# Notes de cours « Preuves et Types », DEA MDFI

Version préliminaire

27 janvier 2006

# Table des matières

# Chapitre 1

# Théorèmes de Gödel

CE CHAPITRE reproduit pour l'essentiel le premier chapitre du livre de Girard [2] dont il suit grosso modo le plan. Le lecteur pourra aussi se reporter au Shoenfield [4] pour élucider ou compléter l'exposé.

## 1.1 Le langage de l'arithmétique.

Le langage de l'arithmétique est défini par

- les symboles de fonctions $\underline{0}$ (symbole de constante, *i.e.*, fonction d'arité 0), $\underline{S}$ (symbole de fonction unaire), $\underline{+}$ et $\underline{.}$ (symboles de fonction binaire) ;
- les symboles de prédicats $=$ et $<$.

En toute rigueur les termes et les formules de l'arithmétique sont construits en utilisant la notation *polonaise préfixée* (c'est à dire que la formule qui exprime que $x$ est égale à $y$ s'écrit $=xy$). En pratique on utilisera des parenthèses et les symboles binaires en notation infixe.

*Vérité, équivalence de formules.* On dira que deux formules $A$ et $B$ sont *logiquement équivalentes* si le séquent $\vdash A \leftrightarrow B$ est prouvable dans **LK** (le calcul des séquents). On dira qu'une formule $A$ de l'arithmétique est vraie, ou simplement que l'on a $A$ si $A$ est vraie dans les entiers. En particulier on dira que deux énoncés $A$ et $B$ traitant des entiers sont équivalents s'ils sont équivalents dans les entiers.

*Formules* $\Delta$, $\Sigma$ *et* $\Pi$. Les formules $\Delta$ sont définies par récurrence :

- les formules atomiques sont $\Delta$ ;
- si $A$ et $B$ sont $\Delta$ alors $\neg A$, $A \wedge B$, $A \vee B$ sont $\Delta$ ;
- si $A$ est $\Delta$ et si $t$ est un terme ne contenant pas $x$ alors $\forall x(x < t \rightarrow A)$ (que l'on notera $\forall x < t.\, A$) et $\exists x(x < t \wedge A)$ (que l'on notera $\exists x < t.\, A$) sont $\Delta$.

REMARQUE : Les quantificateurs $\forall x < t.$ et $\exists x < t.$ sont appelés *quantificateurs bornés*. Une formule $\Delta$ est donc une formule dans laquelle tous les quantificateurs sont bornés.

Les formules $\Sigma$ sont données par :

- les formules atomiques et leurs négations sont $\Sigma$ ;
- si $A$ et $B$ sont $\Sigma$ alors $A \wedge B$ et $A \vee B$ sont $\Sigma$ ;
- si $A$ est $\Sigma$ et si $t$ est un terme ne contenant pas $x$ alors $\forall x < t.\, A$ est $\Sigma$ ;
- si $A$ est $\Sigma$ alors $\exists x A$ est $\Sigma$.

Une formule $\Pi$ est la négation d'une formule $\Sigma$.

REMARQUE : L'ensemble des formules $\Delta$ est inclus dans l'ensemble des formules $\Sigma$. L'inclusion est bien entendu stricte, mais il y a tout de même une « réciproque » au sens du lemme suivant.

**Lemme 1.1.1** Si $A$ est $\Sigma$ alors il existe une formule $A_0$ qui est $\Delta$ et telle que $A$ est équivalente à $\exists x A_0$.

Ce lemme très utile, sera souvent utilisé sans référence dans la suite.

*Preuve :* On choisit une variable $x$ qui n'apparait pas dans $A$ (ni libre, ni liée) et on remplace chaque quantificateur non borné $\exists y$ de $A$ par $\exists y < x$. On obtient ainsi une formule $A_0$ qui est $\Delta$ et on voit facilement que $A$ est équivalente à $\exists x\, A_0$. □

*Notations diverses.* Soit $A$ une formule. On écrira $A[x_1, \ldots, x_k]$ (ou $A[\vec{x}]$) pour dire que $x_1$, …, $x_k$ sont des variables libres de $A$ et dans ce cas on écrira $A[t_1, \ldots, t_k]$ (ou $A[\vec{t}]$) pour désigner la formule $A$ dans laquelle chaque variable $x_i$ est remplacée par le terme $t_i$, *i.e.*, $A[\vec{t}] = A[t_1/x_1, \ldots, t_k/x_k]$. A ce propos on rappelle que la notation $A[t_1/x_1, \ldots, t_n/x_n]$ désigne la substitution *sans capture de variable*. On se reportera au premier chapitre du Krivine [3] pour une définition rigoureuse d'ycelle.

Soit $a$ un entier, on note $\underline{a}$ le terme définie par récurrence par :

- si $a = 0$ alors $\underline{a}$ est $\underline{0}$ ;
- si $a = b + 1$ alors $\underline{a}$ est $\underline{S}\,\underline{b}$.

Soit $t$ un terme clos. On note $|t|$ la *valeur* de $t$ définie par récurrence par :

- si $t$ est $\underline{0}$ alors $|t| = 0$ ;
- si $t$ est $\underline{S}\,u$ alors $|t| = |u| + 1$ ;
- si $t$ est $u \underline{+} v$ alors $|t| = |u| + |v|$ ;
- si $t$ est $u \,\underline{.}\, v$ alors $|t| = |u||v|$.

On voit donc que $|\underline{a}| = a$ pour tout entier $a$.

## 1.2 Fonctions (primitive) récursives

*Notations.* Soient $x_1$, …, $x_k$ des objets. Lorsque le contexte est clair on écrira $\vec{x}$ en lieu et place de $x_1, \ldots, x_k$ ; par exemple le $l$-uplet $(a_1, \ldots, a_l)$ s'écrira simplement $(\vec{a})$. On utilisera les lettres $a$, $b$, $c$, $k$, $l$, $n$, $m$ pour désigner des entiers, $t$, $u$, $v$ pour les termes, $x$, $y$, $z$ pour les variables, $F$, $G$, $H$ pour les fonctions, $P$, $Q$, $R$ pour les prédicats.

Si $e$ est une expression entière dépendant (ou pas) des variables $x_1$, …, $x_k$ (par exemple $e = x_1 + x_2$) on notera $\lambda\vec{x}.\, e$ la fonction qui à tout $k$-uplet d'entiers $(\vec{a})$ associe $e(\vec{a})$. En particulier si $n$ est un entier, $\lambda x.\, n$ représente la fonction constante de valeur $n$.

*Prédicats.* On appelle *prédicat* d'arité $k$ toute relation $k$-aire entre entiers où équivalemment, tout sous-ensemble de $\mathbb{N}^k$. Si $P$ est un prédicat $k$-aire on notera indifféremment $(\vec{a}) \in P$ ou $P(\vec{a})$. La *fonction caractéristique* de $P$ est $\chi_P : \mathbb{N}^k \to \mathbb{N}$ définie par $\chi_P(\vec{a}) = 0$ si $P(\vec{a})$ (*i.e.*, si $\vec{a} \in P$), $\chi_P(\vec{a}) = 1$ sinon. Remarquons que l'on prend ici une convention qui semble peu naturelle à savoir que 0 représente la valeur vraie et 1 la valeur faux ; ce choix s'explique pour des raisons techniques que l'on verra apparaître un peu plus bas.

*Fonctions partielles.* Une fonction *partielle* $F$ (d'arité $k$) de $\mathbb{N}^k \to \mathbb{N}$ est donnée par une partie $\mathrm{dom}\, F$ de $\mathbb{N}^k$ appelée le *domaine* de $F$ et une fonction de $\mathrm{dom}\, F \to \mathbb{N}$. Si $\mathrm{dom}\, F = \mathbb{N}^k$ on dit que $F$ est *totale*. On notera $F(\vec{a})\!\downarrow$ (resp. $F(\vec{a})\!\uparrow$) pour $(\vec{a}) \in \mathrm{dom}\, F$ (resp. $(\vec{a}) \notin \mathrm{dom}\, F$).

Si $F$ est partielle d'arité $k$ et $G$ est partielle d'arité $l$, $F(\vec{a}) = G(\vec{b})$ signifie que soit $F(\vec{a})\!\uparrow$ et $G(\vec{b})\!\uparrow$, soit $F(\vec{a})\!\downarrow$ et $G(\vec{b})\!\downarrow$ et $F(\vec{a}) = G(\vec{b})$.

### Fonctions récursives

Une fonction partielle d'arité $k$ de $\mathbb{N}^k \to \mathbb{N}$ est *récursive* si on peut la définir en appliquant un nombre fini de fois les schémas suivant :

**R1** les projections $\Pi_i^k : (\vec{a}) \to a_i$ sont récursives totales (d'arité $k$) ; l'addition, la multiplication et la fonction caractéristique $\chi_<$ sont récursives totales (d'arité 2) ;

**R2 (schéma de composition)** Si $H$ est récursive (d'arité $n$) et $G_1$, …, $G_n$ sont récursives (d'arité $k$) alors la *composée* $F$ :

$$F(\vec{a}) \to H(G_1(\vec{a}), \dots, G_n(\vec{a}))$$

est récursive. Son domaine est défini par ;

$$F(\vec{a})\downarrow \quad \text{ssi} \quad G_i(\vec{a})\downarrow \text{ pour } i = 1, \dots, n \text{ et } H(G_1(\vec{a}), \dots, G_n(\vec{a}))\downarrow$$

**R3 (schéma de minimisation)** Si $G$ est récursive d'arité $k+1$ alors la fonction $F$ :

$$F(\vec{a}) = \mu x.\,(G(x, \vec{a}) = 0)$$

dont la valeur (si elle est définie) est le plus petit $x$ tel que $G(x, \vec{a}) = 0$ et le domaine est donné par :

$$F(\vec{a})\downarrow \quad \text{ssi} \quad \begin{cases} \forall b < a,\ G(b, \vec{a})\downarrow \text{ et } G(b, \vec{a}) > 0 \\ G(a, \vec{a})\downarrow \text{ et } G(a, \vec{a}) = 0 \end{cases}$$

REMARQUE : Le schéma de minimisation est le seul qui peut produire des fonctions non totales. Par exemple la fonction $F(a) = \mu x.\,(\chi_<(x, 0) = 0)$ ($F(a)$ est le premier entier strictement plus petit que 0) est nulle part définie ce qui ne l'empêche pas d'être récursive.

*Prédicats récursifs, récursivement énumérables.* Un prédicat est récursif si sa fonction caractéristique est récursive totale. Un prédicat $P$ est *récursivement énumérable* s'il est le domaine d'une fonction récursive, c'est à dire s'il existe $F$ récursive telle que $P(\vec{a})$ ssi $F(\vec{a})\downarrow$ pour tout $k$-uplet $(\vec{a})$. On utilise communément l'abréviation *r.e.* pour « récursivement énumérable ».

REMARQUE : Un prédicat récursif est récursivement énumérable : soit $F$ définie par $F(a) = \mu x.\,(a = 0)$. Alors $F$ est récursive (pourquoi ?) et on a $F(a)\downarrow$ ssi $a = 0$. Par conséquent $F(\chi_P(\vec{a}))\downarrow$ ssi $P(\vec{a})$. Comme $\chi_P$ est récursive $F \circ \chi_P$ l'est par schéma de composition et donc $P$ est récursivement énumérable.

## DÉFINISSABILITÉ DES FONCTIONS RÉCURSIVES

**Théorème 1.2.1 (Définissabilité)** (i) *Si $F$ est une fonction récursive d'arité $k$ alors il existe une formule $A[\vec{x}, y]$ qui est $\Sigma$ et qui* définit *$F$, c'est à dire que :*

$$F(\vec{a}) = b \text{ ssi } A[\underline{\vec{a}}, \underline{b}]$$

(ii) *Si $P$ est un prédicat r.e. d'arité $k$ alors il existe une formule $A[\vec{x}]$ qui est $\Sigma$ et qui* définit *$P$, c'est à dire que :*

$$P(\vec{a}) \text{ ssi } A[\underline{\vec{a}}]$$

*Preuve :* La deuxième partie du théorème se déduit de la première : si $P$ est r.e. alors il y a une fonction récursive $F$ telle que $P(\vec{a})$ ssi $F(\vec{a})\downarrow$, c'est à dire ssi $\exists y\, y = F(\vec{a})$. D'après *(i)* il y a une formule $A[\vec{x}, y]$ qui définit $F$ et qui est $\Sigma$ ; mais alors $B = \exists y A[\vec{x}, y]$ définit $P$ et comme $A$ est $\Sigma$, $B$ l'est aussi.

La première partie se démontre par récurrence sur la construction de $F$. Si $F$ est la projection $\Pi_i^k$ alors $F$ est définie par $y = x_i$ ; l'addition est définie par $y = x_1 \underline{+} x_2$ et la multiplication par $y = x_1 \underline{.} x_2$. Enfin $\chi_<$ se définit par $(x_1 < x_2 \wedge y = \underline{0}) \vee (x_1 \not< x_2 \wedge y = \underline{S\,0})$ ($x_1 \not< x_2$ est une abréviation pour $\neg(x_1 < x_2)$). Remarquons que pour l'instant toutes ces formules sont $\Delta$.

Si $F(\vec{a}) = H(\vec{G}(\vec{a}))$ où les $G_i$ et $H$ sont récursives alors par hypothèse de récurrence il y a des formules $B_i[\vec{x}, y_i]$ pour $i = 1, \dots, n$ et $C[\vec{y}, y]$ qui sont $\Sigma$ et qui définissent respectivement les $G_i$ et $H$. On pose

$$A = \exists y_1 \dots \exists y_n\, B_1[\vec{x}, y_1] \wedge \dots \wedge B_n[\vec{x}, y_n] \wedge C[\vec{y}, y]$$

Clairement $A$ est $\Sigma$ et définit $F$.

Enfin si $F(\vec{a}) = \mu x.\,(G(x, \vec{a}) = 0)$ où $G$ est récursive alors par hypothèse de récurrence il y a une formule $B[x, \vec{x}, y]$ qui est $\Sigma$ et qui définit $G$. On pose

$$A = \forall y' < y. \exists z\, B[y', \vec{x}, \underline{S}\, z] \wedge B[y, \vec{x}, \underline{0}]$$

$\square$

(Question : pourquoi n'a-t-on pas défini $A = \forall y' < y.\, \neg B[y', \vec{x}, 0] \wedge B[y, \vec{x}, \underline{0}]$ ?)

## Fonctions primitives récursives

Il est bien clair que parmi les fonctions récursives on est plus intéressé par celles qui sont totales : du point de vue informatique la totalité correspond à la notion de *correction* de programme, c'est à dire que lorsqu'on écrit $F(\vec{a})\uparrow$ cela correspond à un programme qui « bugge » sur l'entrée $\vec{a}$ (typiquement en entrant dans une boucle infinie).

Toutefois l'ensemble des fonctions récursives totales n'a pas de bonnes propriétés, par exemple bien qu'il soit visiblement dénombrable, il n'est pas récursivement énumérable.[1] C'est pourquoi on est amené à introduire des classes de fonctions récursives totales bénificiant de meilleures propriétés. On verra plus tard dans le cours les fonctions représentables par des lambda-termes typés mais tout de suite on va introduire les *fonctions primitives récursives*.

Une fonction de $\mathbb{N}^k \to \mathbb{N}$ est primitive récursive (p.r.) si on peut l'obtenir en appliquant un nombre fini de fois les schémas suivant :

**PR1** Les projections $\Pi_i^k$, l'addition, la multiplication, la fonction caractéristique de $<$ *et les fonctions constantes* $\lambda \vec{x}.\, n$ sont primitives récursives ;

**PR2 (schéma de composition)** idem que pour les fonctions récursives ;

**PR3 (schéma de récurrence)** si $G$ est primitive récursive d'arité $k$ et $H$ est primitive récursive d'arité $k+2$ alors $F$ définie par :

$$F(a, \vec{a}) = \begin{cases} G(\vec{a}) & \text{si } a = 0 \\ H(F(a-1, \vec{a}), a, \vec{a}) & \text{si } a > 0 \end{cases}$$

est primitive récursive d'arité $k+1$.

*Prédicat récursif primitif.* Un prédicat est primitif récursif si sa fonction caractéristique l'est.

**Théorème 1.2.2** *Si $F$ est une fonction primitive récursive alors $F$ est récursive (totale). Si $P$ est un prédicat primitif récursif alors $P$ est récursif.*

*Preuve :* La démonstration qui suit est initialement due à Gödel ; c'est une variante de celles que l'on trouvera dans le Shoenfield [4] et/ou le Cori-Lascar [1]. Remarquons tout d'abord que la partie traitant des prédicats est conséquence immédiate de celle sur les fonctions.

On va montrer que chacun des schémas **PR1**, **PR2** et **PR3** peut se « programmer » à l'aide des schémas **R1**, **R2** et **R3**. Pour **PR1** il faut juste voir que les fonctions constantes sont programmables. La fonction $C_0 = \lambda \vec{x}.\, 0$ est donnée par $C_0(\vec{a}) = \mu x.\, (\Pi)_1^{k+1}(x, \vec{a}) = 0$ ; la fonction $C_1 = \lambda \vec{x}.\, 1$ par $C_1(\vec{x}) = \mu x.\, (\chi_)\! < (0, x) = 0$. Remarquons que c'est cette définition qui justifie le choix technique et peu naturel de prendre 0 pour valeur vraie des fonctions caractéristiques. La fonction $C_2 = \lambda \vec{x}.\, 2$ est donnée par $C_2(\vec{x}) = C_1(\vec{x}) + C_1(\vec{x})$, et plus généralement on définit $C_{n+1}$ par $C_{n+1}(\vec{x}) = C_n(\vec{x}) + C_1(\vec{x})$.

[1] On le démontre par *diagonalisation* : on verra un peu plus loin qu'un ensemble $X$ est r.e. ssi il existe une fonction récursive totale $F$ qui *énumère* $X$, c'est à dire telle que $X$ est l'image de $F$. Soit $T_1$ l'ensemble des fonctions récursive totales unaires et $\varphi$ une fonction de $\mathbb{N}^2$ dans $\mathbb{N}$ qui énumère $T_1$, c'est à dire telle que pour toute $F$ dans $T_1$ il existe un $e_F$ tel que $F(a) = \varphi(e_F, a)$ pour tout $a$. On va voir que $\varphi$ ne peut être récursive. Définissons $G$ par $G(a) = \varphi(a, a) + 1$ ; alors $G$ est clairement totale et unaire mais elle ne peut être récursive, sinon elle est dans $T_1$ et il y a un $e_G$ tel que $G(a) = \varphi(e_G, a)$ pour tout $a$ donc en particulier pour $a = e_G$, $G(e_G) = \varphi(e_G, e_G)$ contredisant la définition de $G$. Donc $\varphi$ n'est pas récursive, sinon $G$ le serait également.

Le schéma **PR2** est le même que le schéma **R2** ; reste donc à voir le schéma de récurrence primitive **PR3** et c'est là qu'il va falloir travailler un peu. On commence par dériver de **R1**, **R2** et **R3** des schémas supplémentaires pour programmer des fonctions récursives.

Soit $P(x, \vec{x})$ un prédicat récursif. Alors la fonction définie par $F(\vec{a}) = \mu x.(P(x, \vec{a}))$ est récursive ; en effet comme $P$ est récursif $\chi_P$ est récursive (totale), et donc $F$ est obtenue par **R3** (ici aussi on se sert du choix $\chi_P(a, \vec{a}) = 0$ ssi $P(a, \vec{a})$). Si de plus pour tout $\vec{a}$ il y a un $a$ tel que $P(a, \vec{a})$ alors $F$ est totale.

La fonction $\lambda xy.\, x \dot{-} y$ est récursive (totale) : on l'obtient par $a \dot{-} b = \mu x.(x + b + 1 > a)$.

Soient $P$ et $Q$ des prédicats récursifs ; alors $\neg P$, $P \vee Q$, $P \wedge Q$ sont récursifs. En effet leurs fonctions caractéristiques s'obtiennent au moyen de celles de $P$ et $Q$ par les opérations arithmétiques $+$, . et $\dot{-}$ qui sont tous récursifs totals. Remarquons que si on suppose $P$ seulement r.e. alors $\neg P$ n'est pas r.e. en général (pourquoi ?).

Soit $P(x, \vec{x})$ un prédicat récursif et $F(\vec{x})$ une fonction récursive totale ; alors la fonction $G(\vec{x})$ définie par $G(\vec{a}) = \mu x < F(\vec{a}).(P)(x, \vec{a})$ (le premier $x < F(\vec{a})$ tel que $P(x, \vec{a})$ s'il en existe, $F(\vec{a})$ sinon), est récursive totale. En effet $G(\vec{a})$ est égal à $\mu x.((x < F(\vec{a}) \wedge P(x, \vec{a})) \vee x = F(\vec{a}))$. En particulier le prédicat $\exists x < F(\vec{a}).P(x, \vec{a})$ est récursif puisqu'il est équivalent à $(\mu x < F(\vec{a}).(P(x, \vec{a}))) < F(\vec{a})$. Par négation le prédicat $\forall x < F(\vec{a}).P(x, \vec{a})$ est également récursif.

**Lemme 1.2.3** Il existe une fonction $\beta$ récursive totale vérifiant : pour tout $n$ et tous $a_0, \ldots, a_n$, il existe $c$ tel que pour $i = 0, \ldots, n$ on a $\beta(i, c) = a_i$.

On ne va pas démontrer ce lemme ici et l'on renvoie le lecteur aux références sus-citées ; Lascar-Cori le dérive du lemme chinois et Shoenfield le prouve in extenso au moyen de simples récurrences.

On note $(c)_i = \beta(i, c)$. Muni du lemme on va pouvoir démontrer que le schéma de récurrence primitive **PR3** est récursif.

