# DM de $\lambda$-calcul : standardisation et représentation des fonctions récursives
# DM du 26 novembre 2021

Alexis Saurin

---

***Remarque préliminaire :*** Ce devoir vous permettra de vous exercicer sur le $\lambda$-calcul tout en revoyant quelques notions basiques de calculabilité.

Il n'est pas obligatoire de rendre ce DM qui est optionnel mais nous servira lors des révisions. Si vous le souhaitez, vous pouvez me rendre votre travail pour le 2 décembre, par email, ou au début du cours du 3 décembre.

Si vous souhaitez rendre une copie, celle-ci doit constituer un travail entièrement personnel et ne doit pas, même en partie, être dérivée du travail d'autres étudiants, ni résulter d'une recherche sur internet ou dans un livre ou autres cours liés : mieux vaut sauter une question en indiquant que vous n'êtes pas parvenus à la résoudre – ou mieux : en indiquant ce qui vous bloquait – ou que vous avez eu l'occcasion de prendre connaissance de la solution sur telle ou telle ressources ou auprès d'un ou une camarade, plutôt que de recopier une solution qui n'est pas la vôtre.

Vous pouvez biensûr discuter de certaines questions du DM avec d'autres étudiants du cours mais si vous me rendez une solution pour une question dont vous avez discuté, je vous demanderai de respecter les règles suivantes (ces règles sont recommandées même si vous ne rendez pas de copie !) :

- Il est attendu de vous un travail **personnel** : vous ne devez en aucun cas consulter ou vous inspirer du DM écrit par un ou une autre étudiante.
- Vous devez avoir déjà travaillé sur une partie du DM **avant toute discussion** à ce sujet avec quelqu'un d'autre.
- Si vous discutez du sujet, parlez-en oralement plutôt que par écrit (mail, chat, etc.). En particulier, je vous demande de pas discuter du sujet de DM via *Discord*, à la fois pour respecter la règle d'oralité des échanges et pour permettre à celles et ceux qui souhaitent travailler de manière complètement autonome de ne pas être exposés aux solutions.
- Lors de ces discussion, essayez de prendre le moins de notes possibles pour vous forcer à refaire par vous-même les raisonnements ou exemples qui auront été discutés, plutôt qu'à les recopier. Cette règle, qui est impérative si vous souhaitez rendre une solution pour la partie du DM correspondante, reste recommandée dans tous les cas : c'est votre capacité à refaire par vous même les raisonnements qui auront été discutés qui vous permettra d'apprécier la compréhension que vous aurez acquise.
- Si vous rendez une copie, je vous invite à citer vos sources, y compris les personnes avec lesquelles vous aurez discuté du partiel.

Tout ceci repose avant tout sur la confiance et ces règles visent à rendre l'exercice le plus profitable pour vous.

Je vous souhaite bon travail et bon courage !

*Alexis Saurin*

---

**Table des matières**

## 1. Standardisation sous hypothèse de normalisation forte

On rappelle que dans un $\lambda$-terme qui n'est pas en forme normale, on peut toujours trouver un radical plus à gauche que tous les autres : la réduction gauche consiste à réduire toujours le radical le plus à gauche.

(Pour mémoire, on dit qu'un radical $l$ est ***à gauche*** d'un radical $r$ dans $t$ si la position du motif ($\lambda$ de $l$ est à gauche du motif ($\lambda$ de $r$ dans $t$. Cela correspond au fait que l'occurrence de $l$ est plus petite, pour l'ordre alphabétique, que l'occurrence de $r$.)

On notera $t \longrightarrow_g u$ le fait que $t$ se réduit en $u$ par une étape de réduction gauche. En particulier, la réduction gauche définit une fonction partielle sur les $\lambda$-termes puisque terme est en relation avec au plus un terme par la réduction gauche.

Un résultat important du $\lambda$-calcul est que si $t$ a une forme normale $n$, celle-ci peut être atteinte par la réduction gauche : $t \longrightarrow_g n$, ce qu'on appelle le ***théorème de normalisation gauche***.

La notion de réduction standard est une classe plus large de réductions dans laquelle on ne contraint pas à réduire le radical le plus à gauche mais où on demande que les radicaux soient réduits ***de la gauche vers la droite***. Plus précisément, pour qu'une réduction soit standard, on demande qu'on n'y réduise jamais de résidu d'un radical qui se trouve à gauche d'un radical déjà réduit, ce que la définition ci-dessous traduit formellement :

**Définition 1 (*Réduction standard*)**

*Une réduction $\sigma = (t_i, a_i)_{i\in\lambda}$, $\lambda \in \omega + 1$ (finie ou infinie) est dite* **standard** *si pour tous $i$, $j$ tels que $i < j \in \lambda$, il n'y a pas de radical $b \in \mathsf{RAD}(t_i)$ à gauche de $a_i$ tel que $a_j \in b/(t_k, a_k)_{i\leq k<j}$.*

*S'il existe une dérivation standard $\sigma$, $t \xrightarrow{\sigma} u$, on note $t \xrightarrow{st} u$.*

Le ***théorème de standardisation*** énonce la complétude de la réduction standard : si $t$ se réduit sur $u$, alors il se réduit de manière standard. On va montrer le théorème de standardisation dans un cas particulier : en faisant l'hypothèse que les termes considérés sont fortement normalisables.

