Aspects logiques

Jean Goubault-Larrecq

Le 31 août 1999

Dans la partie précédente, il nous est arrivé de mentionner des équations aux domaines, et de demander que certains objets appartiennent à tel ou tel domaine. En général, on peut avoir envie de considérer un langage comme le λ-calcul, ou plus compliqué, mais avec une *discipline de types* qui assure que les objets manipulés sont bien dans les bons domaines de valeurs. Une discipline de types *statique* sera donnée par un certain nombre de règles de typage, qu'un algorithme de vérification de types pourra appliquer pour s'assurer que les programmes donnés sont bien typés.

L'intérêt en programmation d'une telle discipline est qu'aucune erreur de types à l'exécution ne pourra avoir lieu: c'est notamment la philosophie du typage à la ML [HMT90]. En ML, par exemple, si f est une fonction de type `int` $\rightarrow$ `string`, alors dans tout programme bien typé, toute application fu sera nécessairement telle que u s'évalue en un entier, et le corps de la fonction f n'aura jamais à vérifier que son argument est bien un entier, car cette propriété sera garantie par typage. De plus, la valeur de retour de fu sera toujours une chaîne de caractères (le type `string`).

Le système de typage de ML est particulièrement élégant et pratique. D'autres langages proposent d'autres systèmes de types, souvent moins élégants: Pascal propose un système de types presque aussi strict que celui de ML, tout en étant moins souple; C propose un système de types laxiste (les *casts* permettent toujours de faire croire n'importe quoi au typeur); et des langages comme Scheme ou Lisp en général utilisent des mécanismes de vérification des types à l'exécution, mais pas à la compilation.

Nous verrons quelques systèmes de types à la ML, et les étudierons sous un angle logique. Il existe en effet une correspondance remarquable entre systèmes de types à la ML et systèmes de preuve, appelée correspondance de Curry-Howard, dans laquelle les notions de programmation suivantes (colonne de gauche) sont identifiées avec les notions de théorie de la démonstration correspondantes (colonne de droite):

Discipline de types	$\sim$	Logique
Type	$\sim$	Formule
Programme, terme	$\sim$	Preuve
Réduction, calcul	$\sim$	Élimination des coupures

Nous la découvrirons et en développerons les conséquences dans plusieurs disciplines de types dans la suite. Nous commencerons par la plus simple des disciplines de types utiles, celle des *types simples*, en section 1, et sa relation avec la logique minimale intuitionniste. Le passage à la logique classique propositionnelle sera examiné en section 2, où nous verrons réapparaître les opérateurs de capture de continuation. Nous explorerons des systèmes logiques de plus en plus expressifs par la suite, à commencer par l'arithmétique de Peano-Heyting du premier ordre $\mathbf{HA}_1$ en section 3, pour aller vers la logique et l'arithmétique d'ordre deux et le système F de Girard-Reynolds en section 4. Nous présenterons ensuite une application non triviale de ces techniques en section 5, en l'occurrence le théorème de Kreisel-Friedman.

1 Types simples et logique minimale

1.1 Types simples

Essayons quelques idées simples pour attribuer des types à des λ-termes. Nous revenons ici au λ-calcul pur, avec variables, applications et abstractions uniquement.

Nous nous donnons une algèbre de *types simples*, qui sont des expressions F obéissant à la grammaire:

$$F \hat{=} b \mid F \Rightarrow F$$

où b parcourt un ensemble de *types de base*, typiquement `nat`, `int`, `string`, `bool`, etc. Intuitivement, la sémantique d'un type est un *ensemble de valeurs*, par exemple `nat` pourra représenter l'ensemble $\mathbb{N}$ des entiers naturels. Intuitivement encore, un type de la forme $F_1 \Rightarrow F_2$ dénote l'ensemble de toutes les fonctions de F_1 vers F_2.

Syntaxiquement, nous utiliserons des parenthèses pour désambigüer les expressions de types. Nous noterons $F_1 \Rightarrow F_2 \Rightarrow F_3$ pour $F_1 \Rightarrow (F_2 \Rightarrow F_3)$, le type des fonctions à deux arguments de types respectifs F_1 et F_2, retournant un résultat de type F_3.

Dans une application uv, il est alors naturel de supposer que u a un type de la forme $F_1 \Rightarrow F_2$: l'expression uv n'est légale que si u est une fonction, ici du type F_1 vers F_2. Nous demandons alors que v soit aussi de type F_2, alors uv sera naturellement de type F_2. Ceci mène à une règle provisoire de la forme:

$$\frac{u : F_1 \Rightarrow F_2 \qquad v : F_1}{uv : F_2}$$

Pour typer les variables, c'est plus délicat, et il y a principalement deux solutions. La première est de supposer que toute variable x a un type uniquement déterminé précisé à l'avance; si F est ce type, on notera x_F au lieu de x pour le préciser clairement. Cette solution est celle des *λ-calculs typés*, et on aura la règle suivante pour typer les variables:

$$\frac{}{x_F : F}$$

tandis que pour les abstractions, on aura:

$$\frac{u : F_2}{\lambda x_{F_1} \cdot u : F_1 \Rightarrow F_2}$$

Cette première solution a l'avantage que si un terme est typable, alors il a un type unique, mais présente notamment l'inconvénient d'être incompatible avec l'α-renommage tel qu'il a été présenté jusqu'ici: remplacer une variable liée x_F par une $y_{F'}$ ne doit être autorisé que si $F' = F$. La deuxième solution est d'utiliser des *jugements de typages*, ou *séquents*, de la forme $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$. La partie gauche $\Gamma \hat{=} x_1 : F_1, \ldots, x_n : F_n$ est appelée le *contexte* de typage, et enregistre les hypothèses de typage sur les variables libres de u (et éventuellement d'autres variables). Rendre explicite ce contexte aura d'autres avantages, comme on le verra par la suite. Formellement:

Définition 1 *Un* contexte *de typage Γ est un ensemble fini de* liaisons *$x : F$, où les variables x sont deux à deux distinctes. Le* domaine $\operatorname{dom}\Gamma$ *de Γ est l'ensemble des x, lorsque $x : F$ parcourt Γ. On note Γ, Δ l'union de Γ et de Δ lorsqu'ils sont de domaines disjoints. On note $x : F$ le contexte ne contenant que la liaison $x : F$, en particulier $\Gamma, x : F$ est l'union de Γ et de la liaison $x : F$.*

Les règles de typage simple *sont:*

$$\frac{}{\Gamma, x : F \vdash x : F}\ (Var)$$

$$\frac{\Gamma \vdash u : F_1 \Rightarrow F_2 \quad \Gamma \vdash v : F_1}{\Gamma \vdash uv : F_2}\ (App) \qquad \frac{\Gamma, y : F_1 \vdash u[x := y] : F_2 \quad (y \notin \operatorname{dom}\Gamma)}{\Gamma \vdash \lambda x \cdot u : F_1 \Rightarrow F_2}\ (Abs)$$

Un typage *de u est un jugement dérivable de la forme $\Gamma \vdash u : F$. S'il existe un typage de u, on dira que u est* typable, *et* de type F dans le contexte Γ.

La règle (Var) se lit: x est de type F dans tout contexte contenant la liaison $x : F$. En effet, un tel contexte est exactement un contexte de la forme $\Gamma, x : F$ pour un certain Γ. Dans la règle (*Abs*), l'introduction de y dans la prémisse permet d'inclure le α-renommage dans les règles de typage. Nous ferons cependant toujours l'hypothèse (abusive) que les termes sont vus à α-renommage près, alors (*Abs*) se simplifie en:

$$\frac{\Gamma, x : F_1 \vdash u : F_2}{\Gamma \vdash \lambda x \cdot u : F_1 \Rightarrow F_2} \, (Abs)$$

en supposant que x peut toujours être renommée implicitement de sorte à ne pas être dans le domaine de Γ.

Exercice 1 *Quels sont les typages de $\lambda x \cdot x$? En déduire qu'un terme typable n'a pas en général de typage unique.*

Exercice 2 *Montrer que le terme xx n'est pas typable. (Observer que l'égalité des types est* textuelle, *en particulier $F_1 \neq F_1 \Rightarrow F_2$ pour tous F_1, F_2.) En déduire que $\lambda x \cdot xx$, Ω, Y, Θ, ne sont pas typables.*

Exercice 3 *Montrer que si $\Gamma \vdash u : F$ est un typage de u, alors F est déterminé de façon unique par Γ et u. Autrement dit, dans un contexte donné, le type est unique (s'il existe).*

Exercice 4 *Montrer que tous les entiers de Church sont de type $(F \Rightarrow F) \Rightarrow (F \Rightarrow F)$, pour tout type F et dans n'importe quel contexte Γ. Soit $\mathtt{nat}_F \hat{=} (F \Rightarrow F) \Rightarrow (F \Rightarrow F)$. Vérifier que $\Gamma \vdash S : \mathtt{nat}_F \Rightarrow \mathtt{nat}_F$, $\Gamma \vdash \lceil + \rceil : \mathtt{nat}_F \Rightarrow \mathtt{nat}_F \Rightarrow \mathtt{nat}_F$, $\Gamma \vdash \lceil \times \rceil : \mathtt{nat}_F \Rightarrow \mathtt{nat}_F \Rightarrow \mathtt{nat}_F$. Montrer par contre qu'on n'a $\Gamma \vdash \lceil exp \rceil : \mathtt{nat}_F \Rightarrow \mathtt{nat}_F \Rightarrow \mathtt{nat}_F$ pour aucun type F, mais que l'on a $\Gamma \vdash \lceil exp \rceil : \mathtt{nat}_{F \Rightarrow F} \Rightarrow \mathtt{nat}_F \Rightarrow \mathtt{nat}_F$.*

Exercice 5 *Posons $F_1 \times_F F_2 \hat{=} (F_1 \Rightarrow F_2 \Rightarrow F) \Rightarrow F$. Montrer que les règles suivantes sont admissibles (i.e., que toute instance de ces règles est déductible des règles de la définition 1):*

$$\frac{\Gamma \vdash u : F_1 \quad \Gamma \vdash v : F_2}{\Gamma \vdash \langle u, v \rangle : F_1 \times_F F_2} \qquad \frac{}{\Gamma \vdash \pi_1 : F_1 \times_{F_1} F_2 \Rightarrow F_1} \qquad \frac{}{\Gamma \vdash \pi_2 : F_1 \times_{F_2} F_2 \Rightarrow F_1}$$

Montrer par contre que, de façon surprenante, les règles suivantes ne sont pas *admissibles en général:*

$$\frac{}{\Gamma \vdash \pi_1 : F_1 \times_F F_2 \Rightarrow F_1} \qquad \frac{}{\Gamma \vdash \pi_2 : F_1 \times_F F_2 \Rightarrow F_1}$$

Exercice 6 *Par analogie avec l'exercice 4, comment définiriez-vous $\mathtt{bool}_F$?*

Exercice 7 *En s'aidant des exercices précédents, montrer que P est de type $\mathtt{nat}_F \Rightarrow \mathtt{nat}_F$ dans tout contexte.*

Exercice 8 **( )** *Montrer que le terme de Maurey $G_{\mathbf{V},\mathbf{F}}$ est typable, mais que bien que l'on ait:*

$$G_{\mathbf{V},\mathbf{F}} \lceil m \rceil \lceil n \rceil \to^* \begin{cases} \mathbf{V} & \textit{si } m \leq n \\ \mathbf{F} & \textit{sinon} \end{cases}$$

$G_{\mathbf{V},\mathbf{F}}$ n'a le type $\mathtt{nat}_{F_1} \Rightarrow \mathtt{nat}_{F_2} \Rightarrow \mathtt{bool}_{F_3}$, pour aucun types F_1, F_2, F_3, et dans aucun contexte.

Une propriété importante que les λ-termes possèdent est la suivante, qui exprime que les programmes ne changent pas de type pendant l'exécution. Cela peut paraître trivial, mais il faut le vérifier.

Lemme 1 (Auto-réduction) *Si $\Gamma \vdash u : F$ est dérivable et $u \to^*_{\beta\eta} v$, alors $\Gamma \vdash v : F$ est dérivable aussi.*

Preuve : Par récurrence d'abord sur la longueur de la réduction de u vers v, ensuite sur la profondeur du rédex contracté dans u. Les cas de récurrence sont triviaux, traitons donc des cas de base, où u est lui-même le rédex contracté.

(β) u est de la forme $(\lambda x \cdot s)t$ et $v = s[x := t]$. Le typage de u se termine donc forcément par:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_1\\ \Gamma, x : F_1 \vdash s : F\end{array}}{\Gamma \vdash \lambda x \cdot s : F_1 \Rightarrow F}\,(Abs) \qquad \begin{array}{c}\vdots\,\pi_2\\ \Gamma \vdash t : F_1\end{array}}{\Gamma \vdash (\lambda x \cdot s)t : F}\,(App)$$

Il ne reste plus qu'à montrer que $\Gamma \vdash s[x := t] : F$ est dérivable aussi. Nous montrons plus généralement que si $\Gamma, x : F_1, \Delta \vdash s : F$ a une dérivation π_1, alors $\Gamma, \Delta \vdash s[x := t] : F$ est dérivable aussi, par récurrence structurelle sur π_1.

– Si π_1 se termine par une règle (Var), alors soit $s \neq x$, donc s est une variable y telle que $y : F$ est dans Γ, Δ, $s[x := t] = y$, et nous dérivons $\Gamma, \Delta \vdash s[x := t] : F$ par la règle (Var); soit $s = x$. Dans ce dernier cas, $F = F_1$, et nous devons dériver $\Gamma, \Delta \vdash t : F$, sachant que nous avons déjà une preuve π_2 de $\Gamma \vdash t : F_1$. C'est une conséquence du lemme suivant, qui dit essentiellement qu'on peut toujours ajouter des hypothèses de typage inutiles:

  Lemme 2 (Affaiblissement) *Si $\Gamma \vdash t : F$ est dérivable, alors $\Gamma, \Delta \vdash t : F$ aussi.*

  Preuve : Récurrence structurelle facile sur la dérivation de typage donnée. Intuitivement, on ajoute Δ à gauche de tous les jugements dans la dérivation. $\diamondsuit$

– Si π_1 se termine par une règle (App), alors s est de la forme $s_1 s_2$ et π_1 se termine par:

$$\dfrac{\begin{array}{c}\vdots\,\pi'_1\\ \Gamma, x : F_1, \Delta \vdash s_1 : F'_1 \Rightarrow F\end{array} \quad \begin{array}{c}\vdots\,\pi'_2\\ \Gamma, x : F_1, \Delta \vdash s_2 : F'_1\end{array}}{\Gamma, x : F_1, \Delta \vdash s_1 s_2 : F}\,(App)$$

  Par récurrence, $\Gamma, \Delta \vdash s_1[x := t] : F'_1 \Rightarrow F$ et $\Gamma, \Delta \vdash s_2[x := t] : F'_1$ sont dérivables, donc aussi $\Gamma, \Delta \vdash s[x := t] : F$ par (App), puisque $s[x := t] = (s_1[x := t])(s_2[x := t])$.

– Si π_1 se termine par une règle (Abs), alors s est de la forme $\lambda y \cdot s_1$, F est de la forme $F'_1 \Rightarrow F'_2$ et π_1 se termine par:

$$\dfrac{\begin{array}{c}\vdots\,\pi'_1\\ \Gamma, x : F_1, \Delta, y : F'_1 \vdash s_1 : F'_2\end{array}}{\Gamma, x : F_1, \Delta \vdash \lambda y \cdot s_1 : F'_1 \Rightarrow F'_2}\,(Abs)$$

  Par récurrence (avec Δ remplacé par $\Delta, y : F'_1$), on peut dériver $\Gamma, \Delta, y : F'_1 \vdash s_1[x := t] : F'_2$, donc $\Gamma, \Delta \vdash \lambda y \cdot (s_1[x := t]) : F'_1 \Rightarrow F'_2$ par (Abs). Mais $\lambda y \cdot (s_1[x := t]) = (\lambda y \cdot s_1)[x := t] = s[x := t]$ car y n'est pas libre dans t et $x \neq y$. En effet, y n'est pas libre dans t car toutes les variables libres de t sont dans $\operatorname{dom}\Gamma$ (ceci vient du lemme suivant), et y n'est pas dans $\operatorname{dom}\Gamma$ par construction.

  Lemme 3 *Si $\Gamma \vdash t : F$ est dérivable, alors $\mathrm{fv}(t) \subseteq \operatorname{dom}\Gamma$.*

  Preuve : Récurrence structurelle triviale sur la dérivation donnée. $\diamondsuit$

(η) u est de la forme $\lambda x \cdot vx$ avec x non libre dans v, F est de la forme $F_1 \Rightarrow F_2$, et le typage de u doit se terminer par:

$$\dfrac{\dfrac{\overset{\vdots\,\pi_1}{\Gamma, x : F_1 \vdash v : F_1 \Rightarrow F_2} \quad \dfrac{}{\Gamma, x : F_1 \vdash x : F_1}\,(Var)}{\Gamma, x : F_1 \vdash vx : F_2}\,(App)}{\Gamma \vdash \lambda x \cdot vx : F_1 \Rightarrow F_2}\,(Abs)$$

Mais la sous-dérivation π_1 conclut déjà $\Gamma, x : F_1 \vdash v : F_1 \Rightarrow F_2$. On conclut par le lemme:

Lemme 4 (Amincissement) *Si $\Gamma, x : F_1 \vdash v : F$ est dérivable et x n'est pas libre dans v, alors $\Gamma \vdash v : F$ est dérivable.*

Preuve : Récurrence structurelle triviale sur la dérivation donnée. ♢

♢

Le lemme 1 dit que si u se réduit en v, v a tous les types que u possède, mais v peut en acquérir de nouveaux:

Exercice 9 *Soit $u \hat{=} \lambda x \cdot yx$, $v \hat{=} y$ (donc $u \to_\eta v$), où y est une variable différente de x. Montrer que v a des typages que u n'a pas. De même pour $u \hat{=} (\lambda x_1, x_2 \cdot x_2)(\lambda x \cdot yx)y$, $v \hat{=} y$ (donc $u \to_\beta v$).*

Exercice 10 *Montrer que les règles suivantes sont acceptables pour ce qui est du typage, au sens où elles préservent la propriété d'auto-réduction, et préciser les conditions sur les variables libres dans w et v qu'il faut vérifier pour cela:*

$$\begin{array}{lll} (\theta) & (\lambda x \cdot u)vw & \to \quad (\lambda x \cdot uw)v \\ (\gamma) & (\lambda x \cdot \lambda y \cdot u)v & \to \lambda y \cdot (\lambda x \cdot u)v \end{array}$$

*Il s'agit de réductions supplémentaires pour le λ-calcul, qui défissent la relation de σ-*équivalence *[Rég94].*

Le λ-calcul simplement typé est alors confluent:

Corollaire 1 (Church-Rosser) *Pour tous λ-termes simplement typables u, u_1 et u_2 tels que $u \to^* u_1$ et $u \to^* u_2$, il existe il un λ-terme simplement typable v tel que $u_1 \to^* v$ et $u_2 \to^* v$.*

Preuve : Il existe un λ-terme v tel que $u_1 \to^* v$ et $u_2 \to^* v$ parce que le λ-calcul (non typé) est confluent. Ce terme v est alors lui-même typable, car tout typage de u en est un de v, par le lemme 1. ♢

La propriété la plus intéressante du λ-calcul simplement typé, et qui le distingue du λ-calcul non typé est le théorème suivant:

Théorème 1 *Tout λ-terme simplement typable est fortement normalisant pour les relations $\to_{\beta\eta}$ et $\to_\beta$.*

Preuve : Il suffit de le démontrer pour $\to_{\beta\eta}$. L'argument le plus simple est dû à W. W. Tait, cf. [GLT89], que l'on peut présenter comme une version corrigée d'une tentative de preuve qui ne fonctionne pas. (Cette façon de présenter est due à Jean Gallier.)

Soit SN l'ensemble de tous les λ-termes fortement normalisants (pour "Strongly Normalizing"). On voudrait démontrer que si $\Gamma \vdash u : F$ est dérivable, alors $u \in SN$. Essayons de le démontrer par récurrence structurelle sur u. Si u est une variable, c'est évident car u est normal. Si u est une abstraction $\lambda x \cdot u_1$, alors les seules réductions partant de u sont dans u_1 (ces réductions sont finies, par récurrence) ou bien de la forme $u \to_\eta v \to^* \ldots$, avec $u_1 = vx$, x non libre dans v, auquel cas l'hypothèse de récurrence s'applique encore et les réductions obtenues sont encore finies. Toutes les réductions partant de u sont finies, donc

$u \in SN$. Malheureusement, dans le dernier cas où u est une application $u_1 u_2$, on ne peut pas conclure, et cette démonstration échoue.

Dans cette tentative de démonstration, il y a un ingrédient que nous n'avons pas utilisé: le fait que u était typable. Nous allons donc introduire une notion, dite de *réductibilité* mais n'ayant que peu de choses à voir avec l'existence de réductions, faisant intervenir les types et qui impliquera la normalisation forte.

Disons que u est *réductible de type* F, en abrégé $u \in RED_F$ si et seulement si:

- F est un type de base b et $u \in SN$;
- ou F est de la forme $F_1 \Rightarrow F_2$, et pour tout $v \in RED_{F_1}$, $uv \in RED_{F_2}$.

Il s'agit d'une définition licite, par récurrence structurelle sur le type F.

Montrons maintenant que tout terme réductible est fortement normalisant. C'est la propriété (CR1) ci-dessous, les autres sont des propriétés que nous devons montrer par récurrence simultanée sur les types F. Un terme est dit *neutre* s'il n'est pas une abstraction.

(CR1) Si $u \in RED_F$, alors $u \in SN$.

(CR2) Si $u \in RED_F$ et $u \rightarrow_{\beta\eta} u'$, alors $u' \in RED_F$.

(CR3) Si u est neutre et pour tout u' tel que $u \rightarrow_{\beta\eta} u'$, $u' \in RED_F$, alors $u \in RED_F$.

Démontrons donc ces propriétés par récurrence simultanée sur F. Si F est un type de base, (CR1) est vérifiée par définition; (CR2) est évidente, car toute sous-suite d'une réduction finie est finie; pour (CR3), supposons que pout tout u' tel que $u \rightarrow_{\beta\eta} u'$, $u' \in SN$: toute réduction partant de u passe après une étape par un tel u' et est donc finie, donc $u \in SN = RED_F$.

Considérons maintenant le cas inductif, où F est de la forme $F_1 \Rightarrow F_2$:

- (CR1): soit $u \in RED_F$. Par définition, pour tout $v \in RED_{F_1}$, $uv \in RED_{F_2}$. Posons $v \hat{=} x$, une variable. Par hypothèse de récurrence sur F_1, cas (CR3), $x \in RED_{F_1}$; en effet, x est neutre et ne se réduit en aucun terme. Donc $ux \in RED_{F_2}$. Par hypothèse de récurrence sur F_2, cas (CR1), $ux \in SN$. Mais toute réduction partant de u induit une réduction correspondante dans ux, et est donc finie. Ainsi $u \in SN$.
- (CR2): soit $u \in RED_F$, $u \rightarrow_{\beta\eta} u'$. Fixons un $v \in RED_{F_1}$ arbitraire. Par définition, $uv \in RED_{F_2}$. Mais $uv \rightarrow_{\beta\eta} u'v$, donc par hypothèse de récurrence sur F_2, cas (CR2), $u'v \in RED_{F_2}$. Comme $v \in RED_{F_1}$ est arbitraire, $u' \in RED_F$.
- (CR3): soit u neutre tel que pour tout u' tel que $u \rightarrow_{\beta\eta} u'$, $u' \in RED_F$. Pour montrer que $u \in RED_F$, fixons un $v \in RED_{F_1}$ arbitraire. Le terme uv ne peut se réduire en une étape que vers un terme de la forme $u'v$, avec $u \rightarrow_{\beta\eta} u'$, parce que u est neutre. Mais par hypothèse chacun de ces u' est tel que $u' \in RED_F$, donc $u'v \in RED_{F_2}$. Par hypothèse de récurrence sur F_2, cas (CR3), $uv \in RED_{F_2}$. Comme $v \in RED_{F_1}$ est arbitraire, par définition de la réductibilité $u \in RED_F$.

Avant de continuer, définissons $\nu(u)$, pour tout $u \in SN$, comme la longueur de la plus grande réduction partant de u. Ceci est bien défini parce que tout terme n'a qu'un nombre fini de rédexes: s'il n'y avait pas de plus longue réduction partant de u, il y aurait des réductions arbitrairement longues partant de u. Si $u_1^1, \ldots, u_n^1$ sont les réduits en une étape de u, ceci implique qu'il existerait i_1, $1 \leq i_1 \leq n$, et des réductions arbitrairement longues partant de $u_{i_1}^1$ — si toutes les réductions partant de tous les u_i^1 étaient de longueur bornée, il en serait de même pour u. On construit ainsi par récurrence sur k une suite $u_{i_k}^k$ telle qu'il existe des réductions arbitrairement longues partant de $u_{i_k}^k$, en itérant le procédé. Mais la réduction $u \rightarrow_{\beta\eta} u_{i_1}^1 \rightarrow_{\beta\eta} \ldots \rightarrow_{\beta\eta} u_{i_k}^k \rightarrow_{\beta\eta} \ldots$ serait alors infinie, contredisant $u \in SN$.

Montrons maintenant par récurrence sur les termes que tout terme typable est réductible. Plus précisément, montrons que si $\Gamma \vdash u : F$ est dérivable, alors $u \in RED_F$. Examinons d'abord chaque construction possible de typage:

1. Si $u = x$, alors $u \in RED_F$ par (CR3), puisque x est neutre et normal.

2. Si u est de la forme $u_1 u_2$, avec $u_1 \in RED_{F_1 \Rightarrow F}$ et $u_2 \in RED_{F_1}$, alors par définition de la réductibilité, $u \in RED_F$. Le cas le plus compliqué est donc devenu le plus simple.

3. Si u est de la forme $\lambda x \cdot u_1$, $F = F_1 \Rightarrow F_2$, et que $u_1[x := v] \in RED_{F_2}$ pour tout $v \in RED_{F_1}$, alors $u \in RED_F$. (L'hypothèse $u_1 \in RED_{F_2}$ ne suffit pas, comme vous pourrez le vérifier vous-mêmes. Cependant, comme x lui-même est dans RED_{F_1} par (CR3), l'hypothèse $u_1[x := v] \in RED_{F_2}$ pour tout $v \in RED_{F_1}$ implique lorsque $v = x$ que u_1 lui-même soit dans RED_{F_2}.) Pour montrer que $u \in RED_F$, fixons un $v \in RED_{F_1}$ arbitraire et montrons que $uv \in RED_{F_2}$. Ceci se fera par récurrence sur $\nu(u_1) + \nu(v)$ — cette quantité étant bien définie parce que u_1 et v sont réductibles donc dans SN par (CR1).

   Mais uv est neutre et ne peut donc se réduire en une étape que vers: (a) $u_1[x := v]$ (que ce soit par β-réduction de uv ou par η-réduction éventuelle de $\lambda x \cdot u_1$); ou (b) $(\lambda x \cdot u_1')v$ avec $u_1 \to_{\beta\eta} u_1'$; ou (c) $(\lambda x \cdot u_1)v'$ avec $v \to_{\beta\eta} v'$. Le terme (a) est dans RED_{F_2} par hypothèse, les termes (b) et (c) par hypothèse de récurrence. Donc par (CR3), $uv \in RED_{F_2}$. Comme $v \in RED_{F_1}$ est arbitraire, $u \in RED_F$.

À cause de la condition complexe dans le cas des abstractions, nous ne pourrons pas montrer directement que $\Gamma \vdash u : F$ dérivable implique $u \in RED_F$. Montrons à la place le résultat plus général que:

$$(\clubsuit) \text{ si } \Gamma, x_1 : F_1, \ldots, x_n : F_n \vdash u : F \text{ et } v_1 \in RED_{F_1}, \ldots, v_n \in RED_{F_n}, \text{ alors } u[x_1 := v_1, \ldots, x_n := v_n] \in RED_F$$

La notation $u[x_1 := v_1, \ldots, x_n := v_n]$ dénote ici la substitution *en parallèle* de x_1 par v_1, …, de x_n par v_n. À α-renommage près, elle est définie par: $x_i[x_1 := v_1, \ldots, x_n := v_n] \hat{=} v_i$, $y[x_1 := v_1, \ldots, x_n := v_n] \hat{=} y$ si $y \notin \{x_1, \ldots, x_n\}$, $(uv)[x_1 := v_1, \ldots, x_n := v_n] \hat{=} (u[x_1 := v_1, \ldots, x_n := v_n])(v[x_1 := v_1, \ldots, x_n := v_n])$, $(\lambda x \cdot u)[x_1 := v_1, \ldots, x_n := v_n] \hat{=} \lambda x \cdot (u[x_1 := v_1, \ldots, x_n := v_n])$.

Montrons $(\clubsuit)$ par récurrence structurelle sur u. Si u est une variable x_i, $1 \leq i \leq n$, alors par hypothèse $u[x_1 := v_1, \ldots, x_n := v_n] = v_i \in RED_{F_i}$, et $F_i = F$ par typage, donc $u[x_1 := v_1, \ldots, x_n := v_n] \in RED_F$. Si u est une autre variable, alors c'est par le point 1. ci-dessus. Si u est une application, c'est par récurrence et le point 2. ci-dessus. Si $u = \lambda x \cdot u_1$, $F = F_1' \Rightarrow F_2'$, alors par récurrence $u[x_1 := v_1, \ldots, x_n := v_n, x := v] \in RED_{F_2'}$ pour tout $v \in RED_{F_1'}$, donc par le point 3. ci-dessus, $u[x_1 := v_1, \ldots, x_n := v_n] \in RED_F$.

Dans le cas $n = 0$, on en déduit le cas particulier $\Gamma \vdash u : F$ dérivable implique $u \in RED_F$, qui implique $u \in SN$ par (CR1). ♢

En somme, un programme simplement typé finit toujours par s'arrêter. Ce n'est pas le cas dans la plupart des langages de programmation, y compris Pascal et OCaml, qui sont pourtant des langages typés. La raison principale est que ces langages fournissent tous une construction primitive de boucles ou de récursion (`let rec` en OCaml). De façon analogue, on peut voir cette capacité de récursion comme l'existence dans ces langages d'un opérateur de point fixe Y de type $(F \Rightarrow F) \Rightarrow F$, avec comme règle de réduction $Yf \to f(Yf)$. Le λ-calcul simplement typé enrichi avec un tel opérateur Y n'est alors plus fortement normalisant.

La propriété de normalisation forte, pour étrange qu'elle paraisse dans un cadre de langage de programmation, sera centrale dans la section qui suit.

1.2 Logique minimale

Si l'on regarde les règles de typage simple du λ-calcul, et que l'on efface les λ-termes et les variables des jugements de typage, on obtient les règles suivantes, formant le système **NJm**:

$$\frac{}{\Gamma, F \vdash F} (Ax)$$

$$\frac{\Gamma \vdash F_1 \Rightarrow F_2 \quad \Gamma \vdash F_1}{\Gamma \vdash F_2} (\Rightarrow E) \qquad \frac{\Gamma, F_1 \vdash F_2}{\Gamma \vdash F_1 \Rightarrow F_2} (\Rightarrow I)$$

Or si on lit maintenant ces règles en interprétant les types F comme des formules, le symbole $\Rightarrow$ comme l'implication logique, et les jugements $F_1, \ldots, F_n \vdash F$ comme des affirmations de la forme "sous les hypothèses F_1, ..., F_n, la formule F est vraie", alors ces règles fournissent un système de déduction logique tout à fait correct. La première règle dit que l'on peut toujours déduire une formule faisant partie de l'ensemble d'hypothèses, la deuxième est le *modus ponens*, ou *élimination de l'implication*, qui permet les déductions logiques: si F_1 implique F_2, et d'autre part F_1 est vraie, alors F_2 l'est aussi; la troisième est le *théorème de la déduction*, ou *introduction de l'implication*: pour prouver que F_1 implique F_2 (conclusion de la règle), il suffit de poser F_1 en nouvelle hypothèse et de démontrer F_2 (prémisse).

Un tel système de déduction est appelé système de *déduction naturelle*, et a été inventé par Gerhard Gentzen dans les années 30. Un tel système est caractérisé par le fait que toutes les règles agissent sur les formules à droite du symbole $\vdash$.

On ne peut pas démontrer toutes les formules valides de la logique propositionnelle classique en **NJm**. Par exemple, $((F_1 \Rightarrow F_2) \Rightarrow F_1) \Rightarrow F_1$, la loi de Peirce, n'est pas prouvable en **NJm**, bien qu'elle soit vraie en logique classique (exercice: tester la valeur de la formule quand F_1 et F_2 parcourent l'ensemble des booléens, et vérifier cette dernière assertion). La logique que le système de déduction **NJm** définit est une logique différente, appelée *logique intuitionniste minimale*, ou *logique minimale*.

