta \ulcorner n \urcorner =_\beta \ulcorner[\![t_n \ulcorner n \urcorner]\!]\urcorner$ pour tout $n \in \mathbb{N}$;

(b) et donc, pour chaque λ-terme u, un λ-terme B_u tels que $B_u \ulcorner n \urcorner =_\beta u \ulcorner[\![t_n \ulcorner n \urcorner]\!]\urcorner$ pour tout $n \in \mathbb{N}$ (on explicitera B_u en fonction de δ) ;

(c) enfin, pour chaque λ-terme u, un λ-terme A_u tel que $A_u =_\beta u \ulcorner[\![A_u]\!]\urcorner$ (poser $n \stackrel{\text{def}}{=} [\![B_u]\!]$) ; l'existence de ce terme pour chaque u est le *deuxième théorème de point fixe*.

Exercice 71 On dit qu'un ensemble X de λ-termes est *β-saturé* si et seulement pour tout $u \in X$, pour tout $v =_\beta u$, v est dans X. On dit que deux ensembles X et Y de λ-termes sont *séparés* par un ensemble L si et seulement si $X \subseteq L$ et $Y \cap L = \emptyset$. X et Y sont *récursivement séparables* si et seulement si X et Y sont séparés par un ensemble décidable de λ-termes.

(a) On suppose X et Y séparés par un ensemble décidable L. Pourquoi existe-t-il un λ-terme D tel que pour tout entier $n \in \mathbb{N}$, $D \ulcorner n \urcorner =_\beta \mathbf{V}$ si et seulement si $t_n \in L$, et $D \ulcorner n \urcorner =_\beta \mathbf{F}$ si et seulement si $t_n \notin L$?

(b) En utilisant le deuxième théorème de point fixe, pour tous λ-termes t, u, v, construire un λ-terme j tel que :

— si $t \ulcorner[\![j]\!]\urcorner =_\beta \mathbf{V}$, alors $j =_\beta v$;

— et si $t \ulcorner[\![j]\!]\urcorner =_\beta \mathbf{F}$, alors $j =_\beta u$.

(c) En déduire le *théorème de Scott-Curry* : deux ensembles β-saturés non vides de λ-termes ne sont jamais récursivement séparables. (Supposer $u \in X$, $v \in Y$, poser $t \stackrel{\text{def}}{=} D$, et appliquer les questions précédentes.) C'est un analogue λ-calculatoire du théorème de Rice.

23 Indices de de Bruijn

La notation de de Bruijn (le « ui » néerlandais se prononce « euil ») permet d'éviter le α-renommage : on y remplacer toute occurrence de variable liée par un numéro, son *indice de de Bruijn*, qui compte combien de symboles λ on doit parcourir en remontant dans le λ-terme jusqu'à celui qui lie la variable.

La syntaxe des *termes de de Bruijn* est :

$$\begin{array}{rr} M, N, \cdots ::= x, y, z, \cdots & \text{variable libre} \\ \mid 1, 2, 3, \cdots & \text{indice de de Bruijn} \\ \mid MN & \text{application} \\ \mid \lambda M & \text{abstraction.} \end{array}$$

On notera que la variable liée par une abstraction (x dans $\lambda x.u$) disparaît dans ces termes.

On traduit chaque λ-terme t en un terme de de Bruijn t^* comme suit. On utilise une traduction auxilaire $t^*(\ell)$, définie par récurrence sur t, où ℓ est une liste de variables ; l'idée est que si $\ell = [x_1; \cdots, x_n]$, on traduit t tout en remplaçant la

variable x_i par l'indice de de Bruijn i.

$$x^*([x_1;\cdots;x_n]) \stackrel{\text{def}}{=} i \qquad \text{si } i \text{ est le plus petit entier tel que } x = x_i$$
$$x^*([x_1;\cdots;x_n]) \stackrel{\text{def}}{=} x \qquad \text{si } x \neq x_1,\cdots,x_n$$
$$(uv)^*(\ell) \stackrel{\text{def}}{=} u^*(\ell)(v^*(\ell))$$
$$(\lambda x.u)^*(\ell) \stackrel{\text{def}}{=} \lambda(u^*(x :: \ell)) \qquad \text{où } x \text{ est fraîche (pas dans } \ell).$$

On pose finalement $t^* \stackrel{\text{def}}{=} t^*([])$.