**Théorème 2 (*Standardisation dans le cas SN*)**

*Soient $t, u \in \Lambda$, $t$ fortement normalisant. Si $t \longrightarrow_\beta^\star u$ alors $t \xrightarrow{st} u$.*

g

**Définition 3 ($\mathsf{Val}(t)$)**

*Soit $t \in \Lambda$. Si $t \longrightarrow_g^\star \lambda x.u$, on note $\mathsf{Val}(t)$ le premier terme commençant par une abstraction qui apparaît dans cette réduction.*

**▷ Question 1 (★)**

*Pourquoi la définition précédente est-elle bien définie ?*

**▷ Question 2 (★★)**

*Soient $t, u \in \Lambda$ et $\sigma$ une réduction standard de $t$ à $\lambda x.u$.*

*Montrer qu'il existe des réductions $\rho$ et $\tau$ telles que $\sigma = \rho \cdot \tau$ où $t \xrightarrow{\rho} \mathsf{Val}(t)$, $\mathsf{Val}(t) \xrightarrow{\tau} u$, $\rho$ est une réduction de tête et $\tau$ est standard.*

**▷ Question 3 (★★)**

*Montrer que si $t, u \in \Lambda$, $t$ fortement normalisant, sont tels que $t \longrightarrow_\beta^\star u$, alors $t \xrightarrow{st} u$.*

*On procèdera par induction : la mesure n'est pas immédiate à trouver et une indication est donnée ci-dessous.*

### Indications

1. Pas d'indication particulière.
2. Pas d'indication particulière.
3. Ni la structure de $t$, ni la longueur de la plus longue réduction issue de $t$ ne suffisent, à elles seules, pour mener à bien l'induction. Tentez de combiner les deux.

## 2. Représentation des fonctions récursives en $\lambda$-calcul

L'objectif de cette section est de démontrer le théorème suivant :

**Théorème 4**

*Les fonctions récursives (partielles) sont fortement représentables par des $\lambda$-termes.*

On va commencer par rappeler quelques définitions sur les fonctions récursives, puis on définira ce que signifie, pour un $\lambda$-terme, de représenter (fortement) une fonction. On procédera ensuite en plusieurs étapes :

— On verra comment certaines fonctions arithmétiques élémentaires sont représentables.
— On montrera ensuite que toutes les fonctions récursives primitives sont représentables.
— On étendra ensuite ce résultats aux fonctions récursives totales.
— On montrera finalement traiter la partialité : toutes les fonctions récusives partielles sont représentables.

Mais avant toute chose, on va revenir brièvement sur les constructions de points fixes vues en cours car celles-ci sont essentielles pour représenter les fonctions récursives, comme on peut s'y attendre.

### *2.1. Constructions de points fixes en $\lambda$-calcul*

On a vu les résultats suivants en cours :

**Théorème 5 (*Premier théorème de points-fixes*)**

*En $\lambda$-calcul, toute fonction admet un point-fixe :*

$$\forall t \in \Lambda, \exists u \in \Lambda, (t)u =_\beta u.$$

**Théorème 6**

*Il y a, en $\lambda$-calcul, des $\lambda$-termes qui engendre des points-fixes pour tous les $\lambda$-termes, c-es-à-dire :*

$$\exists Y \in \Lambda, \forall t \in \Lambda, (Y)t =_\beta (t)(Y)t.$$

Ces termes sont appelés combinateurs de points-fixes (ou FPC).

On peut même caractériser les combinateurs de points-fixes comme les solutions d'une certaines équation de points-fixes : la proposition suivante fournit un terme dont les points fixes sont exactement les combinateurs de point fixe.

**Proposition 7**

*Soit $g = \lambda xf.(f)(x)f$. Alors on a :*

*Un $\lambda$-terme $t$ est un FPC si, et seulement si,*
*c'est un point fixe de $g$, c'est-à-dire si $(g)t =_\beta t$.*

On donne ci-dessous deux exemples de combinateurs de points-fixes : les combinateurs de Curry et de Turing :

**Définition 8 (*Combinateurs de points-fixes de Curry et Turing*)**

— $Y_C = \lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x$
— $Y_\mathcal{T} = (\lambda x.\lambda y.(y)(x)xy)\lambda x.\lambda y.(y)(x)xy$

On rappelle qu'on a $(Y_C)t =_\beta (t)(Y_C)t$ pour tout terme $t$. On notera que le point-fixe de Turing "réduit" :