Comme on le voit, il y a correspondance bijective entre les règles de déduction de logique minimale et les règles de typage du λ-calcul. Ceci implique qu'il y a *correspondance* entre preuves en logique minimale et λ-termes simplement typés: on obtient un λ-terme u tel que $x_1 : F_1, \ldots, x_n : F_n \vdash F$ à partir d'une preuve de $F_1, \ldots, F_n \vdash F$ en **NJm** en remplaçant les axiomes $F_1, \ldots, F_n, \ldots, F_k \vdash F_i$ par une variable x_i, les éliminations de l'implication par des applications et les introductions de l'implication par des abstractions.

Réciproquement, tout λ-terme u dénote un squelette de preuve, qui peut être complété en une preuve réelle si et seulement si u est simplement typable. Ces considérations nous permettront de voir les dérivations de typage comme des preuves en **NJm**, décorées par des λ-termes décrivant le squelette de la preuve.

Que signifient alors les propriétés d'auto-réduction et de normalisation forte ? La β-réduction réécrit des λ-termes, donc des squelettes de preuve. Un β-rédex correspond à une preuve de la forme suivante:

$$\dfrac{\dfrac{\begin{matrix}\vdots\,\pi_1\\ \Gamma, x : F_1 \vdash u : F\end{matrix}}{\Gamma \vdash \lambda x \cdot u : F_1 \Rightarrow F} \quad \begin{matrix}\vdots\,\pi_2\\ \Gamma \vdash v : F_1\end{matrix}}{\Gamma \vdash (\lambda x \cdot u)v : F}$$

et son β-réduit est, par auto-réduction, encore une preuve de F sous les hypothèses Γ. Le processus de β-réduction est donc un processus de *transformation* de preuves. Par la propriété de normalisation forte, ce processus s'arrête toujours. La β-normalisation est donc un processus de *simplification*, de mise en forme normale de preuves.

Il a pour effet d'éliminer des *détours*. Considérons la preuve correspondant au β-rédex ci-dessus. Elle consiste à prouver F en commençant par prouver F sous l'hypothèse supplémentaire F_1 (par la preuve π_1), et à prouver F_1 d'autre part (par la preuve π_2). Réduire le β-rédex consiste à éliminer le détour via le lemme auxiliaire F_1, et à démontrer F directement.

Formellement, un *détour* est une règle ($\Rightarrow I$) introduisant le connecteur $\Rightarrow$ dans la prémisse majeure (celle de gauche, qui contient le connecteur $\Rightarrow$ éliminé) d'une règle ($\Rightarrow E$). On obtient ainsi:

Lemme 5 (Prawitz) *Si $\Gamma \vdash F$ est prouvable en* **NJm**, *alors il a une preuve sans détour.*

Preuve : Soit $\Gamma \hat{=} F_1, \ldots, F_n$. Si $\Gamma \vdash F$ est prouvable en **NJm**, alors il existe un terme u tel que $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$ soit dérivable. La forme normale de u existe par la propriété de normalisation forte, et correspond donc à une preuve sans détour de $\Gamma \vdash F$ en **NJm**. $\diamondsuit$

Les preuves sans détour, à leur tour, ont leurs squelettes décrits par des λ-termes en forme normale. Ce sont donc des termes de la forme:

$$\lambda x_1, \ldots, x_m \cdot x u_1 \ldots u_n$$

où x est la variable de tête et u_1, …, u_n sont eux-mêmes normaux. (Voir le théorème de standardisation en partie I, et la notion d'arbres de Böhm.)

Une preuve sans détour a donc la forme (où les variables du contexte sont supposées être x_{m+1}, …, x_p, d'autre part x est x_i pour un certain i, $1 \leq i \leq p$, et F_i est de la forme $F_{i1} \Rightarrow \ldots \Rightarrow F_{im} \Rightarrow F$):

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma, x_1 : F_1, \ldots, x_m : F_m \vdash u_1 : F_{i1}\end{array} \quad \ldots \quad \begin{array}{c}\vdots\\ \Gamma, x_1 : F_1, \ldots, x_m : F_m \vdash u_n : F_{in}\end{array}}{\Gamma, x_1 : F_1, \ldots, x_m : F_m \vdash x_i u_1 \ldots u_n : F}\,(Ax)\text{ et }(\Rightarrow E)\ n\text{ fois}}{\Gamma \vdash \lambda x_1, \ldots, x_m \cdot x u_1 \ldots u_n : F_1 \Rightarrow \ldots \Rightarrow F_m \Rightarrow F}\,(\Rightarrow I)\ m\text{ fois}$$

En éliminant les λ-termes de la preuve, on a donc effectué une étape de preuve de la forme:

$$\dfrac{\begin{array}{c}\vdots\\ \Delta \vdash F_{i1}\end{array} \quad \ldots \quad \begin{array}{c}\vdots\\ \Delta \vdash F_{im}\end{array}}{\Delta \vdash F}\,(BC)$$

où Δ est $F_1, \ldots, F_p$ et contient $F_i \hat{=} F_{i1} \Rightarrow \ldots \Rightarrow F_{im} \Rightarrow F$, suivie éventuellement d'une de la forme:

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma, F_1, \ldots, F_m \vdash F\end{array}}{\Gamma \vdash F_1 \Rightarrow \ldots \Rightarrow F_m \Rightarrow F}\,(\Rightarrow^+ I)$$

où $m \geq 1$.

Si la dernière n'est qu'une généralisation de la règle d'introduction de l'implication (à droite de $\vdash$), la première est plus intéressante et agit en partie à gauche du signe $\vdash$: pour démontrer F sachant qu'une hypothèse prouve $F_{i1} \Rightarrow \ldots \Rightarrow F_{im} \Rightarrow F$, il suffit de prouver F_{i1}, …, F_{im}. Il s'agit d'une forme de *chaînage arrière* (*backchaining*; ceci généralise naturellement le mécanisme d'exécution de Prolog, cf. [MNPS91]).

Corollaire 2 *Le système* **NJm** *est* cohérent*: il existe des formules F non prouvables, c'est-à-dire telles que $\vdash u : F$ pour aucun u.*

Preuve : Soit F n'importe quel type de base. S'il existe une preuve $\vdash u : F$, alors on peut supposer que u est normal. Alors la dernière règle était soit (BC) soit $(\Rightarrow^+ I)$. Mais (BC) ne s'applique pas, puisqu'on n'a pas d'hypothèse à gauche de $\vdash$, et $(\Rightarrow^+ I)$ ne s'applique pas, parce que F est un type de base. $\diamondsuit$

Ce résultat pouvait être démontré de façon beaucoup plus simple:

Exercice 11 *En se rappelant que toute formule F prouvable en* **NJm** *est valide (vraie quelles que soient les valeurs de vérité, vrai ou faux, prises par les types de base), montrer que $\vdash F$ n'est pas prouvable pour aucun type de base F.*

Exercice 12 *On définit la forme η-*longue *$\eta_{\Gamma \vdash F}[u]$ d'un terme $u \hat{=} \lambda x_1, \ldots, x_m \cdot x u_1 \ldots u_n$ en forme normale, de type $F \hat{=} F_1 \Rightarrow \ldots \Rightarrow F_p \Rightarrow b$ (b type de base, $p \geq m$) dans le contexte Γ, où x est supposée de type $G \hat{=} G_1 \Rightarrow \ldots \Rightarrow G_n \Rightarrow F_{m+1} \Rightarrow \ldots \Rightarrow F_p \Rightarrow b$ dans Γ ($q \geq n$), par:*

$$\eta_{\Gamma \vdash F}[u] \hat{=} \lambda x_1, \ldots, x_m, x_{m+1}, \ldots, x_p \cdot x(\eta_{\Delta \vdash G_1}[u_1]) \ldots (\eta_{\Delta \vdash G_n}[u_n])(\eta_{\Delta \vdash F_{m+1}}[x_{m+1}]) \ldots (\Delta \vdash \eta_{F_p}[x_p])$$

où $\Delta = \Gamma, x_1 : F_1, \ldots, x_p : F_p$.

On notera que le type de x est déterminé en partie par le type F. D'autre part, la définition est par récurrence sur la paire $(|u|, |F|)$ ordonnée lexicographiquement, où $|u|$ est la taille de u, et $|F|$ la taille du type F. Intuitivement, $\eta_{\Gamma \vdash F}[u]$ est obtenu à partir de u en appliquant la règle η à l'envers autant qu'on le peut sans créer de β-rédex.

Montrer que $\eta_{\Gamma \vdash F}[u] \rightarrow^* u$ et que si $\Gamma \vdash u : F$ est dérivable, alors $\Gamma \vdash \eta_{\Gamma \vdash F}[u] : F$ aussi. Par l'examen de la forme de $\eta_{\Gamma \vdash F}[u]$, en déduire que pour chercher si $\Gamma \vdash F$ est prouvable, on peut se restreindre à appliquer (BC) uniquement lorsque F est un type de base, et à appliquer $(\Rightarrow^+ I)$ avec F un type de base.

D'autres conséquences de la technique d'élimination des détours sont moins évidents, comme nous allons le voir. On obtient le système **NJf** pour le *fragment implicatif de la logique intuitionniste* en ajoutant au langage des types une constante $\perp$ dénotant le faux (en tant que valeur de vérité), et en ajoutant la règle:

$$\frac{\Gamma \vdash u : \perp}{\Gamma \vdash \nabla u : F} \; (\perp E)$$

D'un point de vue logique, $(\perp E)$ exprime que du faux on peut tout déduire. Il n'y a pas de règle d'introduction du faux, car il est dans notre intention que $\perp$ ne soit pas prouvable. D'un point de vue calculatoire, on ajoute la construction ∇u à la grammaire du λ-calcul. Intuitivement, si $u : \perp$, alors l'évaluation de u ne termine pas, et on peut donc convertir le résultat inexistant de l'évaluation de u à n'importe quel type, en $\nabla u : F$.

Cette construction n'introduit pas de nouveau détour (introduction d'un connecteur en prémisse majeure suivie de son élimination), mais elle permet à certaines autres simplifications de se produire:

$$\frac{\dfrac{\dfrac{\vdots\, \pi_1}{\Gamma \vdash u : \perp}}{\Gamma \vdash \nabla u : F_1 \Rightarrow F_2}\,(\perp E) \qquad \dfrac{\vdots\, \pi_2}{\Gamma \vdash v : F_1}}{\Gamma \vdash \nabla uv : F_2}\,(\Rightarrow E) \quad \rightarrow \quad \frac{\dfrac{\vdots\, \pi_1}{\Gamma \vdash u : \perp}}{\Gamma \vdash \nabla u : F_2}\,(\perp E)$$

Ceci est appelé une *conversion commutative*, et mène à la règle de calcul $\nabla uv \rightarrow \nabla u$. (Nous avons déjà vu quelques conversions commutatives à l'exercice 10.) Appelons le calcul combinant (β) et la conversion commutative ci-dessus le $\lambda\nabla$-calcul.

Théorème 2 *Le $\lambda\nabla$-calcul simplement typé est fortement normalisant.*

Preuve : Même preuve que pour le théorème 1, en considérant que $\perp$ est un type de base; les seules différences sont dans la définition des termes neutres — que l'on définira maintenant comme les termes qui ne commencent pas par un λ ou un ∇ —, et qu'il faut vérifier que si $u \in RED_\perp$, alors $\nabla u \in RED_F$ pour tout type F. C'est par récurrence structurelle sur F. Comme $u \in RED_\perp$, $u \in SN$. Si F est un type de base, alors ∇u ne peut se réécrire qu'à l'intérieur de u, et est donc dans SN, donc dans $RED_\perp$. Si F est de la forme $F_1 \Rightarrow F_2$, alors par hypothèse de récurrence $\nabla u \in RED_{F_2}$ (*). On montre par récurrence sur $\nu(u) + \nu(v)$ que pour tout $v \in RED_{F_1}$, ∇uv est dans RED_{F_2}. C'est par (CR3), en regardant où se déroule la première réduction: si elle s'effectue dans u ou dans v, on utilise l'hypothèse de récurrence, et si c'est $\nabla uv \rightarrow \nabla u$, alors c'est par (*). Donc $\nabla uv \in RED_{F_2}$, et v étant quelconque, $\nabla u \in RED_F$. $\diamondsuit$

Corollaire 3 *Le système* **NJf** *est cohérent.*

Preuve : Comme pour le corollaire 2, les formes normales u, où $\vdash u : b$ est dérivable, sont nécessairement de la forme $hu_1 \ldots u_n$, où h est une variable ou le symbole ∇ (auquel cas $n = 1$). Le seul ingrédient nouveau est que nous effectuons une récurrence structurelle sur u. Si u est tel que $\vdash u : b$ est dérivable, pour b un type de base, alors $m = 0$ car b est un type de base, et h ne peut pas être une variable car le côté gauche du jugement est vide. Donc h est ∇ et $n = 1$. Mais alors on a une dérivation plus petite de $\vdash u_1 : \perp$, ce qui est impossible par hypothèse de récurrence. $\diamondsuit$

On a démontré en particulier que le faux $\perp$ n'était pas prouvable. Mais on peut dire mieux. Notons $\neg F$ pour $F \Rightarrow \perp$: il s'agit de la *négation* de F, représentée par le fait que $\neg F$ est vraie exactement lorsque F implique une contradiction $\perp$. En logique classique, $\neg\neg F$ et F sont équivalentes, mais pas en **NJf**:

Lemme 6 $\vdash \neg\neg F \Rightarrow F$ *n'est pas prouvable en* **NJf** *en général.*

Preuve : Prenons F un type de base b différent de $\bot$. Soit u un terme normal clos de type $\neg\neg b \Rightarrow b$. Par typage et comme u est clos, u est nécessairement de la forme $\lambda x \cdot u_1$, où $x : \neg\neg b \vdash u_1 : b$. Le symbole de tête de u_1 ne peut être que x ou ∇. Si c'est x, u_1 est de la forme x ou xu_2; mais aucun des deux cas n'est possible, le premier impliquant $\neg\neg b = b$, et le second impliquant $\bot = b$. Donc $u_1 = \nabla u_2$, avec $x : \neg\neg b \vdash u_2 : \bot$. Par des arguments sémantiques comme dans l'exercice 11, on peut conclure directement qu'une telle preuve du faux à partir de $\neg\neg b$ n'existe pas.

Mais montrons comment des arguments syntaxiques mènent à la même conclusion. Nous allons montrer qu'il n'y a pas de terme normal v tel que $\Gamma \vdash v : \bot$, pour aucun Γ ne contenant que des liaisons de la forme $x : \neg\neg b$ ou $y : b$. Supposons le contraire, et choisissons v de taille minimale. Alors v ne peut que s'écrire xw pour une variable x de type $\neg\neg b$ dans Γ et un terme w tel que $\Gamma \vdash w : \neg b$, ou $\nabla v'$ avec $\Gamma \vdash v' : \bot$. Mais le deuxième cas contredirait la minimalité de la taille de v, donc seul le premier cas est possible. Si le terme w ne commence pas par λ, alors il s'écrit xw' pour un $x : \neg\neg b$ ou bien y pour un $y : b$ dans Γ, mais ceci donnerait à w le type $\bot$ ou le type b, ce qui est impossible. Donc w est de la forme $\lambda z \cdot v'$, avec $\Gamma, z' : b \vdash v' : \bot$. Mais ceci contredit l'hypothèse de minimalité de la taille de v, encore une fois. $\diamondsuit$

Exercice 13 *Montrer que $F \Rightarrow \neg\neg F$ est prouvable en* **NJf***, et donner un λ-terme de ce type.*

On voit donc que la logique **NJf** n'est pas la logique classique, puisque $\neg\neg F \Rightarrow F$ est vraie que F soit vraie ou fausse. Il s'agit d'une logique *intuitionniste*, dont les principes remontent au mathématicien hollandais Luitzen Egbertus Jan Brouwer au début du vingtième siècle. La logique intuitionniste est constructive au sens où une formule est vue comme l'ensemble de ses preuves, $\bot$ est l'ensemble vide et $F \Rightarrow G$ est l'ensemble des algorithmes prenant en entrée une preuve de F et retournant une preuve de G. Elle joue un grand rôle en informatique théorique.

Exercice 14 () *On peut aussi donner une sémantique plus fine de la logique. Soit $(\mathcal{W}, \leq)$ un ensemble ordonné quelconque dit de* mondes*, et ρ un environnement envoyant chaque type de base b vers un ensemble de mondes $\rho(b)$ tel que si $w \in \rho(b)$ et $w \leq w'$, alors $w' \in \rho(b)$. On définit la relation $w, \rho \models F$, où $w \in \mathcal{W}$, par:*

- *$w, \rho \models b$ si et seulement si $w \in \rho(b)$ (autrement dit, $\rho(b)$ est l'ensemble des mondes où b est vrai);*
- *$w, \rho \models \bot$ est toujours faux;*
- *$w, \rho \models F \Rightarrow G$ si et seulement si pour tout $w' \geq w$, si $w' \models F$ alors $w' \models G$.*

Montrer que si $\vdash u : F$ est dérivable, alors $w, \rho \models F$ pour tout $w \in \mathcal{W}$, pour tout ρ comme ci-dessus, pour tout $(\mathcal{W}, \leq)$. Montrer qu'il existe $(\mathcal{W}, \leq)$, ρ et $w \in \mathcal{W}$ tels que $w, \rho \models \neg\neg b \Rightarrow b$ ne soit pas vrai. En déduire une autre preuve du lemme 6. (La sémantique ci-dessus est la sémantique de Kripke *de la logique intuitionniste; elle est complète au sens où si $w, \rho \models F$ pour tout $w \in \mathcal{W}$, pour tout ρ comme ci-dessus, pour tout $(\mathcal{W}, \leq)$, alors F est prouvable en* **NJf***.)*

On peut encore enrichir la logique et obtenir le système **NJi** de *logique intuitionniste* en ajoutant au langage des types des constructions $F_1 \wedge F_2$ (conjonction) et $F_1 \vee F_2$. Nous le mentionnons ici, mais repartirons de **NJm** dans les sections suivantes, car **NJm** contient déjà l'essentiel. Nous ajoutons les règles:

$$\frac{\Gamma \vdash u : F_1 \wedge F_2}{\Gamma \vdash \pi_i u : F_i} (\wedge E_i), i = 1, 2 \qquad \frac{\Gamma \vdash u : F_1 \quad \Gamma \vdash v : F_2}{\Gamma \vdash \langle u, v \rangle : F_1 \wedge F_2} (\wedge I)$$

$$\frac{\Gamma \vdash u : F_1 \vee F_2 \quad \Gamma, x_1 : F_1 \vdash v_1 : F \quad \Gamma, x_2 : F_2 \vdash v_2 : F}{\Gamma \vdash \text{case } u \left\{ \begin{array}{c} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} : F} (\vee E) \qquad \frac{\Gamma \vdash u : F_i}{\Gamma \vdash \iota_i u : F_1 \vee F_2} (\vee I_i), i = 1, 2$$

Les règles $(\wedge E_1)$ et $(\wedge E_2)$ permettent de déduire F_1, respectivement F_2 à partir de $F_1 \wedge F_2$; de même, $(\vee E_1)$ et $(\vee E_2)$ permettent de déduire $F_1 \vee F_2$ à partir de F_1 ou de F_2. La règle $(\wedge I)$ permet de prouver $F_1 \wedge F_2$ en prouvant F_1 et F_2 séparément, et la règle $(\vee E)$ permet le *raisonnement par cas*: si $F_1 \vee F_2$ est vrai, et que l'on sait prouver F en supposant soit F_1 (cas 1) soit F_2 (cas 2), alors c'est que F est vraie dans tous les cas.

À ces règles de preuves correspondent les constructions indiquées venant enrichir le λ-calcul. Les termes sont maintenant définis par la grammaire:

$$\Lambda^+ ::= \mathcal{V} \mid \Lambda^+\Lambda^+ \mid \lambda\mathcal{V} \cdot \Lambda^+ \mid \pi_1\Lambda^+ \mid \pi_2\Lambda^+ \mid \langle \Lambda^+, \Lambda^+ \rangle \mid \iota_1\Lambda^+ \mid \iota_2\Lambda^+ \mid \text{case } \Lambda^+ \left\{ \begin{array}{c} \iota_1\mathcal{V} \mapsto \Lambda^+ \\ \iota_2\mathcal{V} \mapsto \Lambda^+ \end{array} \right\} \mid \nabla\Lambda^+$$

La sémantique intuitive des nouvelles constructions est la suivante: $\langle u, v \rangle$ est le couple formé de u et de v, π_1 récupère la première composante et π_2 la seconde; la conjonction $F_1 \wedge F_2$ est donc un *produit cartésien* de F_1 et F_2. La disjonction $F_1 \vee F_2$ peut-être vue comme une *somme disjointe* de F_1 et de F_2: $\iota_1 u$ est une conversion permettant de voir $u \in F_1$ comme un élément de $F_1 \vee F_2$, et similairement pour ι_2 et F_2; réciproquement, tout u dans $F_1 \vee F_2$ est soit de la forme $\iota_1 t_1$, t_1 dans F_1, soit de la forme $\iota_2 t_2$, t_2 dans F_2, alors case $u \left\{ \begin{array}{c} \iota_1 x_1 \mapsto v_1 \\ \iota_1 x_2 \mapsto v_2 \end{array} \right\}$ retourne la valeur de $v_1[x_1 := t_1]$ dans le premier cas, et celle de $v_2[x_2 := t_2]$ dans le second.

Outre la règle (β), l'élimination des détours demande à faire les transformations suivantes sur les preuves:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_1 \\ \Gamma \vdash u_1 : F_1\end{array} \quad \begin{array}{c}\vdots\,\pi_2 \\ \Gamma \vdash u_2 : F_2\end{array}}{\Gamma \vdash \langle u_1, u_2 \rangle : F_1 \wedge F_2}\,(\wedge I)}{\Gamma \vdash \pi_i \langle u_1, u_2 \rangle : F_i}\,(\wedge E_i) \qquad \to \qquad \begin{array}{c}\vdots\,\pi_i \\ \Gamma \vdash u_i : F_i\end{array}$$

ce qui se traduit dans notre λ-calcul enrichi par les règles $\pi_1\langle u_1, u_2 \rangle \to u_1$ et $\pi_2\langle u_1, u_2 \rangle \to u_2$, et:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_0 \\ \Gamma \vdash u : F_i\end{array}}{\Gamma \vdash \iota_i u : F_1 \vee F_2}\,(\vee I_i) \quad \begin{array}{c}\vdots\,\pi_1 \\ \Gamma, x_1 : F_1 \vdash v_1 : F\end{array} \quad \begin{array}{c}\vdots\,\pi_2 \\ \Gamma, x_2 : F_2 \vdash v_2 : F\end{array}}{\Gamma \vdash \text{case } \iota_i u \left\{ \begin{array}{c} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} : F}\,(\vee E) \qquad \to \qquad \begin{array}{c}\vdots \\ \Gamma \vdash v_i[x_i := u] : F\end{array}$$

où la dérivation de droite est obtenue comme dans le cas (β) du lemme 1. Ceci mène à la règle:

$$\text{case } \iota_i u \left\{ \begin{array}{c} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} \to v_i[x_i := u]$$

On a aussi des *conversions commutatives*, dont la liste complète est:

$$
\begin{array}{rcll}
\nabla u v & \to & \nabla u & \\
\pi_i(\nabla u) & \to & \nabla u & (i = 1, 2) \\
\iota_i(\nabla u) & \to & \nabla u & (i = 1, 2) \\
\text{case } \nabla u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} & \to & \nabla u & \\
\left(\text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} \right) w & \to & \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 w \\ \iota_2 x_2 \mapsto v_2 w \end{array} \right\} & \\
\pi_i \left(\text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} \right) & \to & \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \pi_i v_1 \\ \iota_2 x_2 \mapsto \pi_i v_2 \end{array} \right\} & (i = 1, 2) \\
\iota_i \left(\text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} \right) & \to & \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_i v_1 \\ \iota_2 x_2 \mapsto \iota_i v_2 \end{array} \right\} & (i = 1, 2) \\
\text{case } \left(\text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} \right) \left\{ \begin{array}{l} \iota_1 y_1 \mapsto w_1 \\ \iota_2 y_2 \mapsto w_2 \end{array} \right\} & \to & \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } v_1 \left\{ \begin{array}{l} \iota_1 y_1 \mapsto w_1 \\ \iota_2 y_2 \mapsto w_2 \end{array} \right\} \\ \iota_2 x_2 \mapsto \text{case } v_2 \left\{ \begin{array}{l} \iota_1 y_1 \mapsto w_1 \\ \iota_2 y_2 \mapsto w_2 \end{array} \right\} \end{array} \right\} &
\end{array}
$$

Exercice 15 *À quelles transformation sur les preuves correspondent les conversions commutatives ? En déduire le théorème d'auto-réduction pour ce λ-calcul enrichi.*

Exercice 16 *On enrichit la sémantique de Kripke de l'exercice 14 par:*

- $w, \rho \models F_1 \wedge F_2$ *si et seulement si* $w, \rho \models F_1$ *et* $w, \rho \models F_2$*;*
- $w, \rho \models F_1 \vee F_2$ *si et seulement si* $w, \rho \models F_1$ *ou* $w, \rho \models F_2$*.*

Montrer que $F \vee \neg F$ n'est pas prouvable en démontrant d'abord qu'il existe $(\mathcal{W}, \leq)$, w et ρ tels que $w, \rho \models F \vee \neg F$ soit faux.

2 Logique classique et continuations

Comme $\neg\neg F \Rightarrow F$ n'est pas prouvable en **NJf**, il est tentant d'ajouter une règle de déduction permettant de le prouver. Cette règle correspondra à un nouveau terme à ajouter au λ-calcul. Nous obtiendrons alors un système de démonstration pour la logique classique.

En fait, nous connaissons déjà le nouveau terme à ajouter ! Il s'agit de l'opérateur $\mathcal{C}$ de Felleisen [FFKD87], comme l'a découvert Griffin [Gri90]. Raisonnons en effet intuitivement, au niveau des types: $\mathcal{C}$ prend une fonction f en argument, et cette fonction prend une continuation k en argument. Une continuation est juste une fonction de type $F \Rightarrow \bot$, où F est le type de la valeur V à retourner, puisque kV ne retourne pas (donc kV est de type $\bot$). La fonction f elle-même ne retourne pas non plus, puisque fk est censée toujours appeler une continuation, k ou autre, pour retourner son résultat; donc f est de type $(F \Rightarrow \bot) \Rightarrow \bot = \neg\neg F$. Et le résultat de $\mathcal{C}f$ est de type F, le type de la valeur passée à la continuation. En somme, $\mathcal{C}$ est de type $\neg\neg F \Rightarrow F$, pour tout F.

Pour des raisons esthétiques, on va préférer considérer que $\mathcal{C}$ n'est pas une constante, mais un opérateur unaire du langage, tel que si $u : \neg\neg F$ alors $\mathcal{C}u : F$. Une fois qu'on a $\mathcal{C}$, la construction ∇ n'est plus nécessaire: il suffit de poser $\nabla u = \mathcal{C}(\lambda z \cdot u)$, où z n'est pas libre dans u. (On notera que ∇ est essentiellement l'opérateur "abort" $\mathcal{A}$ de [FFKD87].) En effet, on a la dérivation:

$$
\dfrac{\dfrac{\Gamma, z : \neg F \vdash u : \bot}{\Gamma \vdash \lambda z \cdot u : \neg\neg F}}{\Gamma \vdash \mathcal{C}(\lambda z \cdot u) : F}
$$

où la première ligne est obtenue par affaiblissement du typage donné $\Gamma \vdash u : \bot$ de u.

On obtient le système de déduction naturelle **ND** suivant pour la logique classique implicationnelle:

$$\frac{\Gamma \vdash u : \neg\neg F}{\Gamma \vdash \mathcal{C}u : F}\,(\neg\neg E) \qquad \frac{}{\Gamma, x : F \vdash x : F}\,(Ax)$$

$$\frac{\Gamma \vdash u : F_1 \Rightarrow F_2 \quad \Gamma \vdash v : F_1}{\Gamma \vdash uv : F_2}\,(\Rightarrow E) \qquad \frac{\Gamma, x : F_1 \vdash u : F_2}{\Gamma \vdash \lambda x \cdot u : F_1 \Rightarrow F_2}\,(\Rightarrow I)$$

Ce système de déduction n'offre plus la symétrie entre règles d'introduction et d'élimination de **NJm**, **NJf** et **NJi**. Ceci est réparé par le $\lambda\mu$-calcul de Michel Parigot [Par92], que nous ne présenterons cependant pas ici.

En attendant, la syntaxe du $\lambda\mathcal{C}$-*calcul* est:

$$\Lambda^c ::= \mathcal{V} \mid \Lambda^c\Lambda^c \mid \lambda\mathcal{V} \cdot \Lambda^c \mid \mathcal{C}\Lambda^c$$

et ses règles de réduction sont:

$$\begin{array}{llcl}
(\beta) & (\lambda x \cdot u)v & \rightarrow & u[x := v] \\
(\mathcal{C}_L) & \mathcal{C}uv & \rightarrow & \mathcal{C}(\lambda k \cdot u(\lambda f \cdot k(fv))) \\
(\eta\mathcal{C}) & \mathcal{C}(\lambda k \cdot ku) & \rightarrow & u \qquad (k \notin \mathrm{fv}(u))
\end{array}$$

On a laissé de côté la règle $(\mathcal{C}_R)$, à savoir $V(\mathcal{C}u) \rightarrow \mathcal{C}(\lambda k \cdot u(\lambda x \cdot k(Vx)))$ lorsque V est une P-valeur, car elle est inutile logiquement.

Exercice 17 *Montrer le théorème d'auto-réduction pour $\lambda\mathcal{C}$.*

Théorème 3 *Le $\lambda\mathcal{C}$-calcul simplement typé est fortement normalisant.*

Preuve : Même preuve que pour le théorème 1, en considérant que $\bot$ est un type de base, et que les termes neutres sont tous ceux qui ne commencent ni par λ ni par $\mathcal{C}$; la seule différence est qu'il faut vérifier que si $u \in RED_{\neg\neg F}$, alors $\mathcal{C}u \in RED_F$ pour tout type F. C'est, comme au théorème 2, par récurrence structurelle sur F.

Si F est un type de base, c'est par récurrence double sur $\nu(u)$ d'abord, et la taille $|u|$ de u ensuite. En effet, $\mathcal{C}u$ est dans RED_F si et seulement si $u \in SN$. Soit R une réduction partant de $\mathcal{C}u$. Si R commence par une réduction dans u, i.e. $u \rightarrow u'$, alors $\nu(u') < \nu(u)$, donc R est finie par hypothèse de récurrence. Si R commence par une réduction au sommet, alors la seule règle possible était $(\eta\mathcal{C})$, donc u est de la forme $\lambda k \cdot ku'$, k non libre dans u'. Dans ce cas $\nu(u') \leq \nu(u)$ et $|u'| < |u|$, donc l'hypothèse de récurrence est encore applicable, et R est donc finie encore une fois. R étant quelconque, $\mathcal{C}u$ est dans SN, donc dans RED_F.

Si F est de la forme $F_1 \Rightarrow F_2$, montrons que $\mathcal{C}uv \in RED_{F_2}$ pour tout $v \in RED_{F_1}$. Notons d'abord que, par hypothèse de récurrence: $(*)$ pour tout $w \in RED_{\neg\neg F_2}$, $\mathcal{C}w \in RED_{F_2}$. Notons aussi que $\mathcal{C}uv$ est neutre. Montrons alors par récurrence sur $\nu(u) + \nu(v)$ que $\mathcal{C}uv \in RED_{F_2}$, en utilisant (CR3). Le terme $\mathcal{C}uv$ peut se réduire en une étape soit vers (1) $\mathcal{C}u'v$ avec $u \rightarrow u'$ (qui est donc dans RED_{F_2} par hypothèse de récurrence), soit vers (2) $\mathcal{C}uv'$ avec $v \rightarrow v'$ (même argument), soit vers (3) $u'v$ par $(\eta\mathcal{C})$ (avec $u = \lambda k' \cdot k'u'$, k' non libre dans u'), soit vers (4) $\mathcal{C}(\lambda k \cdot u(\lambda f \cdot k(fv)))$.

Dans le cas (4), on a le raisonnement suivant. Rappelons que: $(**)$ si $a[x := b] \in RED_{G_2}$ pour tout $b \in RED_{G_1}$, alors $\lambda x \cdot a \in RED_{G_1 \Rightarrow G_2}$ (voir théorème 1). Supposons que $s \in RED_{\neg F_2}$ et $t \in RED_{F_1 \Rightarrow F_2}$ sont des termes arbitraires, s dénotant une valeur possible de k et t une de f:

1. $v \in RED_{F_1}$ par hypothèse;
2. donc $tv \in RED_{F_2}$, par définition de la réductibilité;
3. donc $s(tv) \in RED_{\bot}$, par définition de la réductibilité;

4. t étant arbitraire, $\lambda f \cdot s(fv) \in RED_{\neg(F_1 \Rightarrow F_2)}$, par $(**)$;
5. donc $u(\lambda f \cdot s(fv)) \in RED_\perp$, par définition de la réductibilité;
6. s étant arbitraire, $\lambda k \cdot u(\lambda f \cdot k(fv)) \in RED_{\neg\neg F_2}$, par $(**)$;
7. donc $\mathcal{C}(\lambda k \cdot u(\lambda f \cdot k(fv))) \in RED_{F_2}$, par $(*)$.