Soit $G(\vec{x})$ et $H(x, y, \vec{x})$ des fonctions récursives totales. On définit $\bar{F}(x, \vec{x})$ par :

$$\bar{F}(a, \vec{a}) = \mu x.\left(\Big((x)_0 = G(\vec{a}) \wedge \forall y < a.\big((x)_{y+1} = H((x)_y, y + 1, \vec{a})\big)\Big)\right).$$

La fonction $\bar{F}$ est clairement récursive et on peut facilement se convaincre par récurrence sur $n$ qu'elle est totale. Soit maintenant $F$ donnée par $F(a, \vec{a}) = (\bar{F}(a, \vec{a}))_a$ ; alors $F$ est également récursive totale et on peut vérifier par récurrence sur $a$ que l'on a $F(0, \vec{a}) = G(\vec{a})$ et $F(a + 1, \vec{a}) = H(F(a, \vec{a}), a, \vec{a})$ pour tout $a$, soit que $F$ est définie par récurrence primitive à partir de $G$ et $H$. □

## Programmation de fonctions primitives récursives.

On va maintenant se donner un certain nombre d'autres schémas pour définir des fonctions et prédicats p.r.

*Itération.* Si $G$ est p.r. d'arité $k + 1$ alors la fonction $F$ définie par :

$$F(a, b, \vec{a}) = G^a(b, \vec{a})$$

est p.r. d'arité $k + 2$. En fait c'est une reformulation du schéma de récursion primitive. Clairement $F$ est donnée par $F(0, b, \vec{a}) = b$ et $F(a + 1, b, \vec{a}) = G(F(a, b, \vec{a}), \vec{a})$.

*Soustraction.* On définit $a \dot{-} 1$ par : $0 \dot{-} 1 = 0$ et $(a + 1) \dot{-} 1 = a$. On définit $a \dot{-} b$ par $a \dot{-} 0 = a$ et $a \dot{-} (b + 1) = (a \dot{-} b) \dot{-} 1$. La fonction binaire $\dot{-}$ est clairement p.r.

*Variante de la récursion primitive.* Soient $G_0$ une fonction p.r. d'arité $k$, $H$ p.r. d'arité $k + 2$ et $G_1, \ldots, G_k$ des fonctions p.r. d'arité $k + 1$ . Alors la fonction $F$ définie par

$$F(0, \vec{a}) = G_0(\vec{a})$$
$$F(a + 1, \vec{a}) = H(F(a, \vec{G}(a, \vec{a})), a, \vec{a})$$

est p.r. Remarquons qu'il s'agit presque du schéma de récurrence p.r. On va le montrer pour $k = 1$, le cas général s'en déduit immédiatement en utilisant des séquences.

Supposons donc que $F$ est définie par

$$\begin{aligned} F(0, b) &= G_0(b) \\ F(a+1, b) &= H(F(a, G_1(a, b)), a, b) \end{aligned}$$

On se donne $g$ :

$$\begin{aligned} g(0, b, a) &= b \\ g(i+1, b, a) &= G_1(a \dot{-} (i+1), g(i, b, a)) \end{aligned}$$

et $h$ :

$$\begin{aligned} h(0, b, a) &= G_0(g(a, b, a)) \\ h(i+1, b, a) &= H(h(i, b, a), i, g(a \dot{-} (i+1), b, a)) \end{aligned}$$

Il est clair que $g$ et $h$ sont toutes les deux p.r. On va montrer que pour tout $b$ :

$$F(a, b) = h(a, b, a) \tag{1.1}$$

Comme $h$ est p.r. on en déduira que $F$ l'est.

On vérifie aisément par récurrence sur $i$ que pour tout $a \geq i$ on a

$$g(i+1, b, a+1) = g(i, G_1(a, b), a) \tag{1.2}$$

puis que

$$h(i, b, a+1) = h(i, G_1(a, b), a) \tag{1.3}$$

On montre (1.1) par récurrence sur $a$. Pour $a = 0$ on a $h(0, b, 0) = G_0(g(0, b, 0)) = G_0(b) = F(0, b)$. Supposons que $F(a, b) = h(a, b, a)$. Alors

$$\begin{aligned} h(a+1, b, a+1) &= H(h(a, b, a+1), a, g(0, b, a+1)) && \text{par définition de } h \\ &= H(h(a, b, a+1), a, b) && \text{par définition de } g \\ &= H(h(a, G_1(a, b), a), a, b) && \text{par l'équation (1.3)} \\ &= H(F(a, G_1(a, b)), a, b) && \text{par hypothèse de récurrence} \\ &= F(a+1, b) && \text{par définition de } F \end{aligned}$$

*Opérations booléennes.* Si $P$ et $Q$ sont p.r. d'arité $k$ alors $\neg P$, $P \vee Q$, $P \wedge Q$ sont p.r. : $\chi_{\neg P}(\vec{a}) = 1 \dot{-} \chi_P(\vec{a})$, $\chi_{P \vee Q}(\vec{a}) = \chi_P(\vec{a})\chi_Q(\vec{a})$, $\chi_{P \wedge Q}(\vec{a}) = 1 \dot{-} (1 \dot{-} \chi_P(\vec{a}))(1 \dot{-} \chi_Q(\vec{a}))$.

*Définitions par cas.* Si $P$ est un prédicat p.r. d'arité $k$, $G$ et $H$ sont p.r. d'arité $k$ alors $F$ définie par :

$$F(\vec{a}) = \begin{cases} G(\vec{a}) & \text{si } P(\vec{a}) \\ H(\vec{a}) & \text{sinon} \end{cases}$$

est p.r. : on l'obtient par $F(\vec{a}) = (1 \dot{-} \chi_P(\vec{a}))G(\vec{a}) + \chi_P(\vec{a})H(\vec{a})$. Plus généralement si $P_1, \ldots, P_n$ sont p.r. et $G_1, \ldots, G_{n+1}$ sont p.r. alors

$$F(\vec{a}) = \begin{cases} G_1(\vec{a}) & \text{si } P_1(\vec{a}) \\ \quad \vdots & \\ G_n(\vec{a}) & \text{si } \neg(P_1(\vec{a}) \vee \cdots \vee P_{n-1}(\vec{a})) \wedge P_n(\vec{a}) \\ G_{n+1}(\vec{a}) & \text{sinon} \end{cases}$$

est p.r.

*Quelques prédicats p.r.* Par définition $<$ est p.r. C'est également le cas de $\not<$, $=$, $\neq$, $\leq$, $\not\leq$, $>$, $\not>$, $\geq$, $\not\geq$ : par exemple $\chi_{\not<}(a,b) = 1 \dot{-} \chi_{<}(a,b)$, $\chi_{=} = \chi_{\not< \wedge \not>}$, ...

*Minimisation bornée.* Si $P$ est p.r. alors $F$ donnée par $F(a,\vec{a}) = \mu x < a.\,(P(x,\vec{a}))$ dont la valeur est le premier $b < a$ tel que $P(b,\vec{a})$ s'il en existe, $a$ sinon, est p.r. On la définit par $F(a,\vec{a}) = G(a \dot{-} 1, a, \vec{a})$ où $G$ est donnée par :

$$
\begin{array}{rll}
G(0,b,\vec{a}) = 0 & & \text{si } P(0,\vec{a}) \\
G(0,b,\vec{a}) = b & & \text{si } \neg P(0,\vec{a}) \\
G(a+1,b,\vec{a}) = G(a,a+1,\vec{a}) & & \text{si } P(a+1,\vec{a}) \\
G(a+1,b,\vec{a}) = G(a,b,\vec{a}) & & \text{sinon}
\end{array}
$$

*Quantificateurs bornés.* Si $P$ est un prédicat p.r. d'arité $k+1$ alors $\forall x < a.\, P(x,\vec{a})$ et $\exists x < a.\, P(x,\vec{a})$ sont p.r. : le second est défini par : $\exists x < a.\, P(x,\vec{a})$ ssi $\mu x < a.\,(P(x,\vec{a})) < a$ (*i.e.*, $\chi_{\exists x < a.\, P}(\vec{a}) = \chi_{<}(\mu x < a.\,(P(x,\vec{a})), a)$), le premier par $\forall x < a.\, P$ ssi $\neg\exists x < a.\, \neg P(x,\vec{a})$ (*i.e.*, $\chi_{\forall x < a.\, P} = 1 \dot{-} \chi_{\exists x < a.\, \neg P}$).

*Arithmétique.* Les fonctions binaires $\lambda xy.\, x/y$ (quotient de $x$ par $y$ ) et $\lambda xy.\, x\%y$ (reste de $x$ modulo $y$) sont p.r. : la première est donnée par $a/b = (\mu x < a+1.\,(a < xb)) \dot{-} 1$ et la seconde par $a\%b = a \dot{-} (a/b)b$. Le prédicat $a \mid b$ ($a$ divise $b$) se définit par $a\%b = 0$ et est donc p.r.

*Suites d'entiers.* On définit la fonction $\lambda xy.\, \langle x,y\rangle$ par

$$\langle a,b\rangle = ((a+b)(a+b+1))/2 + a$$

Il est clair qu'elle est p.r.

On définit $\pi_1$ et $\pi_2$ par :

$$
\begin{aligned}
\pi_1 c &= \mu x < c+1.\,(\exists y < c+1.\, \langle x,y\rangle = c) \\
\pi_2 c &= \mu y < c+1.\,(\exists x < c+1.\, \langle x,y\rangle = c)
\end{aligned}
$$

$\pi_1$ et $\pi_2$ sont donc p.r. et, comme $\lambda xy.\, \langle x,y\rangle$ est bijective on a $\langle \pi_1 a, \pi_2 a\rangle = a$ pour tout $a$.

Pour chaque $k$ on pose :

$$\langle a_1 \ldots, a_k\rangle_k = \langle k, \langle a_1, \ldots \langle a_{k-1}, a_k\rangle \ldots\rangle\rangle$$

Plus précisément les fonction $\lambda\vec{x}.\, \langle\vec{x}\rangle_k$ sont définies par :

$$
\begin{aligned}
\langle\rangle_0 &= 0 \\
\langle a_0, \vec{a}\rangle_{k+1} &= \langle k+1, \langle a_0, \pi_2\langle\vec{a}\rangle_k\rangle\rangle
\end{aligned}
$$

On voit donc (par récurrence sur $k$) que chacune d'elles est p.r.

- L'ensemble des (codes de) suites est caractérisé par le prédicat `Seq` : $\texttt{Seq}(a)$ ssi $\pi_1 a \neq 0 \vee a = 0$.
- La longueur d'une suite est $\texttt{Lg}\, a = \pi_1 a$.
- Le $i$-ème élément de la suite $a$ est $(a)_i = \pi_1 \pi_2^i a$ si $i < \texttt{Lg}\, a$, $\pi_2^{\texttt{Lg}\, a} a$ sinon.
  La concaténation $a * b$ des suites $a$ et $b$ est donnée par $a * b = \langle \texttt{Lg}\, a + \texttt{Lg}\, b, F(\texttt{Lg}\, a, \pi_2 a, \pi_2 b)\rangle$ où $F$ est définie par $F(0,a,b) = b$, $F(1,a,b) = \langle a,b\rangle$ et $F(l+1,a,b) = \langle \pi_1 a, F(l, \pi_2 a, b)\rangle$.
- La suite d'une liste est donnée par $\texttt{Suite}\, a = 0$ si $\texttt{Lg}\, a = 0$, $\texttt{Suite}\, a = \mu x < a.\,(\langle (a)_1\rangle_1 * x = a)$ sinon. Donc on a

  $$\texttt{Suite}\langle a_1, \ldots, a_n\rangle_n = \langle a_2, \ldots, a_n\rangle_{n-1}$$

  En particulier si $\texttt{Lg}\, a > 0$ alors $\texttt{Suite}\, a < a$. Ceci nous permettra de définir des fonctions par récurrence sur des listes d'entiers.

REMARQUE : A cause de la bijectivité de $\lambda xy.\, \langle x,y\rangle$, si $a$ et $b$ sont deux (codes de) suites telles que $\texttt{Lg}\, a = \texttt{Lg}\, b$ et pour tout $i \leq \texttt{Lg}\, a$, $(a)_i = (b)_i$ alors $a = b$.

Toutes ces fonctions (et prédicats) sont clairement p.r.

*Récurrence générale.* Soit $F$ une fonction p.r. d'arité $k+1$. On note $\bar{F}$ la fonction donnée par :

$$\bar{F}(a,\vec{a}) = \langle F(0,\vec{a}), \ldots, F(a-1,\vec{a})\rangle_a$$

Alors $\bar{F}$ est p.r. En effet on l'obtient par : $\bar{F}(0,\vec{a}) = \langle\rangle_0$ et $\bar{F}(a+1,\vec{a}) = \bar{F}(a,\vec{a}) * \langle F(a)\rangle_1$.

Soit maintenant $G$ une fonction p.r. d'arité $k+1$. Alors la fonction $F$ définie par (récurrence généralisée) :

$$F(a,\vec{a}) = G(\bar{F}(a,\vec{a}),\vec{a})$$

est p.r. Soit $H$ donnée par $H(0,\vec{a}) = \langle\rangle_0$ et $H(a+1,\vec{a}) = H(a,\vec{a}) * \langle G(H(a,\vec{a}),\vec{a})\rangle_1$ qui est donc p.r. On montre facilement par récurrence sur $a$ que $H(a,\vec{a}) = \bar{F}(a,\vec{a})$ d'où l'on déduit que $F$ est p.r. en considérant que $F(a,\vec{a}) = (\bar{F}(a+1,\vec{a}))_{a+1}$.

*Une autre récurrence.* Soit $G_1$, …, $G_n$ des fonctions p.r. d'arité $k+1$ telles que $G_i(a,\vec{a}) < a$ dès que $a > 0$, $H_0$ une fonction p.r. d'arité $k$ et $H_1$ une fonction p.r. d'arité $n+k+2$. Alors la fonction $F$ donnée par :

$$F(a,\vec{a}) = \begin{cases} H_0(\vec{a}) & \text{si } a = 0 \\ H_1(F(G_1(a,\vec{a}),\vec{a}), \ldots, F(G_n(a,\vec{a}),\vec{a}), a, \vec{a}) & \text{sinon} \end{cases}$$

est p.r. On définit $H$ par

$$H(b,a,\vec{a}) = \begin{cases} H_0(\vec{a}) & \text{si } a = 0 \\ H_1((b)_{G_1(a,\vec{a})}, \ldots, (b)_{G_n(a,\vec{a})}, a, \vec{a}) & \text{sinon} \end{cases}$$

La fonction $H$ est clairement p.r., d'arité $k+2$ et $F$ est obtenue par récurrence généralisée sur $H$ :

$$F(a,\vec{a}) = H(\bar{F}(a,\vec{a}), a, \vec{a})$$

## 1.3 Codage de Gödel

ON VA MAINTENANT utiliser le petit « langage de programmation » des fonctions p.r. pour représenter le formalisme du calcul des prédicats dans les entiers. A chaque objet $X$ du formalisme (variable, terme, formule, démonstration) on va associer un entier unique $\ulcorner X \urcorner$ appelé son *code de Gödel*. On construit ensuite un ensemble de fonctions p.r. qui agissent sur les codes de Gödel et implémentent toutes les transformations syntaxiques standard, typiquement la substitution d'un ensemble de termes à un ensemble de variable dans une formule ou un terme.

### CODAGE DU FORMALISME

Soit $L$ un langage, c'est à dire que $L$ est déterminé par sa signature $\Sigma_L = (F_L, P_L)$ où $F_L$ est l'ensemble (au plus dénombrable) des symboles de fonctions de $L$ et $P_L$ est l'ensemble (au plus dénombrable) des symboles de prédicats. On suppose que $L$ contient le langage de l'arithmétique c'est à dire que $\underline{0}$, $\underline{S}$, $\underline{+}$ et $\underline{.}$ sont dans $F_L$ et que $=$ et $<$ sont dans $P_L$.

*Représentation des symboles.* Comme $F_L$ et $P_L$ sont au plus dénombrables on peut les énumérer par des suites $(f_i)_{i<\sigma_L}$ et $(P_i)_{i<\tau_L}$ où $\sigma_L$ et $\tau_L$ sont les cardinaux respectifs (possiblement infinis) de $F_L$ et $P_L$. De même comme l'ensemble des (noms de) variables du calcul des prédicats est dénombrable on peut supposer que celui-ci est complètement énuméré par la suite $(x_i)_{i\in\mathbb{N}}$. Si $s$ est un symbole ou une variable, on définit alors

$$\ulcorner s \urcorner = \begin{cases} \langle 0,0,i,0\rangle_4 & \text{si } s = x_i \\ \langle 0,1,i,n\rangle_4 & \text{si } s = f_i \text{ et } f_i \text{ est d'arité } n \\ \langle 0,2,i,n\rangle_4 & \text{si } s = P_i \text{ et } P_i \text{ est d'arité } n \end{cases}$$

*Représentation des termes.* Soit $t$ un terme du langage $L$ ; on définit $\ulcorner t \urcorner$ par :

$$\ulcorner t \urcorner = \begin{cases} \langle 1, 0, \ulcorner x \urcorner, 0 \rangle_4 & \text{si } t \text{ est la variable } x \\ \langle 1, 1, \ulcorner f \urcorner, \langle \ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner \rangle_n \rangle_4 & \text{si } t = f(t_1, \ldots, t_n) \end{cases}$$

*Représentation des formules.* Soit $A$ une formule du langage $L$. On définit $\ulcorner A \urcorner$ par :

$$\ulcorner A \urcorner = \begin{cases} \langle 2, 0, \ulcorner P \urcorner, \langle \ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner \rangle_n \rangle_4 & \text{si } A \text{ est la formule atomique } P(t_1, \ldots, t_n) \\ \langle 2, 1, \ulcorner A' \urcorner, 0 \rangle_4 & \text{si } A = \neg A' \\ \langle 2, 2, \ulcorner A' \urcorner, \ulcorner A'' \urcorner \rangle_4 & \text{si } A = A' \wedge A'' \\ \langle 2, 3, \ulcorner A' \urcorner, \ulcorner A'' \urcorner \rangle_4 & \text{si } A = A' \vee A'' \\ \langle 2, 4, \ulcorner A' \urcorner, \ulcorner A'' \urcorner \rangle_4 & \text{si } A = A' \to A'' \\ \langle 2, 5, \ulcorner x \urcorner, \ulcorner A' \urcorner \rangle_4 & \text{si } A = \forall x A' \\ \langle 2, 6, \ulcorner x \urcorner, \ulcorner A' \urcorner \rangle_4 & \text{si } A = \exists x A' \end{cases}$$

**Lemme 1.3.1** Soit $X$ un objet du langage (terme ou formule) et $Y$ un sous-objet de $X$ : par exemple $X$ est la formule $\underline{S}\, x < \underline{0}$ et $Y$ est la variable $x$. Alors $\ulcorner Y \urcorner < \ulcorner X \urcorner$.

La preuve est immédiate par récurrence sur $X$ et en considérant pour tout $(a, b) \neq (0, 0), (0, 1)$ on a $a, b < \langle a, b \rangle$. Ce lemme servira dans ce qui suit à définir des fonctions ou prédicats sur les (codes d') objets du formalisme par récurrence sur leurs (codes) de sous-objets.

*Prédicats de la signature.* On définit les prédicats p.r. $\mathtt{Var}(a)$ : « $a$ est le code d'une variable », $\mathtt{Fun}(a, n)$ : « $a$ est le code d'un symbole de fonction d'arité $n$ », $\mathtt{Pred}(a, n)$ : « $a$ est le code d'un symbole de prédicat d'arité $n$ » par :

$$\begin{aligned} \mathtt{Var}(a) \text{ ssi } & \mathtt{Lg}\, a = 4 \wedge (a)_1 = 0 \wedge (a)_2 = 0 \\ \mathtt{Fun}(a, n) \text{ ssi } & \mathtt{Lg}\, a = 4 \wedge (a)_1 = 0 \wedge (a)_2 = 1 \wedge (a)_4 = n \\ \mathtt{Pred}(a, n) \text{ ssi } & \mathtt{Lg}\, a = 4 \wedge (a)_1 = 0 \wedge (a)_2 = 2 \wedge (a)_4 = n \end{aligned}$$

*Objets du langage.* On définit $\mathtt{Term}(a)$ : « $a$ est le code d'un terme », $\mathtt{Atom}(A)$ : « $a$ est le code d'une formule atomique » et $\mathtt{Form}(a)$ : « $a$ est le code d'une formule » par :

$$\mathtt{Term}(a) \text{ ssi } \mathtt{Lg}\, a = 4 \wedge (a)_1 = 1 \wedge \begin{cases} (a)_2 = 0 \wedge \mathtt{Var}((a)_3) \quad \text{ou} \\ (a)_2 = 1 \wedge \mathtt{Fun}((a)_3, \mathtt{Lg}(a)_4) \wedge \forall i < \mathtt{Lg}(a)_4.\ \mathtt{Term}(((a)_4)_{i+1} \end{cases}$$

$$\mathtt{Atom}(a) \text{ ssi } \mathtt{Lg}\, a = 4 \wedge (a)_1 = 2 \wedge (a)_2 = 0 \wedge \mathtt{Pred}((a)_3, \mathtt{Lg}(a)_4) \wedge \forall i < \mathtt{Lg}(a)_4.\ \mathtt{Term}(((a)_4)_{i+1})$$

$$\mathtt{Form}(a) \text{ ssi } \mathtt{Lg}\, a = 4 \wedge (a)_1 = 2 \wedge \begin{cases} \mathtt{Atom}(a) & \text{ou} \\ (a)_2 = 1 \wedge (a)_4 = 0 \wedge \mathtt{Form}(a)_3 & \text{ou} \\ (a)_2 = 2 \wedge \mathtt{Form}(a)_3 \wedge \mathtt{Form}(a)_4 & \text{ou} \\ (a)_2 = 3 \wedge \mathtt{Form}(a)_3 \wedge \mathtt{Form}(a)_4 & \text{ou} \\ (a)_2 = 4 \wedge \mathtt{Form}(a)_3 \wedge \mathtt{Form}(a)_4 & \text{ou} \\ (a)_2 = 5 \wedge \mathtt{Var}(a)_3 \wedge \mathtt{Form}(a)_4 & \text{ou} \\ (a)_2 = 6 \wedge \mathtt{Var}(a)_3 \wedge \mathtt{Form}(a)_4 & \end{cases}$$

## FONCTIONS ET PRÉDICATS MANIPULANT LE FORMALISME

On définit une fonction $\mathtt{Num}$ par

$$\begin{aligned} \mathtt{Num}(0) &= \ulcorner \underline{0} \urcorner \\ \mathtt{Num}(a+1) &= \langle 1, 1, \ulcorner \underline{S} \urcorner, \langle \mathtt{Num}(a) \rangle_1 \rangle_4 \end{aligned}$$

$\mathtt{Num}$ est clairement p.r. et vérifie :

$$\mathtt{Num}(a) = \ulcorner \underline{a} \urcorner$$

pour tout entier $a$.