**▷ Question 4**

*Soit $t$ un $\lambda$-terme. Montrer qu'on a :*

$$(Y_\mathcal{T})t \longrightarrow_\beta^\star (t)(Y_\mathcal{T})t$$

*2.2. Rappel sur les fonctions récursives*

**Définition 9 (*Fonctions partielles ayant leurs arguments et leurs valeurs dans* $\mathbb{N}$)**

*Soit $p \geq 0$ ; on note $\mathcal{N}^p$ l'ensemble des fonctions partielles à $p$ arguments entiers et à valeurs entières et $\mathcal{N} = \cup_{k\in\mathbb{N}}\mathcal{N}^k$. Si $f \in \mathcal{N}^k$, on notera (i) $f(n_1, \ldots, n_k) \uparrow$ le fait que $f$ ne soit pas définie en $(n_1, \ldots, n_k)$ et (ii) $f(n_1, \ldots, n_k) \downarrow m$ le fait que $f$ soit définie en $(n_1, \ldots, n_k)$ et que $f(n_1, \ldots, n_k) = m$*

**Définition 10 (*Fonction récursives primitives*)**

*L'ensemble des fonctions récursives primitives, $\mathcal{REC}_p$, est le plus petit ensemble de fonctions de $\mathcal{N}$ telle que :*

- *la fonction de $\mathcal{N}^0$, toujours égale à 0, est primitive récursive ;*
- *la fonction successeur, $\sigma \in \mathcal{N}^1$, est dans $\mathcal{REC}_p$ ;*
- *la fonction caractéristique de la relation $\leq$, $\chi_\leq$ est dans $\mathcal{REC}_p$ ;*
- *pour tout $p \geq 1$ et $k \leq p$, la fonction de projection $\pi^p_k$, éléments de $\mathcal{N}^p$, définie de la manière suivante, est primitive récursive :*
$$\pi^p_k(m_1, \ldots, m_p) = m_k;$$
- *l'ensemble $\mathcal{REC}_p$ est clos par composition : si $n, p \geq 1$, $g \in \mathcal{N}^p$, $k_1, \ldots k_p \in \mathcal{N}^n$ sont récursives primitives, alors la fonction $f \in \mathcal{N}^n$ définie de la manière suivante est primitive récursive :*
$$f(m_1, \ldots, m_n) = g(h_1(m_1, \ldots, m_n), \ldots h_p(m_1, \ldots, m_n));$$
- *l'ensemble $\mathcal{REC}_p$ est clos par récursion primitive : si $n \geq 0$, $g \in \mathcal{N}^n$, $h \in \mathcal{N}^{n+2}$ alors la fonction $f \in \mathcal{N}^{n+1}$ définie de la manière suivante est primitive récursive :*
$$f(0, m_1, \ldots, m_n) = g(m_1, \ldots, m_n)$$
$$f(k+1, m_1, \ldots, m_n) = h(k, f(k, m_1, \ldots, m_n), m_1, \ldots, m_n)$$

**Définition 11 (*Fonction récursives*)**

*L'ensemble des fonctions récursives primitives, $\mathcal{REC}$, est la plus petite classe de fonctions de $\mathcal{N}$ qui est close pour les mêmes opérations que $\mathcal{REC}_p$ ainsi que pour le schéma de minimisation $\mu$ défini comme suit :*
*si $k \in \mathbb{N}$ et $f \in \mathcal{N}^{k+1}$ alors $\mu f$ est la fonction de $\mathcal{N}^k$ définie par*

$$\begin{cases} \mu f(n_1, \ldots, n_k) \downarrow m & \text{si } m \text{ est tel que pour tout } l < m,\ f(n_1, \ldots, n_k, l) \downarrow j \neq 0 \text{ et} \\ & f(n_1, \ldots, n_k, m) \downarrow 0 \\ \mu f(n_1, \ldots, n_k) \uparrow & \text{sinon.} \end{cases}$$

**Définition 12 (*Fonction récursives totales*)**

*L'ensemble des fonctions récursives totales est la partie de $\mathcal{REC}$ constituée des fonctions totales, ie qui sont définies en tout point.*

*2.3. Représentation de fonctions par des $\lambda$-termes*

On définit maintenant une notion de représentabilité d'une fonction de $\mathbb{N}^k$ dans $\mathbb{N}$ par un $\lambda$-terme. Plus précisément, on définit deux notions de représentabilité, qui coincident sur les fonctions totales et ne diffèrent que pour le traitement de la partialité.

Puisqu'on souhaite représenter en $\lambda$-calcul des fonctions des entiers dans les entiers, il nous faut tout d'abord disposer d'une représentation des entiers en $\lambda$-calcul. On va pour cela rappeler la définition des entiers de Church.

L'idée va être de représenter l'entier $n$ comme une fonction à deux arguments qui itère ($n$ fois) son premier argument sur le second.