Dans le cas (3) montrons d'abord que: $(***)$ si $\lambda k' \cdot k'u' \in RED_{\neg\neg F}$ et k' n'est pas libre dans u', alors $u' \in RED_F$. Ceci se démontre par récurrence structurelle sur F. Si F est un type de base, alors par (CR1) $\lambda k' \cdot k'u' \in SN$, donc $u' \in SN = RED_F$. Si F est de la forme $F_1 \Rightarrow F_2$, soit $v \in RED_{F_1}$ quelconque, et considérons le terme $\lambda k'' \cdot (\lambda k' \cdot k'u')(\lambda x \cdot k''(xv))$. Ce terme est dans $RED_{\neg\neg F_2}$, en effet, pour tout $s \in RED_{\neg F_2}$ (valeur de k''), pour tout $t \in RED_F$ (valeur de x):

1. $tv \in RED_{F_2}$ par définition;
2. donc $s(tv) \in RED_\perp$ par définition;
3. t étant arbitraire, par $(**)$ $\lambda x \cdot s(xv) \in RED_{\neg F}$;
4. puisque par hypothèse $\lambda k' \cdot k'u' \in RED_{\neg\neg F}$, $(\lambda k' \cdot k'u')(\lambda x \cdot s(xv)) \in RED_\perp$;
5. s étant arbitraire, $\lambda k'' \cdot (\lambda k' \cdot k'u')(\lambda x \cdot k''(xv)) \in RED_{\neg\neg F_2}$.

Mais ce terme se réduit en $\lambda k'' \cdot k''(u'v)$, qui est dans $RED_{\neg\neg F_2}$ aussi, par (CR2). Par hypothèse de récurrence, on en conclut $u'v \in RED_{F_2}$. Comme v est arbitraire, $u' \in RED_F$. On a donc démontré $(***)$. On traite maintenant le cas (3) en remarquant que si $u = \lambda k' \cdot k'u' \in RED_{\neg\neg F}$, alors $u' \in RED_F$ par $(***)$, donc le réduit $u'v$ du cas (3) est dans RED_{F_2}.

En somme, tous les réduits en une étape de $\mathcal{C}uv$ sont dans RED_{F_2}. Par (CR3), $\mathcal{C}uv$ est donc dans RED_{F_2}, et comme v est arbitraire, $\mathcal{C}u \in RED_{F_1 \Rightarrow F_2}$.

Le reste de la démonstration consiste à redémontrer la propriété (♣) du théorème 1, par récurrence structurelle sur les termes. ♢

Exercice 18 *Pourquoi la preuve de normalisation forte de $\lambda\mathcal{C}$ plus la règle:*

$$(\mathcal{C}_R) \quad V(\mathcal{C}u) \quad \rightarrow \quad \mathcal{C}(\lambda k \cdot u(\lambda x \cdot k(Vx)))$$

où V est une P-valeur, est-elle plus difficile ?

Exercice 19 *Montrer que les formes normales en $\lambda\mathcal{C}$ sont de la forme $\lambda x_1, \ldots, x_m \cdot hu_1 \ldots u_n$, où $u_1, \ldots, u_n$ sont normaux, où h est soit une variable soit le symbole $\mathcal{C}$, et que dans ce dernier cas $n = 1$.*

En déduire une procédure de recherche automatique de preuve pour la logique classique, en généralisant les règles $(\rightarrow^+ I)$ et (BC).

Exercice 20 *En utilisant les exercices 12 et 19, montrer qu'on peut encore restreindre l'usage de la règle $(\rightarrow^+ I)$ dans la procédure de l'exercice 19 au cas où F est un type de base, comme dans l'exercice 12. À quoi sert la règle $(\eta\mathcal{C})$ dans ce cadre ?*

Exercice 21 *En logique classique, on peut définir le "et" par: $F_1 \wedge F_2 \hat{=} (F_1 \Rightarrow F_2 \Rightarrow \perp) \Rightarrow \perp$. Vérifier cette identité sur les tables de vérité des deux côtés. On pose alors:*

$$\begin{array}{rcl} \langle u, v \rangle & \hat{=} & \lambda k \cdot kuv \\ \pi_1 u & \hat{=} & \mathcal{C}(\lambda k \cdot u(\lambda x, y \cdot kx)) \\ \pi_2 u & \hat{=} & \mathcal{C}(\lambda k \cdot u(\lambda x, y \cdot ky)) \end{array}$$

Montrer que ce codage définit bien les paires et les deux projections, au sens où les règles de typage $(\wedge I)$, $(\wedge E_1)$ *et* $(\wedge E_2)$ *sont vérifiées, et où on a les réductions:*

$$\begin{array}{rcl} \pi_1\langle u,v\rangle & \to^+ & u \\ \pi_2\langle u,v\rangle & \to^+ & v \end{array}$$

Comparer ce résultat avec l'exercice 5.

Exercice 22 *En logique classique, on peut définir le "ou" par:* $F_1 \vee F_2 \hat{=} \neg F_1 \Rightarrow \neg F_2 \Rightarrow \bot$. *Montrer que l'on peut définir* ι_1, ι_2 *et la construction case par:*

$$\begin{array}{rcl} \iota_1 u & \hat{=} & \lambda k_1, k_2 \cdot k_1 u \\ \iota_2 u & \hat{=} & \lambda k_1, k_2 \cdot k_2 u \\ \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} & \hat{=} & \mathcal{C}(\lambda k \cdot u(\lambda x_1 \cdot kv_1)(\lambda x_2 \cdot kv_2)) \end{array}$$

où k, k_1, k_2 *sont des variables fraîches. Montrer que ce codage définit bien les deux injections et l'analyse de cas, au sens où les règles de typage* $(\vee I_1)$, $(\vee I_2)$ *et* $(\vee E)$ *sont vérifiées, et où on a les réductions:*

$$\begin{array}{rcl} \text{case } \iota_1 u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} & \to^+ & v_1[x_1 := u] \\ \text{case } \iota_2 u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} & \to^+ & v_2[x_2 := u] \end{array}$$

Montrer par contre que la conversion commutative:

$$\text{case } \left(\text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} \right) \left\{ \begin{array}{l} \iota_1 y_1 \mapsto w_1 \\ \iota_2 y_2 \mapsto w_2 \end{array} \right\} =_{\lambda\mathcal{C}} \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } v_1 \left\{ \begin{array}{l} \iota_1 y_1 \mapsto w_1 \\ \iota_2 y_2 \mapsto w_2 \end{array} \right\} \\ \iota_2 x_2 \mapsto \text{case } v_2 \left\{ \begin{array}{l} \iota_1 y_1 \mapsto w_1 \\ \iota_2 y_2 \mapsto w_2 \end{array} \right\} \end{array} \right\}$$

n'est pas démontrable. (On admettra que le $\lambda\mathcal{C}$*-calcul est confluent.)*

3 Arithmétique de Peano-Heyting

Si l'on peut noter les preuves de la logique propositionnelle par des λ-termes typés, éventuellement légèrement étendus, on peut faire de même pour des logiques plus expressives, incluant notamment des quantifications ($\forall$, $\exists$) et des principes de récurrence.

3.1 Logique du premier ordre

Nos formules, ou types, seront dans cette section non plus des formules propositionnelles mais des formules *du premier ordre*. Les types de base, ou formules atomiques, ne seront plus nécessairement de simples lettres b, mais de la forme $P(t_1, \ldots, t_n)$, où P est un *symbole de prédicat*, et où t_1, …, t_n sont n termes. Si $n = 0$, on retrouve le cas de la logique propositionnelle, et nous noterons P au lieu de $P()$. Les *termes* servent à dénoter des valeurs sur lesquelles nous voulons raisonner. En arithmétique par exemple, les termes dénoteront des entiers naturels, et notre seul symbole de prédicat sera l'égalité $\approx$ (que nous notons ainsi pour ne pas confondre ce symbole avec la relation d'égalité $=$). Nous écrirons d'ailleurs par confort $t_1 \approx t_2$ plutôt que $\approx (t_1, t_2)$. Si les formules atomiques sont décrites à l'aide de symboles de prédicats, les termes sont décrits à l'aide de variables i, j, k, …, sur lesquelles on pourra quantifier, et de *symboles de fonctions* f, g, h, … En arithmétique par exemple, $+$ sera un symbole de fonction binaire, et on notera $t_1 + t_2$ au lieu de $+(t_1, t_2)$; $\mathtt{0}$ sera un symbole de fonction nullaire (sans argument), c'est-à-dire une *constante*.

En général, un *langage* $\mathcal{L}$ *du premier ordre* est la donnée d'une famille d'ensembles $\mathcal{F}_n$ dits de *symboles de fonctions d'arité* n, et d'une famille d'ensembles $\mathcal{P}_n$ dits de *symboles de prédicats d'arité* n, tous ces ensembles étant disjoints deux à deux. Étant donné un ensemble $\mathcal{X}$ dit de *variables*, l'ensemble $\mathcal{T}(\mathcal{L}, \mathcal{X})$ des *termes* de $\mathcal{L}$ à variables dans $\mathcal{X}$ est le plus petit tel que:

- toute variable dans $\mathcal{X}$ est dans $\mathcal{T}(\mathcal{L}, \mathcal{X})$;
- pour tout $n \in \mathbb{N}$, tout $f \in \mathcal{F}_n$, tous $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{L}, \mathcal{X})$, le terme $f(t_1, \ldots, t_n)$ est dans $\mathcal{T}(\mathcal{L}, \mathcal{X})$ (où $f(t_1, \ldots, t_n)$ est une notation pour le $(n+1)$-uplet $(f, t_1, \ldots, t_n)$).

En d'autres termes, $\mathcal{T}(\mathcal{L}, \mathcal{X})$ est le langage T défini par la grammaire:

$$T ::= \mathcal{X} \mid \mathcal{F}_0() \mid \mathcal{F}_1(T) \mid \mathcal{F}_2(T, T) \mid \ldots$$

Les *formules atomiques* sont les $P(t_1, \ldots, t_n)$, où $P \in \mathcal{P}_n$ et $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{L}, \mathcal{X})$ (ceci représentant encore un $(n+1)$-uplet $(P, t_1, \ldots, t_n)$).

Les *formules* F sont définies par la grammaire suivante, où A désigne n'importe quelle formule atomique:

$$F ::= A \mid \bot \mid F \Rightarrow F \mid \forall \mathcal{X} \cdot F$$

Comme avant, on définit $\neg F$ par $F \Rightarrow \bot$. La variable i dans $\forall i \cdot F$ est liée par $\forall$, et nous supposons que $\forall i \cdot F$ et $\forall j \cdot (F[i := j])$ dénotent la même formule, lorsque j n'est pas libre dans F. On a ici exactement les mêmes difficultés pour définir les variables libres dans une formule ou un terme, l'α-renommage entre formules, et la notion de substitution de termes à des variables dans des termes ou formules, que dans le λ-calcul pur. Nous supposerons donc le problème réglé.

De même que nous avons étendu le langage des types, nous étendons le langage **NJf** des preuves par les règles suivantes, où t est un terme quelconque dans $(\forall E)$:

$$\frac{\Gamma \vdash \forall i \cdot F}{\Gamma \vdash F[i := t]}\ (\forall E) \qquad \qquad \frac{\Gamma \vdash F}{\Gamma \vdash \forall i \cdot F}\ (\forall I)$$

$$(i \text{ non libre dans aucune formule de } \Gamma)$$

On obtient ainsi le système $\mathbf{NJf}_1$ de *logique minimale du premier ordre.*

La condition sur i dans la règle $(\forall I)$ est indispensable. Si on ne l'imposait pas, on pourrait, à partir de $P(i) \vdash P(i)$, déduire $P(i) \vdash \forall i \cdot P(i)$ par $(\forall I)$, c'est-à-dire pour plus de clarté $P(i) \vdash \forall j \cdot P(j)$, et à partir de là:

$$\cfrac{\cfrac{\cfrac{\begin{array}{c}\vdots \\ P(i) \vdash \forall j \cdot P(j)\end{array}}{\vdash P(i) \Rightarrow \forall j \cdot P(j)}\ (\Rightarrow I)}{\vdash \forall i \cdot P(i) \Rightarrow \forall j \cdot P(j)}\ (\forall I)}{\vdash P(\mathsf{0}) \Rightarrow \forall j \cdot P(j)}\ (\forall E)$$

par exemple, ce qui est absurde: ceci signifie que si P est vraie de 0, alors P est vraie de toute valeur, et ceci pour n'importe quelle propriété P.

On veut trouver des constructions étendant le λ-calcul permettant de donner des termes de preuves pour les nouvelles règles. Une façon simple de faire est de considérer qu'une preuve de $\forall i \cdot P(i)$ est une fonction qui à une valeur de i associe une preuve de $P(i)$. Il n'y a donc quasiment pas à étendre le langage: si u est une preuve de $P(i)$, nous prendrons $\lambda i \cdot u$ comme preuve de $\forall i \cdot P(i)$: $(\forall I)$ introduira donc un λ, comme $(\Rightarrow I)$, et de même $(\forall E)$ appliquera un λ-terme à un terme du premier ordre.

Pour ceci, on enrichit le λ-calcul par des termes du premier ordre:

$$\Lambda_1 ::= \mathcal{V} \mid \Lambda_1 \Lambda_1 \mid \lambda \mathcal{V} \cdot \Lambda_1 \mid \nabla \Lambda_1 \mid \Lambda_1 T \mid \lambda \mathcal{X} \cdot \Lambda_1$$

Pour qu'il n'y ait aucune ambiguïté, on supposera qu'aucun terme du premier ordre ne peut être confondu avec un λ-terme ordinaire, en particulier que $\mathcal{X}$ ne contient aucune des variables du λ-calcul ordinaire, dans $\mathcal{V}$. Et les règles supplémentaires de $\mathbf{NJf}_1$ par rapport à **NJf** sont les suivantes, où t est un terme et $i \in \mathcal{X}$:

$$\frac{\Gamma \vdash u : \forall x \cdot F}{\Gamma \vdash ut : F[x := t]}\ (\forall E) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \lambda i \cdot u : \forall i \cdot F}\ (\forall I)$$
$$(i \text{ non libre dans aucune formule de } \Gamma)$$

Les règles de calcul sont celles de **NJf**, soit celles du $\lambda\nabla$-calcul:

$$\begin{array}{lrcl} (\beta) & (\lambda x \cdot u)v & \to & u[x := v] \\ (\beta_1) & (\lambda i \cdot u)t & \to & u[i := t] \\ (\nabla) & \nabla uv & \to & \nabla u \end{array}$$

où dans (β_1) $i \in \mathcal{X}$ et $t \in \mathcal{T}(\mathcal{L}, \mathcal{X})$.

Lemme 7 (Auto-réduction) *Si $\Gamma \vdash u : F$ est dérivable en* $\mathbf{NJf}_1$ *et $u \to^*_{\beta\eta} v$, alors $\Gamma \vdash v : F$ est dérivable en* $\mathbf{NJf}_1$ *aussi.*

Preuve : Comme au lemme 1. Le cas suivant que nous devons traiter est celui d'un (β_1)-redex:

$$\dfrac{\dfrac{\stackrel{\vdots\ \pi_1}{\Gamma \vdash u : F}}{\Gamma \vdash \lambda i \cdot u : \forall i \cdot F}\ (\forall I)}{\Gamma \vdash (\lambda i \cdot u)t : F[i := t]}\ (\forall E)$$

et nous devons montrer que l'on peut transformer la preuve π_1 en une dérivation de $\Gamma \vdash u[i := t] : F[i := t]$, sachant que i n'est libre dans aucune des formules de Γ. Pour ceci, il suffit de montrer le lemme auxiliaire suivant:

Lemme 8 *Si $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$ est dérivable en* $\mathbf{NJf}_1$*, alors $x_1 : F_1[i := t], \ldots, x_n : F_n[i := t] \vdash u[i := t] : F[i := t]$ aussi.*

Preuve : On montre le résultat par récurrence sur la preuve donnée π de $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$. Plus généralement, montrons que l'on peut produire une preuve de $x_1 : F_1[i := t], \ldots, x_n : F_n[i := t] \vdash u[i := t] : F[i := t]$ de même taille que π. Il n'y a aucune difficulté, sauf éventuellement lorsque la preuve se termine par une règle $(\forall I)$. Alors u est de la forme $\lambda j \cdot v$ et F est de la forme $\forall j \cdot G$, et on a dérivé $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$ à partir d'une preuve π_1 de $x_1 : F_1, \ldots, x_n : F_n \vdash v : G$ plus courte que π, où j n'est pas libre dans $F_1, \ldots, F_n$. Notons que j peut être libre dans t. Soit k une variable qui n'est ni libre dans $F_1, \ldots, F_n$ ni dans t. Par hypothèse de récurrence, on produit une preuve π_1' de $x_1 : F_1[j := k], \ldots, x_n : F_n[j := k] \vdash v[j := k] : G[j := k]$, c'est-à-dire de $x_1 : F_1, \ldots, x_n : F_n \vdash v[j := k] : G[j := k]$. Comme π_1' est de même taille que π_1, on peut encore appliquer l'hypothèse de récurrence et produire une preuve π_1'' de $x_1 : F_1[i := t], \ldots, x_n : F_n[i := t] \vdash v[j := k][i := t] : F[j := k][i := t]$. Comme k n'est pas libre dans $F_1[i := t], \ldots, F_n[i := t]$ par construction (au contraire de j qui pourrait être libre dans t), on en déduit une preuve π' de $x_1 : F_1[i := t], \ldots, x_n : F_n[i := t] \vdash \lambda k \cdot (v[j := k][i := t]) : \forall k \cdot (F[j := k][i := t])$ par $(\forall I)$, qui est de même taille que π. Il ne reste qu'à observer que, à α-renommage près, $\lambda k \cdot (v[j := k][i := t])$ est $(\lambda j \cdot v)[i := t]$ et que $\forall k \cdot (F[j := k][i := t])$ est $(\forall j \cdot F)[i := t]$. ♢

♢

Théorème 4 *Tout $\lambda\nabla$-terme typable en* $\mathbf{NJf}_1$ *est fortement normalisant.*

Preuve : C'est la même preuve qu'au théorème 2. Il suffit de modifier la définition des termes réductibles de type F du théorème 1. Pour ceci, on définit RED_ι comme étant l'ensemble des termes fortement normalisants du premier ordre—c'est-à-dire de tous les termes du premier ordre. D'autre part, $RED_{\forall i \cdot F}$ est défini comme l'ensemble des λ-termes u tels que pour tout $t \in RED_\iota$, $ut \in RED_{F[i:=t]}$. Cette définition de RED_F est toujours une définition valide, cette fois-ci non pas sur la structure de F, mais sur la structure du *squelette* $Sk(F)$ de F. On définit ce squelette par: $Sk(A) \hat{=} *$ pour toute formule atomique A, où $*$ est un symbole fixé, $Sk(F \Rightarrow G) \hat{=} Sk(F) \Rightarrow Sk(G)$, $Sk(\forall i \cdot F) \hat{=} \forall i \cdot Sk(F)$. Il est nécessaire de passer par le squelette: dans la définition de $RED_{\forall i \cdot F}$ en fonction de $RED_{F[i:=t]}$, $F[i := t]$ n'est pas en général une sous-expression de $\forall i \cdot F$; par contre, $Sk(F[i := t]) = Sk(F)$ est une sous-expression stricte de $Sk(\forall i \cdot F) = \forall i \cdot Sk(F)$, ce qui justifie la récurrence utilisée dans la définition.

La preuve du théorème se déroule ensuite comme au théorème 1, modifié comme au théorème 2, sans différence essentielle.

Mais il y a une façon plus simple d'établir le même résultat, qui consiste à effacer toute information du premier ordre dans les types et les λ-termes. Le slogan ici est que, du point de vue de la réduction des preuves, tout ce qui est terme et quantification du premier ordre ne sert à rien.

La fonction d'effacement sur les formules est définie par:

$$E(P(t_1, \ldots, t_n)) \hat{=} P \quad E(F_1 \Rightarrow F_2) \hat{=} E(F_1) \Rightarrow E(F_2) \quad E(\forall i \cdot F) \hat{=} E(F) \quad E(\bot) \hat{=} \bot$$

et fournit donc des types simples à partir de formules, où les types de base sont simplement les symboles de prédicats. Sur les termes, on définit:

$$\begin{array}{ccc} E(x) \hat{=} x & E(uv) \hat{=} E(u)E(v) & E(\lambda x \cdot u) \hat{=} \lambda x \cdot E(u) \\ & E(ut) \hat{=} E(u) & E(\lambda i \cdot u) \hat{=} E(u) \end{array}$$

où t est un terme du premier ordre. Il est facile de voir que si $\Gamma \vdash u : F$ est dérivable en $\mathbf{NJf}_1$, alors $E(\Gamma) \vdash E(u) : E(F)$ est dérivable en $\mathbf{NJf}$, où $E(\Gamma)$ est défini par $E(x_1 : F_1, \ldots, x_n : F_n) \hat{=} x_1 : E(F_1), \ldots, x_n : E(F_n)$. D'autre part, si $u \to v$ par (β) ou (∇), alors $E(u) \to E(v)$ par (β) ou (∇) respectivement; et si $u \to v$ par (β_1), alors $E(u) = E(v)$. Donc s'il existe une réécriture infinie partant d'un terme u typé en $\mathbf{NJf}_1$, alors comme toute réduction partant de $E(u)$ est finie par le théorème 2, c'est qu'il existe un réduit v de u à partir duquel on peut mener une réduction infinie en n'utilisant que la règle (β_1). Mais cette règle fait diminuer de 1 le nombre d'applications de λ-termes à des termes du premier ordre, et ne peut donc être appliquée qu'un nombre fini de fois à la suite. D'où le théorème. $\diamondsuit$

Corollaire 4 *Le système* $\mathbf{NJf}_1$ *est cohérent.*

Preuve : Exactement comme au Corollaire 3. $\diamondsuit$

Il n'y a donc pas de contradiction en logique intuitionniste du premier ordre.

Exercice 23 *La logique intuitionniste du premier ordre inclut en fait des constructions pour le $\wedge$, le $\vee$ et le quantificateur $\exists$. Les deux premiers sont traités comme en section 1.2. Le quantificateur existentiel a les règles de déduction suivantes:*

$$\frac{\Gamma \vdash u : \exists i \cdot F \quad \Gamma, x : F \vdash v : G}{\Gamma \vdash \text{case } u \, \{\iota i x \mapsto v\} : G} \, (\exists E) \qquad \frac{\Gamma \vdash u : F[i := t]}{\Gamma \vdash \iota t u : \exists i \cdot F} \, (\exists I)$$

(i non libre dans G et dans aucune formule de Γ)

Les notations ι et case *sont censées rappeler les constructions de preuves pour les règles du $\vee$, mais observer que $\iota t u$ est essentiellement un couple formé d'un terme t et d'une preuve u de $F[i := t]$, et que* case $w \, \{\iota i x \mapsto v\}$ *prend un w qui doit être un tel couple, récupère le terme t dans i, la preuve u dans x et ensuite évalue v. La règle de réduction correspondante est:*

$$(\exists\text{case}) \quad \text{case } \iota t u \, \{\iota i x \mapsto v\} \to v[i := t][x := u]$$

Soit $\mathbf{NJf}_1^\exists$ *le système obtenu à partir de* $\mathbf{NJf}_1$ *en ajoutant les règles de déduction $(\exists E)$ et $(\exists I)$. Les règles de réduction de* $\mathbf{NJf}_1^\exists$ *sont (β), (β_1), (∇) et* $(\exists\text{case})$. *Montrer que cette notion de réduction est fortement normalisante sur les termes bien typés. (Étendre la technique d'effacement du théorème 4. Pourquoi la technique par réductbilité est-elle plus difficile à adapter ?) En déduire que* $\mathbf{NJf}_1^\exists$ *est cohérente.*

Exercice 24 *On définit le système de déduction* $\mathbf{ND}_1$ *à partir de* $\mathbf{ND}$ *comme on a défini* $\mathbf{NJf}_1$ *à partir de* $\mathbf{NJf}$*: les règles de typage sont* (Ax)*,* $(\Rightarrow E)$*,* $(\Rightarrow I)$*,* $(\neg\neg E)$ *ainsi que* $(\forall I)$ *et* $(\forall E)$*. Le* λ*-calcul que nous utilisons est défini par la grammaire:*

$$\Lambda_1^c ::= \mathcal{V} \mid \Lambda_1^c\Lambda_1^c \mid \lambda\mathcal{V}\cdot\Lambda_1^c \mid \mathcal{C}\Lambda_1^c \mid \Lambda_1^c T \mid \lambda\mathcal{X}\cdot\Lambda_1^c$$

où T *est* $\mathcal{T}(\mathcal{L},\mathcal{X})$*. Les règles de réduction sont* (β)*,* $(\mathcal{C}_L)$*,* $(\eta\mathcal{C})$ *et* (β_1)*. Montrer, en s'inspirant des résultats déjà démontrés, l'auto-réduction et la normalisation forte de ce calcul. En déduire que la logique classique (implicationnelle) du premier ordre est cohérente.*

Un point qui se dégage ici est que le quantificateur universel se comporte presque exactement comme l'implication. Ceci peut être formalisé en utilisant des *types dépendants*. Ceci a aussi le mérite de rendre plus esthétique le système de types de $\mathbf{ND}_1$. On aboutit ainsi à une logique qui s'appelle LF, dans laquelle le constructeur fondamental de types n'est plus l'implication $\Rightarrow$ mais le *produit dépendant* $\Pi x : F_1 \cdot F_2$. Les termes du premier ordre sont codés comme des λ-termes d'un type spécial que l'on notera ι. Lorsque $F_1 = \iota$, $\Pi x : \iota \cdot F$ représente $\forall x \cdot F$, et lorsque F_1 est une formule et x n'est pas libre dans F_2, $\Pi x : F_1 \cdot F_2$ représente $F_1 \Rightarrow F_2$. Consulter [HHP87] pour plus de renseignements sur LF.

3.2 Arithmétique de Heyting

L'arithmétique de Peano est la façon usuelle de formaliser l'ensemble des entiers naturels $\mathbb{N}$ et ses propriétés. Il s'agit d'une théorie du premier ordre, dont les symboles de fonction sont $\mathtt{0}$ (constante, représentant 0), $\mathtt{S}$ (fonction unaire, dénotant la fonction qui à n associe $n+1$), $+$ et $*$ (fonctions binaires dénotant l'addition et la multiplication respectivement), et dont le seul symbole de prédicat est $\approx$, qui est binaire. Le type ι dénote l'ensemble des entiers naturels, et les axiomes de Peano sont:

1. $\forall i \cdot i \approx i$ (réflexivité de l'égalité);
2. $\forall i \cdot \forall j \cdot i \approx j \Rightarrow F \Rightarrow F[i := j]$ (remplacement des équivalents: on peut toujours remplacer i par un j égal à i dans toute formule F);
3. $\forall i \cdot \neg\mathtt{0} \approx \mathtt{S}(i)$;
4. $\forall i \cdot \forall j \cdot \mathtt{S}(i) \approx \mathtt{S}(j) \Rightarrow i \approx j$;
5. $\forall i \cdot i{+}\mathtt{0} \approx i$ (définition de l'addition, 1);
6. $\forall i \cdot \forall j \cdot i{+}\mathtt{S}(j) \approx \mathtt{S}(i{+}j)$ (définition de l'addition, 2);
7. $\forall i \cdot i{*}\mathtt{0} \approx \mathtt{0}$ (définition de la multiplication, 1);
8. $\forall i \cdot i{*}\mathtt{S}(j) \approx i{*}j{+}i$ (définition de la multiplication, 2);

plus le *schéma de récurrence*:

$$F[i := \mathtt{0}] \Rightarrow (\forall j \cdot F[i := j] \Rightarrow F[i := \mathtt{S}(j)]) \Rightarrow \forall i \cdot F$$

qui exprime qu'une propriété $F(i)$ est vraie pour tout i dès qu'elle est vraie en $i = 0$ et que $F(i)$ implique $F(i+1)$.

La version intuitionniste de cette théorie s'appelle l'arithmétique de Heyting, et a les mêmes axiomes qui ci-dessus, mais est codée comme une extension de $\mathbf{NJf}_1$ plutôt que $\mathbf{ND}_1$. Nous étudions l'arithmétique de Heyting parce que les transformations de preuve seront un peu plus simples que dans l'arithmétique de Peano.

La principale difficulté ici est de trouver une nouvelle construction de λ-termes interprétant le schéma de récurrence. Intuitivement, il s'agit d'un terme R de type le schéma de récurrence lui-même. Si u est un terme de type $F[i := \mathtt{0}]$, si v est un terme de type $\forall j \cdot F[i := j] \Rightarrow F[i := \mathtt{S}(j)]$, c'est-à-dire une fonction qui prend un entier j en argument, un autre argument de type $F[i := j]$ et retourne un terme de type $F[i := \mathtt{S}(j)]$, alors les règles de réduction suivantes semblent raisonnables, parce qu'elles satisfont la propriété d'auto-réduction, autrement ce sont des transformations de preuves correctes. D'abord, $Ruv\mathtt{0} \to u$; ensuite, $Ruv(\mathtt{S}(j)) \to v(\mathtt{S}(j))(Ruvj)$.

Ce qui est intéressant, d'un point de vue calculatoire, c'est que Ruv définit une fonction de type $\forall i \cdot F$, des entiers i vers des preuves de F, cette fonction g étant définie par les règles: $g(\mathtt{0}) \to u$, $g(\mathtt{S}(j)) \to v(\mathtt{S}(j))(g(j))$. La fonction g est définie par cas: son argument est soit 0 soit un successeur. De plus, la définition de g est récursive, mais il n'y a qu'un appel récursif à g, et $g(j + 1)$ est définie en fonction de $g(j)$ uniquement. Une telle définition s'appelle une *définition par récurrence primitive*, et g est une *fonction primitive récursive*.

Nous allons aussi ajouter un terme r_0 de type $\mathtt{0} \approx \mathtt{0}$, pour modéliser en partie l'axiome 1, mais il est beaucoup plus naturel de ne pas ajouter de termes pour représenter les autres axiomes de Peano. À la place, on va définir une relation de simplification des formules de l'arithmétique par:

$$
\begin{array}{rrcl}
(\mathtt{0} \approx \mathtt{S}) & \mathtt{0} \approx \mathtt{S}(t) & \to_{\mathbb{N}} & \bot \\
(\mathtt{S} \approx \mathtt{S}) & \mathtt{S}(s) \approx \mathtt{S}(t) & \to_{\mathbb{N}} & s \approx t \\
(+\mathtt{0}) & s+\mathtt{0} & \to_{\mathbb{N}} & s \\
(+\mathtt{S}) & s+\mathtt{S}(t) & \to_{\mathbb{N}} & \mathtt{S}(s+t) \\
(*\mathtt{0}) & s*\mathtt{0} & \to_{\mathbb{N}} & \mathtt{0} \\
(*\mathtt{S}) & s*\mathtt{S}(t) & \to_{\mathbb{N}} & s*t+s
\end{array}
$$

Ceci est l'approche naturelle en Coq [BBC+99], et est aussi une instance de *déduction modulo* [DHK98].

Notons $\leftrightarrow^*_{\mathbb{N}}$ la plus petite relation réflexive symétrique et transitive contenant $\to_{\mathbb{N}}$: ceci va nous permettre de considérer deux formules en relation par la relation d'équivalence $\leftrightarrow^*_{\mathbb{N}}$ comme interchangeables.