*Variables libres.* On définit le prédicat p.r. $\mathtt{VarLibres}(a,b)$ : « $a$ est une séquence de variables contenant toutes les variables libres de l'objet codé par $b$ », par :

$$\mathtt{VarLibres}(a,b) \text{ ssi } \forall i < \mathtt{Lg}\, a.\ \mathtt{Var}((a)_{i+1}) \wedge$$
$$\begin{cases} \mathtt{Term}(b) \wedge (b)_2 = 0 \wedge \exists i < \mathtt{Lg}\, a.\, (b)_3 = (a)_{i+1} & \text{ou} \\ \mathtt{Term}(b) \wedge (b)_2 = 1 \wedge \forall i < \mathtt{Lg}(b)_4.\ \mathtt{VarLibres}(a, ((b)_4)_{i+1}) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 0 \wedge \forall i < \mathtt{Lg}(b)_4.\ \mathtt{VarLibres}(a, ((b)_4)_{i+1}) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 1 \wedge \mathtt{VarLibres}(a, (b)_3) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 2, 3 \text{ ou } 4 \wedge (\mathtt{VarLibres}(a, (b)_3) \wedge \mathtt{VarLibres}(a, (b)_4)) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 5 \text{ ou } 6 \wedge \mathtt{VarLibres}(a * \langle (b)_3 \rangle_1, (b)_4) \end{cases}$$

REMARQUE : La dernière clause concerne une formule de la forme $A = Qx\, B$ où $Q$ est l'un des quantificateurs ; dans ce cas on dit que $X$ est l'ensemble des variables libres de $A$ si $X \cup \{x\}$ est l'ensemble des variables libres de $B$.

On définit la fonction p.r. $\mathtt{VL}(a)$ par :

$$\mathtt{VL}(a) = \begin{cases} 0 & \text{si } \neg\mathtt{Term}(a) \vee \neg\mathtt{Form}(a) \\ \mu x < \langle a, \ldots, a \rangle_a.\ (\mathtt{VarLibres}(x, a)) & \text{sinon} \end{cases}$$

On a donc pour tout objet $X$

$$\mathtt{VL}(\ulcorner X \urcorner) = \langle \ulcorner x_1 \urcorner, \ldots, \ulcorner x_n \urcorner \rangle_n$$

où $x_1$, …, $x_n$ sont toutes les variables libres de $X$.

On définit le prédicat p.r. $\mathtt{Libre}(a,b)$ (« $a$ est le code d'une variable libre dans l'objet codé par $b$ ») par :

$$\mathtt{Libre}(a,b) \text{ ssi } \mathtt{Var}(a) \wedge \begin{cases} \mathtt{Term}(b) \wedge (b)_2 = 0 \wedge a = (b)_3 & \text{ou} \\ \mathtt{Term}(b) \wedge (b)_2 = 1 \wedge \exists i < \mathtt{Lg}(b)_4.\ \mathtt{Libre}(a, ((b)_4)_{i+1}) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 0 \wedge \exists i < \mathtt{Lg}(b)_4.\ \mathtt{Libre}(a, ((b)_4)_{i+1}) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 1 \wedge \mathtt{Libre}(a, (b)_3) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 2, 3 \text{ ou } 4 \wedge (\mathtt{Libre}(a, (b)_3) \vee \mathtt{Libre}(a, (b)_4)) & \text{ou} \\ \mathtt{Form}(b) \wedge (b)_2 = 5 \text{ ou } 6 \wedge a \neq (b)_3 \wedge \mathtt{Libre}(a, (b)_4) \end{cases}$$

*Objets clos.* On définit $\mathtt{Clos}(a)$ (« $a$ est le code d'un objet clos ») par

$$\mathtt{Clos}(a) \text{ ssi } \mathtt{VarLibres}(\langle\rangle_0, a)$$

*Clôture universelle d'une formule.* Soit $A$ une formule. La *clôture universelle* $\overline{A}$ de $A$ est la formule $\forall x_1 \ldots \forall x_n\, A$ où $x_1$, …$x_n$ sont toutes les variables libres de $A$. On définit $\mathtt{Clot}(a) = F(\mathtt{VL}(a), a)$ où $F$ est donnée par

$$F(l,a) = \begin{cases} a & \text{si } \mathtt{Lg}\, l = 0 \\ \langle 2, 5, (l)_1, \mathtt{Clot}(\mathtt{Suite}\, l, a) \rangle_4 & \text{sinon} \end{cases}$$

Clairement $\mathtt{Clot}$ est p.r. vérifie

$$\mathtt{Clot}(\ulcorner A \urcorner) = \ulcorner \overline{A} \urcorner$$

pour toute formule $A$.

*Substitution.* On va maintenant définir la fonction p.r. Sub telle que :

$$\texttt{Sub}(\ulcorner X \urcorner, \ulcorner x \urcorner, \ulcorner t \urcorner) = \ulcorner X[t/x] \urcorner$$

pour tout terme ou formule $X$, et toute variable $x$ et terme $t$. Toutefois, pour résoudre le problème de « capture de variable », on va passer par une fonction légèrement plus générale ParSub qui prend en argument (le code d') un objet $X$ et (le code d') une séquence de (code de) couples $(x_i, t_i)$ et qui effectue la substitution *en parallèle* de chaque $x_i$ par $t_i$ dans $X$. L'idée est, dans le cas où l'on substitue la variable $x$ par le terme $t$ dans une formule quantifiée, par exemple $Qy\,A$, d'exploiter le fait que :

$$(Qy\,A)[t/x] \leftrightarrow Qz\,A[z/y, t/x]$$

où $z$ est une variable qui n'est pas libre dans $t$.

On définit I par $\texttt{I}(a,b) = \mu i < \operatorname{Lg} b + 1.\, ((i > 0 \wedge \pi_1(b)_i = a))$ ; donc si $b = \langle\langle a_1, b_1\rangle, \ldots, \langle a_n, b_n\rangle\rangle_n$ alors $\texttt{I}(a,b)$ est le premier $i$ tel que $a_i = a$, ou $n+1$ s'il n'existe aucun tel $i$.

On définit ParSub par :

$$\begin{array}{ll}
\multicolumn{2}{l}{\texttt{ParSub}(a,b) =} \\
0 & \text{si } \neg((\texttt{Term}(a) \vee \texttt{Form}(a)) \wedge \\
 & \quad \forall i < \operatorname{Lg} b.\, (\texttt{Var}(\pi_1(b)_i) \wedge \texttt{Term}(\pi_2(b)_i))) \\
\pi_2(b)_{\texttt{I}(a,b)} & \text{si } \texttt{Term}(a) \wedge (a)_2 = 0 \wedge \texttt{I}(a,b) \leq \operatorname{Lg} b \\
a & \text{si } \texttt{Term}(a) \wedge (a)_2 = 0 \wedge \texttt{I}(a,b) > \operatorname{Lg} b \\
\langle 1, 1, (a)_3, \langle a_1, \ldots, a_n\rangle_n\rangle_4 & \text{si } \texttt{Term}(a) \wedge (a)_2 = 1 \wedge n = \texttt{Lg}(a)_4 \wedge \\
 & \quad \forall i < n+1.\, (i = 0 \vee a_i = \texttt{ParSub}(((a)_4)_i, b)) \\
\langle 2, 0, (a)_3, \langle a_1, \ldots, a_n\rangle_n\rangle_4 & \text{si } \texttt{Form}(a) \wedge (a)_2 = 0 \wedge n = \texttt{Lg}(a)_4 \wedge \\
 & \quad \forall i < n+1.\, (i = 0 \vee a_i = \texttt{ParSub}(((a)_4)_i, b)) \\
\langle 2, 1, \texttt{ParSub}((a)_3, b), 0\rangle_4 & \text{si } \texttt{Form}(a) \wedge (a)_2 = 1 \\
\langle 2, (a)_2, \texttt{ParSub}((a)_3, b), \texttt{ParSub}((a)_4, b)\rangle_4 & \text{si } \texttt{Form}(a) \wedge (a)_2 = 3, 4 \text{ ou } 5 \\
\langle 2, (a)_2, V(a,b), \texttt{ParSub}((a)_4, \langle\langle (a)_3, V(a,b)\rangle\rangle_1 * b)\rangle_4 & \text{si } \texttt{Form}(a) \wedge (a)_2 = 5 \text{ ou } 6
\end{array}$$

où $V(a,b) = \langle 0, 0, ab, 0\rangle_4$ est le code de la variable de numéro $ab$. On s'assure ainsi que cette variable n'apparait ni dans la formule codée par $a$, ni dans aucun des termes dont les codes sont listés dans $b$. Remarquons que l'on a en particulier :

$$\texttt{ParSub}(\ulcorner Qx\,A \urcorner, \langle\langle \ulcorner x \urcorner, \ulcorner t \urcorner\rangle\rangle_1) = \ulcorner Qv\ \texttt{ParSub}(\ulcorner A \urcorner, \langle\langle \ulcorner x \urcorner, \ulcorner v \urcorner\rangle, \langle \ulcorner x \urcorner, \ulcorner t \urcorner\rangle\rangle_2) \urcorner$$

si bien que la variable $x$ est substituée par la nouvelle variable $v$ dans $A$, et non par $t$, ce qui est conforme à la définition de la substitution.

REMARQUE : Le prédicat ParSub est p.r. car il utilise essentiellement (dans la dernière clause) le schéma intitulé « variante de la récursion primitive ».

On définit finalement Sub par :

$$\texttt{Sub}(a,b,c) = \texttt{ParSub}(a, \langle\langle b, c\rangle\rangle_1)$$

*Quantificateurs bornés.* On définit le prédicat $\texttt{QuantBorn}(a)$ (« $a$ est le code d'une formule $\forall x < t.\, A$ ou $\exists x < t.\, A$ ») par :

$$\texttt{QuantBorn}(a) \text{ ssi } \texttt{Form}(a) \wedge \begin{cases} (a)_2 = 5 \wedge P((a)_4, 4, (a)_3) & \text{ou} \\ (a)_2 = 6 \wedge P((a)_4, 2, (a)_3) \end{cases}$$

où $P(a,b,c)$ est le prédicat « $a$ est le code d'une formule $x < t\,\alpha\,A'$ où $\alpha = \rightarrow$ si $b = 4$, $\alpha = \wedge$ si $b = 2$ et où $x$ est la variable de code $c$ et $t$ est un terme ne contenant pas $x$ » défini par :

$$P(a,b,c) \text{ ssi } \texttt{Form}(a) \wedge (a)_2 = b \wedge ((a)_3)_2 = 0 \wedge ((a)_3)_3 = \ulcorner < \urcorner \wedge (((a)_3)_4)_1 = c \wedge \neg\,\texttt{Libre}(c, (((a)_3)_4)_2)$$

**Théorème 1.3.2** *Il existe une fonction p.r.* `Val` *telle que pour tout terme clos t de l'arithmétique*

$$\mathtt{Val}(\ulcorner t \urcorner) = |t|$$

*Il existe un prédicat p.r.* `Vr` *telle que pour toute formule A de l'arithmétique,* $\Delta$ *et close*

$$\mathtt{Vr}(\ulcorner A \urcorner) \text{ ssi } A$$

*Preuve :* La fonction `Val` est définie par :

$$\mathtt{Val}(a) = \begin{cases} 0 & \text{si } (a)_3 = \ulcorner \underline{0} \urcorner \\ \mathtt{Val}(((a)_4)_1) + 1 & \text{si } (a)_3 = \ulcorner \underline{S} \urcorner \\ \mathtt{Val}(((a)_4)_1) + \mathtt{Val}(((a)_4)_2) & \text{si } (a)_3 = \ulcorner \underline{+} \urcorner \\ \mathtt{Val}(((a)_4)_1)\, \mathtt{Val}(((a)_4)_2) & \text{si } (a)_3 = \ulcorner \underline{.} \urcorner \\ 0 & \text{sinon} \end{cases}$$

Le prédicat `Vr` est défini par :

$$\mathtt{Vr}(a) \text{ ssi } \mathtt{Form}(a) \wedge \mathtt{Clos}(a) \wedge \begin{cases} \mathtt{Val}(((a)_4)_1) = \mathtt{Val}(((a)_4)_2) & \text{si } (a)_2 = 0 \text{ et } (a)_3 = \ulcorner = \urcorner \\ \mathtt{Val}(((a)_4)_1) < \mathtt{Val}(((a)_4)_2) & \text{si } (a)_2 = 0 \text{ et } (a)_3 = \ulcorner < \urcorner \\ \neg\, \mathtt{Vr}((a)_3) & \text{si } (a)_2 = 1 \\ \mathtt{Vr}((a)_3) \wedge \mathtt{Vr}((a)_4) & \text{si } (a)_2 = 2 \\ \mathtt{Vr}((a)_3) \vee \mathtt{Vr}((a)_4) & \text{si } (a)_2 = 3 \\ \neg\, \mathtt{Vr}((a)_3) \vee \mathtt{Vr}((a)_4) & \text{si } (a)_2 = 4 \\ \forall i < \mathtt{Val}(((((a)_4)_3)_4)_2).\ \mathtt{Vr}(\mathtt{Sub}(((a)_4)_4, (a)_3, \mathtt{Num}(i))) & \text{si } \mathtt{QuantBorn}(a) \wedge (a)_2 = \\ \exists i < \mathtt{Val}(((((a)_4)_3)_4)_2).\ \mathtt{Vr}(\mathtt{Sub}(((a)_4)_4, (a)_3, \mathtt{Num}(i))) & \text{si } \mathtt{QuantBorn}(a) \wedge (a)_2 = \end{cases}$$

La condition $x$ n'est pas variable libre de $t$ dans la définition des quantificateurs bornés nous assure d'une part que $t$ est clos, si bien que l'on est fondé à appliquer `Val` à son code, et d'autre part que $\mathtt{Sub}(((a)_4)_4, (a)_3, \mathtt{Num}(i))$ est bien le code d'une formule close. □

REMARQUE : La définition de `Vr` n'entre pas exactement dans le cadre des schémas de définitions de prédicats p.r. vus dans la section précédente. Le problème est que dans les deux derniers cas (correspondant aux quantificateurs bornés), on définit $\mathtt{Vr}(\ulcorner \forall x < t.\, A \urcorner)$ en fonction de $\mathtt{Vr}(\mathtt{Sub}(\ulcorner A \urcorner, \ulcorner x \urcorner, \mathtt{Num}(i))$. Or pour que ceci soit vraiment p.r. il faudrait que $\mathtt{Sub}(\ulcorner A \urcorner, \ulcorner x \urcorner, \mathtt{Num}(i)) < \ulcorner \forall x < t.\, A \urcorner$ ce qui est en général faux. En fait il faut définit un prédicat $V(a, b)$ à deux paramètres en utilisant la variante de la récursion primitive récursive : le paramètre $a$ est le code de la formule sur laquelle on récurre, mais il ne s'agit pas d'une formule close ; le paramètre $b$ est (le code d') un ensemble de substitutions qui remplace toutes les variables libres de $A$ par des termes clos. Par exemple, l'avant-dernière clause ci-dessus devient alors : $V(\ulcorner \forall x < t.\, A \urcorner, b)$ ssi

$$\forall i < \mathtt{Val} \ulcorner t \urcorner.\ V(\ulcorner A \urcorner, b :: [\mathtt{Num}(i)/x])$$

où $b :: [\mathtt{Num}(i)/x]$ représente (le code de) la substitution $b$ augmentée de la substitution $\mathtt{Num}(i)/x$.

## RÉSULTATS ÉLÉMENTAIRES SUR LES FONCTIONS RÉCURSIVES

Les fonctions de codage vont nous permettre d'établir quelques résultats de base en théorie des fonctions récursives. L'idée fondamentale est d'associer un *programme*, c'est à dire un objet formel identifié par un entier, à une fonction récursive. Le programme d'une fonction récursive $F$ sera ici une formule $A$ de l'arithmétique définissant la fonction et le calcul de $F(\vec{a})$ revient à rechercher un $b$ tel que $A[\underline{\vec{a}}, b]$.

**Théorème 1.3.3 (Forme normale de Kleene)** *Pour tout k il existe un prédicat p.r. $T_k$ d'arité $k+2$ tel que si F est une fonction récursive d'arité k alors il existe un entier e appelé* l'indice *de F tel que*

$$F(\vec{a}) = \pi_1 \mu x. \,(T_k(e, \vec{a}, x)) \tag{1.4}$$

REMARQUE : C'est un théorème de factorisation du schéma de minimisation.

*Preuve :* Soit donc $F$ une fonction récursive d'arité $k$. D'après le théorème 1.2.1 il y a une formule $A[\vec{x}, y, z]$ de l'arithmétique qui est $\Delta$ et telle que :

$$b = F(\vec{a}) \text{ ssi } \exists z\, A[\underline{\vec{a}}, \underline{b}, z]$$

Mais d'après le théorème 1.3.2 de la partie précédente, pour tout entier $c$, comme $A[\underline{\vec{a}}, \underline{b}, \underline{c}]$ est une formule close, on a $A[\underline{\vec{a}}, \underline{b}, \underline{c}]$ ssi $\mathtt{Vr}(\ulcorner A[\underline{\vec{a}}, \underline{b}, \underline{c}]\urcorner)$. Par définition de la fonction `ParSub` on obtient donc

$$A[\underline{\vec{a}}, \underline{b}, \underline{c}] \text{ ssi } \mathtt{Vr}(\mathtt{ParSub}(\ulcorner A\urcorner, \langle\langle \ulcorner x_1\urcorner, \mathtt{Num}(a_1)\rangle, \ldots, \langle \ulcorner x_k\urcorner, \mathtt{Num}(a_k)\rangle, \langle \ulcorner y\urcorner, \mathtt{Num}(b)\rangle, \langle \ulcorner z\urcorner, \mathtt{Num}(c)\rangle\rangle_{k+2}))$$

On définit

$$T_k(e, \vec{a}, a) \text{ ssi } \mathtt{Vr}(\mathtt{ParSub}(e, \langle\langle \ulcorner x_1\urcorner, \mathtt{Num}(a_1)\rangle, \ldots, \langle \ulcorner x_k\urcorner, \mathtt{Num}(a_k)\rangle, \langle \ulcorner y\urcorner, \mathtt{Num}(\pi_1 a)\rangle, \langle \ulcorner z\urcorner, \mathtt{Num}(\pi_2 a)\rangle\rangle_{k+2}))$$

Le prédicat $T_k$ est clairement p.r. et vérifie $T_k(\ulcorner A\urcorner, \vec{a}, a)$ ssi $A[\underline{\vec{a}}, \underline{\pi_1 a}, \underline{\pi_2 a}]$. Supposons qu'il y a un $b$ tel que $b = F(\vec{a})$. Donc par définition de $A$ il y a un $c$ tel que $A[\underline{\vec{a}}, \underline{b}, \underline{c}]$. Si on prend $a = \langle b, c\rangle$ on a montré l'existence d'un $a$ tel que $T_k(\ulcorner A\urcorner, \vec{a}, a)$. Réciproquement, si il existe un tel $a$, alors on a $A[\underline{\vec{a}}, \underline{\pi_1 a}, \underline{\pi_2 a}]$ et donc par définition de $A$ on a $\pi_1 a = F(\vec{a})$. Donc le théorème est démontré avec $e = \ulcorner A\urcorner$ où $A$ est une formule qui définit $F$. □

**Théorème 1.3.4** *Un prédicat P est récursif ssi P et $\neg P$ sont tous deux récursivement énumérables.*

*Preuve :* On a déja vu que si $P$ est récursif alors il est r.e. Si $P$ est récursif alors $\neg P$ est aussi récursif et donc $\neg P$ est également r.e. Réciproquement supposons que $P$ et $\neg P$ sont tous deux r.e. Soient $F_1$ et $F_2$ deux fonctions récursives telles que $P(\vec{a})$ ssi $F_1(\vec{a})\downarrow$ et $\neg P(\vec{a})$ ssi $F_2(\vec{a})\downarrow$ et soient $e_1$ et $e_2$ des indices pour $F_1$ et $F_2$. On a donc $F_i(\vec{a}) = \pi_1\mu x. \,(T_k(e_i, \vec{a}, x))$ et en particulier $F_i(\vec{a})\downarrow$ ssi il existe un $a$ tel que $T_k(e_i, \vec{a}, a)$. Donc

$$\chi_P(\vec{a}) = \pi_1\mu x. \,((T_k(e_1, \vec{a}, \pi_2 x) \wedge \pi_1 x = 0) \vee (T_k(e_2, \vec{a}, \pi_2 x) \wedge \pi_1 x = 1))$$

Comme pour tout $\vec{a}$ on a $F_i(\vec{a})\downarrow$ pour exactement un $i$ on déduit que la fonction $\chi_P$ est totale. □

**Théorème 1.3.5** *Soit $P(\vec{a})$ un prédicat r.e. d'arité* 1 *et non vide (il y a un $a \in \mathbb{N}$ tel que $P(a)$). Alors il existe une fonction récursive* totale *G telle que P est l'image de G, c'est à dire :*

$$P(a) \text{ ssi } \exists n,\, a = F(n)$$

*Preuve :* Par définition de r.e. il y a une fonction récursive $F$ dont $P$ est le domaine c'est à dire : $P(a)$ ssi $F(a)\downarrow$. Soit $e$ un indice de $F$. Par le théorème de Kleene on a donc :

$$P(a) \text{ ssi } \exists x\, T_1(e, a, x)$$

Soit $G$ la fonction définie par :

$$G(c) = \begin{cases} \pi_1 c & \text{si } T_1(e, \pi_1 c, \pi_2 c) \\ \pi_1\mu x. \,(T_1(e, \pi_1 x, \pi_2 x)) & \text{sinon} \end{cases}$$

La fonction $G$ est alors récursive et comme on a supposé $P$ non vide elle est totale. De plus, par construction $P$ est l'image de $G$. □

*Accolades de Kleene (Kleene's brackets).* Soit $e$ un entier. On définit

$$\{e\}(\vec{a}) = \pi_1 \mu x.\, (T_k(e, \vec{a}, x))$$

REMARQUE : La fonction $\lambda e\vec{a}.\,\{e\}(\vec{a})$ est donc récursive d'arité $k+1$. Si $F$ est $k$-récursive et $e$ est un indice pour $F$ alors

$$F(\vec{a}) = \{e\}(\vec{a})$$

**Théorème 1.3.6 (Théorème $S^n_m$)** *Pout tous entiers $m$ et $n$, il existe une fonction p.r. $S^n_m$ d'arité $m+1$ telle que*

$$\{e\}(a_1, \ldots, a_{m+n}) = \{S^n_m(e, a_{n+1}, \ldots, a_{m+n})\}(a_1, \ldots, a_n)$$

*Preuve :* On ne traite que le cas $m = 1$, le reste s'en déduit par récurrence sur $m$. On définit $S^n$ par :

$$S^n(e, a) = \mathtt{Sub}(e, \ulcorner x_{n+1} \urcorner, \mathtt{Num}(a))$$

On voit immédiatement par définition des accolades de Kleene et du prédicat $T_n$ que l'on a le résultat attendu : $\{e\}(a_1, \ldots, a_{n+1}) = \{S^n(e, a_{n+1})\}(a_1, \ldots, a_n)$. □

Nous terminerons cette partie avec une conséquence importante du théorème $S^n_m$.