**Définition 13 (*Entiers de Church*)**

*Les entiers de Church sont définis comme :*

$$\overline{n} = \lambda f.\lambda x.(f)^n x = \lambda f.\lambda x.(f)(f)\ldots(f)x \qquad n \geq 0$$

*où $(f)^n x$ correspond à $n$ itérations de $f$ sur $x$, c'est-à-dire :*

— $(f)^0 x = x$ ;

— $(f)^{n+1} x = (f)(f)^n x$.

On a ainsi :

— $\overline{n} = \lambda f.\lambda x.(f)(f)\dots(f)x$

— $\overline{0} = \lambda f x.x$

— $\overline{1} = \lambda f x.(f)x \sim_\eta \lambda f.f$

**Définition 14 (*Fonction représentable*)**

*Soit $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ une fonction partielle de $\mathbb{N}^k$ dans $\mathbb{N}$.*

*On dit qu'un terme $t \in \Lambda$* **représentee** *(resp.* **représente fortement***) $f$ si pour tous $n_1, \dots, n_k \in \mathbb{N}$, on a :*

$$\begin{array}{lll} f(n_1,\dots,n_k)\downarrow n & \Rightarrow & (t)\,\overline{n_1}\dots\overline{n_k} =_\beta \overline{n} \\ f(n_1,\dots,n_k)\uparrow & \Rightarrow & (t)\,\overline{n_1}\dots\overline{n_k} \quad \textit{n'est pas normalisable (resp. n'est pas résoluble)} \end{array}$$

**Remarque 1**

*Par les théorèmes de normalisation gauche et de normalisation de tête, on a une formulation équivalente de la (forte) représentabilité :*

— *On dit qu'un terme $t \in \Lambda$* **représente** *$f$ si pour tous $n_1, \dots, n_k \in \mathbb{N}$, on a :*

$$\begin{array}{lll} f(n_1,\dots,n_k) = n & \Rightarrow & (t)\,\overline{n_1}\dots\overline{n_k} \Downarrow_g \overline{n} \\ f(n_1,\dots,n_k)\uparrow & \Rightarrow & (t)\,\overline{n_1}\dots\overline{n_k} \Uparrow_g \end{array}$$

— *On dit qu'un terme $t \in \Lambda$* **représente fortement** *$f$ si pour tous $n_1, \dots, n_k \in \mathbb{N}$, on a :*

$$\begin{array}{lll} f(n_1,\dots,n_k) = n & \Rightarrow & (t)\,\overline{n_1}\dots\overline{n_k} \Downarrow_g \overline{n} \\ f(n_1,\dots,n_k)\uparrow & \Rightarrow & (t)\,\overline{n_1}\dots\overline{n_k} \Uparrow_h \end{array}$$

*Cette formulation fournit une stratégie déterministe pour calculer $f(n_1, \dots, n_k)$.*

*On notera que pour la représentabilité forte, on demande, dans le cas où $f$ n'est pas défini que le terme associé ne converge par pour la réduction de tête, c'est-à-dire non seulement qu'il n'est pas de forme normale, mais pas non plus de forme normale de tête.*

**Lemme 2**

*Soit $t \in \Lambda$ représentant une fonction $f$. Alors :*

— *Tout terme $\beta$-équivalent à $t$ représente aussi $f$ ;*

— *Si $f$ est définie en au moins un point, alors $t$ est résoluble ;*

— *Si $f$ est définie est au point un point, alors il existe un terme en forme normale de tête qui représente $f$.*

**▷ Question 5**

*Montrer le lemme.*

### *2.4. Un peu d'arithmétique en λ-calcul*

On va maintenant considérer comment représenter quelques fonctions arithmétiques élémentaires.

La manière dont il faut comprendre le codage des entiers que nous venons d'introduire est celui d'un codage unaire avec zéro et successeur ***mais avec la possibilité d'instancier la fonction zéro et la fonction successeur de la manière que l'on souhaite par des fonctions concrètes qui pourront faire du calcul***. Ce dernier fait va être la clé du codage des principales fonctions arithmétiques que l'on va voir maintenant :

**Proposition 15 (*Fonction successeur*)**

*Soit $\mathsf{Succ} = \lambda n.\lambda f.\lambda x.(f)(n)fx$ (resp. $\mathsf{Succ}' = \lambda n.\lambda f.\lambda x.(n)f(f)x$). Ces deux $\lambda$-termes représentent la fonction successeur :*

$$(\mathsf{Succ})\,\overline{n} \longrightarrow_\beta^\star \overline{n+1} \qquad \textit{(idem pour } \mathsf{Succ}'\textit{)}.$$