Les termes de preuves de l'arithmétique de Heyting du premier ordre sont les $\lambda\nabla R$-termes, définis par la grammaire suivante, où T est $\mathcal{T}(\mathcal{L}, \mathcal{X})$ et $\mathcal{L}$ est le langage contenant les symboles $\approx$, $\mathtt{0}$, $\mathtt{S}$, $+$ et $*$ comme décrit plus haut:

$$\Lambda_{\mathbb{N},1} ::= \mathcal{V} \mid \Lambda_{\mathbb{N},1}\Lambda_{\mathbb{N},1} \mid \lambda\mathcal{V} \cdot \Lambda_{\mathbb{N},1} \mid \nabla\Lambda_{\mathbb{N},1} \mid \Lambda_{\mathbb{N},1}T \mid \lambda\mathcal{X} \cdot \Lambda_{\mathbb{N},1} \mid r_0 \mid R\Lambda_{\mathbb{N},1}\Lambda_{\mathbb{N},1}T$$

Les règles de $\mathbf{HA}_1$, l'*arithmétique de Heyting du premier ordre*, sont alors les suivantes:

$$\frac{\Gamma \vdash u : \bot}{\Gamma \vdash \nabla u : F}\ (\bot E) \qquad \frac{}{\Gamma, x : F \vdash x : F}\ (Ax)$$

$$\frac{\Gamma \vdash u : F_1 \Rightarrow F_2 \quad \Gamma \vdash v : F_1}{\Gamma \vdash uv : F_2}\ (\Rightarrow E) \qquad \frac{\Gamma, x : F_1 \vdash u : F_2}{\Gamma \vdash \lambda x \cdot u : F_1 \Rightarrow F_2}\ (\Rightarrow I)$$

$$\frac{\Gamma \vdash u : \forall i \cdot F}{\Gamma \vdash ut : F[i := t]}\ (\forall E) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \lambda i \cdot u : \forall i \cdot F}\ (\forall I)$$

(i non libre dans aucune formule de Γ)

$$\frac{}{\Gamma \vdash r_0 : \mathtt{0} \approx \mathtt{0}}\ (Refl_0) \qquad \frac{\Gamma \vdash u : F_1 \quad F_1 \leftrightarrow^*_{\mathbb{N}} F_2}{\Gamma \vdash u : F_2}\ (\leftrightarrow^*_{\mathbb{N}})$$

$$\frac{\Gamma \vdash u : F[i := \mathtt{0}] \quad \Gamma \vdash v : \forall j \cdot F[i := j] \Rightarrow F[i := \mathtt{S}(j)]}{\Gamma \vdash Ruvt : F[i := t]}\ (Rec)$$

L'originalité de ce système est la règle ($\leftrightarrow^*_{\mathbb{N}}$), qui permet de remplacer une formule par une formule équivalente; pas n'importe quelle formule équivalente, d'ailleurs, juste une qui est *calculatoirement* équivalente, où l'équivalence calculatoire est définie par les règles de calcul $\to_{\mathbb{N}}$. On a ici deux niveaux de calculs: un dans les formules, par $\to_{\mathbb{N}}$, et un dans les preuves, par les règles de réduction de notre $\lambda\nabla R$-calcul. Ces règles sont les suivantes (noter que les quatre dernières sont aussi des règles de $\to_{\mathbb{N}}$):

$$\begin{array}{crcl}
(\beta) & (\lambda x \cdot u)v & \to & u[x := v] \\
(\beta_1) & (\lambda i \cdot u)t & \to & u[i := t] \\
(\nabla) & \nabla uv & \to & \nabla u \\
(R\mathtt{0}) & Ruv\mathtt{0} & \to & u \\
(R\mathtt{S}) & Ruv(\mathtt{S}(t)) & \to & vt(Ruvt) \\
(+\mathtt{0}) & s+\mathtt{0} & \to & s \\
(+\mathtt{S}) & s+\mathtt{S}(t) & \to & \mathtt{S}(s+t) \\
(*\mathtt{0}) & s*\mathtt{0} & \to & \mathtt{0} \\
(*\mathtt{S}) & s*\mathtt{S}(t) & \to & s*t+s
\end{array}$$

L'axiome 1 se démontre à l'aide de la règle ($\mathtt{0} \approx \mathtt{0}$) et du schéma de récurrence (Rec). Intuitivement, la preuve est la suivante. Montrons $s \approx s$ par récurrence sur s. Le cas de base est $\mathtt{0} \approx \mathtt{0}$, ce qui est démontré par la règle ($Refl_0$). Le cas récurrent demande à établir $\mathtt{S}(s) \approx \mathtt{S}(s)$ sous l'hypothèse $s \approx s$. Mais la conclusion $\mathtt{S}(s) \approx \mathtt{S}(s)$ est équivalente à l'hypothèse, à cause de la règle de calcul ($\mathtt{S} \approx \mathtt{S}$). Formellement:

$$\dfrac{\dfrac{}{\vdash r_0 : \mathtt{0} \approx \mathtt{0}}\,(Refl_0) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x : j \approx j \vdash x : j \approx j}\,(Ax)}{x : j \approx j \vdash x : \mathtt{S}(j) \approx \mathtt{S}(j)}\,(\leftrightarrow^*_{\mathbb{N}})}{\vdash \lambda x \cdot x : j \approx j \Rightarrow \mathtt{S}(j) \approx \mathtt{S}(j)}\,(\Rightarrow I)}{\vdash \lambda j \cdot \lambda x \cdot x : \forall j \cdot j \approx j \Rightarrow \mathtt{S}(j) \approx \mathtt{S}(j)}\,(\forall I)}{\vdash R\, r_0\, (\lambda j \cdot \lambda x \cdot x)s : s \approx s}\,(Rec)$$

Le schéma d'axiomes de remplacement des équivalents 2 peut lui aussi entièrement se démontrer dans $\mathbf{HA}_1$, ou plutôt chacune de ses instances peut se démontrer en $\mathbf{HA}_1$ (cf. exercice 27). On dit que ce schéma est *admissible* en $\mathbf{HA}_1$.

Exercice 25 *En supposant qu'il existe un terme de type $\forall i \cdot \forall j \cdot i \approx j \Rightarrow F \Rightarrow F[i := j]$ pour toute formule F de $\mathbf{HA}_1$ (voir l'exercice 27 pour une preuve de cette affirmation), montrer que le système $\mathbf{HA}_1$ permet de démontrer tous les axiomes de Peano 1–8 ainsi que toutes les instances du schéma de récurrence.*

Exercice 26 *Montrer, réciproquement, que si $\Gamma \vdash u : F$ est dérivable en $\mathbf{HA}_1$, alors il existe une preuve en logique du premier ordre de F à partir des hypothèses de Γ plus des axiomes de Peano 1–8 et du schéma de récurrence. Si vous souhaitez être formels, montrer qu'il existe une preuve de $\Gamma, \Delta \vdash v : F$ pour un $\lambda\nabla$-terme v à trouver en fonction de la preuve de $\mathbf{HA}_1$ donnée, et où Δ est une liste d'hypothèses comprenant les axiomes de Peano et un nombre fini d'instances du schéma de récurrence.*

Exercice 27 (☡) *1. Montrer que le terme $\lambda x \cdot x$ est un terme de preuve de $s \approx t \Rightarrow \mathtt{S}(s) \approx \mathtt{S}(t)$.*

2. *Montrer qu'on peut démontrer $s_1 \approx t_1 \Rightarrow s_2 \approx t_2 \Rightarrow s_1{+}t_1 \approx s_2{+}t_2$, par récurrence sur l'entier s_2 puis sur l'entier t_2. (Les courageux montreront que $Ru(\lambda j \cdot \lambda z \cdot v)s_2$ en est un terme de preuve, où $u \hat{=} R(\lambda x{\cdot}\lambda y{\cdot}x)(\lambda t_2{\cdot}\lambda z_1{\cdot}\lambda x{\cdot}\lambda y{\cdot}\nabla y)t_2 : s_1 \approx t_1 {\Rightarrow} \mathtt{0} \approx t_2 {\Rightarrow} s_1 \approx t_1{+}t_2$ et $v \hat{=} R(\lambda x{\cdot}\lambda y{\cdot}\nabla y)(\lambda t_2{\cdot}\lambda z_2{\cdot}z)t_2 : s_1 \approx t_1 \Rightarrow \mathtt{S}(j) \approx t_2 \Rightarrow \mathtt{S}(s_1{+}j) \approx t_1{+}t_2$ dans le contexte $z : s_1 \approx t_1 \Rightarrow s_2 \approx t_2 \Rightarrow s_1{+}s_2 \approx t_1{+}t_2$.)*

3. *Montrer qu'on peut démontrer $s_1 \approx t_1 \Rightarrow s_2 \approx t_2 \Rightarrow s_1{*}t_1 \approx s_2{*}t_2$, par récurrence sur s_2 puis sur t_2, en utilisant le point précédent.*

4. *En déduire que pour tout terme t, on peut montrer $s_1 \approx s_2 \Rightarrow t[i := s_1] \approx t[i := s_2]$. (Appliquer une récurrence structurelle sur t — pas une récurrence (Rec) sur la valeur entière de t, notez bien.)*

5. *Montrer qu'on peut démontrer la symétrie de l'égalité en* $\mathbf{HA}_1$*:* $\forall i \cdot \forall j \cdot i \approx j \Rightarrow j \approx i$*. (Appliquer une récurrence sur* i*, et dans chaque cas appliquer une récurrence sur* j *et simplifier par* $\to_{\mathbb{N}}$*. Bonus si vous arrivez à montrer qu'un terme de preuve possible est* $\lambda i \cdot R(\lambda j \cdot R\ id_0(\lambda i \cdot \lambda x \cdot id_\perp)j)(\lambda i \cdot \lambda y \cdot R\ id_\perp(\lambda k \cdot \lambda z \cdot yk)j)i$*, où* id_0 *est le terme* $\lambda x_0 \cdot x_0$ *de type* $\mathtt{0} \approx \mathtt{0} \Rightarrow \mathtt{0} \approx \mathtt{0}$ *et* $id_\perp$ *est le terme* $\lambda x_1 \cdot x_1$ *de type* $\perp \Rightarrow \perp$*.)*

6. *Montrer qu'on peut démontrer la transitivité de l'égalité en* $\mathbf{HA}_1$*:* $\forall i \cdot \forall j \cdot \forall j \cdot i \approx j \Rightarrow j \approx k \Rightarrow i \approx k$*. (Indication: effectuer des récurrences sur* i*,* j*,* k *imbriquées dans cet ordre. Bonus si vous arrivez à fournir un terme de preuve explicite !)*

7. *Déduire des points 4, 5 et 6 que l'on peut montrer* $s_1 \approx t_1 \Rightarrow F[i := s_1] \Rightarrow F[i := t_1]$ *pour toute formule* F *qui est une égalité* $s \approx t$*.*

8. *En déduire que l'on peut montrer* $s_1 \approx t_1 \Rightarrow F[i := s_1] \Rightarrow F[i := t_1]$ *pour toute formule* F*, par récurrence structurelle sur* F*. En conséquence, toute instance du principe de remplacement des équivalents est démontrable en* $\mathbf{HA}_1$*.*

À cause de la règle $(\leftrightarrow^*_{\mathbb{N}})$, la propriété d'auto-réduction est plus compliquée à démontrer qu'avant. Le lemme d'affaiblissement (cf. lemme 2) est toujours aussi facile. Par contre, le lemme suivant est plus difficile:

Lemme 9 *Si* $\Gamma, x : F_1 \vdash u : F_2$ *et* $\Gamma \vdash v : F_1$ *sont dérivables en* $\mathbf{HA}_1$*, alors* $\Gamma \vdash u[x := v] : F_2$ *aussi.*

Preuve : Plus généralement, soit π_2 une preuve de $\Gamma, x : F_1, \Delta \vdash u : F_2$ et π_1 une de $\Gamma \vdash v : F_1$. On construit une preuve π de $\Gamma, \Delta \vdash u[x := v] : F_2$ par récurrence structurelle sur π_2.

Si π_2 se termine par la règle (Ax), alors soit u est x, donc $F_1 = F_2$, et on prend pour π un affaiblissement idoine de π_1, soit u est une autre variable y et π est la preuve qui produit $\Gamma, \Delta \vdash y : F_2$ par (Ax). Si π_2 se termine par $(\Rightarrow E)$ ou $(\Rightarrow I)$, c'est comme au lemme 1, cas (App) et (Abs). Les cas où π_2 se terminent par $(\perp E)$, $(\forall E)$, $(\forall I)$ sont faciles. Si π_2 se termine par $(Refl_0)$, alors $u = r_0$ et $u[x := v]$ est encore r_0: on définit π comme la preuve qui déduit $\Gamma, \Delta \vdash r_0 : \mathtt{0} \approx \mathtt{0}$ par $(Refl_0)$.

Si π_2 se termine par $(\leftrightarrow^*_{\mathbb{N}})$, alors on a dérivé $\Gamma, x : F_1, \Delta \vdash u : F_2$ à partir d'une preuve plus courte de $\Gamma, x : F_1, \Delta \vdash u : F_2'$, avec $F_2 \leftrightarrow^*_{\mathbb{N}} F_2'$. Par hypothèse de récurrence, on a une preuve de $\Gamma, \Delta \vdash u[x := v] : F_2'$, d'où l'on déduit une de $\Gamma, \Delta \vdash u[x := v] : F_2$ par $(\leftrightarrow^*_{\mathbb{N}})$.

Si π_2 se termine par (Rec), alors on a dérivé $\Gamma, x : F_1, \Delta \vdash u : G[i := t]$, avec $F_2 = G[i := t]$ et $u = Ru'v't$, à partir de preuves plus courtes de $\Gamma, x : F_1, \Delta \vdash u' : G[i := \mathtt{0}]$ et de $\Gamma, x : F_1, \Delta \vdash v' : \forall j \cdot G[i := j] \Rightarrow G[i := \mathtt{S}(j)]$. Par hypothèse de récurrence, on peut déduire $\Gamma, \Delta \vdash u'[x := v] : G[i := \mathtt{0}]$ et $\Gamma, \Delta \vdash v'[x := v] : \forall j \cdot G[i := j] \Rightarrow G[i := \mathtt{S}(j)]$, donc $\Gamma, \Delta \vdash u : G[i := t]$. ◇

L'auto-réduction exprime alors que les règles de réduction de $\mathbf{HA}_1$ sont bien des règles de transformations de preuves de séquents en des preuves des mêmes séquents:

Théorème 5 (Auto-réduction) *Si* $\Gamma \vdash u : F$ *est dérivable en* $\mathbf{HA}_1$ *et* $u \to v$*, alors* $\Gamma \vdash v : F$ *est dérivable en* $\mathbf{HA}_1$*.*

Preuve : Le cas de la règle (β) est par le lemme 9. Celui de la règle (β_1) est comme au lemme 8. Le cas de (∇) est évident. La règle $(R\mathtt{0})$ correspond à réécrire la preuve:

$$\dfrac{\begin{array}{c}\vdots\,\pi\\ \Gamma \vdash u : G[i := \mathtt{0}]\end{array} \quad \begin{array}{c}\vdots\\ \Gamma \vdash v : \forall j \cdot G[i := j] \Rightarrow G[i := \mathtt{S}(j)]\end{array}}{\Gamma \vdash Ruv\mathtt{0} : G[i := \mathtt{0}]}\ (Rec)$$

en π, et $(R\mathtt{S})$ correspond à réécrire la preuve:

$$\dfrac{\dfrac{\vdots\ \pi_1}{\Gamma \vdash u : G[i := \mathtt{0}]} \quad \dfrac{\vdots\ \pi_2}{\Gamma \vdash v : \forall j \cdot G[i := j] \Rightarrow G[i := \mathtt{S}(j)]}}{\Gamma \vdash Ruv(\mathtt{S}(t)) : G[i := \mathtt{S}(t)]}\,(Rec)$$

en:

$$\dfrac{\dfrac{\dfrac{\vdots\ \pi_2}{\Gamma \vdash v : \forall j \cdot G[i := j] \Rightarrow G[i := \mathtt{S}(j)]}}{\Gamma \vdash vt : G[i := t] \Rightarrow G[i := \mathtt{S}(t)]}\,(\forall E) \quad \dfrac{\dfrac{\vdots\ \pi_1}{\Gamma \vdash u : G[i := \mathtt{0}]} \quad \dfrac{\vdots\ \pi_2}{\Gamma \vdash v : \forall j \cdot G[i := j] \Rightarrow G[i := \mathtt{S}(j)]}}{\Gamma \vdash Ruvt : G[i := t]}\,(Rec)}{\Gamma \vdash vt(Ruvt) : G[i := \mathtt{S}(t)]}\,(\Rightarrow E)$$

$\diamondsuit$

Théorème 6 *Tout $\lambda\nabla R$-terme typé en $\mathbf{HA}_1$ est fortement normalisant.*

Preuve : Montrons d'abord que tout terme t du premier ordre termine. Interprétons les termes t dans les entiers par: $[\mathtt{0}]\hat{=}1$, $[\mathtt{S}(t)]\hat{=}[t]+1$, $[s+t]\hat{=}[s]+2[t]$, $[s*t]\hat{=}[s](3[t]+1)$. Nous montrons que: $(*)$ si $t_1 \to t_2$, alors $[t_1] > [t_2]$; ceci impliquera qu'il ne peut exister aucun réduction infinie à partir de t_1. On montre $(*)$ par récurrence sur la profondeur du rédex contracté dans t_1. Si t_1 est lui-même le rédex, remarquons que $[s+\mathtt{0}] = [s]+2 > [s]$, $[s+\mathtt{S}(t)] = [s]+2[t]+2 > [s]+2[t]+1 = [\mathtt{S}(s+t)]$, $[s*\mathtt{0}] = 4[s] > 1 = [\mathtt{0}]$ (car $[s] \geq 1$ pour tout s, comme une récurrence facile sur s le montre), et $[s*\mathtt{S}(t)] = [s](3[t]+4) > [s](3[t]+3) = [s](3[t]+1)+2[s] = [s*t+s]$. Si t_1 n'est pas lui-même le rédex, alors c'est par l'hypothèse de récurrence. Par exemple, si t_1 est de la forme $\mathtt{S}(t'_1)$, avec $t'_1 \to t'_2$ et $t_2 = \mathtt{S}(t'_2)$, alors $[t_1] = [t'_1]+1 > [t'_2]+1 = [t_2]$.

Un $\lambda\nabla R$-terme est dit *neutre* s'il ne commence pas par λ ou ∇. Soit SN l'ensemble des $\lambda\nabla R$-termes fortement normalisants, $RED_\bot$ et $RED_{s\approx t}$ sont définis comme étant SN, $RED_{F\Rightarrow G}$ est l'ensemble des $\lambda\nabla R$-termes u tels que pour tout $v \in RED_F$, $uv \in RED_G$, et $RED_{\forall i \cdot F}$ est l'ensemble des $\lambda\nabla R$-termes u tels que pour tous termes t du premier ordre (qui terminent), $ut \in RED_{F[i:=t]}$. Ceci est une définition de RED_F par récurrence structurelle sur le squelette $Sk(F)$ de F. Comme d'habitude, montrons:

(CR1) Si $u \in RED_F$, alors $u \in SN$.

(CR2) Si $u \in RED_F$ et $u \to u'$, alors $u' \in RED_F$.

(CR3) Si u est neutre et pour tout u' tel que $u \to u'$, $u' \in RED_F$, alors $u \in RED_F$.

Ceci est par récurrence structurelle sur $Sk(F)$. Si F est de la forme $\bot$ ou $s \approx t$, ces propriétés sont évidentes. Si F est de la forme $G \Rightarrow H$, alors si $u \in RED_F$, pour tout $v \in RED_G$, $uv \in RED_H$, donc par hypothèse de récurrence (CR1), $uv \in SN$; par hypothèse de récurrence (CR3), la variable x de type G est dans RED_G, donc $ux \in SN$; donc $u \in SN$: (CR1) est vraie. Si $u \in RED_F$ et $u \to u'$, par définition pour tout $v \in RED_G$, $uv \in RED_H$ et de plus $uv \to u'v$, donc par hypothèse de récurrence (CR2) $u'v \in RED_H$; comme v est arbitraire, $u' \in RED_H$, établissant (CR2). Si u est neutre et pour tout u' tel que $u \to u'$, $u' \in RED_F$, montrons que pour tout $v \in RED_G$, uv ne se réduit qu'à des termes de RED_H, par récurrence sur $\nu(v)$, la longueur de la plus grande réduction partant de v (qui est bien défini car par hypothèse de récurrence (CR1) $v \in SN$). Mais comme u est neutre, uv ne se réduit qu'à des termes de la forme $u'v$ avec $u \to u'$ (donc $u'v \in RED_H$ car par hypothèse $u' \in RED_F$) ou de la forme uv' avec $\nu(v') < \nu(v)$ (donc $uv' \in RED_H$ par hypothèse de récurrence). Comme uv est lui-même neutre, par hypothèse de récurrence (CR3), $u \in RED_F$, ce qui établit (CR3).

On montre maintenant que:

$$(\clubsuit)\ \text{si } \Gamma, x_1 : F_1, \ldots, x_n : F_n \vdash u : F \text{ et } v_1 \in RED_{F_1}, \ldots, v_n \in RED_{F_n}, \text{ alors}$$
$$u[x_1 := v_1, \ldots, x_n := v_n] \in RED_F$$

par récurrence structurelle sur u. Ceci impliquera le théorème. Si u est une variable, une application ou une abstraction, c'est comme au théorème 1. Si u est de la forme ∇v, alors c'est comme au théorème 2. Le cas intéressant est celui où u est de la forme $Rvwt$, et découle du résultat auxiliaire: ($\circ$) si $v \in RED_{F[i:=\mathtt{0}]}$ et $w \in RED_{\forall j \cdot F[i:=j] \Rightarrow F[i:=\mathtt{S}(j)]}$, alors $Rvwt \in RED_{F[i:=t]}$. Nous montrons ceci par récurrence sur $(\nu(v) + \nu(w) + \nu(t), |t|)$ ordonné lexicographiquement, où ν est la fonction "longueur de la plus grande réduction" et $|t|$ est la taille du terme t (le nombre de symboles dans t); la quantité $\nu(v) + \nu(w) + \nu(t)$ est bien définie sur t car tout terme du premier ordre termine, et sur v et w car par (CR1) v et w sont dans SN. $Rvwt$ est neutre, donc par (CR3) il suffit de montrer que tous les réduits immédiats de $Rvwt$ sont dans $RED_{F[i:=t]}$. Mais $Rvwt$ peut se réduire en $Rv'wt$ avec $v \to v'$ (qui est dans $RED_{F[i:=t]}$ par hypothèse de récurrence), en $Rvw't$ avec $w \to w'$ (idem), en $Rvwt'$ avec $t \to t'$ (idem), ou en $vt'(Rvwt')$ avec $t = \mathtt{S}(t')$: dans ce dernier cas, $\nu(v) + \nu(w) + \nu(t') \leq \nu(v) + \nu(w) + \nu(t)$ et $|t'| < |t|$, donc l'hypothèse de récurrence s'applique et $Rvwt' \in RED_{F[i:=t']}$; par définition de la réductibilité, $vt'(Rvwt')$ est dans $RED_{F[i:=\mathtt{S}(t')]} = RED_{F[i:=t]}$. $\diamondsuit$

Corollaire 5 *L'arithmétique de Heyting du premier ordre* $\mathbf{HA}_1$ *est cohérente.*

Preuve : Supposons que u soit un terme normal tel que $\vdash u : \bot$ soit dérivable. On peut supposer sans perte de généralité que u ne contient aucune variable libre du premier ordre, car si $i_1, \ldots, i_k$ sont ces variables, alors $\vdash u[i_1 := \mathtt{0}, \ldots, i_k := \mathtt{0}] : \bot$ est dérivable, et on peut remplacer u par toute forme normale de $u[i_1 := \mathtt{0}, \ldots, i_k := \mathtt{0}]$, qui n'a aucune variable variable libre du premier ordre.

Supposons donc que u est un terme normal sans variable libre du premier ordre tel que $\vdash u : \bot$ soit dérivable, de taille minimale. Alors u est nécessairement de la forme $hu_1 \ldots u_n$, où h est une variable ou le symbole ∇ (auquel cas $n = 1$), ou de la forme Ru_1u_2t. Le premier cas est impossible car le côté gauche du jugement est vide. Le second cas implique que $\vdash u_1 : \bot$, ce qui contredit la minimalité de u. Dans le dernier cas, t étant normal et sans variable libre, t est de la forme $\mathtt{S}(\ldots(\mathtt{S}(\mathtt{0})))$; c'est par récurrence structurelle sur t: si $t = \mathtt{0}$, c'est clair; si t est de la forme $\mathtt{S}(t_1)$, c'est par hypothèse de récurrence; et t ne peut pas être ni de la forme t_1+t_2 ni de la forme t_1*t_2, car par hypothèse de récurrence t_2 commencerait par $\mathtt{0}$ ou par $\mathtt{S}$, ce qui contredirait l'hypothèse de normalité de t. Mais comme $t = \mathtt{S}(\ldots(\mathtt{S}(\mathtt{0})))$, Ru_1u_2t n'est pas normal, ce qui montre que le dernier cas est impossible lui aussi. $\diamondsuit$

Ceci est le premier résultat de cohérence non complètement trivial que nous avons démontré. Il peut apparaître trivial, cependant, et en voici une démonstration élémentaire. Pour toute fonction ρ de $\mathcal{X}$ vers $\mathbb{N}$ (une *valuation* du premier ordre), pour tout terme t, définissons $[\![t]\!]\rho$ par: $[\![\mathtt{0}]\!]\rho \hat{=} 0$, $[\![\mathtt{S}(t)]\!]\rho \hat{=} [\![t]\!]\rho + 1$, $[\![s+t]\!]\rho \hat{=} [\![s]\!]\rho + [\![t]\!]\rho$, $[\![s*t]\!]\rho \hat{=} [\![s]\!]\rho \times [\![t]\!]\rho$. Définissons la relation $\rho \models F$ ("F est vraie dans la valuation ρ") par: $\rho \models \bot$ est faux, $\rho \models s \approx t$ si et seulement si $[\![s]\!]\rho = [\![t]\!]\rho$, $\rho \models F \Rightarrow G$ si et seulement si $\rho \models F$ implique $\rho \models G$, et $\rho \models \forall i \cdot F$ si et seulement si $\rho[i := n] \models F$ pour tout $n \in \mathbb{N}$, où $\rho[i := n]$ envoie i vers n et toute autre variable j vers $\rho(j)$. Il est facile de voir que pour toute preuve de $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$, pour toute valuation ρ, $\rho \models F_1$ et $\ldots$ et $\rho \models F_n$ impliquent $\rho \models F$. $\mathbf{HA}_1$ est alors cohérent parce que par exemple $\vdash u : \bot$ n'est prouvable pour aucun $\lambda\nabla R$-terme u, sinon $\rho \models \bot$ serait vrai.

Cette démonstration sémantique est beaucoup plus simple que la démonstration syntaxique par normalisation des $\lambda\nabla R$-termes typés, mais en fait elle ne dit pas grand-chose: pour démontrer que pour toute preuve de $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$, pour toute valuation ρ, $\rho \models F_1$ et $\ldots$ et $\rho \models F_n$ impliquent $\rho \models F$, nous utilisons les règles usuelles de l'arithmétique. Autrement dit, on utilise essentiellement les mêmes règles que celles de $\mathbf{HA}_1$ pour montrer que les règles de $\mathbf{HA}_1$ sont correctes: c'est un cercle vicieux.

En général, on peut démontrer la cohérence de n'importe quelle théorie par ce moyen, même de théories incohérentes. Par exemple, considérons un langage de termes dans lequel on peut écrire $\{x \mid F(x)\}$ pour toute formule $F(x)$, dont la sémantique est l'ensemble de toutes les valeurs pour x qui rendent $F(x)$ vraie. On adopte une règle de déduction qui exprime que $t \in \{x \mid F(x)\}$ si et seulement $F(t)$ est vraie. Cette règle est valide, et par le même argument sémantique que ci-dessus, ce système est cohérent. Mais on verra au lemme 14 que cette théorie — la *théorie naïve des ensembles* — est en réalité incohérente. La raison pour laquelle la démonstration sémantique est incorrecte ici, c'est qu'elle suppose que l'on peut toujours définir l'ensemble de toutes les valeurs vérifiant une formule donnée (voir la définition de la sémantique de $\{x \mid F(x)\}$). Le fait que la théorie naïve des ensembles soit incohérente montre que, justement, ceci n'a pas de sens.

C'est pour éviter ce cercle vicieux, dans lequel finalement on ramène la cohérence d'une théorie à elle-même, que David Hilbert a proposé au début du vingtième siècle d'établir la cohérence des systèmes logiques utilisés en mathématiques par des moyens *finitistes*, c'est-à-dire n'utilisant des moyens de raisonnement qui ne portent que sur des objets finis: entiers, arbres finis (donc aussi formules, preuves), mais pas les ensembles quelconques d'entiers ou les arbres infinis, par exemple; et des principes de récurrence structurelle ou sur les entiers, mais pas de quantification sur l'ensemble (infini) de tous les entiers ou de tous les arbres finis par exemple. (Le système logique correspondant s'appelle l'arithmétique primitive récursive **PRA** et est beaucoup moins expressif, donc aussi beaucoup moins suspect d'incohérence, que $\mathbf{PA}_1$.)

La preuve de cohérence de $\mathbf{HA}_1$ que nous avons donnée est alors presque finitiste: les notions de formules et de preuves sont finitistes, de même que les transformations de preuves définies par les règles de réduction du $\lambda\nabla R$-calcul et que l'argument du corollaire 5 montrant qu'il n'y a pas de $\lambda\nabla R$-calcul clos de type $\perp$. Le seul argument non finitiste dans la preuve est l'argument de normalisation du théorème 6, où la définition par récurrence structurelle sur F du prédicat $u \in RED_F$ repose sur des quantifications universelles sur des domaines infinis: $u \in RED_{F \Rightarrow G}$ si et seulement si, *pour tout* $v \in RED_F$, $uv \in RED_G$. (Le fait que l'on raisonne sur des ensembles RED_F n'est pas non plus finitiste, mais comme la démonstration n'utilise RED_F que comme un prédicat $u \in RED_F$, ceci n'est pas une entorse grave au finitisme.)

En un sens, on ne peut pas faire mieux, et donc on ne peut pas échapper au cercle vicieux mentionné plus haut: le second théorème d'incomplétude de Gödel peut en effet être invoqué pour prouver qu'on ne peut montrer la cohérence de $\mathbf{HA}_1$ que dans un système logique au moins aussi fort que $\mathbf{HA}_1$ lui-même. (Nous ne décrirons pas formellement ce théorème; informellement, il énonce qu'on ne peut pas montrer la cohérence d'une théorie mathématique T dans T elle-même dès que T est cohérente, récursivement axiomatisable et contient l'arithmétique de Peano du premier ordre — ou de Heyting, ce qui ne change rien.)

Toutefois, la démonstration du corollaire 5 montre que que la seule partie non finitiste, et ce donc d'une façon essentielle, est l'argument de normalisation du $\lambda\nabla R$-calcul typé. En résumé, on a réduit la question de la cohérence de $\mathbf{HA}_1$ à la question de la seule terminaison d'une classe de programmes informatiques.

Exercice 28 () *L'arithmétique de Heyting du premier ordre inclut en fait aussi le $\wedge$, le $\vee$ et le quantificateur $\exists$. Considérons le système $\mathbf{HA}_1^{\exists}$, obtenu à partir de $\mathbf{HA}_1$ en ajoutant les constructions de l'exercice 23. Montrer que le $\lambda\nabla R$-calcul enrichi par les constructions ιtu et* case $u\,\{\iota ix \mapsto v\}$ *avec les règles de typage correspondantes est fortement normalisant. (Adapter la technique de réductibilité. Pourquoi la technique d'effacement ne fonctionne-t-elle pas ? Montrer cependant que la technique d'effacement permet de montrer que $\mathbf{HA}_1^{\exists}$ normalise* faiblement*: toute stratégie qui normalise d'abord par (β), (β_1), (∇), $(R\mathtt{0})$, $(R\mathtt{S})$,* ($\exists$case)*, puis normalise le terme obtenu par $\to_{\mathbb{N}}$ aboutit à une forme normale pour les règles de $\mathbf{HA}_1^{\exists}$. Voir aussi l'exercice 30 et l'exercice 37.) En conclure que $\mathbf{HA}_1^{\exists}$ est cohérent.*

Exercice 29 *L'arithmétique de Peano du premier ordre $\mathbf{PA}_1$ est à $\mathbf{HA}_1$ ce que $\mathbf{ND}_1$ est à $\mathbf{NJf}_1$, à savoir sa version classique. (Voir exercice 24.) Plus précisément, les règles de typage de $\mathbf{PA}_1$ sont (Ax), $(\Rightarrow E)$, $(\Rightarrow I)$, $(\neg\neg E)$ ainsi que $(\forall I)$ et $(\forall E)$, $(Refl_0)$, $(\leftrightarrow^*_{\mathbb{N}})$ et (Rec). Le $\lambda\mathcal{C}R$-calcul qui est le λ-calcul correspondant est défini par la grammaire:*

$$\Lambda^c_{\mathbb{N},1} ::= \mathcal{V} \mid \Lambda^c_{\mathbb{N},1}\Lambda^c_{\mathbb{N},1} \mid \lambda\mathcal{V}\cdot\Lambda^c_{\mathbb{N},1} \mid \mathcal{C}\Lambda^c_{\mathbb{N},1} \mid \Lambda^c_{\mathbb{N},1}T \mid \lambda\mathcal{X}\cdot\Lambda^c_{\mathbb{N},1} \mid r_0 \mid R\Lambda^c_{\mathbb{N},1}\Lambda^c_{\mathbb{N},1}T$$

où T est le langage des termes du premier ordre de $\mathbf{HA}_1$. Les règles de réduction sont celles de $\mathbf{HA}_1$ moins (∇), plus $(\mathcal{C}_L)$ et $(\eta\mathcal{C})$. Montrer, en s'inspirant des résultats déjà démontrés, l'auto-réduction et la normalisation forte de ce calcul. En déduire que $\mathbf{PA}_1$ est cohérente.