**Théorème 1.3.7 (Théorème de récursivité de Kleene)** *Soit $F$ une fonction récursive d'arité $k+1$. Il existe un entier $e$ tel que*

$$\{e\}(\vec{a}) = F(\vec{a}, e)$$

C'est ce théorème qui permet de montrer que toute fonction définie par un système d'équations récursives (au sens informatique) est récursive. Un exemple notable est la fonction d'Ackermann définie par :

$$\begin{aligned} A(0, p) &= p + 1 \\ A(n+1, 0) &= A(n, 1) \\ A(n+1, p+1) &= A(n, A(n+1, p)) \end{aligned}$$

Pour montrer qu'elle est récursive on considère la fonction $H$ définie par

$$H(n, p, a) = \begin{cases} p+1 & \text{si } n = 0 \\ \{a\}(n \dot{-} 1, 1) & \text{si } n > 0 \text{ et } p = 0 \\ \{a\}(n \dot{-} 1, \{a\}(n, p \dot{-} 1)) & \text{sinon} \end{cases}$$

Alors $H$ est récursive et en applicant le théorème de récursivité de Kleene, on déduit l'existence d'un $a$ tel que $\{a\}(n, p) = H(n, p, a)$. Un tel $a$ est un indice pour la fonction $A$.

*Preuve :* Soit $G$ la fonction récursive définie par $G(\vec{a}, a) = F(\vec{a}, S^k(a, a))$ et soit $f$ un indice pour $G$. Alors on a $\{f\}(\vec{a}, a) = F(\vec{a}, S^k(a, a))$ ; mais par le théorème $S^n_m$ on en déduit que $\{S^k(f, a)\}(\vec{a}) = F(\vec{a}, S^k(a, a))$. Si on prend $a = f$ on obtient le résultat souhaité avec $e = S^k(f, f)$. □

## 1.4 Le prédicat de prouvabilité

*Codage des séquents* On rappelle qu'un séquent n'est rien d'autre qu'un couple de deux suites finies de formules ; soit $S = A_1, \ldots, A_m \vdash B_1, \ldots, B_n$ un séquent. Le code de $S$ est :

$$\ulcorner S \urcorner = \langle 3, 0, \langle \ulcorner A_1 \urcorner, \ldots, \ulcorner A_m \urcorner \rangle_m, \langle \ulcorner B_1 \urcorner, \ldots, \ulcorner B_n \urcorner \rangle_n \rangle_4$$

*Théorie.* Une théorie $T$ est un ensemble de formules closes appelées les *axiomes de* $T$, dans un langages $L$ donné. Une formule $A$ *prouvable dans la théorie* $T$ si il existe un sous-ensemble fini $T_0$ de $T$ tel que le séquent $T_0 \vdash A$ est prouvable. On dit alors que $A$ est *un théorème de* $T$ ce que l'on note $T \vdash A$.

Une théorie est *cohérente* si le séquent vide n'y est pas démontrable, c'est à dire si pour tout sous-ensemble fini $T_0$ de $T$ le séquent $T_0 \vdash$ n'est pas prouvable. Une théorie est *primitive récursive* si le prédicat

$$\mathtt{Ax}_T(a) \text{ ssi } a \text{ est le code d'un axiome de } T$$

est primitif récursif.

On dit qu'une théorie $T'$ *étend* une théorie $T$ si le langage $L'$ de $T'$ contient le langage $L$ de $T$ et si $T'$ contient $T$. On dit de plus que $T'$ est une *extension primitive récursive* de $T$ si $T$ et $T'$ sont primitive récursives

*Codage des preuves.* On se donne maintenant une numérotation des règles du calcul des séquents : 0 pour la règle axiome, 1 pour la règle de coupure, 2 et 3 pour les règles d'échanges gauche et droite, 4 et 5 pour les règles d'affaiblissement gauche et droite, ... On définit le code d'une démonstration $\pi$ du calcul des séquents dans la théorie p.r. $T$ par :

$$\ulcorner \pi \urcorner = \begin{cases} \langle 4, 0, \ulcorner A \urcorner, 0 \rangle_4 & \text{si } \pi \text{ est la démonstration réduite au seul séquent } A \vdash A \\ \langle 4, i, \ulcorner \pi_0 \urcorner, 0 \rangle_4 & \text{si } \pi \text{ est obtenue en appliquant la règle unaire de numéro } i \text{ à la preuve } \pi_0 \\ \langle 4, i, \ulcorner \pi_1 \urcorner, \ulcorner \pi_2 \urcorner \rangle_4 & \text{si } \pi \text{ est obtenue en appliquant la règle binaire de numéro } i \text{ à } \pi_1 \text{ et } \pi_2 \end{cases}$$

**Théorème 1.4.1** *Soit $T$ une théorie p.r. Il existe un prédicat p.r.* $\mathtt{Pr}_T$ *tel que :*

$\mathtt{Pr}_T(a, p)$ *ssi* $p = \ulcorner \pi \urcorner$ *et* $a = \ulcorner A \urcorner$ *où* $\pi$ *est une démonstration de* $T_0 \vdash A$ *pour un sous-ensemble fini* $T_0$ *de* $T$

Essentiellement le théorème établit que l'on peut vérifier de manière calculatoire (et même par une fonction primitive récursive ce qui est à peine plus fort) la correction d'une preuve. Il serait trop fastidieux de le démontrer ici. Il s'agit encore une fois de programmer tout un ensemble de fonctions et prédicats p.r.

*Le prédicat de prouvabilité.* On note $\mathtt{Thm}_T$ le prédicat de prouvabilité de la théorie $T$, c'est à dire que $\mathtt{Thm}_T$ est défini par :

$$\mathtt{Thm}_T(a) \text{ ssi } a \text{ est le code d'un théorème de } T$$

Comme on a $\mathtt{Thm}_T(a)$ ssi $\exists b\ \mathtt{Pr}_T(a, b)$ on voit que $\mathtt{Thm}_T$ est r.e. (dès que $T$ est p.r.).

*Théories complètes, décidables.* Une théorie $T$ est *complète* si pour toute formule $A$ close on a soit $T \vdash A$, soit $T \vdash \neg A$. Une théorie $T$ est *décidable* si $\mathtt{Thm}_T$ est un prédicat récursif.

**Théorème 1.4.2** *Si $T$ est une théorie p.r. cohérente et complète alors $T$ est décidable.*

*Preuve :* Remarquons que pour toute formule $A$ on a $T \vdash A$ ssi $T \vdash \overline{A}$. Par conséquent si $T$ est cohérente et complète alors $T \nvdash A$ ssi $T \vdash \neg\overline{A}$. Donc on a $\neg\,\mathtt{Thm}_T(a)$ ssi soit $a$ n'est pas le code d'une formule, soit $\mathtt{Thm}_T(\mathtt{Neg}(\mathtt{Clot}(a)))$. Comme la fonction $\lambda x.\ \mathtt{Neg}(\mathtt{Clot}(x))$ est p.r. et comme $\mathtt{Thm}_T$ est r.e. on voit que $\neg\,\mathtt{Thm}_T$ est r.e. d'où l'on déduit par le théorème 1.3.4 que $\mathtt{Thm}_T$ est récursif. □

## L'ARITHMÉTIQUE ÉLÉMENTAIRE

L'arithmétique élémentaire (**EA**) est l'ensemble fini (des clôtures universelles) des axiomes suivants (dans le langage de l'arithmétique :

1. Axiomes d'égalité :
   (a) $x = x$;;

(b) $x = x' \wedge y = y' \wedge x = y \to x' = y'$ ;
(c) $x = x' \to S(x) = S(x')$ ;
(d) $x = x' \wedge y = y' \to x + x' = y + y'$ :
(e) $x = x' \wedge y = y' \to x \cdot x' = y \cdot y'$ ;
(f) $x = x' \wedge y = y' \wedge x < y \to x' < y'$ ;

2. Axiomes de définitions :
   (a) $\neg S(x) = 0$ ;
   (b) $S(x) = S(y) \to x = y$ ;
   (c) $x + 0 = x$ ;
   (d) $x + S(y) = S(x + y)$ ;
   (e) $x \cdot 0 = 0$ ;
   (f) $x \cdot S(y) = x \cdot y + x$ ;
   (g) $\neg x < 0$ ;
   (h) $x < S(y) \leftrightarrow x < y \vee x = y$ ;
   (i) $x < y \vee x = y \vee y < x$ ;

**Lemme 1.4.3** L'arithmétique élémentaire est une théorie primitive récursive.

On note $\mathtt{Pr}_0$ le prédicat $\mathtt{Pr}_{\mathbf{EA}}$ et $\mathtt{Thm}_0$ le prédicat $\mathtt{Thm}_{\mathbf{EA}}$.

**Théorème 1.4.4** *Si $A$ est une formule de l'arithmétique qui est $\Delta$ et close alors :*

$$A \to \boldsymbol{EA} \vdash A$$
$$\neg A \to \boldsymbol{EA} \vdash \neg A$$

*Si $A$ est $\Sigma$ et close alors :*

$$A \to \boldsymbol{EA} \vdash A$$

*Preuve :* Le cas d'une formule $\Delta$ se déduit du lemme suivant :

**Lemme 1.4.5** Pour tous entiers $n$ et $m$ et tout terme $t$ du langage de l'arithmétique on a

$$\begin{aligned}
&\mathbf{EA} \vdash \underline{n} \underline{+} \underline{m} = \underline{m+n}\\
&\mathbf{EA} \vdash \underline{n} \,\underline{.}\, \underline{m} = \underline{mn}\\
|t| = n \to\ &\mathbf{EA} \vdash t = \underline{n}\\
n = m \to\ &\mathbf{EA} \vdash \underline{n} = \underline{m}\\
n \neq m \to\ &\mathbf{EA} \vdash \neg(\underline{n} = \underline{m})\\
n < m \to\ &\mathbf{EA} \vdash \underline{n} < \underline{m}\\
n \not< m \to\ &\mathbf{EA} \vdash \neg(\underline{n} < \underline{m})\\
&\mathbf{EA} \vdash x < \underline{n} \leftrightarrow x = \underline{0} \vee \cdots \vee x = \underline{n-1}
\end{aligned}$$

Chaque clause du lemme se démontre par récurrence sur $n$ et/ou $m$ et/ou $t$. Pour fixer les idées on va montrer la dernière clause : $\mathbf{EA} \vdash x < \underline{n} \leftrightarrow x = \underline{0} \vee \cdots \vee x = \underline{n-1}$. Si $n = 0$ cette équivalence est conséquence de l'axiome $x < 0 \vdash$ de $\mathbf{EA}$ : en effet la disjonction est alors vide donc fausse donc équivalente à $x < 0$ puisque cette formule est supposée fausse dans $\mathbf{EA}$. Supposons le résultat démontré pour $n$. On a $\mathbf{EA} \vdash x < \underline{n+1} \leftrightarrow x < \underline{n} \vee x = \underline{n}$ comme conséquence des axiomes de $\mathbf{EA}$ et du fait que $\mathbf{EA} \vdash \underline{S}\,\underline{n} = \underline{n+1}$. Or par récurrence sur $n$ on a $\mathbf{EA} \vdash x < \underline{n} \leftrightarrow x = \underline{0} \vee \cdots \vee x = \underline{n-1}$ ; on en déduit immédiatement le résultat, *i.e.*, $\mathbf{EA} \vdash x < \underline{n+1} \leftrightarrow x = \underline{0} \vee \cdots \vee x = \underline{n}$.

Le théorème se déduit du lemme par induction sur la formule $A$. Le cas d'une formule atomique est traitée presqu'entièrement par le lemme. Le seul cas compliqué est est en fait celui des formules à quantificateur borné. Supposons donc que $A = \forall x < t.\, B$. Comme $A$ est close, $t$ est clos et $B = B[x]$ a $x$ pour seule variable libre.

Si $A$ est vraie alors pour tout $a$ inférieur à $|t|$, $B[\underline{a}]$ est vraie. Par hypothèse d'induction on en déduit que $\mathbf{EA} \vdash B[\underline{a}]$ pour tout $a < |t|$. Donc $\mathbf{EA} \vdash x = \underline{a} \to B$ pour tout $a < |t|$. Notons $n = |t|$. On a $\mathbf{EA} \vdash x < \underline{n} \leftrightarrow x = \underline{0} \vee \cdots \vee x = \underline{n-1}$. Donc $\mathbf{EA} \vdash x < \underline{n} \to B$. Comme $\mathbf{EA} \vdash t = \underline{n}$ (puisque $n = |t|$) on obtient finalement le résultat souhaité : $\mathbf{EA} \vdash x < t \to B$.

Si $A$ est fausse alors il y a un $a < |t|$ tel que $B[\underline{a}]$ est fausse. Par hypothèse d'induction on a $\mathbf{EA} \vdash \neg B[\underline{a}]$, mais comme $a < |t|$ on a également $\mathbf{EA} \vdash \underline{a} < t$ d'où l'on déduit finalement $\mathbf{EA} \vdash \exists x < t.\, \neg B$. Mais cette dernière formule est logiquement équivalente à $\neg A$.

Le cas d'un $\exists$ borné se déduit de la remarque que $\mathbf{EA} \vdash \neg \forall x < t.\, B \leftrightarrow \exists x < t.\, \neg B$ (cette équivalence est d'ailleurs prouvable dans le calcul des séquents sans axiome) et du cas d'un $\forall$ borné.

Si maintenant $A$ est une formule $\Sigma$ close et que $A$ est vraie on démontre que $\mathbf{EA} \vdash A$ par induction sur $A$. Le seul cas intéressant de l'induction est $A = \exists x\, B$ où $B = B[x]$ a comme seule variable libre $x$. Comme $A$ est vraie il existe un $a$ tel que $B[\underline{a}]$ est vraie, donc par induction $\mathbf{EA} \vdash B[\underline{n}]$ d'où l'on déduit immédiatement que $\mathbf{EA} \vdash A$. □

*Fonctions et prédicats représentables dans **EA**.* Soit $F$ une fonction d'arité $k$. On dit que $F$ est *représentable* si il existe une formule $A[\vec{x}, y]$ de l'arithmétique et telle que :

$$b = F(\vec{a}) \to \mathbf{EA} \vdash A[\vec{\underline{a}}, y] \leftrightarrow y = \underline{b}$$

Soit $P$ un prédicat d'arité $k$. On dit que $P$ est *représentable* si il existe une formule $A[\vec{x}]$ telle que :

$$P(\vec{a}) \text{ ssi } \mathbf{EA} \vdash A[\vec{\underline{a}}]$$
$$\neg P(\vec{a}) \text{ ssi } \mathbf{EA} \vdash \neg A[\vec{\underline{a}}]$$

**Théorème 1.4.6 (Représentabilité)** *Si $F$ est est une fonction récursive alors $F$ est représentable. Si $P$ est un prédicat récursif alors $P$ est représentable.*

*Preuve :* Le cas d'un prédicat récursif est conséquence de celui d'une fonction. Si $P$ est récursif alors $\chi_P$ est récursive totale, donc il y a une formule $A[\vec{x}, y]$ qui représente $\chi_P$. Mais alors la formule $B[\vec{x}] = A[\vec{x}, \underline{0}]$ représente $P$. Supposons que $P(\vec{a})$, c'est à dire que $\chi_P(\vec{a}) = 0$. Alors on a $\mathbf{EA} \vdash A[\vec{\underline{a}}, y] \leftrightarrow y = \underline{0}$ d'où l'on déduit aisément en appliquant les axiomes d'égalité que $\mathbf{EA} \vdash B[\vec{\underline{a}}]$. Si au contraire on a $\neg P(\vec{a})$ c'est à dire $\chi_P(\vec{a}) = 1$ alors $\mathbf{EA} \vdash A[\vec{\underline{a}}, y] \leftrightarrow y = \underline{S\,0}$. En particulier $\mathbf{EA} \vdash A[\vec{\underline{a}}, \underline{0}] \leftrightarrow \underline{0} = \underline{S\,0}$. Comme $\mathbf{EA} \vdash \neg(\underline{0} = \underline{S\,0})$ on voit que $\mathbf{EA} \vdash \neg B[\vec{\underline{a}}]$.

On montre maintenant le cas d'une fonction récursive. C'est une conséquence de :

**Lemme 1.4.7** Si $F$ est récursive alors il existe une formule $A[\vec{x}, y]$ qui est $\Sigma$, qui définit $F$ et telle que

$$\mathbf{EA} \vdash A[\vec{x}, y] \wedge A[\vec{x}, y'] \to y = y'$$

Commençons par finir le théorème. Supposons $b = F(\vec{a})$. Comme la formule $A$ du lemme définit $F$ on a $A[\vec{\underline{a}}, \underline{b}]$ ; comme elle est $\Sigma$, par le théorème 1.4.4 on déduit que $\mathbf{EA} \vdash A[\vec{\underline{a}}, \underline{b}]$. En utilisant les axiomes d'égalité on voit immédiatement qu'alors $\mathbf{EA} \vdash y = \underline{b} \to A[\vec{\underline{a}}, y]$. Par ailleurs, d'après le lemme on a que $\mathbf{EA} \vdash A[\vec{\underline{a}}, y] \wedge A[\vec{\underline{a}}, \underline{b}] \to y = \underline{b}$. Mais comme $\mathbf{EA} \vdash A[\vec{\underline{a}}, \underline{b}]$, on voit que $\mathbf{EA} \vdash A[\vec{\underline{a}}, y] \to y = \underline{b}$.

Démontrons maintenant le lemme. Soit $A$ une formule $\Sigma$ qui définit $F$. On va construire une formule $D[\vec{x}, y]$ qui est $\Sigma$, équivalente à $A$ (donc qui définit $F$) et telle que $\mathbf{EA} \vdash D[\vec{x}, y] \wedge$

$D[\vec{x}, y'] \to y = y'$. Comme $A$ est $\Sigma$ on sait qu'il existe une formule $A_0[\vec{x}, y, z]$ qui est $\Delta$ et telle que

$$A \leftrightarrow \exists z\, A_0$$

On note $x \leq y$ pour $x < \underline{S}\, y$. On définit $B_0[\vec{x}, y, z] = \exists w \leq z\, (y \leq z \wedge z = y \,\underline{+}\, w \wedge A_0[\vec{x}, y, w])$. Clairement $B_0$ est $\Delta$ et $\exists z\, B_0$ est équivalente à $A$. Mais en plus on a

$$\mathbf{EA} \vdash B_0[\vec{x}, y, z] \to y \leq z$$

Soit maintenant $C_0 = B_0 \wedge \forall z' < z \forall y' \leq z' \neg B_0[\vec{x}, y', z']$. Alors $C_0$ est $\Delta$ et on a encore $\exists z\, C_0$ équivalente à $A$. Mais en plus on a

$$\mathbf{EA} \vdash C_0[\vec{x}, y, z] \wedge C_0[\vec{x}, y', z'] \to z = z'$$

On raisonne dans **EA** : soient $z$, $z'$ et $y$, $y'$ tels que $C_0[\vec{x}, y, z]$ et $C_0[\vec{x}, y', z']$. Supposons que $z < z'$. Comme $C_0[\vec{x}, y', z']$ on a $\forall u < z' \forall v \leq u \neg B_0[\vec{x}, v, u]$ donc en particulier pour $u = z$, $\forall v \leq z \neg B_0[\vec{x}, v, z]$. Mais comme $C_0[\vec{x}, y, z]$ on a $B_0[\vec{x}, y, z]$ et donc $y \leq z$. En prenant $v = y$ on obtient une contradiction. Donc $\neg(z < z')$. Similairement $\neg(z' < z)$ et les axiomes de **EA** entrainent alors que $z = z'$.