On remarque qu'avec l'interprétation précédente des entiers de Church, le deuxième terme pour le successeur consiste à construire un terme où l'on prend un entier $\overline{n}$ en entrée et on reconstruit un nouvel entier à partir de $\overline{n}$ en instanciant le "zéro" de $\overline{n}$ avec $(f)x$ c'est à dire ce qui correspond au 1. Cela se voit plus précisément en remarquant que :

$$\mathsf{Succ}' =_\beta \lambda n.\lambda f.\lambda x.(n)f(\overline{1})fx.$$

**Proposition 16 (*Addition*)**

*Soit* $\mathsf{Add} = \lambda m.\lambda n.\lambda f.\lambda x.(m)f(n)fx$. *Ce* $\lambda$*-terme représente l'addition :*

$$(\mathsf{Add})\,\overline{m}\,\overline{n} \longrightarrow_\beta^\star \overline{m+n}\,.$$

**Proposition 17 (*Multiplication*)**

*Soit* $\mathsf{Mult} = \lambda m.\lambda n.\lambda f.\lambda x.((m)(n)f)x$. *Ce* $\lambda$*-terme représente la multiplication :*

$$(\mathsf{Mult})\,\overline{m}\,\overline{n} \longrightarrow_\beta^\star \overline{m\times n}\,.$$

*On écrira aussi* $\mathsf{Mult} =_\eta \lambda m.\lambda n.\lambda f.(m)(n)f$.

On remarque dans le terme extensionnellement équivalent à Mult n'est autre que le terme correspondant à la composition.

**Proposition 18 (*Exponentiation*)**

*Soit* $\mathsf{Exp} = \lambda m.\lambda n.\lambda f.\lambda x.(m)nfx$. *Ce* $\lambda$*-terme représente l'exponentiation :*

$$(\mathsf{Exp})\,\overline{m}\,\overline{n} \longrightarrow_\beta^\star \overline{m^n}\,.$$

*On écrira :* $\mathsf{Exp} =_\eta \lambda m.\lambda n.(n)m$.

**▷ Question 6**

*Montrer les trois propositions précédentes.*

*Représentation de la fonction prédécesseur*

Malgré sa simplicité, il est bien plus compliqué (et peu efficace) de définir la fonction prédécesseur avec les entiers de Church... Pour cela, on va utiliser une idée simple consistant à utiliser des paires pour calculer $(predn, n)$, en posant, par convention, que le prédécesseur de 0 vaut 0. l'idée est de considérer la fonction qui à $\phi = (x, y) \mapsto (y, y+1)$ et à l'itérer $n$ fois à partir de $(0, 0)$ : en itérant une fois $\phi$ sur $(0, 0)$, on obtient $(0, 1)$, en l'itérant 2 fois, on obtient $(1, 2)$, et on montre aisément par induction qu'en itérant $n+1$ fois $\phi$ sur $(0, 0)$, on obtient $(n, n+1)$. Le prédécesseur est donc stocker dans la première composante de la paire.

**Définition 19 (*Encodage de la paire*)**

*On considère les termes suivants :*

- $paire = \lambda x.\lambda y.\lambda p.(p)xy$
- $\pi_1 = \lambda y.(y)\lambda x_1.\lambda x_2.x_1$
- $\pi_2 = \lambda y.(y)\lambda x_1.\lambda x_2.x_2$

*Par souci de lisibilité, on notera* $\langle t, u\rangle$ *le terme* $(paire)tu$.

**Proposition 20**

*Pour tous* $t_1, t_2 \in \Lambda$, *et pour* $i \in \{1, 2\}$, *on a :*

$$\pi_i\langle t_1, t_2\rangle \longrightarrow^\star t_i.$$

**▷ Question 7**

*Montrer la proposition 20.*

**Proposition 21 (*Représentation du prédecesseur*)**

*Soit* $\mathsf{Pred} = \lambda n.(\pi_1)((n)\lambda p.(\lambda p_2.\langle p_2, (\mathsf{Succ})p_2\rangle)(\pi_2)p)\langle\overline{0},\overline{0}\rangle$.

Pred *représente fortement la fonction prédécesseur (avec la convention que le prédécesseur de* 0 *est* 0*).*

**▷ Question 8**

*Montrer la proposition 21.*

**Remarque 3**

*Comme on le voit, il est beaucoup plus compliqué de définir la fonction prédécesseur avec les entiers de Church que les fonctions précédentes. En particulier, pour calculer le prédécesseur d'un entier de Church il faut un nombre d'étapes de* $\beta$*-réduction proportionnel à la valeur de l'entier : cet encodage calcul le prédécesseur en temps linéaire !*

*Une manière alternative est d'utiliser une autre représentation des entiers, comme les entiers de Scott :*

— $\underline{0} = \lambda x.x$ ;

— $\underline{n+1} = \lambda x.\lambda y.(y)\underline{n}$;

*On obtient un prédécesseur en temps constant mais on perd sur d'autres plans.*

*2.5. Représentation des fonction récursives primitives*

Dans cette section, on montre, que toute fonction récursive primitive est fortement représentable par un $\lambda$-terme. La preuve se fait par induction sur une preuve du fait que la fonction considérée est récursive primitive.