Exercice 30 *Montrer qu'on peut définir $\exists i\cdot F$ en $\mathbf{PA}_1$ comme $\neg\forall i\cdot\neg F$. Autrement dit, cette construction obéit aux règles de typage $(\exists E)$ et $(\exists I)$ de l'exercice 23, pour une définition à trouver de ιtu et de* case $u\,\{\iota ix \mapsto v\}$*. Montrer que* case $\iota tu\,\{\iota ix \mapsto v\}$ $\to^+ v[i := t][x := u]$*. En déduire que la notion de réduction de $\mathbf{HA}_1^{\exists}$ de l'exercice 28 normalise fortement.*

4 Logique intuitionniste d'ordre deux et système F

L'arithmétique de Peano du premier ordre $\mathbf{PA}_1$, ou bien celle de Heyting $\mathbf{HA}_1$, est en fait un système logique relativement faible. Il se trouve que, même si de nombreuses vérités arithmétiques sont démontrables en $\mathbf{PA}_1$ ou en $\mathbf{HA}_1$, il en existe de relativement simples qui ne sont pas démontrables en $\mathbf{PA}_1$ [PH78].

4.1 Logique et arithmétique d'ordre deux

Un système beaucoup plus complet est l'*arithmétique de Peano du second ordre* $\mathbf{PA}_2$, et sa contrepartie intuitionniste $\mathbf{HA}_2$. Nous allons nous intéresser à $\mathbf{HA}_2$, qui est défini comme $\mathbf{HA}_1$ à partir de la logique du premier ordre, mais à partir de la logique du second ordre. La nouveauté apportée par la logique du second ordre est que nous disposons maintenant de *variables de prédicats* X, Y, ..., que l'on peut appliquer à des termes du premier ordre, et sur lesquelles on peut quantifier.

Formellement, étant donné un langage du premier ordre $\mathcal{L}$ donné par des familles de symboles de fonctions $\mathcal{F}_n$, $n \geq 0$, et des familles de symboles ou *variables de prédicats* $\mathcal{P}_n$, $n \geq 0$, on définit les *formules du second ordre* par la grammaire:

$$F ::= A \mid \bot \mid F \Rightarrow F \mid \forall \mathcal{X} \cdot F \mid \forall \mathcal{P}_n \cdot F$$

où A parcourt l'ensemble des formules atomiques, et les formules sont vues à α-conversion près. La différence avec les formules du premier ordre est l'ajout de la quantification $\forall P \cdot F$ d'ordre deux. Par commodité, nous supposerons que chaque ensemble $\mathcal{P}_n$ est infini.

L'autre différence avec la logique d'ordre un est la notion de substitution. Pour toute variable du premier ordre i et pour tout terme t du premier ordre, $F[i := t]$ est défini comme d'habitude. Fixons une fois pour toutes une suite de variables k_1, k_2, ..., alors $F[P(k_1, \ldots, k_n) := G]$, où P est une variable de prédicat n-aire et G est une formule est défini par récurrence structurelle sur F par:

$$\begin{array}{rcll}
P(t_1, \ldots, t_n)[P(k_1, \ldots, k_n) := G] & \hat{=} & G[k_1 := t_1, \ldots, k_n := t_n] & \\
Q(t_1, \ldots, t_m)[P(k_1, \ldots, k_n) := G] & \hat{=} & Q(t_1, \ldots, t_m) & (P \neq Q) \\
\bot[P(k_1, \ldots, k_n) := G] & \hat{=} & \bot & \\
(F_1 \Rightarrow F_2)[P(k_1, \ldots, k_n) := G] & \hat{=} & (F_1[P(k_1, \ldots, k_n) := G]) \Rightarrow (F_2[P(k_1, \ldots, k_n) := G]) & \\
(\forall i \cdot F)[P(k_1, \ldots, k_n) := G] & \hat{=} & \forall i \cdot (F[P(k_1, \ldots, k_n) := G]) & (i \text{ non libre dans } G, k_1, \ldots, k_n) \\
(\forall Q \cdot F)[P(k_1, \ldots, k_n) := G] & \hat{=} & \forall Q \cdot (F[P(k_1, \ldots, k_n) := G]) & (Q \neq P, Q \text{ non libre dans } G)
\end{array}$$

où dans la première ligne, côté droit, la substitution est une substitution du premier ordre. Intuitivement, la substitution $[P(k_1, \ldots, k_n) := G]$ consiste à remplacer $P(k_1, \ldots, k_n)$ par G, et donc aussi $P(t_1, \ldots, t_n) = P(k_1, \ldots, k_n)[k_1 := t_1, \ldots, k_n := t_n]$ par $G[k_1 := t_1, \ldots, k_n := t_n]$.

Les règles de déduction de la logique d'ordre deux sont celles de la logique d'ordre un plus:

$$\frac{\Gamma \vdash u : \forall P \cdot F}{\Gamma \vdash uG : F[P(k_1, \ldots, k_n) := G]} (\forall_2 E) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \lambda P \cdot u : \forall P \cdot F} (\forall_2 I)$$

$(P$ non libre dans aucune formule de $\Gamma)$

où $P \in \mathcal{P}_n$. De même, $\mathbf{HA}_2$ est $\mathbf{HA}_1$ plus ces deux règles, et $\mathbf{PA}_2$ est $\mathbf{PA}_1$ plus ces deux règles.

Exercice 31 *Soit X une variable de prédicat 0-aire. Démontrer $(\forall X \cdot X) \Rightarrow \bot$ et $\bot \Rightarrow (\forall X \cdot X)$ en logique d'ordre deux. En déduire qu'on n'a pas besoin de construction spéciale pour le faux en logique d'ordre deux.*

Notre λ-calcul est le même que le $\lambda\nabla R$-calcul, plus les constructions uG (application d'un terme à un type) et $\Lambda P \cdot u$ (abstraction sur une variable de prédicat), et est défini par la grammaire:

$$\Lambda_{\mathbb{N},2} ::= \mathcal{V} \mid \Lambda_{\mathbb{N},2}\Lambda_{\mathbb{N},2} \mid \lambda\mathcal{V} \cdot \Lambda_{\mathbb{N},2} \mid \nabla\Lambda_{\mathbb{N},2} \mid \Lambda_{\mathbb{N},2}T \mid \lambda\mathcal{X} \cdot \Lambda_{\mathbb{N},2} \mid r_0 \mid R\Lambda_{\mathbb{N},2}\Lambda_{\mathbb{N},2}T \mid \Lambda_{\mathbb{N},2}F \mid \lambda\mathcal{P}_n \cdot \Lambda_{\mathbb{N},2}$$

où T est le langage des termes du premier ordre de l'arithmétique, et F est celui des formules du second ordre.

Les règles de réduction sont celles de $\mathbf{HA}_1$, plus:

$$(Beta) \quad (\lambda P \cdot u)F \quad \rightarrow \quad u[P(k_1, \ldots, k_n) := F]$$

où P est n-aire, et la substitution est étendue aux termes de preuve de façon immédiate.

4.2 Système F_2

Les termes et les réductions de $\mathbf{HA}_2$ sont complexes, et nous allons commencer par étudier une restriction drastique, un système de typage du λ-calcul appelé le *système* F_2 et dû à Jean-Yves Girard. Nous verrons plus tard que le système F_2 contient l'essentiel des caractéristiques logiques de $\mathbf{HA}_2$.

On obtient le système F_2 d'une part en supprimant dans les formules tous les termes et toutes les quantifications du premier ordre, en ne gardant que des variables de prédicats 0-aires, et en ne gardant dans le $\lambda\nabla R$-calcul que le λ-calcul plus les applications et abstractions du second ordre sur les variables de prédicats 0-aires — on supprime récurseur, application et abstraction du premier et du second ordre non 0-aire.

Ce qui reste est un langage de types de la forme:

$$F \hat{=} \mathcal{P}_0 \mid F \Rightarrow F \mid \forall \mathcal{P}_0 \cdot F$$

La substitution sur les types est la substitution usuelle (regarder la définition de la substitution d'ordre deux lorsque la variable de prédicat est 0-aire).

Le λ-calcul du système F_2 est défini par:

$$\Lambda_\forall ::= \mathcal{V} \mid \Lambda_\forall \Lambda_\forall \mid \lambda \mathcal{V} \cdot \Lambda_\forall \mid \Lambda_\forall F \mid \lambda \mathcal{P}_0 \cdot \Lambda_\forall$$

où F_2 parcourt l'ensemble des formules ci-dessus; et les règles de typage sont:

$$\frac{}{\Gamma, x : F \vdash x : F} \, (Var)$$

$$\frac{\Gamma \vdash u : F_1 \Rightarrow F_2 \quad \Gamma \vdash v : F_1}{\Gamma \vdash uv : F_2} \, (App) \qquad \frac{\Gamma, y : F_1 \vdash u[x := y] : F_2}{\Gamma \vdash \lambda x \cdot u : F_1 \Rightarrow F_2} \, (Abs)$$

$$\frac{\Gamma \vdash u : \forall P \cdot F}{\Gamma \vdash uG : F[P := G]} \, (TApp) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \lambda P \cdot u : \forall P \cdot F} \, (TAbs)$$

(P non libre dans aucune formule de Γ)

D'un point de vue logique, ce système de typage est un système de déduction naturelle pour la *logique intuitionniste minimale propositionnelle quantifiée.*

Les règles de réduction définissant $\rightarrow_{\beta\eta}$ sont:

$$\begin{array}{rrcll}
(\beta) & (\lambda x \cdot u)v & \rightarrow & u[x := v] & \\
(\eta) & \lambda x \cdot ux & \rightarrow & u & (x \text{ non libre dans } v) \\
(Beta) & (\lambda P \cdot u)F & \rightarrow & u[P := F] & \\
(Eta) & \lambda P \cdot uP & \rightarrow & u & (P \text{ non libre dans aucun type d'aucune variable libre de } u)
\end{array}$$

Exercice 32 *Montrer l'auto-réduction pour $\rightarrow_{\beta\eta}$ et le système F_2. (Dans le cas de (Eta), montrer d'abord la propriété d'amincissement, voir lemme 4.)*

Théorème 7 *Tout λ-terme typé dans le système F_2 est fortement normalisant.*

Preuve : Bien que la preuve ne soit pas très longue, elle est beaucoup plus subtile que dans les cas précédents. Essayons d'adapter la méthode de réductibilité. Le problème est de définir $RED_{\forall P \cdot F}$: on est tenté de le définir comme étant l'ensemble des λ-termes u tels que, pour toute formule G, uG soit dans $RED_{F[P:=G]}$. Mais la formule $F[P := G]$ n'est non seulement pas en général une sous-expression de $\forall P \cdot F$, mais elle peut en fait être beaucoup plus grosse. Le passage par le squelette comme aux théorèmes 4 et 6 ne fonctionne plus. Prenons en effet par exemple $RED_{\forall P \cdot P}$: si on le définit comme ci-dessus, ce sera l'ensemble des λ-termes u tels que, pour toute formule G, uG est dans RED_G. Autrement dit, pour définir $RED_{\forall P \cdot P}$, on a besoin de disposer déjà par hypothèse de récurrence des définitions des RED_G pour *toutes* les formules G — y compris $\forall P \cdot P$, au passage. La subtilité fondamentale du système F_2, comme on le voit, est que lorsque l'on quantifie sur P pour définir $\forall P \cdot F$, P varie sur toutes les formules, y compris la formule à définir $\forall P \cdot F$ elle-même. Un système logique qui a cette propriété est dit *imprédicatif*.

On contourne le problème comme suit. Considérons des *contextes* C servant à interpréter les variables de types: C est une fonction qui à chaque variable de type P associe un ensemble de λ-termes. Pour toute variable de type P et pour tout ensemble S de λ-termes, notons $C[P := S]$ le contexte qui à P associe S et à tout $Q \neq P$ associe $C(Q)$. On définit alors l'ensemble RED_F^C des termes *réductibles de type F dans le contexte C*. Ceci permettra de contourner le problème en définissant (en gros) $RED_{\forall P \cdot F}^C$ comme l'ensemble des termes u tels que pour tout ensemble S de termes, pour toute formule G, uG est dans $REC_F^{C[P:=S]}$. L'ensemble RED_F^C sera tel, notamment, que pour toute variable de type P, RED_P^C sera simplement l'ensemble $C(P)$. (Dans les preuves précédentes, c'était SN: on peut rejouer ces preuves en utilisant le contexte qui à tout P associe SN.) Mais pour que la preuve de normalisation forte fonctionne, il faudra que nous vérifions les propriétés (CR1), (CR2) et (CR3). Ces propriétés ne sont pas valides sur RED_P^C si $C(P)$ est un ensemble arbitraire de termes. On restreint donc les $C(P)$ a être des ensembles de λ-termes particuliers appelés *candidats de réductibilité*. Définissons donc les candidats de réductibilité en paraphrasant les conditions (CR1), (CR2) et (CR3). Un *candidat de réductibilité* est un ensemble S de λ-termes tels que:

(CR1) Si $u \in S$, alors $u \in SN$.

(CR2) Si $u \in S$ et $u \rightarrow_{\beta\eta} u'$, alors $u' \in S$.

(CR3) Si u est neutre et pour tout u' tel que $u \rightarrow_{\beta\eta} u'$, $u' \in S$, alors $u \in S$.

Dans cette définition, un terme *neutre* est un terme ne commençant pas par λ, et SN est l'ensemble des termes fortement normalisants pour $\rightarrow_{\beta\eta}$.

Reprenons maintenant la définition de RED_F^C. Pour toute variable de prédicat P, on pose donc $RED_P^C \hat{=} C(P)$; $RED_{F_1 \Rightarrow F_2}^C$ est l'ensemble des termes u tels que pour tout $v \in RED_{F_1}^C$, $uv \in RED_{F_2}^C$; et $RED_{\forall P \cdot F}^C$ est l'ensemble des termes u tels que, pour tout candidat de réductibilité S, pour toute formule G, $uG \in RED_F^{C[P:=S]}$. Cette définition de RED_F^C est valide, et procède par récurrence structurelle sur F.

Un contexte C est appelé un *contexte de candidats* si $C(P)$ est un candidat de réductilibité pour tout P. Montrons que RED_F^C vérifie les conditions (CR1) à (CR3), autrement dit que RED_F^C est un candidat de réductibilité pour tout contexte de candidats C et pour toute formule F. Ceci ne présente pas plus de difficulté qu'au théorème 1, et s'effectue par récurrence structurelle sur F. Si F est une variable de type P, $RED_P^C = C(P)$ est un candidat de réductibilité par hypothèse. Si F est de la forme $F_1 \Rightarrow F_2$, c'est comme au théorème 1. Le cas intéressant est celui où F est de la forme $\forall P \cdot F_1$. Remarquons d'abord que SN est lui-même un candidat de réductibilité; en particulier, il existe des candidats de réductibilité.

(CR1) Si $u \in RED_F^C$, alors pour tout candidat S, pour toute formule G, $uG \in RED_{F_1}^{C[P:=S]}$ par définition. En particulier pour $S \hat{=} SN$, $G \hat{=} \forall X \cdot X$ (par exemple). Par hypothèse de récurrence, partie (CR1), uG est dans SN, donc u aussi.

(CR2) Si $u \in RED_F^C$ et $u \rightarrow_{\beta\eta} u'$, alors pour tout candidat S, pour toute formule G, $uG \in RED_{F_1}^{C[P:=S]}$ par définition. Par hypothèse de récurrence (CR2), comme $uG \rightarrow u'G$, $u'G \in RED_{F_1}^{C[P:=S]}$. Comme S et G sont arbitraires, $u' \in RED_F^C$.

(CR3) Si u est neutre et pour tout u' tel que $u \to_{\beta\eta} u'$, $u' \in RED_F^C$, montrons que $u \in RED_F^C$. Pour cela, il suffit, par définition, de montrer que $uG \in RED_{F_1}^{C[P:=S]}$ pour tout candidat S et toute formule G. Soit donc S un candidat arbitraire, G une formule arbitraire. Comme uG est neutre, pour montrer que $uG \in RED_{F_1}^{C[P:=S]}$, il suffit de montrer que tous ses réduits en une étape sont dans $RED_{F_1}^{C[P:=S]}$, par hypothèse de récurrence (CR3). Mais uG ne peut se réduire qu'en $u'G$ avec $u \to_{\beta\eta} u'$ (auquel cas $u' \in RED_F^C$ par hypothèse, donc $u'G \in RED_{F_1}^{C[P:=S]}$) ou en $u_1[P := G]$, à condition que u soit de la forme $\lambda P \cdot u_1$; mais ce dernier cas est impossible, puisqu'alors u ne serait pas neutre.

Observons que: $(*)$ si Q n'est pas libre dans G, alors $RED_G^C = RED_G^{C[Q:=S]}$ pour tout candidat S. Autrement dit, RED_G^C ne dépend que de la restriction de C aux variables de prédicats libres dans G. On montre $(*)$ par récurrence structurelle sur G. Si G est une variable de prédicat P, alors $RED_G^C = C(P)$, alors que $RED_G^{C[Q:=S]} = (C[Q := S])(P) = C(P)$ aussi, car par hypothèse Q n'est pas libre dans G, donc $Q \neq P$. Si G est de la forme $G_1 \Rightarrow G_2$, alors $RED_G^C = \{u \mid \forall v \in RED_{G_1}^C \cdot uv \in RED_{G_2}^C\} = \{u \mid \forall v \in RED_{G_1}^{C[Q:=S]} \cdot uv \in RED_{G_2}^{C[Q:=S]}\}$ (par hypothèse de récurrence) $= RED_G^{C[Q:=S]}$. Si G est de la forme $\forall P \cdot G_1$, alors $RED_G^C = \{u \mid \forall S' \text{ candidat} \cdot \forall G' \cdot uG' \in RED_{G_1}^{C[P:=S']}\} = \{u \mid \forall S' \text{ candidat} \cdot \forall G' \cdot uG' \in RED_{G_1}^{C[P:=S'][Q:=S]}\}$ (par hypothèse de récurrence) $= \{u \mid \forall S' \text{ candidat} \cdot \forall G' \cdot uG' \in RED_{G_1}^{C[Q:=S][P:=S']}\} = RED_G^{C[Q:=S]}$.

Montrons maintenant que: $(**)$ $RED_F^{C[P:=RED_G^C]} = RED_{F[P:=G]}^C$. Ceci est par récurrence structurelle sur F. Si $F = P$, alors $RED_F^{C[P:=RED_G^C]} = (C[P := RED_G^C])(P) = RED_G^C = RED_{F[P:=G]}^C$. Si F est une autre variable de prédicat Q, $RED_F^{C[P:=RED_G^C]} = C(Q) = RED_Q^C = RED_{F[P:=G]}^C$. Si F est de la forme $F_1 \Rightarrow F_2$, alors $RED_F^{C[P:=RED_G^C]} = \{u \mid \forall v \in RED_{F_1}^{C[P:=RED_G^C]} \cdot uv \in RED_{F_2}^{C[P:=RED_G^C]}\} = \{u \mid \forall v \in RED_{F_1[P:=G]}^C \cdot uv \in RED_{F_2[P:=G]}^C\}$ (par hypothèse de récurrence) $= RED_{F_1[P:=G] \Rightarrow F_2[P:=G]}^C = RED_{F[P:=G]}^C$. Si F est de la forme $\forall Q \cdot F_1$, alors $RED_F^{C[P:=RED_G^C]} = \{u \mid \forall S \text{ candidat} \cdot \forall H \cdot uH \in RED_{F_1}^{C[P:=RED_G^C][Q:=S]}\} = \{u \mid \forall S \text{ candidat} \cdot \forall H \cdot uH \in RED_{F_1}^{C[Q:=S][P:=RED_G^C]}\} = \{u \mid \forall S \text{ candidat} \cdot \forall H \cdot uH \in RED_{F_1}^{C[Q:=S][P:=RED_G^{C[Q:=S]}]}\}$ (par $(*)$, car par α-renommage on peut supposer Q non libre dans G) $= \{u \mid \forall S \text{ candidat} \cdot \forall H \cdot uH \in RED_{F_1[P:=G]}^{C[Q:=S]}\}$ (par hypothèse de récurrence) $= RED_{F[P:=G]}^C$.

On a alors: $(+)$ pour tout terme u_1 tel que $u_1[x := v] \in RED_{F_2}^C$ pour tout $v \in RED_{F_1}^C$, alors $\lambda x \cdot u_1 \in RED_{F_1 \Rightarrow F_2}^C$. C'est comme au point 3. de la preuve du théorème 1.

Aussi: $(++)$ pour tout terme u_1 tel que $u_1[P := G] \in RED_{F_1}^{C[P:=S]}$ pour toute formule G et tout candidat S, alors $\lambda P \cdot u_1 \in RED_{\forall P \cdot F_1}^C$. Par hypothèse, en prenant $G \hat{=} P$ et $S \hat{=} C(P)$, u_1 lui-même est dans $RED_{F_1}^C$, donc dans SN par (CR1). Nous démontrons $(++)$ en montrant que $(\lambda P \cdot u_1)G \in RED_{F_1}^{C[P:=S]}$ pour toute formule G et tout candidat S. C'est par récurrence sur $\nu(u_1)$, en utilisant (CR3), puisque $(\lambda P \cdot u_1)G$ est neutre. Mais $(\lambda P \cdot u_1)G$ ne peut se réduire en une étape que vers: $(\lambda P \cdot u_1')G$ avec $u_1 \to_{\beta\eta} u_1'$, qui est dans $RED_{F_1}^{C[P:=S]}$ par hypothèse de récurrence; ou vers $u_1[P := G]$, qui est dans $RED_{F_1}^{C[P:=S]}$ par hypothèse; ou vers u_2G par (Eta), alors que u_1 est de la forme u_2P et P n'est libre dans aucun type d'aucune variable libre de u_2 — mais alors $u_2G = u_1[P := G]$, qui est dans $RED_{F_1}^{C[P:=S]}$ par hypothèse.

Il ne reste plus qu'à démontrer:

> (♣) si $\Gamma, x_1 : F_1, \ldots, x_n : F_n \vdash u : F$ est dérivable en système F_2, pour tout contexte de candidats C, si $v_1 \in RED_{F_1}^C, \ldots, v_n \in RED_{F_n}^C$, alors $u[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] \in RED_F^C$

où la substitution $[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m]$ qui remplace en parallèle les x_i par les v_i et les variables de types P_j par les formules G_j est définie de la façon naturelle.

On montre (♣) par récurrence structurelle sur u. Si u est une variable x_i, $1 \leq i \leq n$, alors $u[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] = v_i$ est dans $RED_{F_i}^C = RED_F^C$ par hypothèse.

Si u est une autre variable x, alors $u[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] = x$; comme x est neutre et normal, et comme RED_F^C est un candidat de réductibilité, par (CR3) $x \in RED_F^C$.

Si u est de la forme $u_1 u_2$, par hypothèse de récurrence $u_1[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m]$ est dans $RED^C_{G \Rightarrow F}$ pour une certaine formule G telle que $u_2[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m]$ est dans RED^C_G, donc $u[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] \in RED^C_F$.

Si u est de la forme $u_1 G$, avec u_1 de type $\forall P \cdot F_1$ et $F = F_1[P := G]$, alors par hypothèse de récurrence $u_1[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] \in RED^C_{\forall P \cdot F_1}$, donc $u[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] \in RED^{C[P:=S]}_{F_1}$ pour tout candidat S. Prenons $S \hat{=} RED^C_G$: par $(**)$, $u[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] \in RED^C_{F_1[P:=G]} = RED^C_F$.

Si u est de la forme $\lambda x \cdot u_1$, avec $F = F_1 \Rightarrow F_2$, alors par hypothèse de récurrence $u_1[x_1 := v_1, \ldots, x_n := v_n, x := v, P_1 := G_1, \ldots, P_m := G_M] \in RED^C_{F_2}$ pour tout $v \in RED^C_{F_1}$, mais $u_1[x_1 := v_1, \ldots, x_n := v_n, x := v, P_1 := G_1, \ldots, P_m := G_m] = (u_1[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m])[x := v]$, donc par $(+)$ $u[x_1 := v_1, \ldots, x_n := v_n, P_1 := G_1, \ldots, P_m := G_m] \in RED^C_F$.

Si u est de la forme $\lambda P \cdot u_1$, avec $F = \forall P \cdot F_1$, alors par hypothèse de récurrence $u_1[x_1 := v_1, \ldots, x_n := v_n, x := v, P_1 := G_1, \ldots, P_m := G_m, P := G]$ est dans $RED^{C'}_{F_1}$, et ce pour tout contexte de candidats C', en particulier pour C' de la forme $C[P := S]$. Comme $u_1[x_1 := v_1, \ldots, x_n := v_n, x := v, P_1 := G_1, \ldots, P_m := G_m, P := G] = (u_1[x_1 := v_1, \ldots, x_n := v_n, x := v, P_1 := G_1, \ldots, P_m := G_m])[P := G]$, et comme S est arbitraire, par $(++)$ $u[x_1 := v_1, \ldots, x_n := v_n, x := v, P_1 := G_1, \ldots, P_m := G_m]$ est dans $RED^C_{\forall P \cdot F_1} = RED^C_F$.

Le théorème de normalisation forte des termes typés dans le système F_2 se déduit de $(\clubsuit)$ dans le cas $n = m = 0$. $\diamondsuit$

Vu sous l'angle des langages de programmation, les règles $(TAbs)$ et $(TApp)$ offrent la notion de fonctions, et en général d'objets *polymorphes*. Prenons par exemple la fonction $id \hat{=} \lambda P \cdot \lambda x \cdot x$, de type $\forall P \cdot P \Rightarrow P$: c'est la fonction identité sur tout type P, au sens où *id* `nat` est la fonction identité sur les entiers, *id* `real` est la fonction identité sur les réels, etc. Il s'agit de *polymorphisme paramétrique*, le comportement de la fonction étant indépendant du type P passé en paramètre. (L'autre forme classique de polymorphisme est le *polymorphisme ad hoc*, où le corps de la fonction teste d'abord le type P; c'est la forme typique de polymorphisme présent dans les langages orientés objets.)

Le polymorphisme paramétrique est typiquement le polymorphisme présent dans le langage ML, en particulier en OCaml. Le polymorphisme est cependant restreint en ML: les seules quantifications universelles sont au sommet, pas à l'intérieur, des types. Formellement, appelons *monotype* tout type du système F_2 dans lequel $\forall$ n'intervient pas (ce sont les types simples), et appelons *polytype* tout type de la forme $\forall \alpha_1, \ldots, \alpha_n \cdot F$, où F est un monotype. Les types ML sont les polytypes. De plus, les jugements de typage en ML sont asymétriques, et sont de la forme $x_1 : F_1, \ldots, F_n \vdash u : \tau$, où $F_1, \ldots, F_n$ sont des polytypes et τ est un monotype. À cause de cette forme de jugement, la règle $(\forall I)$ n'a aucun sens comme règle de typage, et la construction $\lambda P \cdot u$ non plus. La création de polytypes passe par la règle de généralisation (Gen) et la construction de programmes let… in:

$$\frac{\Gamma \vdash u : \tau_1 \quad \Gamma, x : \forall \alpha_1, \ldots, \alpha_n \cdot \tau_1 \vdash v : \tau_2}{\Gamma \vdash \text{let } x = u \text{ in } v : \tau_2} \; (Gen)$$

où $\alpha_1, \ldots, \alpha_n$ sont les variables de types libres dans τ_1 mais pas libres dans aucun polytype de Γ.

ML diffère aussi du système F_2 par d'autres points. D'abord, le polymorphisme est implicite, comme dans le système F (exercice 40): lorsque x est de type $\forall \alpha \cdot F$, on n'écrit pas xG pour obtenir une instance de x de type $F[\alpha := G]$, mais simplement x. Formellement, la règle de typage des variables est:

$$\frac{}{\Gamma, x : \forall \alpha_1, \ldots, \alpha_n \cdot \tau \vdash x : \tau[\alpha_1 := \tau_1, \ldots, \alpha_n := \tau_n]}$$

Enfin ML est un langage de programmation réel, et inclut donc un certain nombre de constructions moins justifiables logiquement. Par exemple, ML possède les définitions par récursion générales (la construction let rec, codable à l'aide d'une constante $Y : \forall \alpha \cdot (\alpha \Rightarrow \alpha) \Rightarrow \alpha$ — ou plutôt $Y : \forall \alpha, \beta \cdot ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \beta)) \Rightarrow (\alpha \Rightarrow \beta))$, des types de données récursifs (généralisant la définition des entiers de Peano construits sur `0` et

S, notamment), des effets de bord, et OCaml a en plus les objets (polymorphisme ad hoc) et encore d'autres constructions.

Exercice 33 *Les entiers de Church en système F_2 sont $\lceil n \rceil \hat{=} \lambda P \cdot \lambda f, x \cdot f^n x$. Montrer qu'ils sont tous de type $\mathbf{N} \hat{=} \forall P \cdot (P \Rightarrow P) \Rightarrow (P \Rightarrow P)$. En s'inspirant des définitions correspondantes en λ-calcul pur, donner des définitions du successeur S de type $\mathbf{N} \Rightarrow \mathbf{N}$, de l'addition $\lceil + \rceil$, de la multiplication $\lceil \times \rceil$ et de la fonction puissance $\lceil exp \rceil$, ces trois dernières de type $\mathbf{N} \Rightarrow \mathbf{N} \Rightarrow \mathbf{N}$. Comparer avec l'exercice 4.*

Exercice 34 *On pose $F_1 \times F_2 \hat{=} \forall P \cdot (F_1 \Rightarrow F_2 \Rightarrow P) \Rightarrow P$. (Comparer avec l'exercice 5.) On pose $\langle u, v \rangle \hat{=} \lambda P \cdot \lambda z \cdot zuv$, et lorsque u est de type $F_1 \times F_2$, $\pi_1 F_1 u \hat{=} u F_1 (\lambda x, y \cdot x)$, $\pi_2 F_2 u \hat{=} u F_2 (\lambda x, y \cdot y)$. Montrer que les règles $(\wedge I)$, $(\wedge E_1)$ et $(\wedge E_2)$ sont admissibles (en remplaçant π_1 par $\pi_1 F_1$ et π_2 par $\pi_2 F_2$), et que $\pi_1 F_1 \langle u, v \rangle \rightarrow^+_{\beta\eta} u$, $\pi_2 F_2 \langle u, v \rangle \rightarrow^+_{\beta\eta} v$. En somme, on peut définir le produit (la conjonction) en système F_2.*

Exercice 35 *On pose $F_1 + F_2 \hat{=} \forall P \cdot (F_1 \Rightarrow P) \Rightarrow (F_2 \Rightarrow P) \Rightarrow P$, et:*

$$\begin{array}{rcl} \iota_1 u & \hat{=} & \lambda P \cdot \lambda k_1, k_2 \cdot k_1 u \\ \iota_2 u & \hat{=} & \lambda P \cdot \lambda k_1, k_2 \cdot k_2 u \\ \langle G \rangle \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} & \hat{=} & uG(\lambda x_1 \cdot v_1)(\lambda x_1 \cdot v_2) \end{array}$$

où k, k_1, k_2 sont des variables fraîches, P est une variable de type fraîche dans les deux premiers cas et v_1 et v_2 sont de types G dans le troisième cas. Montrer que les règles de typage $(\vee I_1)$, $(\vee I_2)$ et $(\vee E)$ sont vérifiées, et que l'on a les réductions:

$$\begin{array}{rcl} \langle G \rangle \text{case } \iota_1 u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} & \rightarrow^+_{\beta\eta} & v_1[x_1 := u] \\ \langle G \rangle \text{case } \iota_2 u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto v_1 \\ \iota_2 x_2 \mapsto v_2 \end{array} \right\} & \rightarrow^+_{\beta\eta} & v_2[x_2 := u] \end{array}$$

En somme, on peut définir la somme (la disjonction) en système F_2. Comparer avec l'exercice 22.