Soit finalement $D_0 = C_0 \wedge \forall y' < y \neg C_0[\vec{x}, y', z]$. Alors $D = \exists z\, D_0$ a les propriétés requises par le lemme. En effet $D_0$ est $\Delta$ et clairement $D$ est équivalente à $A$ donc définit $F$. Supposons que $y$ et $y'$ sont tels que $D[\vec{x}, y]$ et $D[\vec{x}, y']$. On raisonne dans **EA**. Comme $D[\vec{x}, y]$ il y a un $z$ tel que $D_0[\vec{x}, y, z]$ donc en particulier tel que $C_0[\vec{x}, y, z]$. De même, comme $D[\vec{x}, y']$, il y a un $z'$ tel que $C_0[\vec{x}, y', z']$. Mais alors $z = z'$ et donc $D_0[\vec{x}, y, z] \wedge D_0[\vec{x}, y', z]$. Mais comme précédemment on voit que l'on ne peut alors avoir $y < y'$, ni $y' < y$ d'où $y = y'$ comme annoncé. □

## 1.5 Théorèmes d'indécidabilité et d'incomplètude

**Théorème 1.5.1 (Church)** *Soit $T$ une extension p.r. de **EA**. Si $T$ est cohérente alors $T$ est indécidable.*

*Preuve :* On définit un prédicat $Q$ par :

$$Q(a) \text{ ssi } \neg\, \mathtt{Thm}_T(\mathtt{Sub}(a, \ulcorner x \urcorner, \mathtt{Num}(a))$$

Supposons que $Q$ est récursif. Par le théorème de représentation il y a une formule $A[x]$ qui représente $Q$. Posons $a = \ulcorner A \urcorner$.

Si $Q(a)$ est vrai, alors $\mathbf{EA} \vdash A[\underline{a}]$, donc comme $T$ est une extension de **EA**, $T \vdash A[\underline{a}]$. Mais alors $\mathtt{Thm}_T(\ulcorner A[\underline{a}] \urcorner)$ est vrai, *i.e.*, $\mathtt{Thm}_T(\mathtt{Sub}(\ulcorner A \urcorner, \ulcorner x \urcorner, \mathtt{Num}(a)))$, *i.e.*, $\neg Q(a)$ puisque $a = \ulcorner A \urcorner$.

Donc $Q(a)$ est faux, c'est à dire que $\mathtt{Thm}_T(\mathtt{Sub}(a, \ulcorner x \urcorner, \mathtt{Num}(a)))$ est vrai donc, comme $a = \ulcorner A \urcorner$, que $T \vdash A[\underline{a}]$. Mais si $Q(a)$ est faux, comme $A$ représente $Q$ on a $EA \vdash \neg A[\underline{a}]$, donc $T \vdash \neg A[\underline{a}]$ ce qui contredit la cohérence de $T$.

Par conséquent $Q$ ne peut être récursif. Comme `Sub` et `Num` sont toutes les deux p.r., c'est que $\mathtt{Thm}_T$ n'est pas récursif. □

**Corollaire 1.5.2** *Le calcul des prédicats est indécidable.*

*Preuve :* On note $\mathcal{N}$ la suite (finie) des formules de l'arithmétique suivantes :

**Égalité** $x = x$, $x = y \to \underline{S}\, x = \underline{S}\, y$, $x = x' \wedge y = y' \to x \,\underline{+}\, y = x' \,\underline{+}\, y'$, $x = x' \wedge y = y' \to x \,\underline{.}\, y = x' \,\underline{.}\, y'$, $x = x' \wedge y = y' \to x = y \to x' = y'$, $x = x' \wedge y = y' \to x < y \to x' < y'$ ;

**Arithmétique élémentaire** $x \,\underline{+}\, \underline{0} = x$, $x \,\underline{+}\, \underline{S}\, y = \underline{S}(x \,\underline{+}\, y)$, $x \,\underline{.}\, \underline{0} = \underline{0}$, $x \,\underline{.}\, \underline{S}\, y = x \,\underline{.}\, y \,\underline{+}\, x$, $\neg(x < 0)$, $x < \underline{S}\, y \leftrightarrow x < y \vee x = y$, $\neg(\underline{S}\, x = 0)$, $\underline{S}\, x = \underline{S}\, y \to x = y$, $x < y \vee x = u \vee y < x$.

On a alors

**Lemme 1.5.3** Soit $A$ une formule de l'arithmétique. Alors

$$\mathbf{EA} \vdash A \text{ ssi le séquent } \mathcal{N} \vdash A \text{ est prouvable dans le calcul des séquents}$$

La preuve de ce lemme est un (comme d'autres) trop fastidieuse pour être explicitée ici. On la laisse au lecteur.

Soit $\mathtt{Thm}_{\mathbf{LK}}$ le prédicat de prouvabilité du calcul des séquents défini par $\mathtt{Thm}_{\mathbf{LK}}(a)$ ssi $a$ est le code d'un séquent $S$ et $S$ est prouvable dans **LK**. D'après le lemme pour toute formule $A$, $\mathtt{Thm}_{\mathbf{EA}}(\ulcorner A \urcorner)$ est équivalent à $\mathtt{Thm}_{\mathbf{LK}}(\ulcorner \mathcal{N} \vdash A \urcorner)$, donc d'après le théorème de Church, $\mathtt{Thm}_{\mathbf{LK}}$ n'est pas récursif. □

**Corollaire 1.5.4 (Incomplétude)** *Soit $T$ une extension p.r. de **EA**. Si $T$ est cohérente alors $T$ est incomplète.*

*Preuve :* On a vu qu'une théorie cohérente et complète est décidable. □

## 1.6 Le second théorème de Gödel

**Théorème 1.6.1** *Soit $T$ une extension primitive récursive cohérente de **EA**. Il existe une formule $\Pi$ close $G_T$ telle que : $G_T$ est vraie mais $G_T$ n'est pas prouvable dans $T$.*

*Preuve :* On va commencer par énoncer le lemme suivant

**Lemme 1.6.2** Soit $Q$ un prédicat unaire. Si $Q$ est r.e. (resp. si $\neg Q$ est r.e.[2]) alors il existe une formule close $A$ de l'arithmétique qui est $\Sigma$ (resp. $\Pi$) telle que :

$$Q(\ulcorner A \urcorner) \text{ ssi } A$$

Supposons le lemme démontré et prenons $Q(a)$ ssi $\neg\,\mathtt{Thm}_T(a)$. Alors $Q$ est co-r.e. donc, d'après le lemme, il existe une formule close $G_T$ qui est $\Pi$ et telle que :

$$G_T \text{ ssi } \neg\,\mathtt{Thm}_T(\ulcorner G_T \urcorner)$$

Si $G_T$ est fausse on a donc $\mathtt{Thm}_T(\ulcorner G\urcorner_T)$, donc $T \vdash G_T$. Mais si $G_T$ est fausse, $\neg G_T$ est vraie et comme elle est (équivalente à une formule) $\Sigma$, on a $\mathbf{EA} \vdash \neg G_T$ par le théorème 1.4.4. Comme $T$ est une extension de **EA** on a également $T \vdash \neg G_T$, ce qui contredit la cohérence de $T$. Par conséquent $G_T$ est vraie et donc n'est pas prouvable dans $T$.

Démontrons maintenant le lemme. Supposons que $Q$ est r.e. (resp. co-r.e.) et soit $P$ le prédicat binaire défini par $P(a,b)$ ssi $Q(\mathtt{Sub}(a, \ulcorner x \urcorner, \mathtt{Num}(b)))$ (resp. ssi $\neg Q(\mathtt{Sub}(a, \ulcorner x \urcorner, \mathtt{Num}(b)))$). Comme $\mathtt{Sub}$ et $\mathtt{Num}$ sont p.r. $P$ est r.e. Donc il existe une formule de l'arithmétique $B[x,y]$ qui est $\Sigma$ et telle que

$$P(a,b) \text{ ssi } B[\underline{a}, \underline{b}]$$

Posons $A[x] = B[x,x]$ (resp. $A[x] = \neg B[x,x]$) et soit $a = \ulcorner A \urcorner$. Remarquons que $A$ est $\Sigma$ (resp. $\Pi$) et que $A[\underline{a}]$ est close. On a : $A[\underline{a}]$ est équivalente à $P(a,a)$ (resp. à $\neg P(a,a)$), c'est à dire à $Q(\mathtt{Sub}(a, \ulcorner x \urcorner, \mathtt{Num}(a)))$. Mais par définition de $\mathtt{Sub}$ on a $\mathtt{Sub}(a, \ulcorner x \urcorner, \mathtt{Num}(a)) = \ulcorner A[\underline{a}] \urcorner$, donc finalement $A[\underline{a}]$ est équivalente à $Q(\ulcorner A[\underline{a}] \urcorner)$. □

REMARQUE : Quand bien même $G_T$ est vraie, il se pourrait que $\neg G_T$ soit prouvable dans $T$. Il n'a jamais été dit, et c'est d'ailleurs faux, qu'une extension quelconque de **EA** ne démontre que des formules vraies. Typiquement si $A$ est une formule qui exprime la cohérence de **PA** alors $\mathbf{PA} + \neg A$ est cohérente (en vertu du second théorème de Gödel ci-dessous) et démontre $\neg A$ qui est visiblement fausse (nul ne doute que **PA** est cohérente !). On n'a donc pas démontré ici l'incomplètude de $T$ mais un résultat un peu plus faible. C'est assez normal si on considère que l'on n'a utilisé ici que la définissabilité du prédicat $\mathtt{Thm}_T$ alors que pour démontrer l'incomplètude on s'est servi de la représentabilité de $\mathtt{Thm}_T$.

[2]On dit que $Q$ est co-r.e.

*Arithmétique de Peano.* L'arithmétique de Peano **PA** est la théorie constituée des séquents de **EA** auxquels on ajoute le *principe de récurrence*, *i.e.*, tous les séquents de la forme

$$A[\underline{0}], \forall x\,(A[x] \to A[\underline{S}\,x]) \vdash A[x]$$

où $A[x]$ est une formule quelconque.

**Lemme 1.6.3** **PA** est une théorie p.r.

Remarquons que toutes les preuves que nous avons faites jusqu'à maintenant (ainsi d'ailleurs que celles que nous n'avons pas faites) n'utilise comme principe le plus évolué que le principe de récurrence. C'est en particulier le cas du théorème ci-dessus ainsi que des deux faits que l'on a utilisés pour le montrer :

1. le fait que $\mathtt{Thm}_T$ est définissable par une formule $\Sigma$ ;
2. le théorème 1.4.4 (une formule $\Sigma$ vraie est prouvable dans **EA**).

Autrement dit on a le lemme suivant

**Lemme 1.6.4** Le théorème 1.6.1 est entièrement formalisable dans **PA**, c'est à dire que, si $T$ est une extension p.r. de **EA** alors :

$$\mathbf{PA} \vdash \mathtt{Coh}(T) \to G_T \wedge \neg\,\mathtt{Thm}_T(\ulcorner G_T \urcorner)$$

Encore une démonstration fastidieuse que l'on ommet ici.

**Théorème 1.6.5 (Gödel)** *Soit $T$ une extension cohérente p.r. de **PA**. Alors $T$ ne démontre pas sa propre cohérence.*

*Preuve :* Comme $T$ étend **PA** on a $\mathbf{PA} \vdash \mathtt{Coh}(T) \to G_T \wedge \neg\,\mathtt{Thm}_T(\ulcorner G_T \urcorner)$ donc en particulier $T \vdash \mathtt{Coh}(T) \to G_T$. Donc si $T \vdash \mathtt{Coh}(T)$ alors $T \vdash G_T$ et d'après le théorème1.6.1, $T$ n'est pas cohérente. $\square$

# Chapitre 2

# Lambda-calcul

On va procéder ici à une rapide introduction au lambda-calcul. Pour plus ample information, le lecteur est invité à se reporter au livre de Krivine [3] duquel sont empruntés les concepts et résultats exposés ici.

## 2.1 Le lambda-calcul pur

Le lambda-calcul peut être décrit comme une théorie générale des fonctions. Il fournit des notations pour les deux opérations sur les fonctions :

– application d'une fonction à un argument
– définition d'une fonction à partir d'un terme, comme lorsqu'on parle de la fonction qui à $x$ associe $x + 3$. Observer que cela nécessite aussi la présence de *variables* (comme « $x$ » dans l'exemple précédent).

Ce qui peut sembler étrange au premier abord, c'est que la syntaxe du lambda-calcul ne possède que ces deux opérations : il n'y a pas d'objets « de base » (comme les entiers) sur lesquels calculer, ni d'opérations prédéfinies (addition, etc.). Il se trouve que ces objets peuvent être définis dans le lambda-calcul, comme on aura l'occasion de le voir.

On se donne un ensemble infini dénombrable $V$ de *variables*. L'ensemble des lambda-termes $\Lambda$ (ou $\Lambda(V)$ lorsqu'il est nécessaire de préciser les variables) est donné par :

– Si $x \in V$, alors $x \in \Lambda$.
– Si $t, u \in \Lambda$, alors $(t)u \in \Lambda$ (application de la « fonction » $t$ à l'« argument » $u$).
– Si $t \in \Lambda$ et $x \in V$, alors $\lambda x\, t \in \Lambda$. Ce terme représente la fonction qui à $x$ associe $t$ (dans lequel $x$ figure en général).

Il faut voir un (lambda-)terme comme un arbre possédant trois types de noeuds :

– Des feuilles, qui sont étiquetées par des variables.
– Des noeuds *application*, qui ont deux descendants (la fonction et l'argument).
– Des noeuds lambda ou *abstraction*, qui sont étiquetés par une variable et ont un descendant (le corps de la fonction).

Voici deux conventions typographiques très utiles et très répandues dans la littérature sur le sujet :

– Au lieu d'écrire $(\ldots(t)t_1\ldots)t_n$, on écrit $(t)t_1\ldots t_n$.
– Au lieu d'écrire $\lambda x_1 \ldots \lambda x_n\, t$, on écrit $\lambda x_1 \ldots x_n\, t$.

*Sous-termes.* Un *sous-terme* d'un terme $t$ est un terme apparaissant sous un noeud de $t$. Un même sous-terme de $t$ peut apparaître à plusieurs endroits différents de $t$ : il a alors plusieurs *occurrences* dans $t$. Par exemple, le terme $\lambda x\,(y)x$ a deux occurrences différentes dans le terme suivant :

$$(\lambda x\,(y)x)\lambda z\,(z)\lambda x\,(y)x$$

*Variables libres, liées.* Une variable apparaissant dans un terme est *liée* s'il y a au-dessus d'elle, dans l'arbre, un noeud lambda étiqueté par la même variable. Une variable qui n'est pas liée est *libre*. Par exemple, dans

$$t_1 = \lambda x\,(y)x,$$

la variable $x$ est liée, et la variable $y$ est libre.

Plus précisément, on peut définir par récurrence sur la taille des lambda-termes l'ensemble $\mathtt{VL}(t)$ des variables libres d'un terme $t$ :

$$\mathtt{VL}(t) = \begin{cases} \{x\} & \text{si } t = x \\ \mathtt{VL}(u) \cup \mathtt{VL}(v) & \text{si } t = (u)v \\ \mathtt{VL}(u) \setminus \{x\} & \text{si } t = \lambda x\, u \end{cases}$$

Il est clair que le nom d'une variable liée n'a aucune importance : le terme $t_1$ précédent doit être identifié au terme

$$\lambda z\,(y)z$$

Toutefois, lorsqu'on renomme en $y$ la variable liée $x$, dans une occurrence d'un sous-terme $\lambda x\, u$ d'un terme $t$, il faut prendre garde à ce que la nouvelle variable $y$ que l'on veut utiliser ne soit pas déjà libre dans $u$ : dans le terme $t_1$, on ne peut pas renommer $x$ en $y$. On obtiendrait le terme $\lambda y\,(y)y$, qui n'a pas la même signification que $t$.

*Alpha-équivalence.* Deux termes sont dits *$\alpha$-équivalents* (ou simplement *équivalents*) si on peut obtenir le second à partir du premier en renommant des variables liées (en respectant bien sûr les précautions ci-dessus concernant la capture des variables libres). Il s'agit d'une relation d'équivalence.

> A partir de maintenant, on considérera les lambda-termes à $\alpha$-équivalence près, c'est à dire que l'on écrira $t = u$ pour « $t$ est $\alpha$-équivalent à $u$ », ou « $t$ et $u$ ne diffèrent que par les noms de leur variables liées ».

*Substitution.* Une opération très importante sur les lambda-termes est la *substitution* d'une suite de termes $u_1, \ldots, u_n$ à la suite de variables libres $x_1, \ldots, x_n$ d'un terme $t$, ce qu'on note $t[u_1/x_1, \ldots, u_n/x_n]$ (ou lorsque ce n'est pas ambigu $t[\vec{u}/\vec{x}]$), et qu'on définit par récurrence sur la taille de $t$ :

$$t[\vec{u}/\vec{x}] = \begin{cases} t & \text{si } x_i \notin \mathtt{VL}(t) \text{ pour } i = 1, \ldots, n \\ u_i & \text{si } t = x_i \\ (t_1[\vec{u}/\vec{x}])t_2[\vec{u}/\vec{x}] & \text{si } t = (t_1)t_2 \\ \lambda z\, s[z/y, \vec{u}/\vec{x} \setminus y] & \text{si } t = \lambda y\, s \text{ et } z \text{ est une variable } \textit{fraîche} \end{cases}$$

Par « $z$ est une variable fraîche » on entend : $z \notin \mathtt{VL}(u_i)$ pour $i = 1, \ldots, n$ ; par $\vec{u}/\vec{x} \setminus y$ on entend la suite $u_1/x_1, \ldots, u_n/x_n$ amputée de $u_i/x_i$ si $y = x_i$.

Par exemple, si on veut substituer $y$ à $x$ et $t$ à $y$ dans le terme $t = \lambda y\,(y)x$ on obtient :

$$\begin{aligned} (\lambda y\,(y)x)[y/x, t/y] &= \lambda z\,((y)x)[z/y, y/x] \\ &= \lambda z\,(y[z/y, y/x])x[z/y, y/x] \\ &= \lambda z\,(z)y \end{aligned}$$

REMARQUE : Quand on écrit $t[t_1/x_1, \ldots, t_n/x_n]$, les $n$ substitutions sont faites simultanément, et non l'une après l'autre si bien que

$$t[t_1/x_1, \ldots, t_n/x_n] = t[t_{\sigma 1}/x_{\sigma 1}, \ldots, t_{\sigma n}/x_{\sigma n}]$$

pour toute permutation $\sigma$ de $\{1, \ldots, n\}$.

**Lemme 2.1.1** Soit $t, t_1, \ldots, t_n, u_1, \ldots, u_m$ des termes. On a

$$t[\vec{t}/\vec{x}][\vec{u}/\vec{y}] = t[\vec{t}[\vec{u}/\vec{y}]/\vec{x}, \vec{u}/\vec{y}]$$

La preuve de ce lemme est laissée en exercice (simple récurrence sur la taille de $t$).

## Généralités sur les relations

Soit $E$ un ensemble. Pour nous, une relation sur $E$ est un sous-ensemble de $E \times E$ (on ne considère ici que des relations binaires). Soit $R$ une relation sur $E$. On écrit souvent $a\ R\ b$ au lieu de $(a, b) \in R$.

On dit que $R$ est réflexive si

$$\forall a \in E \quad a\ R\ a\ .$$

On dit que $R$ est symétrique si

$$\forall a, b \in E \quad a\ R\ b \rightarrow b\ R\ a\ .$$

On dit que $R$ est transitive si

$$\forall a, b, c \in E \quad a\ R\ b \wedge b\ R\ c \rightarrow a\ R\ c\ .$$

On dit que $R$ est une relation d'équivalence si elle a ces trois propriétés.

*Clôtures symétrique et transitive.* La *clôture symétrique* de $R$ est la plus petite (pour l'inclusion) relation $S$ sur $E$ qui contient $R$ et qui est symétrique. On vérifie facilement que

$$a\ S\ b \leftrightarrow a\ R\ b \vee b\ R\ a\ .$$

La *clôture transitive* de $R$ est la plus petite relation $T$ sur $E$ qui contient $R$ et qui est réflexive et transitive. On vérifie facilement que $a\ T\ b$ si et seulement s'il existe une suite $a_1, \ldots a_n$ telle que $a_1 = a$, $a_n = b$ et, pour chaque $i = 1, \ldots, n-1$, on a $a_i\ R\ a_{i+1}$.

La clôture symétrique transitive de $R$ est la plus petite relation d'équivalence sur $E$ qui contienne $R$. On vérifie facilement que c'est la clôture transitive de la clôture symétrique de $R$.[1]

*Confluence.* La relation $R$ est dite *confluente* si, pour tous $a, a_1, a_2 \in E$ tels que $a\ R\ a_1$ et $a\ R\ a_2$, il existe $b \in E$ tel que $a_1\ R\ b$ et $a_2\ R\ b$.

**Lemme 2.1.2** La clôture transitive d'une relation confluente est confluente.

La preuve est laissée en exercice.

*Congruences.* Soit $R$ une relation sur $\Lambda$. On dit que $R$ est une *congruence* ou que $R$ *passe au contexte* si $R$ est réflexive et vérifie

– $(u)v\ R\ (u')v'$ dès que $u\ R\ u'$ et $v\ R\ v'$ ;
– $\lambda x\, u\ R\ \lambda x\, u'$ dès que $u\ R\ u'$.

## La $\beta$-réduction

Un *rédex* est un lambda-terme de la forme

$$(\lambda x\, t)u$$

La $\beta$-réduction (la seule règle de calcul du lambda-calcul, pour nous du moins) exprime qu'un tel rédex, où qu'il se trouve dans un terme, peut être *contracté* en

$$t[u/x]$$

ce qui se justifie bien, si l'on a présent à l'esprit que l'expression $(\lambda x\, t)u$ représente l'application à $u$ de la fonction qui à $x$ associe $t$ (dans lequel $x$ figure libre en général).

[1] et non pas la clôture symétrique de la clôture transitive de $R$ (cette dernière relation n'a d'ailleurs aucune raison d'être une relation d'équivalence).

*Beta-réduction.* Un terme $t$ se $\beta$-réduit en une étape en un terme $t'$ si $t'$ s'obtient à partir de $t$ en sélectionnant dans $t$ *exactement un* rédex (c'est-à-dire une occurrence de sous-terme de $t$ qui est un rédex) et en contractant ce rédex. Plus précisément, on définit la relation $\beta_0$ par récurrence sur son premier argument :

- $x \ \beta_0 \ u$ n'est jamais vrai quand $x \in V$ ;
- $(\lambda x\, u)v \ \beta_0 \ u[v/x]$ ;
- $(u)v \ \beta_0 \ (u')v$ si $u \ \beta_0 \ u'$ ;
- $(u)v \ \beta_0 \ (u)v'$ si $v \ \beta_0 \ v'$ ;
- $\lambda x\, u \ \beta_0 \ \lambda x\, u'$ si $u \ \beta_0 \ u'$.