Soit $f \in \mathcal{REC}_p$. On construit $\phi_f \in \Lambda^0$ par induction sur la définition inductive de $\mathcal{REC}_p$ de la manière suivante en utilisant la série de propositions suivantes :

**Proposition 22**

— *la fonction nulle est fortement représentée par l'entier de Church* $\overline{0}$ *:* $\phi_0 = \lambda x.\lambda y.y$ ;

— $\sigma$ *est fortement représentée par la fonction successeur sur les entiers de Church :* $\phi_\sigma = \mathsf{Succ}$ ;

— $\chi_\leq$ *est fortement représentée par le terme de Maurey :*

$$\phi_{\chi_\leq} = \lambda m.\lambda n.(((m)A)\lambda d.\overline{0})((n)A)\lambda d.\overline{0} \quad (\text{avec } A = \lambda a.\lambda b.(b)a)$$

— $\pi_k^p$ *est fortement représentée par* $\phi_{\pi_k^p} = \lambda x_1 \ldots x_p.x_k$ ;

— *Si* $f$ *est construite par schéma de composition à partir des fonctions* **totales** $g, h_1, \ldots, h_p$ *fortement représentées par* $\phi_g, \phi_{h_1}, \ldots, \phi_{h_p}$*, alors* $f$ *est fortement représentée par :*

$$\phi_f = \lambda x_1 \ldots \lambda x_n.((\phi_g)(\phi_{h_1})x_1 \ldots x_n) \ldots (\phi_{h_p})x_1 \ldots x_n$$

**▷ Question 9**

*Montrer la proposition précédente.*

Pour représenter des fonctions définies par récursion primitive, on procède comme suit.

On commence par considérer le terme $Z? = \lambda xyz.((z)\lambda x'.y)x$.

$Z?$ est une variante de la paire définie plus haut et qui réalise le test à 0 :

**Proposition 23**

$$\begin{array}{ccl} (Z?)xy\,\overline{0} & \longrightarrow^\star & x \\ (Z?)xy\,\overline{n+1} & \longrightarrow^\star & y \\ (Z?)xyv & & n\text{'est pas résoluble si } v \text{ n'est pas résoluble.} \end{array}$$

**▷ Question 10**

*Prouver la proposition précédente.*

Enfin, si $f$ est une fonction définie par schéma de récursion primitive à partir de $g$ et $h$, toutes deux fortement représentées par $\phi_g, \phi_h$ respectivement, on a :

$$f(0, m_1, \ldots, m_n) = g(m_1, \ldots, m_n)$$
$$f(k+1, m_1, \ldots, m_n) = h(k, f(k, m_1, \ldots, m_n), m_1, \ldots, m_n)$$

ce qui s'exprime encore :

$$f(k, m_1, \ldots, m_n) = \begin{array}{ll} \text{si } k \leq 0 \text{ alors} & g(m_1, \ldots, m_n) \\ \text{sinon} & h(k-1, f(k-1, m_1, \ldots, m_n), m_1, \ldots, m_n) \end{array}$$

**Proposition 24**

*Si $f$ est une fonction définie par schéma de récursion primitive à partir de $g$ et $h$, toutes deux fortement représentées par $\phi_g, \phi_h$ respectivement, alors il existe un terme $\phi_f$ qui représente fortement $f$.*

**▷ Question 11**

*En se rappelant qu'on sait résoudre des équations de points fixes en $\lambda$-calcul et en utilisant la question précédente, démontrer la proposition 24.*

*2.6. Représentation des fonctions récursives totales*

On va maintenant étendre le résultat de représentation en traitant le cas de la ***minimisation pour les fonctions totales***. On aura alors représenté toutes les fonctions récursives totales. La question suivante traitera de la partialité ; en préparation de cette extension, la présente section énonce des résultats plus forts que nécessaires pour la représentation des fonctions récursives totales, de manière à ce que l'extension suivante soit moins importante.

Considérons $g \in \mathcal{N}^{n+1}$ (une fonction totale) telle que pour tous $m_1, \ldots, m_n \in \mathbb{N}$, il existe $k$ tel que $g(m_1, \ldots, m_n, k) = 0$, que l'on suppose (fortement) représentée par $\phi_g$ .

On veut représenter fortement la fonction $f$ définie par minimisation de $g$, c'est-à-dire : $f = \mu g$, ie :

$$f(m_1, \ldots, m_n) = \left\{ \; k \text{ si pour tout } i < k, g(m_1, \ldots, m_n, i) = j \neq 0 \text{ et } g(m_1, \ldots, m_n, k) = 0 \right.$$

Il suffit de trouver un terme $u$ tel que

$$(u)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k} \left\{ \begin{array}{l} \longrightarrow_h{}^\star \; \overline{k} \text{ si } (\phi_g)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k} \longrightarrow_h{}^\star \; \overline{0} \\ \longrightarrow_h{}^\star \; (u)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k+1} \text{ sinon.} \end{array} \right.$$

En effet, dans ce cas, $u$ va tester tous les $k$ possibles et retourner le premier qui annule $x \mapsto g(m_1, \ldots, m_n, x)$.