Exercice 72 On définit $Abs_n(x, M)$, pour tout entier $n \geq 1$, toute variable x et tout terme de de Bruijn M, par :

$$Abs_n(x, x) \stackrel{\text{def}}{=} n$$
$$Abs_n(x, y) \stackrel{\text{def}}{=} y \qquad \text{si } y \text{ est une variable autre que } x$$
$$Abs_n(x, i) \stackrel{\text{def}}{=} i \qquad \text{si } i < n$$
$$Abs_n(x, i) \stackrel{\text{def}}{=} i + 1 \qquad \text{si } i \geq n$$
$$Abs_n(x, MN) \stackrel{\text{def}}{=} Abs_n(x, M)(Abs_n(x, N))$$
$$Abs_n(x, \lambda M) \stackrel{\text{def}}{=} \lambda(Abs_{n+1}(x, M)).$$

Montrer l'égalité $u^*(x :: \ell) = Abs_1(x, u^*(\ell))$ pour toute variable x hors de ℓ. (Vous aurez besoin de démontrer la proposition plus générale $u^*(x_1 :: x_2 :: \cdots :: x_k :: x :: \ell) = Abs_{k+1}(x, u^*(x_1 :: x_2 :: \cdots :: x_k :: \ell))$ si $x \neq x_1, x_2, \cdots, x_k$ et x n'est pas dans ℓ.) En déduire qu'une définition équivalente de t^* est :

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*v^*$$
$$(\lambda x.u)^* \stackrel{\text{def}}{=} \lambda(Abs_1(x, u^*)).$$

Exercice 73 Montrer que pour tous λ-termes u et v, u et v sont α-équivalents si et seulement si u^* et v^* sont *égaux*.

Exercice 74 On définit l'opération de substitution d'indice de de Bruijn $M[n \leftarrow P]$, ainsi que l'opération U_k^n de mise à jour d'indices, par :

$$x[n \leftarrow P] \stackrel{\text{def}}{=} x \qquad U_k^n(x) \stackrel{\text{def}}{=} x$$
$$m[n \leftarrow P] \stackrel{\text{def}}{=} \begin{cases} m & \text{si } m < n \\ U_0^n(P) & \text{si } m = n \\ m-1 & \text{si } m > n \end{cases} \qquad U_k^n(p) \stackrel{\text{def}}{=} \begin{cases} p+n-1 & \text{si } p > k \\ p & \text{sinon} \end{cases}$$
$$(MN)[n \leftarrow P] \stackrel{\text{def}}{=} M[n \leftarrow P](N[n \leftarrow P]) \qquad U_k^n(MN) \stackrel{\text{def}}{=} U_k^n(M)(U_k^n(N))$$
$$(\lambda M)[n \leftarrow P] \stackrel{\text{def}}{=} \lambda(M[n+1 \leftarrow P]) \qquad U_k^n(\lambda M) \stackrel{\text{def}}{=} \lambda(U_{k+1}^n(M)).$$

Montrer l'égalité $(u[x := v])^* = Abs_1(x, u^*)[1 \leftarrow v^*]$. Vous aurez besoin de lemmes intermédiaires, que vous énoncerez et prouverez soigneusement.

Exercice 75 En déduire que la règle suivante :

$$(Beta) \qquad (\lambda M)N \to M[1 \leftarrow N]$$

implémente correctement la β-réduction, au sens où $u \to v$ (en λ-calcul) implique $u^* \to v^*$ (par la règle $(Beta)$).

Exercice 76 On définit une traduction inverse $M^\circ(\ell)$, où ℓ est une liste de λ-termes (qui seront, en pratique, toujours des variables), par :

$$\begin{aligned}
x^\circ(\ell) &\stackrel{\text{def}}{=} x \\
i^\circ([w_1;\cdots;w_n]) &\stackrel{\text{def}}{=} w_i && \text{si } 1 \leq i \leq n \\
&\quad \text{indéfini} && \text{sinon} \\
(MN)^\circ(\ell) &\stackrel{\text{def}}{=} M^\circ(\ell)(N^\circ(\ell)) \\
(\lambda M)^\circ(\ell) &\stackrel{\text{def}}{=} \lambda x.M^\circ(x :: \ell) && \text{où } x \text{ est une variable fraîche.}
\end{aligned}$$

Par « variable fraîche », on entend non libre ni dans M ni dans aucun λ-terme de ℓ. On notera que $M^\circ(\ell)$ n'est définie que si ℓ est suffisamment longue. Montrer que :