Exercice 36 *On pose $\exists P \cdot F \hat{=} \forall Q \cdot (\forall P \cdot F \Rightarrow Q) \Rightarrow Q$, où Q est une variable de type fraîche. On définit $\iota G u \hat{=} \lambda Q \cdot \lambda k \cdot k G u$ et $\text{case } u \, \{\iota P x \mapsto v\} \hat{=} u H (\lambda P \cdot \lambda x \cdot v)$ (où H est le type de v dans un contexte idoine, cf. ci-dessous).*

Montrer que les règles de typage suivantes sont admissibles:

$$\frac{\Gamma \vdash u : \exists P \cdot F \quad \Gamma, x : F \vdash v : H}{\Gamma \vdash \text{case } u \, \{\iota P x \mapsto v\} : H} \, (\exists_2 E) \qquad \frac{\Gamma \vdash u : F[P := G]}{\Gamma \vdash \iota G u : \exists P \cdot F} \, (\exists_2 I)$$

(P non libre dans aucune formule de Γ ni dans G)

En déduire que le quantificateur existentiel du second ordre est définissable dans le système F_2. Montrer qu'on a la règle de réduction:

$$\text{case } \iota G u \, \{\iota P x \mapsto v\} \quad \rightarrow^+_{\beta\eta} \quad v[P := G][x := u]$$

Exercice 37 *Comme à l'exercice 36, montrer qu'on peut définir le quantificateur existentiel du premier ordre en système F_2 par $\exists i \cdot F \hat{=} \forall Q \cdot (\forall i \cdot F \Rightarrow Q) \Rightarrow Q$. En déduire une preuve simple de la normalisation forte de $\mathbf{HA}_1^\exists$ (exercice 28).*

Exercice 38 *Montrer que $P_\forall \hat{=} \lambda k \cdot \pi_2 \mathbf{N}(k(\mathbf{N} \times \mathbf{N})(\lambda s \cdot \langle S(\pi_1 \mathbf{N} s), \pi_1 \mathbf{N} s \rangle) \langle \lceil 0 \rceil, \lceil 0 \rceil \rangle)$, où S et $\lceil 0 \rceil$ sont comme à l'exercice 33 et $\langle \ldots \rangle$, π_1, π_2 et $\times$ sont comme à l'exercice 34, est un terme définissant le prédécesseur, au sens où $P_\forall$ est de type $\mathbf{N} \Rightarrow \mathbf{N}$ et où $P(S \lceil n \rceil) \rightarrow^*_{\beta\eta} \lceil n \rceil$ pour tout entier n. (Effectuer une récurrence sur n.)*

Exercice 39 (☡) *Le but de cet exercice est de montrer qu'on peut définir le récurseur sur les entiers de Church dans le système F_2. Ceci procède comme pour le prédécesseur. On pose* $R_\forall \hat{=} \lambda Q \cdot \lambda x, y, n \cdot \pi_1 Q(n(Q \times \mathbf{N})(\lambda z \cdot \langle y(\pi_2 \mathbf{N} z)(\pi_1 Q z), S(\pi_2 \mathbf{N} z)\rangle)\langle x, \lceil 0 \rceil\rangle)$. *Vérifier que* $R_\forall$ *est de type* $\forall Q \cdot Q \Rightarrow (\mathbf{N} \Rightarrow Q \Rightarrow Q) \Rightarrow \mathbf{N} \Rightarrow Q$, *et que* $R_\forall G u v \lceil 0 \rceil \rightarrow^+_{\beta\eta} u$, $R_\forall G u v \lceil n+1 \rceil \leftrightarrow^*_{\beta\eta} v \lceil n \rceil (R_\forall G u v \lceil n \rceil)$, *où* $\leftrightarrow^*_{\beta\eta}$ *est la clôture réflexive symétrique transitive de* $\rightarrow_{\beta\eta}$. *(Montrer par récurrence sur n que, si l'on pose* $f(Q, u, v, n) \hat{=} n(Q \times \mathbf{N})(\lambda z \cdot \langle v(\pi_2 \mathbf{N} z)(\pi_1 Q z), S(\pi_2 \mathbf{N} z)\rangle)\langle u, \lceil 0 \rceil\rangle$, *alors* $f(Q, u, v, \lceil 0 \rceil) \rightarrow^+_{\beta\eta} \langle u, \lceil 0 \rceil\rangle$, *et que si* $f(Q, u, v, \lceil n \rceil) \leftrightarrow^*_{\beta\eta} \langle w, \lceil n \rceil\rangle$, *alors* $f(Q, u, v, \lceil n+1 \rceil) \leftrightarrow^*_{\beta\eta} \langle v \lceil n \rceil w, \lceil n+1 \rceil\rangle$.*)*

Exercice 40 *Le système F est comme le système F_2, sauf que les règles $(TApp)$ et $(TAbs)$ sont changées en:*

$$\frac{\Gamma \vdash u : \forall P \cdot F}{\Gamma \vdash u : F[P := G]}\ (TApp) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall P \cdot F}\ (TAbs)$$

(P non libre dans aucune formule de Γ)

Autrement dit, on se passe des constructions d'application à des types et d'abstraction sur les types. Noter que tout terme a alors en général une infinité de types, même dans un contexte Γ fixé. Montrer que la propriété d'auto-réduction est vérifiée pour la règle (β), mais pas pour la règle (η).

Exercice 41 (☡) *Montrer que le λ-calcul typé en système F (exercice 40) est fortement normalisant. (Adapter la preuve du théorème 7.) Pourquoi ceci n'est-il pas un cas particulier du théorème 7 ?*

4.3 Normalisation forte de la logique d'ordre deux

Le langage des termes de la logique intuitionniste d'ordre deux est le $\lambda\nabla$-calcul plus les constructions d'application et d'abstraction de prédicats:

$$\Lambda_2 ::= \mathcal{V} \mid \Lambda_2\Lambda_2 \mid \lambda\mathcal{V} \cdot \Lambda_2 \mid \nabla\Lambda_2 \mid \Lambda_2 T \mid \lambda\mathcal{X} \cdot \Lambda_2 \mid \Lambda_2 F \mid \lambda\mathcal{P}_n \cdot \Lambda_2$$

où T est l'ensemble des termes du premier ordre et F est celui des formules du second ordre. Les règles de preuve, c'est-à-dire de typage, sont (Ax), $(\bot E)$, $(\Rightarrow I)$, $(\Rightarrow E)$, $(\forall I)$, $(\forall E)$, $(\forall_2 I)$ et $(\forall_2 E)$. On normalise les preuves par les règles de réduction:

$$\begin{array}{rrcll} (\beta) & (\lambda x \cdot u)v & \rightarrow & u[x := v] & \\ (\beta_1) & (\lambda i \cdot u)t & \rightarrow & u[i := t] & \\ (Beta) & (\lambda P \cdot u)F & \rightarrow & u[P(k_1, \ldots, k_n) := G] & (P \in \mathcal{P}_n) \end{array}$$

La propriété de normalisation forte des preuves de logique intuitionniste d'ordre deux est une conséquence directe de celle du système F_2. En effet, définissons comme au théorème 4 une fonction qui efface tout ce qui est du premier ordre dans les formules et les $\lambda\nabla$-termes. Sur les formules:

$$E(P(t_1, \ldots, t_n)) \hat{=} P \quad E(\bot) \hat{=} \forall P \cdot P \quad E(F_1 \Rightarrow F_2) \hat{=} E(F_1) \Rightarrow E(F_2) \quad E(\forall i \cdot F) \hat{=} E(F) \quad E(\forall P \cdot F) \hat{=} \forall P \cdot E(F)$$

Noter que $P \in \mathcal{P}_n$ sur le côté gauche est vu comme variable 0-aire sur le côté droit. On traduit aussi $\bot$ en $\forall P \cdot P$, cf. exercice 31. Sur les $\lambda\nabla$-termes, l'effacement est défini par:

$$\begin{array}{rcllrcll} E(x) & \hat{=} & x & & E(uv) & \hat{=} & E(u)E(v) & \\ E(\lambda x \cdot u) & \hat{=} & \lambda x \cdot E(u) & & E(\nabla u) & \hat{=} & E(u)E(G) & (\nabla u \text{ de type } G) \\ E(ut) & \hat{=} & E(u) & & E(\lambda i \cdot u) & \hat{=} & E(u) & \\ E(uG) & \hat{=} & E(u)E(G) & & E(\lambda P \cdot u) & \hat{=} & \lambda P \cdot E(u) & \end{array}$$

On a:

Lemme 10 *Si $x_1 : F_1, \ldots, x_m : F_m \vdash u : F$ est dérivable en logique intuitionniste du second ordre, alors $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u) : E(F)$ en système F_2.*

Preuve : Remarquons d'abord que: $(*)$ $E(F'[P(k_1, \ldots, k_n) := G]) = E(F')[P := E(G)$. C'est une récurrence structurelle facile sur F'. Ensuite, que: $(**)$ si P n'est pas libre dans F', alors P n'est pas libre dans $E(F')$. C'est encore une récurrence structurelle facile sur F'.

On démontre le lemme par récurrence structurelle sur la dérivation donnée de $x_1 : F_1, \ldots, x_m : F_m \vdash u : F$. C'est évident si u est une variable x_i, $1 \leq i \leq n$, si u est de la forme $u'v'$, ou $\lambda x \cdot u'$. Si u est de la forme $\nabla u'$, alors par hypothèse de récurrence on a dérivé $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u') : \forall P \cdot P$, donc $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u) : E(F)$ par $(TApp)$. Si u est de la forme $u't$ avec t un terme du premier ordre, alors par hypothèse de récurrence on a dérivé $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u') : E(G)$, où $G = \forall i \cdot F$, donc $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u) : E(F)$, puisque $E(u) = E(u')$ et $E(F) = E(G)$. Si u est de la forme $\lambda i {\cdot} u'$, alors par hypothèse de récurrence on a dérivé $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u') : E(G)$, où $F = \forall i {\cdot} G$, donc $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u) : E(F)$, puisque $E(u) = E(u')$ et $E(F) = E(G)$. Si u est de la forme $u'G$, alors par hypothèse de récurrence on a dérivé $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u') : E(\forall P {\cdot} F')$, où $F = F'[P(k_1, \ldots, k_n) := G]$; on en déduit que $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u')E(G) : E(F')[P := E(G)]$: par $(*)$, $E(F) = E(F')[P := E(G)]$, donc on a dérivé $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u) : E(F)$. Si u est de la forme $\lambda P \cdot u'$, alors par hypothèse de récurrence on a dérivé $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u') : E(F')$, avec $F = \forall P \cdot F'$, où P n'est pas libre dans $F_1, \ldots, F_m$; par $(**)$, on peut appliquer $(TAbs)$ et inférer $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash \lambda P {\cdot} E(u') : \lambda P {\cdot} E(F')$, c'est-à-dire $x_1 : E(F_1), \ldots, x_m : E(F_m) \vdash E(u) : E(F)$. $\diamondsuit$

Lemme 11 *Si $u \to v$ par les règles (β) ou $(Beta)$, alors $E(u) \to E(v)$ par (β) ou $(Beta)$. Si $u \to v$ par (β_1), alors $E(u) = E(v)$.*

Preuve : Montrons d'abord que: $(*)$ $E(u')[x := E(v')] = E(u'[x := v'])$. Aussi, que: $(**)$ $E(u')[P := E(G)] = E(u'[P(k_1, \ldots, k_n) := G])$ pour tout $P \in \mathcal{P}_n$. Enfin, que: $(***)$ $E(u'[i := t]) = E(u')$. Les trois sont par récurrence structurelle sur u', $(**)$ utilisant la propriété $(*)$ démontrée au lemme 10 dans le cas où u' est de la forme $v'G'$, où G' est une formule.

On démontre le lemme par récurrence sur la profondeur du rédex que l'on contracte dans u. Le seul cas intéressant est celui où u est lui-même le rédex. Or $E((\lambda x \cdot u')v') = (\lambda x \cdot E(u'))E(v') \to E(u')[x := E(v')] = E(u'[x := v'])$ (par $(*)$), ce qui règle le cas de (β), $E((\lambda P \cdot u')G) = (\lambda P \cdot E(u'))E(G) \to E(u')[P := E(G)] = E(u'[P(k_1, \ldots, k_n) := G])$ (par $(**)$), ce qui règle le cas de $(Beta)$, et $E((\lambda i \cdot u')t) = E(u') = E(u'[i := t])$ (par $(***)$). $\diamondsuit$

Théorème 8 *Tout $\lambda\nabla$-terme u typé en logique intuitionniste du second ordre est fortement normalisant.*

Preuve : Notons que si $v \to w$ par la règle (β_1), alors la taille $|Sk(v)|$ du squelette de v est strictement supérieure à celle $|Sk(w)|$ du squelette de w. Définissons le squelette d'un terme par: $Sk(x) \hat{=} x$, $Sk(u'v') \hat{=} Sk(u')Sk(v')$, $Sk(\lambda x \cdot u') \hat{=} \lambda x \cdot Sk(u')$, $Sk(\lambda i \cdot u') \hat{=} \lambda i \cdot Sk(u')$, $Sk(u't) \hat{=} Sk(u')$, $Sk(\lambda P \cdot u') \hat{=} \lambda P \cdot Sk(u')$, $Sk(u'G) \hat{=} Sk(u')Sk(G)$, où le squelette de G est défini comme au théorème 4. La taille de $Sk((\lambda i {\cdot} u')t) = \lambda i {\cdot} Sk(u')$ est strictement supérieure à celle de $Sk(u'[i := t]) = Sk(u')$ (comme une récurrence structurelle facile sur u' le montre).

Donc, si on mesure u par le couple $(\nu(E(u)), |Sk(u)|)$, où ν est l'opérateur "longueur de la plus longue réduction" en système F_2 (qui est bien défini par le théorème 7) et que l'on ordonne ces couples par l'ordre lexicographique, si $v \to w$, alors la mesure de v est strictement supérieure à celle de w, en utilisant le lemme 11. L'ordre lexicographique étant bien fondé, le théorème est démontré.

Une preuve peut-être plus intuitive est la suivante. Supposons qu'il existe une réduction infinie partant de u: $u = u_0 \to u_1 \to \ldots \to u_i \to \ldots$. En passant aux effacements, $E(u_0) \to_= E(u_1) \to_= \ldots \to_= E(u_i) \to_= \ldots$ par lemme 11, où $\to_=$ dénote 0 ou une étape de réduction en F_2. Par le théorème 7 il n'y a qu'un nombre fini de ces symboles $\to_=$ qui sont des $\to$, donc il existe un entier i tel qu'à partir de l'indice i on n'ait plus que des réductions par (β_1). Mais il ne peut y avoir qu'un nombre fini consécutif de telles réductions, ce qui contredit le fait que la réduction considérée soit infinie. $\diamondsuit$

Corollaire 6 *La logique intuitionniste d'ordre deux est cohérente: on ne peut pas démontrer* $\vdash u : \bot$ *pour aucun* $\lambda\nabla$*-terme* u.

Preuve : Par le théorème 8, on peut se restreindre à considérer u normal. Prenons u de taille minimale parmi les termes normaux tels que $\vdash u : \bot$ soit dérivable; u est de la forme $hu_1 \dots u_n$, où h est une variable ou le symbole ∇ (auquel cas $n \geq 1$), et $u_1, \dots, u_n$ sont des $\lambda\nabla$-termes, des termes du premier ordre ou des formules. Mais h ne peut pas être une variable, parce que le contexte à gauche de $\vdash$ est vide. Donc h est ∇, mais alors on a $\vdash u_1 : \bot$, ce qui contredit la minimalité de la taille de u. $\diamondsuit$

Exercice 42 *Dans les démonstrations ci-dessus, nous n'avons pas considéré la règle* (∇)*:* $\nabla uv \to \nabla u$*. Pourquoi la technique de preuve que nous avons utilisé ne fonctionne-t-elle plus si on ajoute cette règle ?*

Exercice 43 (☡) *La logique* classique *du second ordre est la logique intuitionniste du second ordre plus l'opérateur* $\mathcal{C}$*, autrement dit plus la règle de typage* $(\neg\neg E)$*. La règles de réduction sont* (β)*,* (β_1)*,* $(Beta)$*,* $(\mathcal{C}_L)$ *et* $(\eta\mathcal{C})$*. Montrer en étendant le système* F_2 *au cas classique, c'est-à-dire en ajoutant la règle de typage:*

$$\frac{\Gamma \vdash u : (F \Rightarrow \forall X \cdot X) \Rightarrow \forall X \cdot X}{\Gamma \vdash \mathcal{C}u : F}$$

et les règles de réduction $(\mathcal{C}_L)$ *et* $(\eta\mathcal{C})$*, que la logique classique du second ordre est cohérente.*

Exercice 44 *On définit l'*égalité de Leibniz *comme suit:* $s \approx t$ *est vu comme une abréviation de* $\forall P \cdot P(s) \Rightarrow P(t)$*. Trouver des termes de preuve montrant que* $\approx$ *est réflexive et que* $s \approx t$ *implique que l'on peut déduire* $F[i := t]$ *à partir de* $F[i := s]$*. En conclure qu'on n'a pas besoin de symbole spécial ou de règle spéciale pour traiter l'égalité en logique d'ordre deux: l'égalité est définissable à l'ordre deux.*

Exercice 45 *Montrer que l'égalité de Leibniz est symétrique, autrement dit trouver* u *tel que* $x : s \approx t \vdash u : t \approx s$*. (Indication: instancier* $P(k_1)$ *par* $Q(k_1) \Rightarrow Q(s)$*.)*

4.4 Retour sur $\mathbf{HA}_2$ et $\mathbf{PA}_2$

On va montrer que $\mathbf{HA}_2$ est cohérent, par une extension de la technique utilisée pour $\mathbf{HA}_1$. La principale innovation est le traitement de la quantification du second ordre, qui sera traitée comme dans le système F_2.

Lemme 12 *Si* $\Gamma, x : F_1 \vdash u : F_2$ *et* $\Gamma \vdash v : F_1$ *sont dérivables en* $\mathbf{HA}_2$*, alors* $\Gamma \vdash u[x := v] : F_2$ *aussi.*

Preuve : Comme au lemme 9, on prouve que pour toute preuve π_2 de $\Gamma, x : F_1, \Delta \vdash u : F_2$ et toute preuve π_1 de $\Gamma \vdash v : F_1$, on peut construire une preuve π de $\Gamma, \Delta \vdash u[x := v] : F_2$ par récurrence structurelle sur π_2. $\diamondsuit$

Lemme 13 *Si* $\Gamma \vdash u : F$ *est dérivable en* $\mathbf{HA}_2$*, et* $P \in \mathcal{P}_n$ *n'est pas libre dans aucune formule de* Γ*, alors* $\Gamma \vdash u[P(k_1, \dots, k_n) := G] : F[P(k_1, \dots, k_n) := G]$ *aussi.*

Preuve : On prouve plus généralement que pour toute preuve π de $x_1 : F_1, \dots, x_m : F_m \vdash u : F$, on peut construire une preuve π de $x_1 : F_1[P(k_1, \dots, k_n) := G], \dots, x_m : F_m[P(k_1, \dots, k_n) := G] \vdash u[P(k_1, \dots, k_n) := G] : F[P(k_1, \dots, k_n) := G]$ par récurrence structurelle sur π. $\diamondsuit$

Nous considérons les règles de réduction suivantes, qui sont celles de $\mathbf{HA}_1$ plus $(Beta)$. On ne considère pas (η) et (Eta), dont nous n'aurons pas besoin.

$$\begin{array}{rrcl}
(\beta) & (\lambda x \cdot u)v & \to & u[x := v] \\
(\beta_1) & (\lambda i \cdot u)t & \to & u[i := t] \\
(Beta) & (\lambda P \cdot u)F & \to & u[P(k_1, \ldots, k_n) := G] \quad (P \in \mathcal{P}_n) \\
(\nabla) & \nabla uv & \to & \nabla u \\
(R\mathtt{0}) & Ruv\mathtt{0} & \to & u \\
(R\mathtt{S}) & Ruv(\mathtt{S}(t)) & \to & vt(Ruvt) \\
(+\mathtt{0}) & s+\mathtt{0} & \to & s \\
(+\mathtt{S}) & s+\mathtt{S}(t) & \to & \mathtt{S}(s+t) \\
(*\mathtt{0}) & s*\mathtt{0} & \to & \mathtt{0} \\
(*\mathtt{S}) & s*\mathtt{S}(t) & \to & s*t+s
\end{array}$$

Théorème 9 (Auto-réduction) *Si $\Gamma \vdash u : F$ est dérivable en $\mathbf{HA}_2$ et $u \to v$, alors $\Gamma \vdash v : F$ est dérivable en $\mathbf{HA}_2$.*

Preuve : Par rapport au théorème 5, le cas de la règle (β) est par le lemme 12, celui de la règle (β_1) est comme au lemme 8, et celui de la règle $(Beta)$ est par le lemme 13. Les autres cas sont comme au théorème 5. $\diamondsuit$

Théorème 10 *Tout $\lambda\nabla R$-terme typé en $\mathbf{HA}_2$ est fortement normalisant.*

Preuve : Comme au théorème 6, notons que tout terme t du premier ordre termine. Appelons encore un $\lambda\nabla R$-terme *neutre* s'il ne commence pas par λ ou ∇. Soit SN l'ensemble des $\lambda\nabla R$-termes fortement normalisants. Un *candidat de réductibilité* est un ensemble S de $\lambda\nabla R$-termes vérifiant les propriétés (CR1), (CR2) et (CR3) du théorème 7. Un *contexte de candidats* C est une fonction des variables de prédicats vers des candidats de réductibilité. On définit ensuite $RED^C_\perp$ et $RED^C_{s\approx t}$ comme étant SN, $RED^C_{P(t_1,\ldots,t_n)}$ est $C(P)$, $RED^C_{F\Rightarrow G}$ est l'ensemble des $\lambda\nabla R$-termes u tels que pour tout $v \in RED^C_F$, $uv \in RED^C_G$, $RED^C_{\forall i \cdot F}$ est l'ensemble des $\lambda\nabla R$-termes u tels que pour tous termes t du premier ordre (qui terminent), $ut \in RED^C_{F[i:=t]}$, et finalement $RED^C_{\forall P \cdot F}$ est l'ensemble des $\lambda\nabla R$-termes u tels que pour toute formule G, pour tout candidat de réductibilité S, $uG \in RED^{C[P:=S]}_F$. Ceci est une définition de RED_F par récurrence structurelle sur le squelette $Sk(F)$ de F, qui est défini comme au théorème 4, étendu par $Sk(\forall P \cdot F) \hat{=} \forall P \cdot Sk(F)$.

Il est facile de vérifier que RED^C_F est un candidat de réductibilité pour tout contexte de candidats C et toute formule F, par récurrence structurelle sur $Sk(F)$, que $(*)$ si Q n'est pas libre dans G, alors $RED^C_G = RED^{C[Q:=S]}_G$ pour tout candidat S (par récurrence sur $Sk(G)$), que $(**)$ $RED^{C[P:=RED^C_G]}_F = RED^C_{F[P(k_1,\ldots,k_n):=G]}$ (par récurrence sur $Sk(F)$), que $(+)$ pour tout terme u_1 tel que $u_1[x := v] \in RED^C_{F_2}$ pour tout $v \in RED^C_{F_1}$, alors $\lambda x \cdot u_1 \in RED^C_{F_1 \Rightarrow F_2}$, et que $(++)$ pour tout terme u_1 tel que $u_1[P(k_1, \ldots, k_n) := G] \in RED^{C[P:=S]}_{F_1}$ pour toute formule G et tout candidat S, alors $\lambda P \cdot u_1 \in RED^C_{\forall P \cdot F_1}$. C'est comme au théorème 7.

Comme au théorème 6, on montre que $(\circ)$ si $v \in RED_{F[i:=\mathtt{0}]}$ et $w \in RED_{\forall j \cdot F[i:=j] \Rightarrow F[i:=\mathtt{S}(j)]}$, alors $Rvwt \in RED_{F[i:=t]}$. Ceci est comme avant par récurrence sur $(\nu(v) + \nu(w) + \nu(t), |t|)$ ordonné lexicographiquement, où ν est la fonction "longueur de la plus grande réduction" et $|t|$ est la taille du terme t.

Les propriétés $(*)$, $(**)$, $(+)$, $(++)$ et $(\circ)$ nous permettent de démontrer:

$(\clubsuit)$ si $\Gamma, x_1 : F_1, \ldots, x_n : F_n \vdash u : F$ est dérivable en $\mathbf{HA}_2$, alors pour tout contexte de candidats C, si $v_1 \in RED^C_{F_1}, \ldots, v_n \in RED^C_{F_n}$, alors
$$u[x_1 := v_1, \ldots, x_n := v_n, P_1(k_1, \ldots, k_{n_1}) := G_1, \ldots, P_m(k_1, \ldots, k_{n_m}) := G_m] \in RED^C_F$$

d'où l'on déduit le théorème. $\diamondsuit$

Corollaire 7 *L'arithmétique de Heyting du second ordre $\mathbf{HA}_2$ est cohérente.*

Preuve : Exactement comme au corollaire 5. ◇

Exercice 46 *Exercez-vous à refaire rigoureusement et en détail la preuve du théorème 10.*

Exercice 47 *L'arithmétique de Peano du premier ordre* $\mathbf{PA}_2$ *est à* $\mathbf{HA}_2$ *ce que* $\mathbf{PA}_1$ *est à* $\mathbf{HA}_1$*, sa version classique. (Voir exercice 29.) On ajoute les règles* $(\forall_2 I)$ *et* $(\forall_2 E)$ *à* $\mathbf{PA}_1$*, et les règles de réduction* (*Beta*)*. Montrer, en s'inspirant des résultats déjà démontrés, l'auto-réduction et la normalisation forte de ce calcul. En déduire que* $\mathbf{PA}_2$ *est cohérente.*

L'arithmétique du second ordre et le système F_2 sont tellement proches qu'en fait les fonctions mathématiques de $\mathbb{N}$ vers $\mathbb{N}$ dont on peut démontrer qu'elles sont totales en arithmétique d'ordre deux sont exactement celles que l'on peut coder comme des λ-termes u de type $\mathbf{N} \Rightarrow \mathbf{N}$ en système F_2. C'est le théorème de Girard, dont on pourra trouver une démonstration au chapitre 15 de [GLT89]. Mais certaines fonctions n'ont que des implémentations inefficaces. L'exemple typique est le prédécesseur, dont l'implémentation la plus efficace en système F_2 est celle de l'exercice 38, qui calcule le prédécesseur de n en temps linéaire en n. Une façon de corriger ce problème est d'ajouter à F_2 un type de base $\mathbb{N}$, plus deux constantes $0 : \mathbb{N}$ et $S : \mathbb{N} \Rightarrow \mathbb{N}$, ainsi qu'un récurseur $R : \forall Q \cdot Q \Rightarrow (\mathbb{N} \Rightarrow Q \Rightarrow Q) \Rightarrow \mathbb{N} \Rightarrow Q$ tel que $Ruv0 \to u$ et $Ruv(Sw) \to vw(Ruvw)$ — essentiellement comme en $\mathbf{HA}_2$, sauf qu'en $\mathbf{HA}_2$ les entiers naturels sont codés comme des termes du premier ordre et non comme des λ-termes de type $\mathbb{N}$. Le prédécesseur est alors $R0(\lambda n, x \cdot n)$.

4.5 Logiques d'ordre supérieur

Il n'y a aucune raison de s'arrêter à l'ordre deux, et l'on peut définir des systèmes logiques d'ordre 3, 4, $\ldots$, ω. La logique et l'arithmétique à tous ordres est encore cohérente. Le but de cette section est d'une part d'indiquer comment le λ-calcul est utilisé ici pour définir non seulement les preuves mais les formules elles-mêmes, et d'autre part d'avertir le lecteur qui aurait l'impression que tous les systèmes logiques que l'on peut définir intuitivement sont cohérents, qu'il existe des systèmes logiques apparemment très proches des systèmes précédents mais qui sont incohérents.

Rappelons que nous avons formalisé l'arithmétique du premier ordre par un λ-calcul dont les types étaient des formules du premier ordre modulo quelques règles de calcul pour définir l'addition et la multiplication. La logique d'ordre supérieure est définie de façon similaire en choisissant comme langage des formules un λ-calcul simplement typé enrichi de quelques constantes représentant l'implication et la quantification universelle, modulo les règles de calcul définies par la $\beta\eta$-conversion.

Les types simples que l'on utilise sont:

$$\mathcal{T} ::= B \mid \mathcal{T} \to \mathcal{T}$$

où l'on note $\to$ ce que l'on notait $\Rightarrow$ auparavant, et B est un ensemble de types dit de base, contenant au moins un type *Prop*, le type des *formules*, et un autre type ι, qui servira de type des termes. On pourrait considérer plusieurs types de termes, $\mathbb{N}$, $\mathbb{R}$, **string**, etc., mais pour simplifier nous ferons comme s'il n'y en avait qu'un.

Les *expressions logiques* sont les λ-termes typables. Ceux de type *Prop* sont appelés les formules, ceux de type ι sont appelés les termes. On considère deux constantes additionnelles servant à fabriquer les formules: pour chaque type τ, $\forall_\tau$ de type $(\tau \to \mathit{Prop}) \to \mathit{Prop}$, qui sert à former les quantifications universelles ($\forall x : \tau \cdot F$ sera codé comme $\forall_\tau(\lambda x \cdot F)$), et *imp* de type $\mathit{Prop} \to \mathit{Prop} \to \mathit{Prop}$, qui sert à former les implications (et on notera $F \Rightarrow G$ au lieu de $\mathit{imp}\ F\ G$). Les symboles de prédicats n-aires seront représentés par des variables P de type $\iota \to \ldots \to \iota \to \mathit{Prop}$, où il y a n ι: si $t_1, \ldots, t_n$ sont n termes (de type ι, donc), $Pt_1 \ldots t_n$ est la formule que nous notions $P(t_1, \ldots, t_n)$ en logique du premier ordre. Les symboles de fonctions n-aires sont représentés par des variables f de type $\iota \to \ldots \to \iota \to \iota$, où il y a $n+1$ ι en tout: si $t_1, \ldots, t_n$ sont n termes, $ft_1 \ldots t_n$ sera donc aussi un terme, celui que nous notions $f(t_1, \ldots, t_n)$ en logique du premier ordre.

Les règles de typage des expressions logiques sont essentiellement celles du λ-calcul simplement typé, plus les règles de typage de $\forall_\tau$ et *imp*:

$$\frac{}{\Gamma \vdash \forall_\tau : (\tau \to Prop) \to Prop}\ (\forall_\tau) \qquad \frac{}{\Gamma \vdash imp : Prop \to Prop \to Prop}\ (imp)$$

La quantification d'ordre un est $\forall_\iota$. Notamment, $\forall x : \iota \cdot F$ est ce que nous notions $\forall i \cdot F[x := i]$ auparavant. La quantification d'ordre deux est $\forall_{\iota \to \ldots \to \iota \to Prop}$: en somme, $\forall x : \iota \to \ldots \to \iota \to Prop \cdot F$, où il y a n ι, est ce que nous notions $\forall P \cdot F[x := P]$ avec $P \in \mathcal{P}_n$ en logique du second ordre.

La substitution du premier ordre est la substitution du λ-calcul, et la substitution du second ordre $F[P(k_1, \ldots, k_n) := G]$ est à $\beta\eta$-équivalence près la substitution $F[P := \lambda k_1, \ldots, k_n \cdot G]$ du λ-calcul.

Dans un contexte de typage simple Γ, soient $F_1, \ldots, F_n$ des formules (des expressions de type $Prop$ dans Γ), alors $x_1 : F_1, \ldots, x_n : F_n$ est un contexte pourvu que les x_i soient des variables différentes deux à deux et en dehors du domaine de Γ. Alors que les contextes de typage seront typiquement notés Γ, les contextes seront typiquement notés Π. Les jugements de preuves seront de la forme $\Gamma; \Pi \vdash u : F$, et leur dérivabilité signifiera que dans le contexte de typage Γ, Π est un contexte et F est une formule, et que de plus u est une preuve de F à partir des hypothèses $F_1, \ldots, F_n$ présentes dans Π. Noter que les jugements de typage $\Gamma \vdash e : \tau$ seront aussi utilisés, et sont des jugements du λ-calcul simplement typé avec les constantes $\forall_\tau$ et imp. On définit la logique d'ordre supérieure par les règles:

$$\frac{\Gamma; \Pi \vdash u : \bot \quad \Gamma \vdash F : Prop}{\Gamma; \Pi \vdash \nabla u : F}\ (\bot E) \qquad \frac{\Gamma \vdash F_1 : Prop \quad \ldots \Gamma \vdash F_n : Prop \quad \Gamma \vdash F : Prop}{\Gamma; x_1 : F_1, \ldots, x_n : F_n, x : F \vdash x : F}\ (Ax)$$

$$\frac{\Gamma; \Pi \vdash u : F_1 \Rightarrow F_2 \quad \Gamma; \Pi \vdash v : F_1}{\Gamma; \Pi \vdash uv : F_2}\ (\Rightarrow E) \qquad \frac{\Gamma; \Pi, x : F_1 \vdash u : F_2}{\Gamma; \Pi \vdash \lambda x \cdot u : F_1 \Rightarrow F_2}\ (\Rightarrow I)$$

$$\frac{\Gamma; \Pi \vdash u : \forall x : \tau \cdot F \quad \Gamma \vdash e : \tau}{\Gamma; \Pi \vdash ut : F[x := e]}\ (\forall E) \qquad \frac{\Gamma, x : \tau; \Pi \vdash u : F}{\Gamma; \Pi \vdash \lambda x : \tau \cdot u : \forall x : \tau \cdot F}\ (\forall I)$$

(x non libre dans aucune formule de Π)

$$\frac{\Gamma; \Pi \vdash u : F_1 \quad \Gamma \vdash F_2 : Prop \quad F_1 \leftrightarrow^*_{\beta\eta} F_2}{\Gamma; \Pi \vdash u : F_2}\ (\leftrightarrow^*_{\beta\eta})$$

Les règles $(\forall E)$, $(\forall I)$ traitent des quantifications d'ordre un, deux, et encore d'autres. Noter la ressemblance entre les règles sur l'implication et les règles sur la quantification universelle. Il se trouve que ce système est encore cohérent, et ceci peut se démontrer en gros comme pour la logique d'ordre deux, en passant par un système similaire au système F_2 appelé F_ω. Ce dernier a aussi été inventé par Jean-Yves Girard, mais le fait qu'il normalise fortement est plus compliqué que pour F_2.