*Forme normale.* Un terme $t$ est dit *normal* s'il ne contient aucun rédex, autrement dit, s'il n'existe aucun terme $u$ tel que $t \ \beta_0 \ u$. Par exemple, le terme

$$\lambda f\,(f)\lambda x\,(f)\lambda d\,x$$

est normal.

On note $\beta$ la clôture transitive de $\beta_0$ et $=_\beta$ la clôture symétrique et transitive de $\beta_0$. Deux termes qui sont reliés par cette relation d'équivalence sont dits *$\beta$-équivalents*.

**Lemme 2.1.3** Les relations $\beta$ et $=_\beta$ passent au contexte.

REMARQUE : En fait la relation $\beta$ (resp. $=_\beta$) est la plus petite relation qui (i) passe au contexte, (ii) est transitive (resp. est une relation d'équivalence) et (iii) relate $(\lambda x\, u)v$ et $u[v/x]$.

*Termes (fortement) normalisables.* Un terme $t$ est *normalisable* s'il existe un terme normal $u$ tel que $t \ \beta \ u$, et alors on dit que $u$ est une *forme normale* de $t$. On dit que $t$ est *fortement normalisable* s'il n'y a pas de suite infinie de terme $(t_i)_{i=1,2,\ldots}$ telle que $t_1 = t$ et $t_i \ \beta_0 \ t_{i+1}$ pour tout $i$.

Clairement, tout terme fortement normalisable est normalisable.

Voici quelques exemples. Soit $\Delta = \lambda x\,(x)x$. Le terme $(\Delta)\Delta$ n'est pas normalisable, puisqu'il se réduit en une étape en $(\Delta)\Delta$. Le terme

$$t = (\lambda d\, z)(\Delta)\Delta$$

est normalisable, mais non fortement normalisable.

## LE THÉORÈME DE CHURCH-ROSSER

*Confluence.* Un même terme peut contenir plusieurs rédex, et donc il y a plusieurs façons de le réduire en général. Par exemple, dans le terme $t$ ci-dessus, la réduction du rédex le plus extérieur aboutit en une étape à une forme normale, alors que la réduction de l'autre rédex aboutit à un terme égal à $t$.

On peut donc se demander si, en réduisant un terme de deux façons différentes, on peut aboutir à deux formes normales distinctes. Ce n'est pas le cas, grâce au théorème de Church-Rosser :

**Théorème 2.1.4 (Church-Rosser)** *La relation $\beta$ sur $\Lambda$ est confluente.*

Admettons ce théorème pour un moment. Soient $t$ un terme et $t_1$, $t_2$ deux termes normaux tels que $t \ \beta \ t_1$ et $t \ \beta \ t_2$. Par le théorème, il existe un terme $t'$ tel que $t_1 \ \beta \ t'$ et $t_2 \ \beta \ t'$. Comme $t_1$ est normal, $t_1 = t'$, et de même, $t_2 = t'$. Donc $t_1 = t_2$ et on vient de montrer le

**Corollaire 2.1.5** *Un terme normalisable du lambda-calcul a une unique forme normale.*

Une autre conséquence intéressante (et tout aussi immédiate) est

**Corollaire 2.1.6** *Soient $t$ et $u$ deux termes. On a $t =_\beta u$ ssi il existe $v$ tel que $t\ \beta\ v$ et $u\ \beta\ v$.*

En particulier deux termes normalisables sont $\beta$-équivalent ssi ils ont la même forme normale.

Le théorème 2.1.4 serait immédiat (grâce au lemme 2.1.2) si la relation $\beta_0$ elle-même était confluente. Ce n'est malheureusement pas le cas, comme le montre l'exemple suivant.

Soit $I = \lambda x\, x$ (l'identité). Soit

$$t = (\Delta)(I_1)I_2$$

où l'on a indicé les deux occurrences de $I$ pour les distinguer. Ce terme contient deux rédex. Selon que l'on réduit l'un ou l'autre on obtient :

$$t_1 = ((I_1)I_2)(I_1)I_2 \qquad \text{ou} \qquad t_2 = (\Delta)I_2$$

Comme $t_2$ n'a qu'un redex, il y a un unique $t_3$ tel que $t_2\ \beta_0\ t_3$, à savoir :

$$t_3 = (I_2)I_2$$

Par contre $t_1$ contient deux rédex, mais la réduction d'aucun de ces deux rédex ne permet (en une étape) d'obtenir $t_3$ à partir de $t_1$. Donc la relation $\beta_0$ n'est pas confluente.

Cet exemple explicite la raison pour laquelle $\beta_0$ n'est pas confluente : le terme $t$ contient deux rédex, et la réduction du rédex le plus extérieur ($t$ lui-même) a pour effet de *dupliquer* l'autre rédex (le terme $(I_1)I_2$). Du coup, il faut faire deux étapes de $\beta_0$ pour passer de $t_1$ à $t_3$ et refermer le diagramme de confluence. Observer toutefois que, pour refermer ce diagramme, on ne réduit que des rédex qui étaient *initialement présents* dans $t$ ; le terme $t_3$ est un rédex qui n'était pas initialement présent dans $t$, mais sa réduction n'est pas utile pour refermer le diagramme de confluence.

*Création de redex.* L'exemple illustre un autre phénomène très important en lambda-calcul, la *création de redex*. Comme $I = \lambda x\, x$, le terme $t_3$ contient un redex qui, comme on l'a déja dit, *n'est pas initialement présent dans $t$*, donc a été créé par la réduction. Le fait qu'une étape de réduction en général crée de nouveaux redex est la clef de la complexité calculatoire du lambda-calcul. C'est en particulier la raison pour laquelle il existe des termes non normalisables : par exemple si l'on fait une étape de réduction sur $(\Delta)\Delta$ alors on crée un nouveau redex qui se trouve être le terme de départ. Pour se convaincre qu'il y a vraiment création de redex, indicer les deux occurrences de $\Delta$, et faire la réduction.

*Réduction parallèle.* Nous allons montrer le théorème de Church-Rosser en définissant une relation $\rho$ (appelé parfois *réduction parallèle*), qui contient $\beta_0$, est confluente, et dont la clôture transitive est $\beta$. L'idée est de supprimer la restriction de $\beta_0$ (*exactement un* redex) qui est comme on vient de le voir, la raison pour laquelle $\beta_0$ n'est pas confluente. Intuitivement on aura donc $t\ \rho\ t'$ si il existe un ensemble (possiblement vide) de redex de $t$ tel que lorsqu'on les réduit tous (en parallèle), on obtient $t'$ ; ou encore si $t$ se réduit en $t'$ sans qu'aucune des étapes de réductions ne choisisse un redex *créé* par les étapes précédentes.

On défini $t\ \rho\ u$ pour tout $u$ par récurrence sur la longueur de $t$ :

- $t\ \rho\ t$ pour tout $t$ ;
- $(\lambda x\, u)v\ \rho\ u'[v'/x]$ si $u\ \rho\ u'$ et $v\ \rho\ b'$ ;
- $(u)v\ \rho\ (u')v'$ si $u\ \rho\ u'$ et $v\ \rho\ b'$ ;
- $\lambda x\, u\ \rho\ \lambda x\, u'$ si $u\ \rho\ u'$.

**Lemme 2.1.7** La relation $\rho$ est la plus petite relation qui passe au contexte et telle que si $u\ \rho\ u'$ et $v\ \rho\ v'$ alors $(\lambda x\, u)v\ \rho\ u'[v'/x]$.

La démonstration est immédiate.

**Lemme 2.1.8**

$$\beta_0 \subseteq \rho \subseteq \beta$$

*Preuve :* Clairement $\rho$ contient $\beta_0$. Et comme $\beta$ est une (la plus petite en fait) relation transitive passant au contexte et vérifiant la condition de clôture de $\rho$, $\beta$ contient $\rho$. $\square$

On en déduit le

**Lemme 2.1.9** La clôture transitive de $\rho$ est $\beta$.

**Lemme 2.1.10** Soient $t, t', u, u' \in \Lambda$ tels que $t \; \rho \; t'$ et $u \; \rho \; u'$, et soit $x \in V$. On a $t[u/x] \; \rho \; t'[u'/x]$.

*Preuve :* Par récurrence sur la taille de $t$.

Si $t$ est une variable $y$, alors $t' = y$ et on conclut immédiatement, que $y$ soit ou non égal à $x$.

Si $t = (s)r$ et $t' = (s')r'$ avec $s \; \rho \; s'$ et $r \; \rho \; r'$, on a, par hypothèse de récurrence, $s[u/x] \; \rho \; s'[u'/x]$ et $r[u/x] \; \rho \; r'[u'/x]$, donc

$$(s[u/x])r[u/x] \; \rho \; (s'[u'/x])r'[u'/x]$$

c'est-à-dire (par définition de la substitution), $t[u/x] \; \rho \; t'[u'/x]$.

Si $t = (\lambda y \, s)r$ et $t' = s'[r'/y]$, avec $s \; \rho \; s'$ et $r \; \rho \; r'$, observer d'abord qu'on peut supposer $y \neq x$ (quitte à renommer la variable liée $y$). Par hypothèse de récurrence, on a

$$s[u/x] \; \rho \; s'[u'/x] \quad \text{et} \quad r[u/x] \; \rho \; r'[u'/x]$$

et donc, par définition de $\rho$,

$$(\lambda y \, s[u/x])r[u/x] \; \rho \; s'[u'/x][r'[u'/x]/y] \; ,$$

or par définition de la substitution (vu que $y \neq x$, et qu'on peut supposer que $y \notin \mathtt{VL}(u)$), on a $(\lambda y \, s[u/x])r[u/x] = t[u/x]$, et par le lemme 2.1.1, on a

$$s'[r'/y][u'/x] = s'[u'/x, r'[u'/x]/y] \; ,$$

et comme on peut supposer que $y \notin \mathtt{VL}(u')$,

$$s'[u'/x, r'[u'/x]/y] = s'[u'/x][r'[u'/x]/y]$$

et on conclut que

$$(\lambda y \, s[u/x])r[u/x] \; \rho \; s'[r'/y][u'/x] \; ,$$

ce qu'il fallait prouver.

Le cas où $t = \lambda y \, s$ et $t' = \lambda y \, s'$ (encore une fois, on peut supposer $y \neq x$) est laissé au lecteur en exercice. $\square$

On conclut la preuve du théorème 2.1.4 au moyen du lemme suivant.

**Lemme 2.1.11** La relation $\rho$ est confluente sur $\Lambda$.

*Preuve :* Soient $t, t_1, t_2 \in \Lambda$ tels que $t \; \rho \; t_1$ et $t \; \rho \; t_2$. On prouve par induction sur la longueur de $t$ qu'il existe $u \in \Lambda$ tel que $t_i \; \rho \; u$ pour $i = 1, 2$.

Si $t = x \in V$, alors $t_1 = t_2 = x$ et on peut prendre $u = x$.

Si $t = (u)v$ et $t_i = (u_i)v_i$ avec $u \; \rho \; u_i$, $v \; \rho \; v_i$ alors par hypothèse de récurrence sur il existe des termes $u'$ et $v'$ tels que $u_i \; \rho \; u'$ et $v_i \; \rho \; v'$ pour $i = 1, 2$. Mais si on prend $t' = (u')v'$, comme $\rho$ passe au contexte on voit que $t_i \; \rho \; t'$ pour $i = 1, 2$.

Si $t = (\lambda x \, u)v$ alors il y a deux cas à considérer :

- $t_1 = (\lambda x \, u_1)v_1$ et $t_2 = u_2[v_2/x]$ où $u \; \rho \; u_i$ et $v \; \rho \; v_i$ pour $i = 1, 2$. Par hypothése de récurrence on a alors $u'$ et $v'$ tels que $u_i \; \rho \; u'$ et $v_i \; \rho \; v'$ pour $i = 1, 2$. Par définition de $\rho$ on a donc $t_1 \; \rho \; u'[v'/x]$ et par le lemme de substitution ci-dessus $t_2 \; \rho \; u'[v'/x]$, soit le résultat cherché avec $t' = u'[v'/x]$.
- $t_i = u_i[v_i/x]$ où $u \; \rho \; u_i$ et $v \; \rho \; v_i$ pour $i = 1, 2$. On obtient à nouveau le résultat avec $t' = u'[v'/x]$ en appliquant deux fois le lemme de substitution.

$\square$

### RÉDUCTION DE TÊTE ET RÉDUCTION GAUCHE

Le résultat suivant est une observation très simple, mais très importante, sur la forme générale d'un lambda-terme.

**Lemme 2.1.12** Soit $t \in \Lambda$. Il existe une unique famille de variable $x_1, \ldots, x_n$ (avec $n \geq 0$), une unique famille de termes $u_1, \ldots, u_k \in \Lambda$ (avec $k \geq 0$), et un unique terme $u \in \Lambda$ qui est *soit une variable, soit un rédex*, tels que

$$t = \lambda x_1 \ldots x_n\, (u) u_1 \ldots u_k \,.$$

L'écriture du terme $t$ fournie par ce résultat sera appelée écriture *canonique* de $t$.

*Forme normale de tête, variable de tête.* Avec les notations du lemme, si $u$ est un rédex, on dit que c'est le *rédex de tête* de $t$. Si $u$ est une variable, on dit que c'est la *variable de tête* de $t$, et que $t$ est en *forme normale de tête* ou est une *forme normale de tête*.

Un terme en forme normale est évidemment aussi en forme normale de tête, mais la réciproque est fausse. Par exemple, le terme

$$\lambda x\, (x)(\Delta)\Delta$$

est en forme normale de tête, mais il n'est pas en forme normale.

*Réduction de tête.* Soient $t$ et $t'$ deux termes et $\lambda\vec{x}(t_0)\vec{t}$ l'écriture canonique de $t$. On dit que $t$ *se réduit de tête* en $t'$ en une étape, et on écrit $t\ \beta_0^t\ t'$ si $t_0$ est un rédex $(\lambda x\, u)v$ et

$$t' = \lambda\vec{x}(u[v/x])\vec{t}.$$

La réduction de tête est la clôture transitive de la réduction de tête en une étape. Elle sera notée $\beta^t$. Il est clair que $\beta_0^t \subseteq \beta_0$ et que $\beta^t \subseteq \beta$ (exercice : montrer par des contre-exemples que ces inclusions sont strictes).

Remarquer que dans un terme, il y a au plus un rédex de tête, donc contrairement à la $\beta$-réduction, la $\beta$-réduction de tête est une stratégie de réduction (à chaque étape, on n'a qu'une seule possibilité pour continuer).

Sur un terme en forme normale de tête, il est bien sûr impossible d'exécuter une réduction de tête.

*Formes normales de tête et beta-équivalence.* Observer qu'un terme peut avoir plusieurs formes normales de tête. Toutefois, si deux termes $t$ et $t'$, tous deux en forme normale de tête, sont $\beta$-équivalents, disons :

$$t = \lambda x_1 \ldots x_n\, (y) u_1 \ldots u_k \quad \text{et} \quad t' = \lambda x'_1 \ldots x'_{n'}\, (y') u'_1 \ldots u'_{k'} \,,$$

il faut qu'on ait $n' = n$, $k' = k$ et (au renommage près des variables liées), $x'_1 = x_1$,..., $x'_n = x_n$, $y' = y$ et $u'_1 =_\beta u_1$,...$u_k =_\beta u'_k$, comme le montre le corollaire 2.1.6. En effet, les seules $\beta$-réductions que l'on peut effectuer sur $t$ sont à l'intérieur des $u_i$, et elle ne peuvent pas affecter l'ossature de $t$ (les $\lambda$ extérieurs $\lambda x_1 \ldots x_n$, la variable de tête et le nombre de ses arguments).

*Normalisation de tête* On dira qu'un terme $t$ est *normalisable de tête* si sa réduction de tête termine, autrement dit s'il n'y a pas de suite infinie de terme $t_1, t_2, \ldots$ telle que $t_i\ \beta_0^t\ t_{i+1}$ pour tout $i$. Dans ce cas, le dernier élément de la suite des réduits de tête de $t$ est évidemment une forme normale de tête.

Voici un petit lemme utile sur la normalisabilité de tête.

**Lemme 2.1.13** Soient $t$, $u_1$,..., $u_n$, $v_1$, ..., $v_m$ des termes. Si $(t[\vec{u}/\vec{x}])\vec{v}$ est normalisable de tête alors $t$ l'est.

*Preuve :* Par récurrence sur la longueur de la réduction de tête de $t' = (t[\vec{u}/\vec{z}])\vec{v}$. Si $t'$ est en forme normale de tête alors $t$ l'est ; sinon on a

$$t = \lambda\vec{y}.\, ((\lambda y.\, t_0)t_1)\vec{w}$$

d'où
$$t' = (\lambda\vec{y}.\,((\lambda y.\,t_0[\vec{u}/\vec{x}])t_1[\vec{u}/\vec{x}])\vec{w}[\vec{u}/\vec{x}])\vec{v}$$
et $t'$ n'est clairement pas en forme normale de tête.

Si $t'$ n'est pas en forme normale de tête on raisonne par cas sur $t$. Si $t$ est en forme normale de tête il n'y a rien à dire. Si $t = (r)\vec{w}$ où $r$ est un redex alors $t' = (r[\vec{u}/\vec{x}])\vec{w}[\vec{u}/\vec{x}]\vec{v}$ donc $t' \; \beta_0^t \; t'' = (r'[\vec{u}/\vec{x}])\vec{w}[\vec{u}/\vec{x}]\vec{v}$ où $r'$ est le contracté de $r$. Mais $t \; \beta_0^t \; t''' = (r')\vec{w}$ et $(t'''[\vec{u}/\vec{x}])\vec{v} = t''$ ; par récurrence sur la longueur de la réduction de tête de $t''$ on a que la réduction de tête de $t'''$ termine et donc celle de $t$ puisque $t'''$ est son réduit de tête à une étape.

Enfin si $t = \lambda x.\,t_0$ alors $t' = (\lambda x.\,t_0[\vec{u}/\vec{x}])\vec{v}$ donc $t' \; \beta_0^t \; t'' = (t_0[\vec{u}/\vec{x}, v_1/x])v_2 \dots v_n$. Par récurrence sur la longueur de la réduction de tête de $t''$ on a que la réduction de tête de $t_0$ termine, et donc celle de $t$ par définition de la réduction de tête. □

*Le théorème de normalisation de tête.* Il s'agit de l'un des résultats de base de la théorie du lambda-calcul, au même titre que le théorème de Church-Rosser. On l'énonce ici mais on le démontrera en 2.3.

**Théorème 2.1.14** *Un terme est normalisable de tête si et seulement s'il est $\beta$-équivalent à un terme en forme normale de tête.*

Autrement dit, la réduction de tête est une stratégie qui aboutit toujours à une forme normale de tête, dès lors qu'on l'applique à un terme $\beta$-équivalent à un terme en forme normale de tête.

*Réduction gauche.* Soient $t$ et $t'$ deux termes. On dit que $t$ *se réduit à gauche* en $t'$ en une étape, et on écrit $t \; \beta_0^g \; t'$ si, soit $t$ a un redex de tête et $t'$ est obtenu par contraction d'ycelui, soit $t = \lambda\vec{x}.\,(x)t_1 \dots t_n$ est en forme normale de tête et $t' = \lambda\vec{x}.\,(x)t_1 \dots t_{i-1}t'_i t_{i+1} \dots t_n$ où $t'_i$ est un réduit gauche à une étape de $t_i$ pour un $i$. Autrement dit la réduction gauche est l'extension de la réduction de tête aux arguments de la variable de tête. On note $\beta^g$ la clôture transitive de $\beta_0^g$.

Remarquons que la réduction gauche, contrairement à la réduction de tête, n'est pas une stratégie car elle ne détermine pas un unique redex dans le cas où $t$ n'est pas en forme normale de tête. On pourrait et c'est d'ailleurs souvent ainsi qu'elle est présentée dans la littérature, la déterminiser complètement en demandant que l'on ne réduit dans $t_i$ que si $t_1, \dots, t_{i-1}$ sont normaux. On peut aussi considérer que les redex de chaque $t_i$ sont indépendants et que la réduction gauche les réduit tous en parallèle.

La réduction gauche admet une propriété du même genre que la réduction de tête :

**Théorème 2.1.15** *Un terme est normalisable si et seulement si l'une des ses réductions gauche termine.*

REMARQUE : Il est relativement facile de voir que si l'une des réductions gauche termine alors toutes terminent.

On ne démontrera pas ce théorème dans ces notes. Le lecteur pourra exercer sa compréhension de la méthode de réductibilité exposée en 2.3 en tâchant d'adapter la preuve du théorème précédent. Une autre méthode consiste à utiliser le théorème précédent et le fait que la réduction gauche est une itération de la réduction de tête.

## 2.2 Représentation des fonctions récursives

*Entiers de Church.* Soit $n$ un entier. On note $\underline{n}$ le lambda-terme
$$\underline{n} = \lambda f.\,\lambda x.\,\underbrace{(f)\dots(f)}_{n\times}x$$

Les lambda-termes de la forme $\underline{n}$ sont appelés les *entiers de Church*. Remarquons qu'ils ont une interprétation fonctionnelle très simple : ce sont des itérateurs.

*Représentation des fonctions.* Soit $F : \mathbb{N}^k \to \mathbb{N}$ une fonction partielle. On dira que le lambda-terme $\underline{F}$ *représente* $F$ si pour tous entiers $n_1, \ldots, n_k$ on a :

- si $F(\vec{n})\uparrow$ alors le terme $(\underline{F})\underline{\vec{n}}$ n'a pas de forme normale de tête ;
- si $F(\vec{n}) = m$ alors $(\underline{F})\underline{\vec{n}} =_\beta \underline{m}$

REMARQUE : D'après le théorème 2.1.14, la première condition est équivalente à : si $F(\vec{n})\uparrow$ alors la réduction de tête de $(\underline{F})\underline{\vec{n}}$ ne termine pas.