(a) $M^\circ(\ell)$ est défini si et seulement si $d(M) \leq |\ell|$, où $|\ell|$ est la longueur de ℓ, et la *profondeur de de Bruijn* $d(M)$ d'un terme de de Bruijn M est définie par :

$$\begin{aligned}
d(x) &\stackrel{\text{def}}{=} 0 \\
d(i) &\stackrel{\text{def}}{=} i \\
d(MN) &\stackrel{\text{def}}{=} \max(d(M), d(N)) \\
d(\lambda M) &\stackrel{\text{def}}{=} \max(d(M) - 1, 0).
\end{aligned}$$

(b) pour tout λ-terme u, pour toute liste de variables ℓ, $d(u^*(\ell)) \leq |\ell|$;

(c) pour tout λ-terme u, pour toute liste de variables ℓ, $(u^*(\ell))^\circ(\ell)$ est défini, et α-équivalent à u ;

(d) pour tout terme de de Bruijn M, pour toute liste de variables ℓ telle que $M^\circ(\ell)$ est définie, alors $(M^\circ(\ell))^*(\ell) = M$;

(e) si $M \to N$ par $(Beta)$, alors $d(N) \leq d(M)$;

(f) si $M \to N$ par $(Beta)$, et si $M^\circ(\ell)$ est défini, alors $N^\circ(\ell)$ est défini et $M^\circ(\ell) \to N^\circ(\ell)$ par β-réduction ;

(g) en déduire que la règle $(Beta)$, restreinte à l'ensemble dB_0 termes de de Bruijn de profondeur 0, implémente correctement et fidèlement la β-réduction, au sens suivant : il existe une bijection $t \mapsto t^*$ entre l'ensemble Λ des λ-termes modulo α-équivalence et dB_0, telle que $u \to v$ dans Λ si et seulement si $u^* \to v^*$ dans dB_0.

24 Combinateurs de Curry-Schönfinkel

On définit un langage dits de *termes combinatoires*, comme suit :

$$\begin{aligned}
M, N, P, \cdots ::= x, y, z, \cdots && \text{variables} \\
\mid MN && \text{applications} \\
\mid \mathrm{S} \mid \mathrm{K} \mid \mathrm{I} && \text{combinateurs.}
\end{aligned}$$

(Un combinateur signifie usuellement un λ-terme clos. Le sens est détourné ici.) Les règles de réduction sont :

$$\begin{aligned}
\mathrm{S}MNP &\to MP(NP) \\
\mathrm{K}MN &\to M \\
\mathrm{I}M &\to M
\end{aligned}$$

Elles s'appliquent sous tout contexte. Comme en λ-calcul, MNP signifie $(MN)P$, etc. Une variable x est *libre* dans M si et seulement si elle apparaît dans M. Les termes combinatoires n'ont aucune construction qui lierait des variables, telle que le λ du λ-calcul.

On définit une traduction $u \mapsto u^*$ des λ-termes vers les termes combinatoires par :

$$\begin{array}{llll} x^* \stackrel{\text{def}}{=} x & & [x]x \stackrel{\text{def}}{=} \mathrm{I} & \\ (uv)^* \stackrel{\text{def}}{=} u^*v^* & & [x]M \stackrel{\text{def}}{=} \mathrm{K}M & \text{si } x \text{ pas libre dans } M \\ (\lambda x.u)^* \stackrel{\text{def}}{=} [x]u^* & [x](MN) \stackrel{\text{def}}{=} \mathrm{S}([x]M)([x]N) & & \text{si } x \text{ libre dans } MN. \end{array}$$

La construction $[x]M$ est une construction externe au langage, qui produit un terme combinatoire à partir d'un terme combinatoire M et d'une variable x.

Exercice 77 Montrer que :

(a) $([x]M)N \to^* M[x := N]$ par les règles de réduction des termes combinatoires ;

(b) si $u \to v$ par réduction *faible* en λ-calcul, alors $u^* \to^* v^*$ dans les termes combinatoires ;

(c) donner un contre-exemple montrant que le point prédécent n'est plus vrai si $u \to v$ par une réduction sous un λ ;

(d) montrer que si $u \to_{\mathrm{tf}} v$, alors $u^* \to_{\mathrm{tf}}^* v^*$, où la relation $\to_{\mathrm{tf}}$ sur les termes combinatoires est définie par $M \to_{\mathrm{tf}} N$ si et seulement M s'écrit $LP_1 \cdots P_m$, N s'écrit $RP_1 \cdots P_m$, et $L \to R$ est l'une des trois règles de réduction entre termes combinatoires ;

(e) décrire une machine de Krivine pour les termes combinatoires qui implémente la réduction de tête faible des termes combinatoires.

Exercice 78 Montrer par une technique de réductions parallèles que la réduction des termes combinatoires est confluente.