Comme on veut ultimement traiter le cas des fonctions partielles, c'est-à-dire soit que $g$ n'est pas définie sur tout $\mathbb{N}^n$, soit qu'elle ne satisfait pas $\forall m_1, \ldots, m_n \in \mathbb{N}, \exists k \in \mathbb{N}, g(m_1, \ldots, m_n, k) = 0$, on va renforcer un peu les conditions ci-dessus :

On veut représenter <u>fortement</u> la fonction $f$ définie par minimisation de $g$, c'est-à-dire : $f = \mu g$, ie :

$$f(m_1, \ldots, m_n) = \left\{ \begin{array}{l} k \text{ si pour tout } i < k, g(m_1, \ldots, m_n, i) = j \neq 0 \text{ et } g(m_1, \ldots, m_n, k) = 0 \\ \underline{\text{est indéfinie sinon}} \end{array} \right.$$

Il suffit de trouver un terme $u$ tel que

$$(u)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k} \left\{ \begin{array}{ll} \longrightarrow_h{}^\star \; \overline{k} & \text{si } (\phi_g)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k} \longrightarrow_h{}^\star \; \overline{0} \\ \longrightarrow_h{}^\star \; (u)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k+1} & \text{sinon.} \\ & \underline{\text{et n'est pas résoluble si } (\phi_g)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k}} \; \underline{\text{n'est pas résoluble}} \end{array} \right.$$

Pour cela, posons alors $\Phi_g = \lambda x_1 \ldots x_n y.(Z?)TF(\phi_g)x_1 \ldots x_n y$ où $T = \lambda x.\lambda y.x$ et $F = \lambda x.\lambda y.y$.

**Lemme 4**

*On a*

$$(\Phi_g)\,\overline{m_1} \ldots \overline{m_n}\,\overline{k} \left\{ \begin{array}{l} =_\beta T \text{ si } g(m_1, \ldots, m_n, k) = 0 \\ =_\beta F \text{ si } g(m_1, \ldots, m_n, k) = j \neq 0 \\ \underline{\text{non résoluble si } g(m_1, \ldots, m_n, k) \text{ est indéfini}} \end{array} \right.$$

**▷ Question 12**

*Démontrer le lemme précédent.*

**Lemme 5**

*Soient $t, u, v \in \Lambda$. Si $t =_\beta T$ (resp. $F$), alors $(\langle u, v\rangle)t \longrightarrow_h^\star u$ (resp. $v$) and $(t)uv \longrightarrow_h^\star u$ (resp. $v$).*

**▷ Question 13**

*Montrer le lemme précédent.*

**Lemme 6**

*Il existe un terme clos $D$ tel que pour tous $u, v \in \Lambda$,*

$$(D)uv \longrightarrow_h^\star (u)vv(D)u(\mathsf{Succ})v.$$

**▷ Question 14**

*Construire un tel terme $D$ et montrer qu'il satisfait la propriété énoncée dans le lemme.*

**Définition 25 ($\widetilde{n}$)**

*On pose : $\widetilde{0} = \overline{0}$ et $\widetilde{n+1} = (\mathsf{Succ})\widetilde{n}$.*

**Lemme 7**

*Soit $u \in \Lambda$ et $n \in \mathbb{N}$.*

- *Si $(u)\,\overline{n}$ n'est pas résoluble, alors $(D)u\overline{n}$ est aussi non résoluble ;*
- *Si $(u)\,\overline{n} =_\beta T$, alors $(D)u\overline{n} =_\beta \overline{n}$ ;*
- *Si $(u)\,\overline{n} =_\beta F$, alors $(D)u\widetilde{n} =_\beta (D)u\widetilde{n+1}$ ;*

**▷ Question 15**

*Montrer le lemme précédent.*

**Proposition 26**

*Le terme $\phi_f = \lambda x_1 \dots x_n.((D)(\Phi_g)x_1 \dots x_n)\,\overline{0}$ représente (fortement) la fonction $f = \mu g$.*

**▷ Question 16**

*Montrer la proposition.*

### *2.7. Représentation des fonctions récursives partielles*

Si le traitement du schéma de minimisation des fonctions récursives totales est compatible avec les fonctions partielles, le terme qu'on a donné pour représenter le schéma de composition ne l'est pas.