*Fonctions de base sur les entiers.* On a en particulier $\underline{0} = \lambda f.\,\lambda x.\,x$. On voit facilement que le terme

$$\underline{S} = \lambda n.\,\lambda f.\,\lambda x.\,(f)((n)f)x$$

représente la fonction successeur au sens où $(\underline{S})\underline{n} =_\beta \underline{n+1}$. De même les termes

$$\mathtt{Add} = \lambda n.\,\lambda m.\,\lambda f.\,\lambda x.\,((n)f)((m)f)x$$

et

$$\mathtt{Mult} = \lambda n.\,\lambda m.\,\lambda f.\,\lambda x.\,((n)(m)f)x$$

représente respectivement l'addition et la multiplication

Enfin le terme

$$\mathtt{Inf} = \lambda n.\,\lambda m.\,((((\underline{S})n)\varphi)\lambda d.\,\underline{0})((m)\varphi)\lambda d.\,\underline{1}$$

où $\varphi = \lambda f.\,\lambda g.\,(g)f$ représente la fonction caractéristique de $<$.

*Un terme de mise en mémoire.* On note $T$ le terme

$$T = \lambda f.\,\lambda n.\,(((n)\lambda h.\,\lambda x.\,(h)(\underline{S})x)f)\underline{0}$$

**Lemme 2.2.1** Soit $t$ et $u$ deux termes.

- Si il existe un entier $n$ tel que $u =_\beta \underline{n}$ alors $(T)tu =_\beta (t)\underline{n}$ ;
- Si $u$ n'a pas de forme normale de tête alors $(T)tu$ non plus.

*Preuve :* Commençons à effectuer la réduction de tête de $(T)tu$. Pour alléger le calcul on note $\varphi$ le terme $\lambda h.\,\lambda x.\,(h)(\underline{S})x$.

$$\begin{aligned}(T)tu\ &\beta_0^t\ \lambda n.\,(((n)\varphi)t)\underline{0}\\ &\beta_0^t\ (((u)\varphi)t)\underline{0}\end{aligned}$$

Si $u$ n'a pas de forme normale de tête alors sa réduction de tête est infinie et, par le lemme 2.1.13, celle de $(((u)\varphi)t)\underline{0}$ aussi. Par le théorème 2.1.14 on en déduit que $(T)tu$ n'a pas de forme normale de tête.

Si $u =_\beta \underline{n}$ alors on voit facilement qu'il existe des termes $u', u_0, u_1, \ldots, u_n$ tels que :

$$\begin{aligned}u\ &\beta^t\ \lambda f.\,u'\\ u'\ &\beta^t\ \lambda x.\,u_0\\ u_0\ &\beta^t\ (f)u_1\\ &\vdots\\ u_{n-1}\ &\beta^t\ (f)u_n\\ u_n\ &\beta^t\ x\end{aligned}$$

Alors la réduction de tête de $(T)tu$ se continue ainsi :

$$\begin{aligned}
(((u)\varphi)t)\underline{0} \;&\beta^t\; (((\lambda f.\, u')\varphi)t)\underline{0}\\
&\beta_0^t\; ((u'[\varphi/f])t)\underline{0}\\
&\beta^t\; ((\lambda x.\, u_0')t)\underline{0}\\
&\beta_0^t\; (u_0'')\underline{0}\\
&\beta^t\; ((\varphi)u_1'')\underline{0}\\
&\beta_0^t\; (\lambda x.\, (u_1'')(\underline{S})x)\underline{0}\\
&\beta_0^t\; (u_1'')(\underline{S})\underline{0}\\
&\vdots\\
&\beta^t\; (u_n'')(\underline{S})\ldots(\underline{S})\underline{0}\\
&\beta^t\; (t)(\underline{S})\ldots(\underline{S})\underline{0}
\end{aligned}$$

où $u_0' = u_0[\varphi/f]$ et $u_i'' = u_i[\varphi/f, t/x]$. Mais comme $(\underline{S})\ldots(\underline{S})\underline{0} =_\beta \underline{n}$, le dernier terme est clairement beta-équivalent à $(t)\underline{n}$, comme annoncé. □

REMARQUE : Le terme $T$ fait partie de la classe des *termes de mise en mémoire*. Ce sont des termes qui ont la propriété que l'on vient de voir, à savoir que pour tout $n$, il existe un terme $\theta_n$ (ici $\theta_n = (\underline{S})\ldots(\underline{S})\underline{0}$) tel que $(T)tu\ \beta^t\ (t)\theta_n$ dès que $u =_\beta \underline{n}$. Le terme $T$ applique donc $t$ à $u$ mais après avoir effectué la normalisation complète de $u$ et donc calculé sa valeur $\theta_n$. Si on pense que la réduction de tête est une stratégie *paresseuse*, selon la terminologie des langages fonctionnels, c'est à dire une stratégie qui ne calcule la valeur des arguments que lorsque c'est strictement nécessaire, alors le terme $T$ permet de faire de *l'appel par valeur*, c'est à dire de calculer l'argument avant de le passer à la fonction.

*Le schéma de composition.* Soit $F$ la fonction définie par :

$$F(a) = \mu x.\, (x = 0)$$

Clairement $F$ est représentée par $\underline{F} = \lambda d.\, \underline{0}$. Soit maintenant $G$ donnée par

$$G(a) = \mu x.\, (x + 1 = 0)$$

Tout aussi clairement $G$ est représentée par $\underline{G} = \lambda d.\, (\Delta)\Delta$. Finalement soit $H$ définie par

$$H(a) = F(G(a))$$

Alors par définition du schéma de composition, $H$ est, comme $G$, la fonction nulle part définie. On voit donc que l'on ne peut pas représenter $H$ par le terme $h = \lambda x.\, (\underline{F})(\underline{G})a$. En effet pour tout $n$ on a

$$(h)\underline{n} =_\beta \underline{0}$$

est donc normalisable, en contradiction avec la définition de représentation d'une fonction. Par contre, grâce au lemme de mise en mémoire, on voit que

$$\underline{H} = \lambda a.\, ((T)\underline{F})(\underline{G})a$$

représente $H$ correctement. On va maintenant généraliser un peu cette idée de manière à coder le schéma de composition complet en utilisant $T$.

Soient $t$, $u_1$, …, $u_n$ des termes. On note $\langle t, u_1, \ldots, u_n\rangle$ ou $\langle t, \vec{u}\rangle$ le terme défini par récurrence sur $n$ :

$$\langle t, u_1\rangle = ((T)t)u_1$$
$$\langle t, u_1, \ldots, u_{n+1}\rangle = ((T)\langle t, u_1, \ldots, u_n\rangle)u_{n+1}$$

On voit immédiatement par récurrence sur $n$ que

– si l'un des $u_i$ n'a pas de forme normale de tête alors $(t)\vec{u}$ non plus ;

– si il existe $m_1, \ldots, m_n$ tels que $u_i = \underline{m_i}$ pour $i = 1, \ldots, n$ alors $\langle t, \vec{u}\rangle =_\beta (t)\underline{m}_1, \ldots, \underline{m}_n$.

D'où l'on déduit le

**Lemme 2.2.2 (Composition)** Soient $F$ une fonction partielle d'arité $n$ et $G_1, \ldots, G_k$ des fonctions partielles d'arité $k$ telles qu'il existe des termes $\underline{F}, \underline{G}_1, \ldots, \underline{G}_n$ les représentant. Alors le terme

$$\lambda x_1. \ldots \lambda x_k. \langle \underline{F}, (\underline{G}_1)\vec{a}, \ldots, (\underline{G}_n)\vec{a}\rangle$$

représente la fonction composée $F(G_1(\vec{a}), \ldots, G_n(\vec{a}))$.

*Schéma de minimisation.* On note $Y$ le terme

$$Y = (\lambda x. \lambda f. (f)((x)x)f)\lambda x. \lambda f. (f)((x)x)f$$

**Lemme 2.2.3 (Point fixe)** Pour tout terme $t$ on a

$$(Y)t\ \beta^t\ (t)(Y)t$$

Soit maintenant $\varphi$ et $U$ les termes

$$\varphi = \lambda u. \lambda n. (((f)n)\lambda d. (u)(\underline{S})n)n$$
$$U = \lambda f. ((Y)\varphi)\underline{0}$$

**Lemme 2.2.4** Soit $t$ un terme et $n$ un entier tel que pour tout $i < n$, il existe $m_i > 0$ tel que $(t)\underline{i} =_\beta \underline{m_i}$. Alors on a

– si $(t)\underline{n}$ n'a pas de forme normale de tête alors $(U)t$ non plus ;

– si $(t)\underline{n} =_\beta \underline{0}$ alors $(U)t =_\beta \underline{n}$.

*Preuve :* On note $\varphi'$ le terme $\varphi[t/f]$ et $U_n$ le terme $((Y)\varphi')(\underline{S})\ldots(\underline{S})\underline{0}$ où $\underline{S}$ occure $n$ fois. On a alors

$$\begin{aligned}
(U)t\ &\beta_0^t\ U_0 \\
&\beta^t\ ((\varphi')(Y)\varphi')\underline{0} \\
&\beta^t\ (((t)\underline{0})\lambda d. ((Y)\varphi')(\underline{S})\underline{0})\underline{0} = (((t)\underline{0})\lambda d. U_1)\underline{0} \\
&\beta\ ((\underline{m_0})\lambda d. U_1)\underline{0} \\
&\beta^t\ U_1 \\
&\vdots \\
&\beta^t\ U_n \\
&\beta\ (((t)\underline{n})\lambda d. U_{n+1})\underline{n}
\end{aligned}$$

Si $(t)\underline{n}$ n'a pas de forme normale de tête alors par le lemme 2.1.13 et le théorème 2.1.14, $(U)t$ non plus. Si $(t)\underline{n} =_\beta \underline{0}$ alors $(U)t =_\beta ((\underline{0})\lambda d. U_{n+1})\underline{n}\ \beta^t\ \underline{n}$ comme annoncé. □

Avec ce dernier lemme on a achevé de démontrer le

**Théorème 2.2.5** *Si $F$ est une fonction récursive alors il existe un terme $\underline{F}$ qui représente $F$.*

*Récursion primitive.* Puisque le lambda-calcul représente les fonctions récursives, d'après le théorème 1.2.2 il représente également les fonctions primitive récursives. On va donner une preuve directe de ce fait.

Si $u_1$ et $u_2$ sont des termes on note $\langle u_1, u_2 \rangle$ le terme

$$\langle u_1, u_2 \rangle = \lambda x.\,(x)u_1u_2$$

Si $c$ est un terme et $i = 1$ ou 2, on note $\pi_i c$ le terme

$$\pi_i c = (c)\lambda x._1 \lambda x._2 x_i$$

On vérifie immédiatement que l'on a

$$\pi_i \langle u_1, u_2 \rangle \ \beta^t \ u_i$$

On note $\varphi$ et $R$ les termes

$$\begin{aligned} \varphi &= \lambda c.\,\langle (\underline{S})\pi_1 c, ((f)\pi_1 c)\pi_2 c \rangle \\ R &= \lambda f.\,\lambda b.\,\lambda n.\,\pi_2((n)\varphi)\langle \underline{0}, b \rangle \end{aligned}$$

**Lemme 2.2.6** Pour tous termes $t$ et $u$ on a :

$$\begin{aligned} (R)tu\underline{0} &=_\beta u \\ (R)tu\,\underline{n+1} &=_\beta ((t)\underline{n})(R)tu\,\underline{n} \end{aligned}$$

pour tout entier $n$.

La preuve est laissé en exercice au lecteur. Soient maintenant $G$ une fonction d'arité $k$ et $H$ une fonction d'arité $k+2$ représentées respectivement par les termes $\underline{G}$ et $\underline{H}$. On vérifie facilement que si on définit

$$\begin{aligned} \varphi &= \lambda a.\,\lambda v.\,(\underline{H})va\vec{a} \\ \psi &= (\underline{G})\vec{a} \\ \underline{F} &= \lambda a.\,\lambda \vec{a}.\,(((R)\varphi)\psi)a \end{aligned}$$

alors le terme $\underline{F}$ représente la fonction $F$ définie par récursion primitive à partir de $G$ et $H$.

## 2.3 Le modèle de Engeler

LE MODÈLE $\mathcal{E}(\mathcal{A})$ de Engeler est un cas particulier de domaine de Scott tel qu'il existe deux fonctions :

$$\mathcal{E}(\mathcal{A}) \ \underset{\psi}{\overset{\phi}{\rightleftarrows}} \ \mathcal{E}(\mathcal{A}) \to \mathcal{E}(\mathcal{A})$$

vérifiant $\phi \circ \psi = \mathrm{Id}$. Toutefois nous allons en donner ici une présentation plus concrète que nous utiliserons ensuite pour démontrer le théorème 2.1.14 selon la méthode classique de *réductibilité*.

### FONCTIONS CONTINUES

*Notations et conventions.* Soit $X$ un ensemble. On note $\mathcal{P}(X)$ l'ensemble des parties de $X$. On utilisera $a$, $b$ et $c$ pour dénoter les éléments de $X$, $x$ et $y$ pour dénoter les parties de $X$ (les éléments de $\mathcal{P}(X)$) et $A$, $B$ pour dénoter les parties de $\mathcal{P}(X)$ (les ensembles de parties de $X$). Si $A$ est un ensemble de parties de $X$ on note $\bigcup A$ la réunion des éléments de $A$. Si $f$ est une fonction de $I \to \mathcal{P}(X)$, on note $\bigcup_{i \in I} f(i)$ la réunion des $f(i)$.

*Familles filtrantes.* Soit $A$ un ensemble d'ensembles. On note $\bigcup A$ ou $\bigcup_{x\in A} x$ la réunion des éléments de $A$. On dira que $A$ est *filtrant* si $A$ est non vide et pour tous $x_1$, $x_2$ dans $A$, il existe un $y$ tel que $x_i \subset y$ pour $i = 1, 2$. L'exemple typique de famille filtrante est l'ensemble des parties *finies* d'un ensemble $X$, que l'on notera $\mathcal{P}_{\text{fin}}(X)$. Si $x$ est un ensemble, on écrira $x_0 \subset_{\text{fin}} x$ pour « $x_0$ est une partie *finie* de $x$ ».

*Fonctions continues.* Soient $X$ et $Y$ deux ensembles. Une fonction $F : \mathcal{P}(X) \to \mathcal{P}(Y)$ est dite *continue* si elle est croissante (pour l'inclusion) et pour tout ensemble filtrant $A \subset \mathcal{P}(X)$ on a :

$$F(\bigcup A) = \bigcup_{x\in A} F(x)$$

REMARQUE : L'une des inclusions est réalisée dès que $F$ est croissante, à savoir $\bigcup_{x\in A} F(x) \subset F(\bigcup A)$.

**Lemme 2.3.1 (Finitude)** Soit $F : \mathcal{P}(X) \to \mathcal{P}(Y)$ une fonction croissante. Alors $F$ est continue ssi pour tout $b \in Y$ et tout $x \subset X$ on a la *condition de finitude* :

$$\text{si } b \in F(x) \text{ alors } \exists x_0 \subset_{\text{fin}} x \text{ tel que } b \in F(x_0)$$

Cette condition exprime une propriété fondamentale des programmes : pour pouvoir donner un résultat fini, même partiel, tel le $b$ qui appartient à $F(x)$, la fonction $F$ qui représente un programme n'a besoin que d'utiliser un nombre *fini* d'informations sur son entrée. Clairement un programme qui utiliserait $x$ en entier pour pouvoir répondre que $b$ est un élément du résultat, calculerait en temps infini dès que $x$ l'est. C'est l'un des buts de la *sémantique dénotationnelle* que de trouver des propriétés abstraites des programmes, telles que la condition de finitude, qui permettent de définir des classes de fonctions représentant les programmes.

*Preuve :* Supposons pour commencer que $F$ est continue et soit $b \in F(x)$. On considère la famille $\mathcal{P}_{\text{fin}}(x)$ qui est filtrante ; comme $x = \bigcup \mathcal{P}_{\text{fin}}(x)$ on a par continuité de $F$

$$F(x) = \bigcup_{x_0 \in \mathcal{P}_{\text{fin}}(x)} F(x_0)$$

Comme $b \in F(x)$ on voit qu'il y a un $x_0 \subset_{\text{fin}} x$ tel que $b \in F(x_0)$.

Réciproquement supposons que $F$ est croissante et vérifie la condition de finitude. Soit $A$ un ensemble filtrant inclus dans $\mathcal{P}(X)$. On sait déja, par croissance de $F$ que $\bigcup_{x\in A} F(x) \subset F(\bigcup A)$. Soit $b \in F(\bigcup A)$. Par condition de finitude, on sait qu'il existe un $x_0 \subset_{\text{fin}} \bigcup A$ tel que $b \in F(x_0)$ ; notons $a_1$, …, $a_n$ les éléments de $x_0$. Comme les $a_i$ sont des éléments de $\bigcup A$, il existe $x_1$, …, $x_n$ appartenant à $A$ tels que $a \in x_i$ pour $i = 1, \ldots, n$. Mais comme $A$ est filtrant, on voit facilement (par récurrence sur $n$) qu'il existe un $x$ appartenant à $A$ et tel que $x_i \subset x$ pour $i = 1, \ldots, n$. Donc $x_0 \subset x$ et par croissance de $F$ on obtient $b \in F(x) \subset \bigcup_{x\in A} F(x)$. □

*Trace d'une fonction continue.* Soit $F$ une fonction continue de $\mathcal{P}(X)$ dans $\mathcal{P}(Y)$. On appelle *trace de F* et on note $\operatorname{Tr} F$ l'ensemble inclus dans $\mathcal{P}_{\text{fin}}(X) \times Y$ des couples :

$$\operatorname{Tr} F = \{(x_0, b),\ x_0 \subset_{\text{fin}} X \text{ et } b \in F(x_0)\}$$

Le lemme de finitude nous assure que $\operatorname{Tr} F$ contient toute l'information nécessaire pour calculer $F(x)$. Plus précisément, si $f$ est une partie de $\mathcal{P}_{\text{fin}}(X) \times Y$, notons $\operatorname{Fun} f$ la fonction de $\mathcal{P}(X) \to \mathcal{P}(Y)$ définie par :

$$\operatorname{Fun} f(x) = \{b \in Y,\ \exists x_0 \subset_{\text{fin}} x \text{ tel que } (x_0, b) \in f\}$$

Alors on a

**Lemme 2.3.2** La fonction $\mathrm{Fun}\, f$ est continue de $\mathcal{P}(X)$ dans $\mathcal{P}(Y)$. Réciproquement, si $F$ est une fonction continue de $\mathcal{P}(X)$ dans $\mathcal{P}(Y)$ alors

$$\mathrm{Fun}\,\mathrm{Tr}\, F = F$$

Tout le propos de la théorie des *domaines* est de généraliser les définitions ci-dessus de façon à les appliquer à une classe d'ordre plus générale que les simples $\mathcal{P}(X)$ que l'on considère ici. On parvient alors à munir l'ensemble des fonctions continues d'une structure de domaine et on obtient que les fonctions Tr et Fun sont elles même continues. Mais c'est une autre histoire…

*Fonctions à plusieurs variables.* Soient $X_1, \ldots, X_n$ et $Y$ des ensembles. Une fonction $F : \mathcal{P}(X_1) \times \cdots \times \mathcal{P}(X_n) \to \mathcal{P}(Y)$ est continue si elle l'est séparément sur chacun de ses paramètres. Cette définition est cohérente avec la remarque que $\mathcal{P}(X_1) \times \cdots \times \mathcal{P}(X_n)$ est isomorphe à $\mathcal{P}(X_1 + \cdots + X_n)$ (où $X_1 + X_2$ représente l'union disjointe de $X_1$ et $X_2$).

Soit maintenant $F : \mathcal{P}(X_1) \times \cdots \times \mathcal{P}(X_{n+1}) \to \mathcal{P}(Y)$ une fonction continue. On note $\mathrm{Tr}_n\, F$ la fonction de $\mathcal{P}(X_1) \times \cdots \times \mathcal{P}(X_n) \to \mathcal{P}(\mathcal{P}_{\mathrm{fin}}(X_{n+1}) \times Y)$ définie par :

$$\mathrm{Tr}_n\, F(x_1, \ldots, x_n) = \mathrm{Tr}\, F_{x_1,\ldots,x_n}$$

où $F_{x_1,\ldots,x_n} : \mathcal{P}(X_{n+1}) \to \mathcal{P}(Y)$ est la fonction : $F_{x_1,\ldots,x_n}(x) = F(x_1, \ldots, x_n, x)$ (Exercice : montrer que $\mathrm{Tr}_n\, F$est continue).

Réciproquement si $F : \mathcal{P}(X_1) \times \cdots \times \mathcal{P}(X_n) \to \mathcal{P}(\mathcal{P}_{\mathrm{fin}}(X_{n+1}) \times Y)$ est continue, on note $\mathrm{Fun}_n\, F : \mathcal{P}(X_1) \times \cdots \times \mathcal{P}(X_{n+1}) \to \mathcal{P}(Y)$ la fonction définie par :

$$\mathrm{Fun}_n\, F(x_1, \ldots, x_{n+1}) = \mathrm{Fun}(F(x_1, \ldots, x_n))(x_{n+1})$$

(Exercice : montrer que $\mathrm{Fun}_n\, F$ est continue). Comme pour le lemme 2.3.2 on a pour $F : \mathcal{P}(X_1) \times \cdots \times \mathcal{P}(X_{n+1}) \to \mathcal{P}(Y)$

$$\mathrm{Fun}_n\, \mathrm{Tr}_n\, F = F$$

## Modèle de Engeler

*Construction du modèle.* Soit $\mathcal{A}$ un ensemble dénombrables dont les éléments seront appelés les *atomes*. On se donne une suite d'ensembles $(D_n)$ définie par récurrence par :

$$\begin{aligned} D_0 &= \mathcal{A} \\ D_{n+1} &= D_n \cup \{(x_0, a),\ x_0 \subset_{\mathrm{fin}} D_n \text{ et } a \in D_n\} \end{aligned}$$

On note $\mathcal{D}(\mathcal{A})$ la réunion de tous les $D_n$. Par construction $\mathcal{D}(\mathcal{A})$ est le plus petit ensemble $D$ tel que $\mathcal{A} \subset D$ et $\mathcal{P}_{\mathrm{fin}}(D) \times D \subset D$.