On peut concevoir des systèmes logiques encore plus expressifs. Utilisant l'égalité de Leibniz $\approx$ (voir exercice 44), on peut modéliser l'arithmétique $\mathbf{HA}_\omega$ d'ordre supérieur par l'ajout de constantes $\mathtt{0}$ de type ι, $\mathtt{S}$ de type $\iota \to \iota$, d'un récurseur R au niveau des preuves:

$$\frac{\Gamma; \Pi \vdash u : F[x := \mathtt{0}] \quad \Gamma; \Pi \vdash v : \forall y : \iota \cdot F[x := y] \Rightarrow F[x := \mathtt{S}(y)] \quad \Gamma \vdash e : \iota}{\Gamma; \Pi \vdash Ruve : F[x := e]}\ (Rec)$$

d'un récurseur au niveau des expressions:

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \iota \to \tau \to \tau \quad \Gamma \vdash e_3 : \iota}{\Gamma \vdash R_\tau e_1 e_2 e_3 : \tau}\ (Rec_\tau)$$

et de termes de preuve vérifiant les règles:

$$\frac{\Gamma \vdash e : \iota}{\Gamma ; \Pi \vdash c_1 e : \mathtt{0} \approx \mathtt{S}(e) \Rightarrow \bot}\ (\mathtt{0} \not\approx \mathtt{S}) \qquad \frac{\Gamma \vdash e_1 : \iota \quad \Gamma \vdash e_2 : \iota}{\Gamma ; \Pi \vdash c_2 e_1 e_2 : \mathtt{S}(e_1) \approx \mathtt{S}(e_2) \Rightarrow e_1 \approx e_2}\ (\mathtt{S} \not\approx \mathtt{S})$$

Ce système, ainsi que sa version classique $\mathbf{PA}_\omega$, sont encore cohérents, pour des raisons similaires aux précédents, par des résultats de normalisation forte.

Si l'on ajoute à ce genre de systèmes des types dépendants à la LF, des types inductifs généralisant le type $\mathbb{N}$ des entiers naturels, plus quelques autres constructions logiques, on aboutit à des systèmes comme le *calcul des constructions inductives*, qui sert de fondement au système Coq [BBC+99], et dont le pouvoir expressif est essentiellement celui de la théorie des ensembles avec une infinité de cardinaux fortement inaccessibles [Wer97, Acz99].

Étendre petit à petit des systèmes logiques semble ainsi une activité anodine, qui produit facilement des logiques cohérentes de plus en plus puissantes. Pourtant, de nombreux mathématiciens célèbres ont été à l'origine de systèmes logiques incohérents, dont on pouvait pourtant croire qu'ils ne poseraient pas de problèmes de cohérence a priori.

L'un de ces premiers systèmes est (une partie de) la théorie naïve des ensembles. Dans ce système qui est une extension de la logique du premier ordre, on a pour tout formule F des termes de la forme $\{i \mid F\}$ dénotant l'ensemble des i vérifiant F, et dont l'ensemble des variables libres est celui de F moins i — c'est très proche de la notation $\lambda i \cdot F$. On a aussi un symbole de prédicat binaire ε dénotant l'appartenance, et deux règles de déduction nouvelles:

$$\frac{\Gamma \vdash u : t\,\varepsilon\,\{i \mid F\}}{\Gamma \vdash \varepsilon_1 u : F[i := t]}\ (\varepsilon E) \qquad \frac{\Gamma \vdash u : F[i := t]}{\Gamma \vdash \varepsilon_2 u : t\,\varepsilon\,\{i \mid F\}}\ (\varepsilon I)$$

Lemme 14 *On peut déduire* $\vdash u : \bot$ *pour un certain terme* u *dans la théorie naïve des ensembles.*

Preuve : Il s'agit du contre-exemple de Bertrand Russell. Considérons la formule $F(i) \hat{=} \neg(i \varepsilon i)$ qui exprime que i ne s'appartient pas à lui-même, où $\neg G \hat{=} G \Rightarrow \bot$. Notons A le terme $\{i \mid F(i)\}$. On a:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \neg(A\,\varepsilon\,A) \vdash x : F(A)}\ (Ax)}{x : \neg(A\,\varepsilon\,A) \vdash \varepsilon_2 x : A\,\varepsilon\,A}\ (\varepsilon I)}{x : \neg(A\,\varepsilon\,A) \vdash x(\varepsilon_2 x) : \bot}\ (\Rightarrow E)}{\vdash \lambda x \cdot x(\varepsilon_2 x) : \neg\neg(A\,\varepsilon\,A)}\ (\Rightarrow I) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x : A\,\varepsilon\,A \vdash A\,\varepsilon\,\{i \mid \neg(i\,\varepsilon\,i)\}}\ (Ax)}{x : A\,\varepsilon\,A \vdash \varepsilon_1 x : \neg(A\,\varepsilon\,A)}\ (\varepsilon E)}{x : A\,\varepsilon\,A \vdash \varepsilon_1 x x : \bot}\ (\Rightarrow E)}{\vdash \lambda x \cdot \varepsilon_1 x x : \neg(A\,\varepsilon\,A)}\ (\Rightarrow I)}{\vdash (\lambda x \cdot x(\varepsilon_2 x))(\lambda x \cdot \varepsilon_1 x x) : \bot}\ (\Rightarrow E)$$

$\diamondsuit$

En somme, toute formule est déductible dans cette théorie. Noter la ressemblance entre le terme $(\lambda x \cdot x(\varepsilon_2 x))(\lambda x \cdot \varepsilon_1 x x)$ et le terme $(\lambda x \cdot x x)(\lambda x \cdot x x)$ qui boucle en λ-calcul pur. La ressemblance n'est pas complètement fortuite. Si on voulait démontrer que la théorie naïve des ensembles était cohérente comme nous l'avons fait jusqu'ici pour d'autres théories, nous essaierions de normaliser les preuves, et ceci se ferait naturellement en utilisant la règle $\varepsilon_1(\varepsilon_2 u) \to u$. Mais alors $(\lambda x \cdot x(\varepsilon_2 x))(\lambda x \cdot \varepsilon_1 x x) \to (\lambda x \cdot \varepsilon_1 x x)(\varepsilon_2(\lambda x \cdot \varepsilon_1 x x)) \to \varepsilon_1(\varepsilon_2(\lambda x \cdot \varepsilon_1 x x))(\varepsilon_2(\lambda x \cdot \varepsilon_1 x x)) \to (\lambda x \cdot \varepsilon_1 x x)(\varepsilon_2(\lambda x \cdot \varepsilon_1 x x))$, qui boucle. C'est heureux, car le système est justement incohérent.

En un sens, c'est parce que $\{i \in F\}$ agit comme un $\lambda i \cdot F$ non typé. On peut éviter le problème en le typant par des types simples, et alors on obtiendra essentiellement la logique d'ordre supérieur.

On peut se demander si on ne pourrait pas enrichir le système de types des expressions logiques de la logique d'ordre supérieur, en remplaçant les types simples par les types du système F_2, par exemple. On obtiendra ainsi le *système U^-*. Définissons-le formellement. On note α, β, $\ldots$, les variables de types, Π ce que nous notions $\forall$ en système F_2, autrement dit les types sont:

$$\mathcal{T} ::= B \mid TVar \mid \mathcal{T} \to \mathcal{T} \mid \Pi TVar \cdot \mathcal{T}$$

où B est un ensemble de types de base et $TVar$ dénote l'ensemble des variables de types. Mais, bien que cette extension du langage des formules ait l'air innocente, on a:

Théorème 11 ([Coq94]) *Le système U^- est incohérent: on peut déduire $\vdash u : \bot$ pour un certain terme u en U^-.*

L'argument est élaboré et ne sera pas donné ici: voir [Coq94], qui donne aussi quelques intuitions et quelques applications.

5 A-traduction et théorème de Kreisel-Friedman

Nous avons souvent parlé des versions classiques des systèmes logiques intuitionnistes que nous avons étudié: $\mathbf{PA}_1$ correspondant à $\mathbf{HA}_1$, $\mathbf{PA}_2$ à $\mathbf{HA}_2$, etc. Les systèmes intuitionnistes étaient plus faciles à étudier, notamment ils nécessitaient moins de règles de normalisation des preuves. Cependant, les systèmes classiques correspondants sont cohérents eux aussi, et de plus ils sont d'un usage beaucoup plus courant en mathématiques. On peut donc se demander quels sont les rapports précis entre les variantes intuitionnistes et classiques des systèmes logiques étudiés.

5.1 Non-non traductions et style par passage de continuations

Une partie claire du rapport entre logiques intuitionnistes et classiques est que toute preuve intuitionniste est une preuve classique, mais qu'il existe des preuves classiques non intuitionnistes. Cependant, ceci n'entame en rien le pouvoir expressif des logiques intuitionnistes comparées à leurs variantes classiques. Ce résultat est dû à Kurt Gödel, mais les traductions que nous utiliserons se rapprochent davantage de traductions dues à Kolmogorov ou à Kuroda.

Commençons par la logique propositionnelle avec $\Rightarrow$ et $\bot$ comme connecteurs logiques. Pour toute formule F, on définit sa *$\neg\neg$-traduction* $F^{\neg\neg}$ comme suit:

$$\begin{array}{rcll} F^{\neg\neg} & \hat{=} & \neg\neg F^\circ & \\ A^\circ & \hat{=} & A & \quad (A \text{ type de base, y compris } \bot) \\ (F \Rightarrow G)^\circ & \hat{=} & F^\circ \Rightarrow G^{\neg\neg} & \end{array}$$

Ceci peut se justifier comme suit: en logique classique, $F \Rightarrow G$ est non F ou G, c'est-à-dire aussi $\neg(F \wedge \neg G)$; or $F \wedge \neg G$ est équivalent à $(F \Rightarrow \neg G \Rightarrow \bot) \Rightarrow \bot = \neg(F \Rightarrow \neg\neg G)$. Ceci mène à l'idée que l'on peut définir $F^{\neg\neg}$ en remplaçant récursivement tous les $F \Rightarrow G$ par quelque chose ressemblant à $\neg\neg(F \Rightarrow \neg\neg G)$.

Rappelons que $\neg F$ est défini comme $F \Rightarrow \bot$. En particulier, $(\neg F)^{\neg\neg} = \neg\neg(F^\circ \Rightarrow \neg\neg\bot)$.

Lemme 15 *En logique propositionnelle classique, F et $F^{\neg\neg}$ sont logiquement équivalentes pour toute formule F. De plus, F est prouvable en logique classique si et seulement si $F^{\neg\neg}$ est prouvable en logique intuitionniste.*

Autrement dit, modulo l'ajout de suffisamment de doubles négations saupoudrées dans F, on obtient une formule qui signifie la même chose que F en classique et qui est prouvable en intuitionniste dès que F est prouvable en classique. Par exemple, $\neg\neg A \Rightarrow A$ n'est pas prouvable en intuitionniste, mais $(\neg\neg A \Rightarrow A)^{\neg\neg} = \neg\neg(((A \Rightarrow \neg\neg\bot) \Rightarrow \neg\neg\bot) \Rightarrow \neg\neg A)$ l'est. (On notera au passage que cette traduction n'est pas particulièrement économe en négations. Il en existe des plus économes.)

Preuve : Il peut servir de s'exercer à construire quelques preuves intuitionnistes de base. D'abord, si $u : G$, alors $u^- \hat{=} \lambda k \cdot ku$ est de type $\neg\neg G$. Ensuite, si $u : \neg\neg\neg G$, alors $u^0 \hat{=} \lambda y \cdot u(\lambda z \cdot zy)$ est de type $\neg G$. Et si $u : \neg\neg(G \Rightarrow H)$ et $v : \neg\neg G$, alors $u \star v \hat{=} \lambda k \cdot u(\lambda x \cdot v(\lambda y \cdot k(xy)))$ est de type $\neg\neg H$ (où $k : \neg H$, $x : G \Rightarrow H$, $y : G$).

L'équivalence entre F et $F^{\neg\neg}$ en classique signifie qu'il existe un $\lambda\mathcal{C}$-terme clos u_F de type $F \Rightarrow F^{\neg\neg}$ et un $\lambda\mathcal{C}$-terme clos v_F de type $F^{\neg\neg} \Rightarrow F$. On définit ces termes par récurrence structurelle sur F: pour tout type de base A, $u_A \hat{=} \lambda x \cdot x^-$, $v_A \hat{=} \lambda x \cdot \mathcal{C}x$; enfin $u_{F \Rightarrow G} \hat{=} \lambda x \cdot \lambda k \cdot k(\lambda y \cdot u_G(x(v_F y^-)))$ (où $x : F \Rightarrow G$, $k : \neg(F^\circ \Rightarrow G^{\neg\neg})$, $y : F^\circ$); et $v_{F \Rightarrow G} \hat{=} \lambda x \cdot \lambda y \cdot v_G((x \star u_F y)^0)$ (où $x : \neg\neg(F^\circ \Rightarrow G^{\neg\neg})$, $y : F$; ceci est bien défini car $x \star u_F y$ est de type $\neg\neg G^{\neg\neg} = \neg\neg\neg(\neg G^\circ)$, donc $(x \star u_F y)^0$ est bien défini).

Si $F^{\neg\neg}$ est prouvable en intuitionniste, autrement dit s'il existe un $\lambda\nabla$-terme clos u de type $F^{\neg\neg}$, alors on peut considérer u comme un $\lambda\mathcal{C}$-terme en posant $\nabla v \hat{=} \mathcal{C}(\lambda z \cdot v)$ pour tout v, avec z non libre dans v. Donc $v_F u$ est un $\lambda\mathcal{C}$-terme clos de type F: F est prouvable en classique.

Réciproquement, montrons que si F est prouvable en classique, alors $F^{\neg\neg}$ est prouvable en intuitionniste. Pour ceci, considérons un $\lambda\mathcal{C}$-terme u quelconque tel que $\vdash u : F$, et construisons un $\lambda\nabla$-terme u' tel que $\vdash u' : F^{\neg\neg}$. Plus généralement, construisons par récurrence structurelle sur le $\lambda\mathcal{C}$-terme u tel que $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$, un $\lambda\nabla$-terme u^* tel que $x_1 : F_1^\circ, \ldots, x_n : F_n^\circ \vdash u^* : F^{\neg\neg}$. Comme $F^{\neg\neg} = \neg\neg F^\circ$, on va construire u^* comme $\langle u\rangle^\circ k$, où $x_1 : F_1^\circ, \ldots, x_n : F_n^\circ, k : \neg F^\circ \vdash \langle u\rangle^\circ k : \bot$.

Si u est une variable, alors c'est un x_i, $1 \leq i \leq n$, et on pose $\langle u\rangle^\circ k \hat{=} ku$.

Si u est de la forme $\lambda x \cdot v$ de type $G \Rightarrow H$, alors par hypothèse de récurrence on a pu dériver $x_1 : F_1^\circ, \ldots, x_n : F_n^\circ, x : G^\circ \vdash v^* : H^{\neg\neg}$, donc $\langle u\rangle^\circ k \hat{=} k(\lambda x \cdot v^*) = k(\lambda x \cdot \lambda k' \cdot \langle v\rangle^\circ k)$ convient.

Si u est de la forme vw avec v de type $G \Rightarrow H$ et w de type G, alors par hypothèse de récurrence dans le contexte $x_1 : F_1^\circ, \ldots, x_n : F_n^\circ$, v^* est de type $\neg\neg(G^\circ \Rightarrow H^{\neg\neg})$ et w^* est de type $G^{\neg\neg} = \neg\neg G^\circ$, donc u^* peut être défini comme $(v^* \star w^*)^0$. Plus explicitement, comme $\lambda y \cdot (\lambda k \cdot v^*(\lambda x \cdot w^*(\lambda y \cdot k(xy))))(\lambda z \cdot zy)$, qui se réduit en $\lambda k \cdot v^*(\lambda x \cdot w^*(\lambda y \cdot xyk))$ (à α-conversion près). On pose donc $\langle u\rangle^\circ k \hat{=} \langle v\rangle^\circ(\lambda x \cdot \langle w\rangle^\circ(\lambda y \cdot xyk))$.

Si u est de la forme $\mathcal{C}v$ avec v de type $\neg\neg F$, alors par hypothèse de récurrence dans le contexte $x_1 : F_1^\circ, \ldots, x_n : F_n^\circ$, v^* est de type $\neg\neg((F^\circ \Rightarrow \neg\neg\bot) \Rightarrow \neg\neg\bot)$. Donc si $k : \neg F^\circ$, $v^*(\lambda x \cdot x(\lambda y, k' \cdot ky)(\lambda z \cdot z))$ est de type $\bot$ (où $x : (F^\circ \Rightarrow \neg\neg\bot) \Rightarrow \neg\neg\bot$, $y : F^\circ$, $k' : \neg\bot$, $z : \bot$). On pose donc $\langle u\rangle^\circ k \hat{=} \langle v\rangle^\circ(\lambda x \cdot x(\lambda y, k' \cdot ky)(\lambda z \cdot z))$. ◊

Le terme $\langle u\rangle^\circ k$ est défini par:

$$\begin{array}{rcl}
\langle x\rangle^\circ k & \hat{=} & kx \\
\langle \lambda x \cdot u\rangle^\circ k & \hat{=} & k(\lambda x \cdot \lambda k' \cdot \langle u\rangle^\circ k') \\
\langle uv\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot \langle v\rangle^\circ(\lambda y \cdot xyk)) \\
\langle \mathcal{C}u\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot x(\lambda y, k' \cdot ky)(\lambda z \cdot z))
\end{array}$$

Les trois premières lignes sont connues comme la *traduction par valeur* de Plotkin [Plo76].

Dans cette définition, k est toujours d'un type de la forme $\neg F^\circ$, et joue le rôle d'une *continuation* acceptant des valeurs de retour de type F°. La traduction $u \mapsto u^* = \lambda k \cdot \langle u\rangle^\circ k$ est ce qu'on appelle une *traduction en style de passage de continuations* (*continuation-passing style* ou *CPS* en anglais). Elle exprime en fait directement une sémantique par continuations (comparer avec la section 5.2 en partie I): pour évaluer une variable x dans une continuation k, on retourne (la valeur de) la variable par k; pour évaluer une abstraction $\lambda x \cdot u$, on retourne par la continuation k la fonction qui prend (la valeur de) l'argument x et la continuation k' qui sera courante au moment de l'application de l'abstraction à son argument, et qui retourne la valeur du corps u de l'abstraction dans la continuation k'; pour évaluer une application uv dans la continuation k, on évalue u dans une continuation qui récupère la valeur x de u, évalue ensuite v dans une continuation qui récupère la valeur y de v, enfin applique x à y et la continuation courante k; pour évaluer $\mathcal{C}u$ dans la continuation k, on évalue u dans la continuation qui récupère la valeur x de u: x est censée être une fonction prenant une continuation en argument, ici $\lambda y, k' \cdot ky$ qui jette la continuation courante k' pour renvoyer y par l'ancienne continuation courante k capturée par $\mathcal{C}$; l'argument $\lambda z \cdot z$ est la continuation fournie à x, qui est triviale car l'application de x est obligé d'appliquer une autre continuation et ne retourne pas.

La traduction $u, k \mapsto \langle u\rangle^\circ k$ a donc non seulement un sens logique (le plongement de la logique classique en logique intuitionniste), mais aussi un sens calculatoire. Les exercices suivants précisent cette remarque. (On pourra s'inspirer des résultats de la partie I.)

Exercice 48 *Montrer que* $\langle u\rangle^\circ k$ *est bien défini, au sens où deux termes* α*-équivalents* u *et* v *donnent deux termes* α*-équivalents* $\langle u\rangle^\circ k$ *et* $\langle v\rangle^\circ k$*. (Montrer que* $\langle u[x := y]\rangle^\circ k = \langle u\rangle^\circ k[x := y]$ *dès que ni* x *ni* y *n'est libre dans* k*.)*

Exercice 49 *Rappelons qu'une* P-valeur V *est une variable ou une abstraction. Montrer que pour toute continuation* k*, pour tout* $\lambda\mathcal{C}$*-terme* u*, pour toute P-valeur* V*,* $\langle u[x := V]\rangle^\circ k \;{}^*\!\!\leftarrow \langle V\rangle^\circ(\lambda x \cdot \langle u\rangle^\circ k)$*. (On pourra d'une part s'aider de l'exercice précédent, d'autre part lorsque* $V = \lambda y \cdot v$*, montrer que* $\langle u\rangle^\circ k[x := \lambda y, k' \cdot \langle v\rangle^\circ k'] = \langle u[x := \lambda y \cdot v]\rangle^\circ k$*.)*

Exercice 50 *Rappelons que la règle* (β_v) *est:* $(\lambda x \cdot u)V \to u[x := V]$*, où* V *est une* P-valeur*. Montrer que la traduction* $u, k \mapsto \langle u\rangle^\circ k$ *interprète correctement* (β_v)*, au sens où, si* $u \to v$ *par la règle* (β_v)*, alors* $\langle u\rangle^\circ k \to^+ \langle v\rangle^\circ k$ *pour toute continuation* k*. On montrera d'abord en s'aidant des exercices précédents que* $\langle(\lambda x \cdot u)V\rangle^\circ k \to^+ \langle u[x := V]\rangle^\circ k$ *pour tout* $\lambda\mathcal{C}$*-terme* u *et toute P-valeur* V*.*

Exercice 51 ( ) *On rappelle les règles suivantes:*

$$\begin{array}{lrcl}
(\mathcal{C}_L) & \mathcal{C}uv & \to & \mathcal{C}(\lambda k' \cdot u(\lambda f \cdot k'(fv))) \\
(\mathcal{C}_R) & V(\mathcal{C}u) & \to & \mathcal{C}(\lambda k' \cdot u(\lambda x \cdot k'(Vx))) \\
(\eta\mathcal{C}) & \mathcal{C}(\lambda k \cdot ku) & \to & u \qquad (k \notin \mathrm{fv}(u))
\end{array}$$

Montrer que si $u \to v$ *par* $(\mathcal{C}_L)$ *ou* $(\mathcal{C}_R)$ *ou* $(\eta\mathcal{C})$*, alors* $\langle u\rangle^\circ k =_{\beta\eta} \langle v\rangle^\circ k$*. Peut-on se passer de la règle* (η) *?*

Exercice 52 *Montrer qu'en général le fait que* u *se réduise en* v *par* (β) *ou* (η) *n'implique pas que* $\langle u\rangle^\circ k =_{\beta\eta} \langle v\rangle^\circ k$*. (On considérera le* (β)*-rédex* $(\lambda x, y \cdot x)(zz')$ *et l'*(η)*-rédex* $\lambda x \cdot yzx$*, où* x*,* y*,* z*,* z' *sont des variables distinctes.)*

Exercice 53 *La* traduction par nom *de Plotkin est définie sur les formules par:*

$$F^* \hat{=} \neg\neg F^\bullet \qquad A^\bullet \hat{=} A \quad (A \text{ type de base}) \qquad (F \Rightarrow G)^\bullet \hat{=} F^* \Rightarrow G^*$$

On a donc $(\neg F)^\bullet = F^* \Rightarrow \neg\neg\bot$*, donc* $(\neg\neg F)^\bullet = \neg\neg(F^* \Rightarrow \neg\neg\bot) \Rightarrow \neg\neg\bot$*. Montrer que le lemme 15 est encore valide en remplaçant* $F^{\neg\neg}$ *par* F^**. Vérifier en particulier que la traduction obtenue des* $\lambda\mathcal{C}$*-termes en* $\lambda\nabla$*-termes est:*

$$\begin{array}{rcl}
\langle x\rangle^\bullet k & \hat{=} & xk \\
\langle \lambda x \cdot u\rangle^\bullet k & \hat{=} & k(\lambda x \cdot \lambda k' \cdot \langle u\rangle^\bullet k') \\
\langle uv\rangle^\bullet k & \hat{=} & \langle u\rangle^\bullet(\lambda f \cdot f(\lambda k' \cdot \langle v\rangle^\bullet k')k) \\
\langle \mathcal{C}u\rangle^\bullet k & \hat{=} & \langle u\rangle^\bullet(\lambda x \cdot x(\lambda k' \cdot k'(\lambda y, k'' \cdot yk))(\lambda z \cdot z))
\end{array}$$

Exercice 54 ( ) *Montrer, en s'inspirant de l'exercice 48, que* $\langle u\rangle^\bullet k$ *est bien défini modulo* α*-conversion; en s'inspirant de l'exercice 49, que* $\langle u\rangle^\bullet k[x := \lambda k' \cdot \langle v\rangle^\bullet k'] \to^* \langle u[x := v]\rangle^\bullet k$ *pour tous termes* u*,* v*; en s'inspirant de l'exercice 51, montrer que si* $u \to v$ *par* (β) *ou par* $(\eta\mathcal{C})$*, alors* $\langle u\rangle^\bullet k \to^+ \langle v\rangle^\bullet k$ *(même si* v *n'est pas une P-valeur), si* $u \to v$ *par* $(\mathcal{C}_L)$*, alors* $\langle u\rangle^\bullet k =_\beta \langle v\rangle^\bullet k$*, mais que les règles* $(\mathcal{C}_R)$ *et* (η) *ne donnent en général pas lieu à des égalités valides à travers la traduction.*

On peut étendre ces $\neg\neg$-traductions et les traductions en style par passage de continuations correspondantes à la logique et à l'arithmétique du premier ordre. La traduction par valeur $F^{\neg\neg}$ s'étend ainsi comme suit (mais il y a de nombreuses autres possibilités) aux autres constructions de la logique du premier ordre:

$$\begin{array}{rcl}
(F \wedge G)^\circ & \hat{=} & F^\circ \wedge G^\circ \\
(F \vee G)^\circ & \hat{=} & F^\circ \vee G^\circ \\
(\forall i \cdot F)^\circ & \hat{=} & \forall i \cdot F^{\neg\neg} \\
(\exists i \cdot F)^\circ & \hat{=} & \exists i \cdot F^\circ
\end{array}$$

et la traduction en style par passage de continuations $u, k \mapsto \langle u\rangle^\circ k$ s'étend alors en:

$$\begin{array}{rcl}\langle\langle u, v\rangle\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot \langle v\rangle^\circ(\lambda y \cdot k\langle x, y\rangle))\\ \langle\pi_1 u\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot k(\pi_1 x))\\ \langle\pi_2 u\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot k(\pi_2 x))\end{array}$$

pour le fragment conjonctif ($\wedge$), en:

$$\begin{array}{rcl}\langle\iota_1 u\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot k(\iota_1 x))\\ \langle\iota_2 u\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot k(\iota_2 x))\\ \left\langle \text{case } u \left\{ \begin{array}{l}\iota_1 x_1 \mapsto v_1\\ \iota_2 x_2 \mapsto v_2\end{array}\right\}\right\rangle^\circ k & \hat{=} & \langle u\rangle^\circ\left(\lambda x \cdot \text{case } x \left\{ \begin{array}{l}\iota_1 x_1 \mapsto \langle v_1\rangle^\circ k\\ \iota_2 x_2 \mapsto \langle v_2\rangle^\circ k\end{array}\right\}\right)\end{array}$$

pour le fragment disjonctif ($\vee$), en:

$$\begin{array}{rcl}\langle\lambda i \cdot u\rangle^\circ k & \hat{=} & k(\lambda i \cdot \lambda k' \cdot \langle u\rangle^\circ k')\\ \langle ut\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot xtk)\end{array}$$

pour la quantification universelle du premier ordre, en:

$$\begin{array}{rcl}\langle\iota t u\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda x \cdot k(\iota t x))\\ \langle\text{case } u\,\{\iota i x \mapsto v\}\rangle^\circ k & \hat{=} & \langle u\rangle^\circ(\lambda y \cdot \text{case } y\,\{\iota i x \mapsto \langle v\rangle^\circ k\})\end{array}$$

pour la quantification existentielle du premier ordre.

Désormais, nous considérerons les versions des logiques classique et intuitionniste, ainsi que des arithmétiques $\mathbf{PA}_1$ et $\mathbf{HA}_1$ qui incluent $\wedge$, $\vee$, et $\exists$, obéissant aux règles de déduction $(\wedge I)$, $(\wedge E_1)$, $(\wedge E_2)$, $(\vee I_1)$, $(\vee I_2)$, $(\vee E)$, $(\exists I)$, $(\exists E)$.