En effet, pour les fonctions partielles, on a

— la fonction $f$ obtenue par schéma de composition à partir de $g, h_1, \dots, h_n$ est définie en $\vec{m}$ et égale à $g(h_1(\vec{m}), \dots, h_n(\vec{m}))$ si $h_i(\vec{m})$ est définie pour tout $i$ et si $g$ est définie en $(h_1(\vec{m}), \dots, h_n(\vec{m}))$, dans tous les autres cas $f$ n'est pas définie en $\vec{m}$.

La manière dont on a défini la composition au début de ce texte ne fonctionne pas car la notion de convergence utilisée pour les fonctions récursives partielle est intrinsèquement "en appel par valeur" : pour définir la composition, il faut d'abord que tous les arguments convergent et que la fonction $g$ soit définie en ce point, tandis que la notion de convergence du $\lambda$-calcul est naturelle "en appel par nom", c'est-à-dire qu'une fonction qui ne dépend pas (intentionnellement) de son argument pourra retourner une valeur même si c'est argument ne converge pas.

La représentation de la composition doit donc être modifier. L'idée va être en quelque sorte de faire en sorte de simuler l'appel par valeur en appel par nom, en forçant l'évaluation des arguments.

On va d'abord se concentrer sur le cas unaire : $f = g \circ \langle h_1, \dots, h_n\rangle$ :

— $f(\vec{m}) \uparrow$ si $\exists i, h_i(\vec{m}) \uparrow$ ou $g(h_1(\vec{m}), \dots, h_n(\vec{m})) \uparrow$

— $f(\vec{m}) = g(h_1(\vec{m}), \ldots, h_n(\vec{m}))$ sinon.

L'idée est qu'on cherche un combinateur $A$ tel que :

— $(A)uv$ est non résoluble si $u$ OU $v$ sont non résoluble ;

— $(A)uv = (u)v$ si $u$ ET $v$ sont TOUS DEUX résolubles et si $v$ est $\beta$-équivalent à un entier de Church.

Pour cela, il va falloir placer $v$ en position de tête de telle sorte que sa potentielle non-résolubilité implique celle de $(A)uv$ et dans le cas où $v =_\beta \overline{n}$ pour un certain $n$, alors on réduira vers $(u)\,\overline{n}$.

On cherche donc $A$ de la forme $\lambda uv.(v)B_1 \ldots B_k$, tel que si $v =_\beta \overline{n}$, $(v)B_1 \ldots B_k =_\beta (u)\,\overline{n}$. On remarque en fait que si $v =_\beta \overline{n}$, $(v)B_1 \ldots B_k =_\beta (B_1)^n B_2 \ldots B_k$ (dès que $k \geq 2$).

On cherche alors des $B_i$ tels que $(B_1)^n B_2 \ldots B_k =_\beta (u)\,\overline{n} =_\beta (u)(\mathsf{Succ})^n\,\overline{0}$ (puisqu'on sait que $\overline{n} =_\beta \widetilde{n}$).

On a le lemme suivant :

**Lemme 8**

*Si $k \geq 1$, alors $(\lambda xy.(x)(s)y)^k u \longrightarrow_h{}^\star\ \lambda y.(u)(s)^k y$.*

**▷ Question 17**

*Montrer le lemme précédent*

Si on considère :

— pour $B_1$ le terme du lemme précédent, avec Succ à la place de $s$,

— pour $B_2$, le terme $u$,

— pour $B_3$, le terme $\overline{0}$,

On a alors :

$$A = \lambda uv.((v)\lambda xy.(x)(\mathsf{Succ})y)u\,\overline{0}$$

qui satisfait les relations voulues.

On pose alors la définition suivante pour définir une composition qui force l'évaluation de ses arguments :

**Définition 27 ($u[v_1 \ldots v_k]$)**

*On considère $u[v_1 \ldots v_k] = ((A)((A) \ldots ((A)(A)uv_1)v_2) \ldots v_{k-1})v_k$, c'est-à-dire :*

— $u[v] = (A)uv$ ;

— $u[v_1 \ldots v_{k+1}] = ((A)u[v_1 \ldots v_k])v_{k+1} = u[v_1 \ldots v_k][v_{k+1}]$.

D'où il vient :

**Lemme 9**

— *Si $u$ ou l'un des $v_i$s est non résoluble, alors $u[v_1 \ldots v_k]$ est non résoluble ;*

— *Si pour tout $i$, $v_i =_\beta \overline{n_i}$, alors $u[v_1 \ldots v_k] \longrightarrow_h{}^\star (u)\widetilde{n_1} \ldots \widetilde{n_k} =_\beta (u)\,\overline{n_1} \ldots \overline{n_k}$.*

**▷ Question 18**

*Montrer le lemme précédent*

**▷ Question 19**

*Montrer que $\lambda x_1 \ldots x_k.\phi_g[(\phi_{h_1})x_1 \ldots x_k, \ldots, (\phi_{h_n})x_1 \ldots x_k]$ représente fortement $f$ construite par composition à partir de $g$ et des $h_i$.*