On note $\mathcal{E}(\mathcal{A})$ l'ensemble $\mathcal{P}(\mathcal{D}(\mathcal{A}))$. Si $F : \mathcal{E}(\mathcal{A}) \to \mathcal{E}(\mathcal{A})$ est continue alors comme $\mathrm{Tr}\, F \subset \mathcal{P}_{\mathrm{fin}}(\mathcal{D}(\mathcal{A})) \times \mathcal{D}(\mathcal{A})$ on a $\mathrm{Tr}\, F \in \mathcal{E}(\mathcal{A})$. Réciproquement si $f \in \mathcal{E}(\mathcal{A})$ on écrira $\mathrm{Fun}\, f$ pour $\mathrm{Fun}(f/\mathcal{A})$ qui est donc une fonction continue de $\mathcal{E}(A) \to \mathcal{E}(\mathcal{A})$. Comme $\mathrm{Fun}\,\mathrm{Tr}\, F = F$ pour toute fonction continue $F$ sur $\mathcal{D}(A)$ on voit que, en notant $\mathcal{D}(\mathcal{A}) \to \mathcal{D}(\mathcal{A})$ l'espace des fonctions continues sur $\mathcal{D}(\mathcal{A})$, on a :

$$\mathcal{E}(\mathcal{A}) \underset{\mathrm{Tr}}{\overset{\mathrm{Fun}}{\rightleftarrows}} \mathcal{E}(\mathcal{A}) \to \mathcal{E}(\mathcal{A})$$

si bien que Fun et Tr réalisent le programme annoncé dans l'introduction (p. 42).

*Interprétation des termes.* Dans toute cette partie, afin d'éviter les confusions on n'utilisera que la lettre $z$ pour dénoter les variables du lambda-calcul. Soient $\ell = (z_1, \ldots, z_n)$ une suite finie de variables et $t$ un terme dont les variables libres sont parmi les éléments de $\ell$. On va associer à $t$ une fonction continue $[\![t]\!]_\ell : \mathcal{E}(A) \times \cdots \times \mathcal{E}(\mathcal{A}) \to \mathcal{E}(\mathcal{A})$ d'arité $n$ définie par récurrence sur $t$ :

- si $t = z_i$ alors $[\![t]\!]_\ell(\vec{x}) = x_i$ ;
- si $t = (u)v$ alors $[\![t]\!]_\ell(\vec{x}) = \mathrm{Fun}_n[\![u]\!]_\ell(\vec{x}, [\![v]\!]_\ell(\vec{x}))$ ;
- si $t = \lambda z.\, u$ alors $[\![t]\!]_\ell(\vec{x}) = \mathrm{Tr}_n[\![u]\!]_{\ell.z}(\vec{x})$ où $\ell.z$ est la suite $(z_1, \ldots, z_n, z)$.

*Le modèle de Engeler vu comme système de typage.* Si on applique les définitions de Fun et Tr on a que

– si $t = (u)v$ alors $[\![t]\!]_\ell(\vec{x}) = \{a,\ \exists x_0 \subset_{\text{fin}} [\![v]\!]_\ell(\vec{x})$ tel que $(x_0, a) \in [\![u]\!]_\ell(\vec{x})\}$.

– si $t = \lambda z.\, u$ alors $[\![t]\!]_\ell(\vec{x}) = \{(x_0, a),\ x_0 \subset_{\text{fin}} \mathcal{D}(\mathcal{A})$ et $a \in [\![u]\!]_{\ell.z}(\vec{x}, x_0)\}$

Notons $z_1 : x_1, \ldots, z_n : x_n \vdash t : a$ le fait que $a \in [\![t]\!]_\ell(\vec{x})$ et notons $x_0 \to a$ le couple $(x_0, a)$. Alors les clauses ci-dessus deviennent :

– $\vec{z} : \vec{x} \vdash z_i : a_i$ si $a_i \in x_i$ ;

– $\vec{z} : \vec{x} \vdash (u)v : a$ si il existe $x_0$ fini tel que pour tout $b$ dans $x_0$ on ait $\vec{z} : \vec{x} \vdash v : b$ et tel que $\vec{z} : \vec{x} \vdash u : x_0 \to a$ ;

– $\vec{z} : \vec{x} \vdash t : x_0 \to a$ si $\vec{z} : \vec{x}, z : x_0 \vdash u : a$.

En fait cette façon de représenter le modèle de Engeler comme système de typage existe sous le nom de *types avec intersections* ou *système D* (voir [3]).

**Théorème 2.3.3 (Correction du modèle de Engeler)** *L'interprétation $[\![t]\!]_\ell$ est* dénotationnelle *c'est à dire que si t et t′ sont deux termes et $\ell$ est une suite de variables contenant toutes les variables libres de t et t′ alors*

$$\text{si } t =_\beta t' \text{ alors } [\![t]\!]_\ell = [\![t']\!]_\ell$$

*Preuve :* Il suffit de démontrer le théorème dans le cas où $t\ \beta_0\ t'$. On raisonne par récurrence sur $t$. On suppose que $\ell = (z_1, \ldots, z_n)$ contient toutes les variables libres de $t$. Les cas $t = z_i$, $t = (u)v$ et $t' = (u')v$, $t = (u)v$ et $t' = (u)v'$, $t = \lambda z.\, u$ et $t' = \lambda z.\, u'$ sont immédiats par hypothèse de récurrence. Ce qui nous amène au seul cas intéressant : $t = (\lambda z.\, u)v$ et $t' = u[v/z]$. Il résulte du lemme suivant

**Lemme 2.3.4 (Substitution)** Pour tous termes $u$ et $v$ et toute suite $\ell = (z_1, \ldots, z_n)$ contenant toutes les variables libres de $t$ et $u$ distinctes de $z$ on a

$$[\![u[v/z]]\!]_\ell(\vec{x}) = [\![u]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x}))$$

Supposons le lemme de substitution démontré. On a

$$\begin{aligned}
[\![t]\!]_\ell(\vec{x}) &= \mathrm{Fun}_n\, \mathrm{Tr}_n [\![u]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x})) && \text{par définition de l'interprétation}\\
&= [\![u]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x})) && \text{par le lemme 2.3.2}\\
&= [\![u[v/z]]\!]_\ell(\vec{x}) && \text{par le lemme de substitution}\\
&= [\![t']\!]_\ell(\vec{x})
\end{aligned}$$

On démontre le lemme de substitution par… récurrence sur $u$. Si $u = z_i$ alors $u[v/z] = u$ et le résultat est immédiat. Si $u = z$ alors $u[v/z] = v$ ; par définition $[\![u]\!]_{\ell.z}(\vec{x}, x) = x$ d'où l'on déduit le résultat en prenant $x = [\![v]\!]_\ell(\vec{x})$.

Si $u = (u_1)u_2$ alors $u[v/z] = (u_1[v/z])u_2[v/z]$ et l'on a

$$\begin{aligned}
[\![u[v/z]]\!]_\ell(\vec{x}) &= \mathrm{Fun}_n [\![u_1[v/z]]\!]_\ell(\vec{x}, [\![u_2[v/z]]\!]_\ell(\vec{x})) && \text{par définition de l'interprétation}\\
&= \mathrm{Fun}([\![u_1[v/z]]\!]_\ell(\vec{x}))([\![u_2[v/z]]\!]_\ell(\vec{x})) && \text{par définition de } \mathrm{Fun}_n\\
&= \mathrm{Fun}([\![u_1]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x})))([\![u_2]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x}))) && \text{par récurrence sur } u_1 \text{ et } u_2\\
&= \mathrm{Fun}_{n+1} [\![u_1]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x}), [\![u_2]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x}))) && \text{par définition de } \mathrm{Fun}_{n+1}\\
&= [\![u]\!]_{\ell.z}(\vec{x}, [\![v]\!]_\ell(\vec{x})) && \text{par définition de l'interprétation}
\end{aligned}$$

Si $u = \lambda z.\, u_0$ alors $u[v/z] = u$ et le résultat est immédiat. Si $u = \lambda z_{n+1}.\, u_0$ où $z_{n+1} \neq z$ et $z_{n+1}$ n'apparaît pas dans $v$ (ce que l'on peut toujours supposer à $\alpha$-conversion près) alors $u[v/z] = \lambda z_{n+1}.\, u_0[v/z]$ et le résultat vient par un raisonnement analogue au cas de l'application. □

## Une application du modèle de Engeler

On va maintenant utiliser le résultat précédent pour démontrer le théorème de réduction de tête 2.1.14. Pour ce faire on va associer à chaque élément $a$ de $\mathcal{E}(\mathcal{A})$ un ensemble $|a|$ de lambda-termes inclus dans l'ensemble des lambda-termes dont la réduction de tête termine. Puis on démontrera le *lemme d'appartenance* qui stipule que si $a \in [\![t]\!]_\ell$ alors $t \in |a|$. On conclut en montrant que si $t$ est en forme normale de tête alors $[\![t]\!]_\ell$ est non vide. Donc si $t =_\beta t_0$ et $t_0$ est en forme normale de tête alors par le théorème de correction du modèle de Engeler on a que $[\![t]\!]_\ell$ est non vide donc par le lemme d'appartenance que sa réduction de tête termine.

*Réalisabilité.* Soit $X$ et $Y$ deux ensembles de lambda-termes. On note $X \to Y$ l'ensemble

$$X \to Y = \{u \text{ tel que } \forall v \in X,\ (u)v \in Y\}$$

Remarquons que dans le cas particulier où $X$ est vide, alors $X \to Y = \Lambda$, l'ensemble de tous les lambda-termes (à $\alpha$-conversion près).

*Saturation.* On dit qu'une partie $X$ de $\Lambda$ est *saturée* si pour tous $u, v, v_1, \ldots, v_n$ on a

$$\text{si } (u[v/z])\vec{v} \in X \text{ alors } (\lambda z.\, u)v\vec{v} \in X$$

**Lemme 2.3.5** Soient $X$ et $Y$ deux parties de $\Lambda$. Si $Y$ est saturée alors $X \to Y$ est saturée. Si de plus $X$ est saturée alors $X \cap Y$ est saturée.

*Preuve :* Supposons que $(u[v/z])\vec{v}$ est un élément de $X \to Y$ et soit $w \in X$. Par définition de $X \to Y$ on a $(u[v/z])\vec{v}w \in Y$ et comme $Y$ est saturée, $(\lambda z.\, u)v\vec{v}w \in Y$. Comme $w$ est quelconque on vient de montrer que $(\lambda z.\, u)v\vec{v}$ appartient à $X \to Y$.

Le cas de l'intersection est immédiat. □

*Termes normalisables de têtes.* On note $\mathcal{N}$ l'ensemble des termes dont la réduction de tête termine et $\mathcal{N}_0$ l'ensemble des termes de la forme $(z)\vec{v}$ où $z$ est une variable et $v_1, \ldots, v_n$ des termes quelconques. En particulier $\mathcal{N}_0$ contient toutes les variables.

**Lemme 2.3.6** Soient $X$ et $Y$ deux parties de $\Lambda$.

- si $\mathcal{N}_0 \subset Y$ alors $\mathcal{N}_0 \subset X \to Y$ ;
- si $\mathcal{N}_0 \subset X$ et $Y \subset \mathcal{N}$ alors $X \to Y \subset \mathcal{N}$ :
- si $X$ et $Y$ contiennent $\mathcal{N}_0$ et/ou sont contenues dans $\mathcal{N}$ alors il en de même de $X \cap Y$.

En particulier, et c'est sous cette forme que l'on utilisera le lemme, si $X$ et $Y$ contiennent tous deux $\mathcal{N}_0$ et sont tous deux contenus dans $\mathcal{N}$ alors il en de même de $X \cap Y$ et $X \to Y$.

*Preuve :* Le cas de l'intersection est immédiat. Supposons que $\mathcal{N}_0 \subset Y$ et soit $u = (z)\vec{v}$ une élément de $\mathcal{N}_0$ et $v$ un élément de $X$. Alors $(u)v = (z)\vec{v}v \in \mathcal{N}_0$ dont $(u)v \in Y$. Comme $v$ est un élément quelconque de $X$ on vient de montrer que $u \in X \to Y$, donc que $\mathcal{N}_0 \subset X \to Y$.

Supposons maintenant que $\mathcal{N}_0 \subset X$ et $Y \subset \mathcal{N}$ et soit $u \in X \to Y$. Prenons un $v$ dans $X$ ; alors $(u)v \in Y$ et comme $Y \subset \mathcal{N}$ la réduction de tête de $(u)v$ termine. Par le lemme 2.1.13 on a que la réduction de tête de $u$ termine, c'est à dire que $u \in \mathcal{N}$. □

*Interprétation par réalisabilité.* On va maintenant associer à chaque point de $\mathcal{D}(\mathcal{A})$ une partie saturée contenant $\mathcal{N}_0$ et contenue dans $\mathcal{N}$. L'intuition est que les points de $\mathcal{D}(\mathcal{A})$ sont des *types* (un point de la forme $(x_0, a)$ représente le type $x_0 \to a$), c'est à dire des spécifications formelles de programmes, et que l'on associe à un point l'ensemble des termes qui *réalisent* ce type, c'est à dire qui respectent la spécification.

On appelle *valuation* une fonction $\rho : \mathcal{A} \to \mathcal{P}(\Lambda)$ qui associe à tout atome $\alpha$ un ensemble $\rho(\alpha)$ saturé, contenant $\mathcal{N}_0$ et contenu dans $\mathcal{N}$. Étant donnée une valuation $\rho$, on définit l'ensemble $|a|_\rho$ des *réalisateurs* d'un élément $a$ de $\mathcal{D}(\mathcal{A})$ par récurrence sur $a$ (ou plus précisément par récurrence sur le plus petit $n$ tel que $a \in D_n$) :

– si $a \in \mathcal{A}$ alors $|a|_\rho = \rho(a)$ ;
– si $a = (x_0, b)$ alors
$$|a|_\rho = |x_0|_\rho \to |b|_\rho$$
où $|x_0|_\rho$ est $\bigcap_{c \in x_0} |c|_\rho$. En particulier si $x_0$ est vide alors $|x_0|_\rho$ est $\Lambda$.

Grâce aux lemmes 2.3.5 et 2.3.6 on a bien que pour tout $a \in \mathcal{D}(\mathcal{A})$ et toute valuation $\rho$, $|a|_\rho$ est une partie de $\Lambda$ saturée contenant $\mathcal{N}_0$ et contenue dans $\mathcal{N}$.

*Le lemme d'appartenance.* C'est le lemme fondamental de toute interprétation par réalisabilité. On le retrouve dans la preuve de forte normalisation du système $F$ mais aussi dans d'autres contexte comme la définition du collapse extensionnel d'un modèle dénotationnel...

**Lemme 2.3.7 (Appartenance)** Soient $t$ un terme dont toutes les variables libres appartiennent à la liste $\ell = (z_1, \ldots, z_n)$, $x_1$, ..., $x_n$ des éléments de $\mathcal{E}(\mathcal{A})$ (des parties de $\mathcal{D}(\mathcal{A})$) et $\rho$ une valuation. Alors pour tout $a \in \mathcal{D}(\mathcal{A})$ et tous termes $u_1$, ..., $u_n$ on a

$$\text{si } \begin{cases} a \in [\![t]\!]_\ell(\vec{x}) \text{ et} \\ u_i \in |x_i|_\rho \text{ pour } i = 1, \ldots, n \end{cases} \qquad \text{alors } t[\vec{u}/\vec{z}] \in |a|_\rho$$

où $|x_i|_\rho$ est $\bigcap_{b \in x_i} |b|_\rho$.

*Preuve :* Le lemme d'appartenance se démontre par récurrence sur $t$. Si $t = z_i$ alors $[\![t]\!]_\ell(\vec{x}) = x_i$, donc $a \in x_i$ et $|x_i|_\rho \subset |a|_\rho$. Comme $u_i \in |x_i|_\rho$ on a $u_i \in |a|_\rho$. Mais $u_i = t[\vec{u}/\vec{z}]$ puisque $t = z_i$.

Si $t = (u)v$ alors $[\![t]\!]_\ell(\vec{x}) = \mathrm{Fun}_n[\![u]\!]_\ell(\vec{x}, [\![v]\!]_\ell(\vec{x}))$, et $a \in [\![t]\!]_\ell(\vec{x})$ entraîne qu'il existe un $x_0 \subset_{\text{fin}} [\![v]\!]_\ell(\vec{x})$ tel que $(x_0, a) \in [\![u]\!]_\ell(\vec{x})$. Par hypothèse de récurrence on a donc $u[\vec{u}/\vec{z}] \in |(x_0, a)|_\rho = |x_0|_\rho \to |a|_\rho$. Mais, comme $x_0 \subset [\![v]\!]_\ell(\vec{x})$, par récurrence sur $v$ on a $v[\vec{u}/\vec{z}] \in |x_0|_\rho = \bigcap_{b \in x_0} |b|_\rho$. Par définition de $\to$ on a donc que $(u[\vec{u}/\vec{z}])v[\vec{u}/\vec{z}] = t[\vec{u}/\vec{z}] \in |a|_\rho$.

Si $t = \lambda z.\, v$ alors $[\![t]\!]_\ell(\vec{x}) = \mathrm{Tr}_n[\![v]\!]_{\ell.z}(\vec{x})$ et $a \in [\![t]\!]_\ell(\vec{x})$ entraîne que $a$ est de la forme $(x_0, b)$ où $b \in [\![v]\!]_{\ell.z}(\vec{x}, x_0)$. Par hypothèse de récurrence, pour tout $u \in |x_0|_\rho$ on a $v[\vec{u}/\vec{z}, u/z] \in |b|_\rho$ ; mais par saturation de $|b|_\rho$ ceci entraîne que $(\lambda z.\, v[\vec{u}/\vec{z}])u \in |b|_\rho$ pour tout $u \in |x_0|_\rho$. Donc $\lambda z.\, v[\vec{u}/\vec{z}] = t[\vec{u}/\vec{z}] \in |x_0|_\rho \to |b|_\rho = |a|_\rho$. □

**Corollaire 2.3.8** *Soient $x_1$, ..., $x_n$ dans $\mathcal{E}(\mathcal{A})$. Si $[\![t]\!]_\ell(\vec{x})$ est non vide alors la réduction de tête de $t$ termine.*

*Preuve :* Comme $\mathcal{N}_0$ contient toutes les variables et est contenu dans tous les $|x_i|_\rho$, on a $z_i \in |x_i|_\rho$ pour $i = 1, \ldots, n$, donc par le lemme d'appartenance $t[\vec{z}/\vec{z}] = t \in |a|_\rho$. Mais $|a|_\rho \subset \mathcal{N}$, donc par définition de $\mathcal{N}$ la réduction de tête de $t$ termine. □

On termine la preuve du théorème 2.1.14 en montrant le lemme suivant :

**Lemme 2.3.9** Si $t$ est beta-équivalent à une forme nomale de tête $t_0$ alors pour toute liste $\ell = (z_1, \ldots, z_n)$ contenant les variables libres de $t$ et de $t_0$, il existe $x_1$, ..., $x_n$ dans $\mathcal{E}(\mathcal{A})$ tels que $[\![t]\!]_\ell(\vec{x})$ est non vide.

Le théorème se déduit alors du corollaire au lemme d'appartenance.

*Preuve :* En vertu du théorème de correction du modèle de Engeler, l'interprétation de $t$ est égale à celle de la forme normale de tête $t_0$. Donc il suffit de montrer le lemme dans le cas où $t$ est en forme normale de tête.

Soit $a \in \mathcal{D}(\mathcal{A})$. On définit $a^m$ par récurrence sur $m$ : $a^0 = a$ et $a^{m+1} = (\emptyset, a^m)$. Autrement dit, en utilisant l'analogie avec les types :

$$a^m = \underbrace{\emptyset \to \cdots \to \emptyset}_{m\,\times} \to a$$

Si $t$ est en forme normale de tête alors on a soit $t = (z_i)u_1 \dots u_m$, soit $t = \lambda z.\, u$ où $u$ est en forme normale de tête. Dans le premier cas, par définition de l'interprétation on a :

$$[\![t]\!]_\ell(\vec{x}) = \{a,\ \exists y_1 \subset_{\text{fin}} [\![u_1]\!](\vec{x}), \dots, \exists y_m \subset_{\text{fin}} [\![u_m]\!](\vec{x}),\ (y_m, \dots, (y_1, a) \dots) \in x_i\}$$

Si on choisit $x_i$ tel que $a^m \in x_i$, comme $y_j = \emptyset \subset_{\text{fin}} [\![u_j]\!](\vec{x})$ pour tout $\vec{x}$ et tout $j$ on a $a \in [\![t]\!]_\ell(\vec{x})$ et donc $[\![t]\!]_\ell(\vec{x})$ est non vide.

Si maintenant $t = \lambda z.\, u$ alors

$$[\![t]\!]_\ell(\vec{x}) = \{(x_0, a) \in \mathcal{P}_{\text{fin}}(\mathcal{D}(\mathcal{A})) \times \mathcal{D}(\mathcal{A}),\ a \in [\![u]\!]_{\ell.z}(\vec{x}, x_0)\}$$

Par récurrence sur $u$ il existe $\vec{x}$ et $x_0$ tel que $[\![u]\!]_{\ell.z}(\vec{x}, x_0)$ est non vide d'où l'on déduit immédiatement que $[\![t]\!]_\ell(\vec{x})$ est non vide. $\square$

# Bibliographie


[1] René Cori and Daniel Lascar. *Logique mathématique*. Axiomes. Masson, 1993.

[2] Jean-Yves Girard. *Proof Theory and Logical Complexity*. Bibliopolis, edizioni di filosofia e scienze, Napoli (Italy), 1987.

[3] Jean-Louis Krivine. *Lambda-Calcul : Types et Modèles*. Études et Recherches en Informatique. Masson, 1990.

[4] Joseph R. Shoenfield. *Mathematical Logic*. Addison Wesley, 1967.