Exercice 55 *Montrer que, en logique classique du premier ordre, F et $F^{\neg\neg}$ sont logiquement équivalentes. De plus, F est prouvable en logique classique du premier ordre si et seulement si $F^{\neg\neg}$ est prouvable en logique intuitionniste du premier ordre. (Étendre la preuve du lemme 15, et utiliser la traduction en CPS ci-dessus.)*

Pour passer à l'arithmétique, il nous faut étendre notre traduction en CPS pour traiter de $r_0 : \mathtt{0} \approx \mathtt{0}$ et le récurseur:

$$\begin{array}{rcl}\langle r_0\rangle^\circ k & \hat{=} & kr_0\\ \langle Ruvt\rangle^\circ k & \hat{=} & \langle v\rangle^\circ(\lambda x \cdot R(\lambda k' \cdot \langle u\rangle^\circ k')(\lambda j \cdot \lambda y, k'' \cdot xj(\lambda x' \cdot y(\lambda y' \cdot x'y'k'')))tk)\end{array}$$

Lemme 16 *F est prouvable en $\mathbf{PA}_1$ si et seulement si $F^{\neg\neg}$ est prouvable en $\mathbf{HA}_1$.*

Preuve : On a déjà la traduction en CPS, il suffit de vérifier qu'elle définit des objets des bons types. Dans le cas de r_0, on doit vérifier que si $k : \neg(\mathtt{0} \approx \mathtt{0})$, alors $kr_0 : \bot$; c'est évident. Dans le cas du récurseur, dans la formule pour $\langle Ruvt\rangle^\circ k$ ci-dessus, on a par hypothèse $k : \neg(F[i := t])^\circ$, et en supposant que $x : \forall j \cdot \neg\neg((F[i := j])^\circ \Rightarrow (F[i := \mathtt{S}(j)])^{\neg\neg})$, que $y : (F[i := j])^{\neg\neg}$, que $k'' : \neg(F[i := \mathtt{S}(j)])^\circ$, que $x' : (F[i := j])^\circ \Rightarrow (F[i := \mathtt{S}(j)])^{\neg\neg}$, que $y' : (F[i := j])^\circ$, on a: $x'y'k''$ est de type $\bot$, donc $\lambda y' \cdot x'y'k''$ est de type $\neg(F[i := j])^\circ$, donc $y(\lambda y' \cdot x'y'k'')$ est de type $\bot$, donc $\lambda x' \cdot y(\lambda y' \cdot x'y'k'')$ est de type $\neg((F[i := j])^\circ \Rightarrow (F[i := \mathtt{S}(j)])^{\neg\neg})$. Comme xj est de type $\neg\neg((F[i := j])^\circ \Rightarrow (F[i := \mathtt{S}(j)])^{\neg\neg})$, $xj(\lambda x' \cdot y(\lambda y' \cdot x'y'k''))$ est de type $\bot$, donc $\lambda k'' \cdot xj(\lambda x' \cdot y(\lambda y' \cdot x'y'k''))$ est de type $\neg\neg(F[i := \mathtt{S}(j)])^\circ = (F[i := \mathtt{S}(j)])^{\neg\neg} = F^{\neg\neg}[i := \mathtt{S}(j)]$. (Une récurrence structurelle facile montre en effet que $G^{\neg\neg}[i := t] = (G[i := t])^{\neg\neg}$.) Comme y est supposé de type $(F[i := j])^{\neg\neg} = F^{\neg\neg}[i := j]$, le terme $v' \hat{=} \lambda j \cdot \lambda y, k'' \cdot xj(\lambda x' \cdot y(\lambda y' \cdot x'y'k''))$ est de type $\forall j \cdot F^{\neg\neg}[i := j] \Rightarrow F^{\neg\neg}[i := \mathtt{S}(j)]$. D'autre part le terme $u' \hat{=} \lambda k' \cdot \langle u\rangle^\circ k'$ est de type $(F[i := \mathtt{0}])^{\neg\neg} = F^{\neg\neg}[i := \mathtt{0}]$. Donc $Ru'v't$ est de type $F^{\neg\neg}[i := t] = (F[i := t])^{\neg\neg}$. Comme $k : \neg(F[i := t])^\circ$, $Ru'v'tk$ est de type $\bot$, donc $\lambda x \cdot Ru'v'tk$ est de type $\neg(\forall j \cdot \neg\neg((F[i := j])^\circ \Rightarrow (F[i := \mathtt{S}(j)])^{\neg\neg}))$. On en déduit que $\langle v\rangle^\circ(\lambda x \cdot Ru'v'tk)$ est de type $\bot$; or $\langle v\rangle^\circ(\lambda x \cdot Ru'v'tk)$ est exactement $\langle Ruvt\rangle^\circ k$. $\Diamond$

Exercice 56 ( ) *La démonstration du lemme 16 est incomplète, et le manque le plus flagrant est que nous n'avons pas traité la règle* $(\leftrightarrow^*_{\mathbb{N}})$*. La traiter directement est difficile, et nous transformons d'abord la preuve de* F *en* $\mathbf{PA}_1$*:*

1. *On dit qu'une instance* $\Gamma, F \vdash F$ *de* (Ax) *est* atomique *si* F *est une formule de la forme* $s \approx t$ *(une formule atomique, à l'exclusion de* $\bot$*). On dit qu'une preuve en* $\mathbf{PA}_1$ *est* η-longue *si tous ses axiomes sont atomiques. Montrer que pour tout séquent* $\Gamma, F \vdash F$ *est prouvable par une preuve* η*-longue. (On effectuera une récurrence sur* F*.)*

2. *En déduire que tout séquent prouvable en* $\mathbf{PA}_1$ *l'est par une preuve* η*-longue.*

3. *Une preuve en* $\mathbf{PA}_1$ *est* spéciale *si les seules instances utilisées de* $(\leftrightarrow^*_{\mathbb{N}})$ *le sont sur les conclusions des règles* $(\bot E)$ *ou* (Ax)*. Montrer qu'on peut transformer n'importe quelle preuve* π *en* $\mathbf{PA}_1$ *d'un séquent en une preuve spéciale* π' *du même séquent. De plus, si* π *est* η*-longue, alors* π' *est* η*-longue.*

4. *Reprendre la preuve du lemme 16, en montrant que si* π *est une dérivation spéciale* η*-longue de* $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$ *en* $\mathbf{PA}_1$*, alors* $x_1 : F_1^\circ, \ldots, x_n : F_n^\circ, k : \neg F^\circ \vdash \langle u \rangle^\circ k : \bot$ *est dérivable en* $\mathbf{HA}_1$*.*

Exercice 57 *Redémontrer que* $\mathbf{PA}_1$ *est cohérente (cf. exercice 29) en utilisant le lemme 16.*

Exercice 58 *Étendre la* $\neg\neg$*-traduction au second ordre (on démontrera au passage que* $F^{\neg\neg}[P(k_1, \ldots, k_n) := G^\circ] = (F[P(k_1, \ldots, k_n) := G])^{\neg\neg}$*), et déduire du théorème 10 que* $\mathbf{PA}_2$ *est cohérente. Comparer avec la preuve de l'exercice 47.*

Exercice 59 (Glivenko) *Par une traduction en CPS modifiée, montrer que si* $x_1 : F_1, \ldots, x_n : F_n \vdash u : F$ *est prouvable en logique propositionnelle classique, alors* $x_1 : \neg\neg F_1, \ldots, x_n : \neg\neg F_n \vdash u^* : \neg\neg F$ *pour un certain term* u^* *à trouver. (Traiter aussi le cas de* $\wedge$*,* $\vee$*. Montrer que ceci s'étend aux quantifications existentielles mais pas aux quantifications universelles.) En déduire le théorème de Glivenko: pour toute formule ne contenant que des quantifications existentielles,* $\neg F$ *est prouvable en logique classique si et seulement si elle est prouvable en logique intuitionniste.*

5.2 Formules décidables et classiques

En arithmétique de Heyting, les formules atomiques (les types de base) sont des égalités $s \approx t$. Il se trouve que ces égalités sont démontrables en $\mathbf{PA}_1$ si et seulement si elles le sont en $\mathbf{HA}_1$ — on n'a pas besoin de $\neg\neg$-traduction. Une question naturelle est alors: quelles sont les formules de l'arithmétique qui sont prouvables en $\mathbf{PA}_1$ si et seulement si elles le sont en $\mathbf{HA}_1$? Georg Kreisel a montré dans les années soixante que les formules Π^0_2 ont cette propriété, et sa démonstration a été simplifiée par la suite par Harvey Friedman [Fri78]. Les formules Π^0_2 (prononcer: "pi zéro deux") sont les formules de la forme $\forall i_1, \ldots, i_m \cdot \exists j_1, \ldots, j_n \cdot F$, où F est sans quantificateur, ou du moins n'a que des instances bénignes des quantificateurs. En général:

Définition 2 *On définit l'ordre* $\leq$ *sur les entiers en* $\mathbf{HA}_1$ *en posant que la notation* $s{\leq}t$ *abrège* $\exists k \cdot k + s \approx t$*. Les* quantifications bornées $\forall i{\leq}t \cdot F$ *et* $\exists i{\leq}t \cdot F$*, lorsque* i *n'est pas libre dans* t*, sont des abréviations pour* $\forall i \cdot i{\leq}t \Rightarrow F$ *et* $\exists i \cdot i{\leq}t \wedge F$*.*

On définit les classes de formules Δ^0_0*,* Σ^0_n *et* Π^0_n *par récurrence sur* $n \geq 0$*:* $\Delta^0_0 = \Sigma^0_0 = \Pi^0_0$ *est la classe des formules dont toutes les quantifications sont bornées.* Σ^0_{n+1} *est la classe des formules de la forme* $\exists i_1, \ldots, i_m \cdot F$ *avec* $m \geq 0$*,* $F \in \Pi^0_n$*;* Π^0_{n+1} *est la classe des formules de la forme* $\forall i_1, \ldots, i_m \cdot F$ *avec* $m \geq 0$*,* $F \in \Sigma^0_n$*.*

Ces classes ainsi que leurs inclusions (sous formes de flèches) sont représentées en figure 1.

Nous allons établir au lemme 19 que toute formule Δ^0_0 est *décidable* en $\mathbf{HA}_1$:

Définition 3 *Une formule* F *est* décidable *en* $\mathbf{HA}_1$ *si et seulement si* $F \vee \neg F$ *est prouvable en* $\mathbf{HA}_1$*. Une formule* F *est* classique *en* $\mathbf{HA}_1$ *si et seulement si* $\neg\neg F \Rightarrow F$ *est prouvable en* $\mathbf{HA}_1$*.*

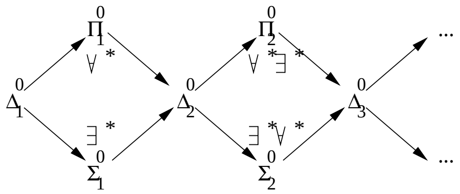

Figure 1: La hiérarchie arithmétique

Notons que:

Lemme 17 *Toute formule décidable est classique en* $\mathbf{HA}_1$.

Preuve : Soit F une formule décidable, et soit u un $\lambda\nabla R$-terme clos de type $F \vee \neg F$. Alors $cl_F(u) \hat{=} \lambda x \cdot \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto x_1 \\ \iota_2 x_2 \mapsto \nabla(x x_2) \end{array} \right\}$ est une preuve de $\neg\neg F \Rightarrow F$. $\diamondsuit$

Lemme 18 *Toute égalité* $s \approx t$ *est décidable en* $\mathbf{HA}_1$.

Preuve : On démontre $\forall i \cdot i \approx t \vee \neg i \approx t$ par récurrence sur t, puis sur i. Définissons:

$$\begin{array}{rcll} a & \hat{=} & \iota_1 r_0 & : \mathtt{0} \approx \mathtt{0} \vee \neg \mathtt{0} \approx \mathtt{0} \\ b(j) & \hat{=} & \iota_2(\lambda x \cdot x) & : \mathtt{S}(j) \approx \mathtt{0} \vee \neg \mathtt{S}(j) \approx \mathtt{0} \\ c & \hat{=} & \lambda i \cdot Ra(\lambda j \cdot \lambda x \cdot b(j))i & : \forall i \cdot i \approx \mathtt{0} \vee \neg i \approx \mathtt{0} \\ d(j) & \hat{=} & \iota_2(\lambda x \cdot x) & : \mathtt{0} \approx \mathtt{S}(j) \vee \neg \mathtt{0} \approx \mathtt{S}(j) \\ e(j) & \hat{=} & \lambda x \cdot \lambda i \cdot R(d(j))(\lambda j' \cdot \lambda y \cdot x j')i & : (\forall i \cdot i \approx j \vee \neg i \approx j) \Rightarrow (\forall i \cdot i \approx \mathtt{S}(j) \vee \neg i \approx \mathtt{S}(j)) \\ u & \hat{=} & Rc(\lambda j \cdot e(j))t & : \forall i \cdot i \approx t \vee \neg i \approx t \end{array}$$

Pour aider à vérifier les types annoncés, vérifier que dans la définition de $e(j)$, xj' est de type $j' \approx j \vee \neg j' \approx j$, donc aussi $\mathtt{S}(j') \approx \mathtt{S}(j) \vee \neg \mathtt{S}(j') \approx \mathtt{S}(j)$. Le terme us est la preuve souhaitée de $s \approx t \vee \neg s \approx t$. $\diamondsuit$

Lemme 19 *Toute formule* F *de* $\mathbf{HA}_1$ *ne contenant que des quantifications bornées est décidable.*

Preuve : Par récurrence structurelle sur $Sk(F)$, on construit un terme d_F de type $F \vee \neg F$.

Si F est une égalité $s \approx t$, alors on définit $d_{s \approx t}$ comme étant le terme us de la preuve du lemme 18.

Si $F = \bot$, alors $d_\bot \hat{=} \iota_2(\lambda x \cdot x)$.

Dans les cas des connecteurs $\Rightarrow$, $\wedge$, $\vee$, l'idée est de décrire leurs tables de vérité. Par exemple, on $G_1 \Rightarrow G_2$ est décidable pour la raison suivante: comme G_1 est décidable par hypothèse de récurrence, G_1 est soit vrai soit faux, de même G_2 est soit vrai soit faux; mais si G_1 est vrai et G_2 faux, alors $G_1 \Rightarrow G_2$ est faux, et $G_1 \Rightarrow G_2$ est vrai dans les trois autres cas. Formellement:

$$\begin{array}{rcll} d_{G_1 \Rightarrow G_2} & \hat{=} & \text{case } d_{G_1} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } d_{G_2} \left\{ \begin{array}{l} \iota_1 y_1 \mapsto \iota_1(\lambda x \cdot y_1) \\ \iota_2 y_2 \mapsto \iota_2(\lambda x \cdot y_2(x x_1)) \end{array} \right\} \\ \iota_2 x_2 \mapsto \iota_1(\lambda x \cdot \nabla(x x_2)) \end{array} \right\} & : (G_1 \Rightarrow G_2) \vee \neg(G_1 \Rightarrow G_2) \\ d_{G_1 \wedge G_2} & \hat{=} & \text{case } d_{G_1} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } d_{G_2} \left\{ \begin{array}{l} \iota_1 y_1 \mapsto \iota_1 \langle x_1, y_1 \rangle \\ \iota_2 y_2 \mapsto \iota_2(\lambda x \cdot y_2(\pi_2 x)) \end{array} \right\} \\ \iota_2 x_2 \mapsto \lambda x \cdot x_2(\pi_1 x) \end{array} \right\} & : (G_1 \wedge G_2) \vee \neg(G_1 \wedge G_2) \\ a & \hat{=} & \lambda k, k', x \cdot \text{case } x \left\{ \begin{array}{l} \iota_1 x_1 \mapsto k x_1 \\ \iota_2 x_2 \mapsto k' x_2 \end{array} \right\} & : \neg G_1 \Rightarrow \neg G_2 \Rightarrow \neg(G_1 \vee G_2) \\ d_{G_1 \vee G_2} & \hat{=} & \text{case } d_{G_1} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1(\iota_1 x_1) \\ \iota_2 x_2 \mapsto \text{case } d_{G_2} \left\{ \begin{array}{l} \iota_1 y_1 \mapsto \iota_1(\iota_2 y_1) \\ \iota_2 y_2 \mapsto \iota_2(a x_2 y_2) \end{array} \right\} \end{array} \right\} & : (G_1 \vee G_2) \vee \neg(G_1 \vee G_2) \end{array}$$

Les cas des quantifications bornées F sont plus difficiles. L'idée est que F ne quantifiant que sur les entiers de 0 à t, on peut prouver F par récurrence, en énumérant tous les entiers de 0 à t. Définissons d'abord quelques termes de preuve auxiliaires démontrant quelques évidences dont nous aurons besoin (on note $G(i)$ une formule quelconque et $G(t)$ la formule $G(i)[i := t]$ pour plus de clarté). Rappelons (exercice 27) qu'il existe en $\mathbf{HA}_1$ un terme de preuve $rep_{k\cdot H(k)}$ pour toute formule $H(k)$, de type $\forall i, j \cdot i \approx j \Rightarrow H(i) \Rightarrow H(j)$.

$$
\begin{array}{rcll}
nlS0(s) & \hat{=} & \lambda x \cdot \text{case } x\,\{\iota k, y \mapsto y\} & : \neg \mathtt{S}(s) \leq \mathtt{0} \\
rep0S(s) & \hat{=} & \lambda x \cdot R(\lambda y \cdot x)(\lambda j \cdot \lambda y, z \cdot \nabla(nlS0(j)z))s & : G(\mathtt{0}) \Rightarrow s \leq \mathtt{0} \Rightarrow G(s) \\
repS0(s) & \hat{=} & R(\lambda x, y \cdot x)(\lambda j \cdot \lambda x, y, z \cdot \nabla(nlS0(j)z))s & : G(s) \Rightarrow s \leq \mathtt{0} \Rightarrow G(\mathtt{0}) \\
l0 & \hat{=} & \iota \mathtt{0} r_0 & : \mathtt{0} \leq \mathtt{0} \\
n0(s) & \hat{=} & Rr_0(\lambda j \cdot \lambda x \cdot x)s & : \mathtt{0} + s \approx s \\
lrefl(s) & \hat{=} & \iota \mathtt{0}(n0(s)) & : s \leq s \\
refl & \hat{=} & \lambda i \cdot Rr_0(\lambda j \cdot \lambda x \cdot x)i & : \forall i \cdot i \approx i \\
nS(s,t) & \hat{=} & R(refl(\mathtt{S}(s)))(\lambda j \cdot \lambda x \cdot x)t & : \mathtt{S}(s) + t \approx \mathtt{S}(s+t) \\
com(s,t) & \hat{=} & R(n0(t))(\lambda j \cdot \lambda x \cdot rep_{k\cdot \mathtt{S}(j)+t \approx \mathtt{S}(k)}(j+t)(t+j)x(nS(j,t)))s & : s+t \approx t+s \\
lch_0(k,s,t) & \hat{=} & R(\lambda x \cdot \iota_2(x))(\lambda j \cdot \lambda x, y \cdot \iota_1(\iota j(rep_{k\cdot \mathtt{S}(k) \approx t}(s+j)(j+s)(com(s,j))y)))k & : s+k \approx t \Rightarrow \mathtt{S}(s) \leq t \vee s \approx t \\
lch_1(s,t) & \hat{=} & \lambda x \cdot \text{case } x\,\{\iota k y \mapsto lch_0(k,s,t)(rep_{i\cdot i \approx t}(k+s)(s+k)(com(k,s))y)\} & : s \leq t \Rightarrow \mathtt{S}(s) \leq t \vee s \approx t \\
lch(s,t) & \hat{=} & lch_1(s, \mathtt{S}(t)) & : s \leq \mathtt{S}(t) \Rightarrow s \leq t \vee s \approx \mathtt{S}(t) \\
sym & \hat{=} & \lambda i, j \cdot \lambda x \cdot rep_{k\cdot k \approx i} ijx(refl\ i) & : \forall i, j \cdot i \approx j \Rightarrow j \approx i \\
reps_{k\cdot H(k)} & \hat{=} & \lambda i, j \cdot \lambda x \cdot rep_{k\cdot H(k)} ji(sym\ i\ j\ x) & : \forall i, j \cdot i \approx j \Rightarrow H(j) \Rightarrow H(i) \\
lS(s,t) & \hat{=} & \lambda x \cdot \text{case } x\,\big\{\iota k y \cdot \iota(\mathtt{S}(k))(reps_{i\cdot i \approx \mathtt{S}(t)}(\mathtt{S}(k)+s)(\mathtt{S}(k+s))(nS(k,s))y)\big\} & : s \leq t \Rightarrow s \leq \mathtt{S}(t)
\end{array}
$$

Lorsque $F = \forall i \leq t \cdot G(i)$, on démontre que F est décidable par récurrence sur t. Le cas de base est démontré par le terme u_1:

$$
\begin{array}{rcll}
u_{11} & \hat{=} & \lambda x \cdot \lambda i \cdot rep0S(i)x & : G(\mathtt{0}) \Rightarrow \forall i \leq \mathtt{0} \cdot G(i) \\
u_{12} & \hat{=} & \lambda x, y \cdot x(y\ \mathtt{0}\ l0) & : \neg G(\mathtt{0}) \Rightarrow \neg \forall i \leq \mathtt{0} \cdot G(i) \\
u_1 & \hat{=} & \text{case } d_{G(\mathtt{0})} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1(u_{11}x_1) \\ \iota_2 x_2 \mapsto \iota_2(u_{12}x_2) \end{array} \right\} & : (\forall i \leq \mathtt{0} \cdot G(i)) \vee \neg(\forall i \leq \mathtt{0} \cdot G(i))
\end{array}
$$

Le cas inductif demande à montrer que, sous l'hypothèse $h : (\forall i \leq j \cdot G(i)) \vee \neg(\forall i \leq j \cdot G(i))$, on a $(\forall i \leq \mathtt{S}(j) \cdot G(i)) \vee \neg(\forall i \leq \mathtt{S}(j) \cdot G(i))$. On considère deux cas. Dans le cas $h_1 : \forall i \leq j \cdot G(i)$, ceci est montré par le terme $u_2(j)$ ci-dessous:

$$
\begin{array}{rcll}
u_{21}(h_1, j) & \hat{=} & \lambda x \cdot \lambda i \cdot \lambda y \cdot & \\
 & & \quad \text{case } lch(i,j)y \left\{ \begin{array}{l} \iota_1 x_1 \mapsto h_1 i x_1 \\ \iota_2 x_2 \mapsto reps_{k\cdot G(k)} i(\mathtt{S}(j))x_2 x \end{array} \right\} & : G(\mathtt{S}(j)) \Rightarrow \forall i \leq \mathtt{S}(j) \cdot G(i) \\
u_{22}(j) & \hat{=} & \lambda x, y \cdot x(y(\mathtt{S}(j))(lrefl(\mathtt{S}(j)))) & : \neg G(\mathtt{S}(j)) \Rightarrow \neg \forall i \leq \mathtt{S}(j) \cdot G(i) \\
u_2(h_1, j) & \hat{=} & \text{case } d_{G(\mathtt{S}(j))} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1(u_{21}(h_1,j)x_1) \\ \iota_2 x_2 \mapsto \iota_2(u_{22}(j)x_2) \end{array} \right\} & : (\forall i \leq \mathtt{S}(j) \cdot G(i)) \vee \neg(\forall i \leq \mathtt{S}(j) \cdot G(i))
\end{array}
$$

Dans le cas $h_2 : \neg \forall i \leq j \cdot G(i)$, ceci est montré par le terme $u_3(j)$ ci-dessous:

$$
\begin{array}{rcll}
u_{31}(j) & \hat{=} & \lambda x \cdot \lambda i \cdot \lambda y \cdot xi(lS(i,j)y) & : (\forall i \leq \mathtt{S}(j) \cdot G(i)) \Rightarrow \forall i \leq j \cdot G(i) \\
u_3(h_2, j) & \hat{=} & \iota_2(\lambda x \cdot h_2(u_{31}(j)x)) & : (\forall i \leq \mathtt{S}(j) \cdot G(i)) \vee \neg(\forall i \leq \mathtt{S}(j) \cdot G(i))
\end{array}
$$

On pose donc:

$$
d_{\forall i \leq t \cdot G(i)} \quad \hat{=} \quad Ru_1(\lambda j \cdot \lambda h \cdot \text{case } h \left\{ \begin{array}{l} \iota_1 h_1 \mapsto u_2(h_1, j) \\ \iota_2 h_2 \mapsto u_3(h_2, j) \end{array} \right\})t \quad : (\forall i \leq t \cdot G(i)) \vee \neg(\forall i \leq t \cdot G(i))
$$

Lorsque $F = \exists i \leq t \cdot G$, avec i non libre dans t, on construit:

$$
\begin{array}{rcll}
v_{11} & \hat{=} & \lambda x \cdot \iota 0\langle l0, x\rangle & : G(0) \Rightarrow \exists i{\leq}0 \cdot G(i) \\
v_{12} & \hat{=} & \lambda x, y \cdot x(\text{case } y \, \{\iota iz \mapsto rep0S(i)(\pi_2 z)(\pi_1 z)\}) & : \neg G(0) \Rightarrow \neg\exists i{\leq}0 \cdot G(i) \\
v_1 & \hat{=} & \text{case } d_{G(0)} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1(v_{11} x_1) \\ \iota_2 x_2 \mapsto \iota_2(v_{12} x_2) \end{array} \right\} & : (\exists i{\leq}0 \cdot G(i)) \vee \neg(\exists i{\leq}0 \cdot G(i))
\end{array}
$$

Sous l'hypothèse $h_1 : \exists i{\leq}j \cdot G(i)$, on a:

$$
v_2(h_1, j) \quad \hat{=} \quad \iota_1(\text{case } h_1 \, \{\iota ix \mapsto \iota i\langle lS(i,j)(\pi_1 x), \pi_2 x\rangle\}) \quad : (\exists i{\leq}\mathtt{S}(j) \cdot G(i)) \vee \neg\exists i{\leq}\mathtt{S}(j) \cdot G(i)
$$

Sous l'hypothèse $h_2 : \neg\exists i{\leq}j \cdot G(i)$, on a:

$$
\begin{array}{rcll}
v_{31}(j) & \hat{=} & \lambda x \cdot \iota(\mathtt{S}(j))\langle lrefl(\mathtt{S}(j)), x\rangle & : G(\mathtt{S}(j)) \Rightarrow \exists i{\leq}\mathtt{S}(j) \cdot G(i) \\
v_{32}(h_2, j) & \hat{=} & \lambda x, y \cdot \text{case } y \left\{ \iota iz \mapsto \text{case } lch(i,j)(\pi_1 z) \left\{ \begin{array}{l} \iota_1 z_1 \mapsto h_2(\iota i\langle z_1, \pi_2 z\rangle) \\ \iota_2 z_2 \mapsto x(rep_{k \cdot G(k)} i(\mathtt{S}(j)) z_2(\pi_2 z)) \end{array} \right\} \right\} & \\
 & & & : \neg G(\mathtt{S}(j)) \Rightarrow \neg\exists i{\leq}\mathtt{S}(j) \cdot G(i) \\
v_3(h_2, j) & \hat{=} & \text{case } d_{G(\mathtt{S}(j))} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1(v_{31}(j) x_1) \\ \iota_2 x_2 \mapsto \iota_2(v_{32}(h_2, j) x_2) \end{array} \right\} & : (\exists i{\leq}\mathtt{S}(j) \cdot G(i)) \vee \neg\exists i{\leq}\mathtt{S}(j) \cdot G(i)
\end{array}
$$

Donc on pose:

$$
d_{\exists i \leq j \cdot G(i)} \quad \hat{=} \quad Rv_1(\lambda j \cdot \lambda h \cdot \text{case } h \left\{ \begin{array}{l} \iota_1 h_1 \mapsto v_2(h_1, j) \\ \iota_2 h_2 \mapsto v_3(h_2, j) \end{array} \right\}) t \quad : (\exists i{\leq}t \cdot G(i)) \vee \neg\exists i{\leq}t \cdot G(i)
$$

$\diamondsuit$

Lemme 20 *En* $\mathbf{HA}_1$*, pour toute formule* F *dans* Δ^0_0*, les formules* F*,* $F^{\neg\neg}$ *et* F° *sont prouvablement équivalentes. Autrement dit, n'importe quelle implication d'une de ces formules vers n'importe quelle autre est prouvable en* $\mathbf{HA}_1$*.*

Preuve : Notons d'abord que, pour toute formule G dans Δ^0_0, si d_G est le terme de type $G \vee \neg G$ construit au lemme 19, alors $c_G \hat{=} cl_G(d_G)$ (cf. lemme 17) est de type $\neg\neg G \Rightarrow G$. On démontre ensuite que F et $F^{\neg\neg}$ sont équivalentes en $\mathbf{HA}_1$, comme au lemme 15 mais en utilisant le terme c_G à la place de $\mathcal{C}$. Nous devons cependant étendre le résultat aux constructions $\wedge$, $\vee$, $\exists$. Pour ceci, nous allons construire par récurrence structurelle sur F des termes clos $u^\circ_F : F \Rightarrow F^\circ$ et $v_F : F^{\neg\neg} \Rightarrow F$. Comme d'autre part $w_F \hat{=} \lambda x, k \cdot kx : F^\circ \Rightarrow F^{\neg\neg}$, ceci suffira pour démontrer le lemme.

En convenant du fait que A dénote n'importe quelle égalité ou $\perp$, on définit (par exemple, il y a de nombreux autres choix possibles):

$$
\begin{array}{rcl}
u^\circ_A & \hat{=} & \lambda x \cdot x \\
u^\circ_{F \Rightarrow G} & \hat{=} & \lambda x, y, k \cdot k(u^\circ_G(x(v_F(\lambda k' \cdot k' y)))) \\
u^\circ_{F \wedge G} & \hat{=} & \lambda x \cdot \langle u^\circ_F(\pi_1 x), u^\circ_G(\pi_2 x)\rangle \\
u^\circ_{F \vee G} & \hat{=} & \lambda x \cdot \text{case } x \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1(u^\circ_F x_1) \\ \iota_2 x_2 \mapsto \iota_2(u^\circ_G x_2) \end{array} \right\} \\
u^\circ_{\forall i \leq t \cdot F} & \hat{=} & \lambda x \cdot \lambda i \cdot \lambda k \cdot k(\lambda y, k' \cdot k'(u^\circ_F(xiy))) \\
u^\circ_{\exists i \leq t \cdot F} & \hat{=} & \lambda x \cdot \text{case } x \, \{\iota iy \mapsto \langle \pi_1 y, u^\circ_F(\pi_2 y)\rangle\}
\end{array}
$$

$$
\begin{array}{rcl}
v_A & \hat{=} & \lambda x \cdot c_A x \\
v_{F \Rightarrow G} & \hat{=} & \lambda x \cdot c_{F \Rightarrow G}(\lambda k \cdot x(\lambda y \cdot k(\lambda z \cdot v_G(y(u^\circ_F z))))) \\
v_{F \wedge G} & \hat{=} & \lambda x \cdot c_{F \wedge G}(\lambda k \cdot x(\lambda z \cdot \\
 & & \quad k\langle v_F(\lambda k' \cdot k'(\pi_1 z)), v_G(\lambda k' \cdot k'(\pi_2 z))\rangle)) \\
v_{F \vee G} & \hat{=} & \lambda x \cdot c_{F \vee G}(\lambda k \cdot x(\lambda y \cdot \\
 & & \quad \text{case } y \left\{ \begin{array}{l} \iota_1 y_1 \mapsto k(\iota_1(v_F(\lambda k' \cdot k' y_1))) \\ \iota_2 y_2 \mapsto k(\iota_2(v_G(\lambda k' \cdot k' y_2))) \end{array} \right\})) \\
v_{\forall i \leq t \cdot F} & \hat{=} & \lambda x \cdot c_{\forall i \leq t \cdot F}(\lambda k \cdot x(\lambda y \cdot k(\\
 & & \quad \lambda i \cdot \lambda z \cdot v_F(\lambda k' \cdot yi(\lambda y' \cdot y' z k'))))) \\
v_{\exists i \leq t \cdot F} & \hat{=} & \lambda x \cdot c_{\exists i \leq t \cdot F}(\lambda k \cdot x(\lambda y \cdot \\
 & & \quad \text{case } y \, \{\iota iz \mapsto k(\iota i\langle \pi_1 z, v_F(\lambda k' \cdot k'(\pi_2 z))\rangle)\}))
\end{array}
$$

$\diamondsuit$

Lemme 21 *En* $\mathbf{HA}_1$*, pour toute formule* Σ^0_1 F*, on peut prouver* $F^\circ \Rightarrow F$*.*

Preuve : Observons que si on peut prouver $G^\circ \Rightarrow G$, alors on peut prouver $(\exists i \cdot G)^\circ \Rightarrow \exists i \cdot G$. En effet, si $u : G^\circ \Rightarrow G$, alors $\lambda x \cdot \text{case } x \{\iota i y \mapsto \iota i(uy)\}$ est de type $(\exists i \cdot G)^\circ \Rightarrow \exists i \cdot G$. Le lemme se déduit ainsi de cette remarque et du lemme 20 par récurrence sur le nombre de quantificateurs $\exists$ au début de F. $\diamondsuit$

Mais on ne peut pas prouver en général $F^{\neg\neg} \Rightarrow F$, car F° n'est pas décidable en général lorsque F est une formule Σ^0_1 — ce qui revient à dire que F elle-même n'est pas en général décidable. Nous ne développerons pas l'argument ici, qui fait appel à la théorie des fonctions calculables.

5.3 A-traduction et théorème de Kreisel-Friedman

On peut en fait aller un peu plus loin. L'astuce de Friedman est de modifier la $\neg\neg$-traduction en observant que le type $\perp$ dans $F^{\neg\neg} = \neg\neg F^\circ = (F^\circ \Rightarrow \perp) \Rightarrow \perp$ (deux occurrences de $\perp$ dans cette dernière formule) pourrait être remplacé par n'importe quelle formule: rien n'oblige à utiliser $\perp$ ici. Du point de vue de la traduction en CPS, le type $\perp$ est utilisé comme type des *réponses* fournies par le programme traduit en CPS (cf. le domaine *Ans* de la partie I). Or on peut utiliser le type de réponses que l'on souhaite, pas seulement $\perp$.

La *A-traduction* de Friedman est ainsi l'analogue de la $\neg\neg$-traduction, mais avec $\perp$ remplacé par une formule quelconque ϕ (que Friedman notait A, d'où le nom de A-traduction). Noter $\neg_\phi F \hat{=} F \Rightarrow \phi$ pour rappeler la négation, et redéfinissons la $\neg\neg$-traduction *relativisée* à ϕ par:

$$\begin{array}{rcll}
{}^\phi F^{\neg\neg} & \hat{=} & \neg_\phi \neg_\phi {}^\phi F^\circ & \\
{}^\phi \perp^\circ & \hat{=} & \phi & \\
{}^\phi A^\circ & \hat{=} & A \vee \phi & \quad (A \text{ type de base, sauf } \perp) \\
{}^\phi (F \Rightarrow G)^\circ & \hat{=} & {}^\phi F^\circ \Rightarrow {}^\phi G^{\neg\neg} & \\
{}^\phi (\forall i \cdot F)^\circ & \hat{=} & \forall i \cdot {}^\phi F^{\neg\neg} & \\
{}^\phi (\exists i \cdot F)^\circ & \hat{=} & \exists i \cdot {}^\phi F^\circ &
\end{array}$$

et la traduction en CPS correspondante est:

$$\begin{array}{rcl}
\langle x \rangle^\phi k & \hat{=} & kx \\
\langle \lambda x \cdot u \rangle^\phi k & \hat{=} & k(\lambda x \cdot \lambda k' \cdot \langle u \rangle^\phi k') \\
\langle uv \rangle^\phi k & \hat{=} & \langle u \rangle^\phi (\lambda x \cdot \langle v \rangle^\phi (\lambda y \cdot xyk)) \\
\langle \mathcal{C} u \rangle^\phi k & \hat{=} & \langle u \rangle^\phi (\lambda x \cdot x(\lambda y, k' \cdot ky)(\lambda z \cdot \nabla z)) \\
\langle \lambda i \cdot u \rangle^\phi k & \hat{=} & k(\lambda i \cdot \lambda k' \cdot \langle u \rangle^\phi k') \\
\langle ut \rangle^\phi k & \hat{=} & \langle u \rangle^\phi (\lambda x \cdot xtk) \\
\langle \iota t u \rangle^\phi k & \hat{=} & \langle u \rangle^\phi (\lambda x \cdot k(\iota t x)) \\
\langle \text{case } u \{\iota i x \mapsto v\} \rangle^\phi k & \hat{=} & \langle u \rangle^\phi (\lambda y \cdot \text{case } y \left\{\iota i x \mapsto \langle v \rangle^\phi k\right\}) \\
\langle r_0 \rangle^\phi k & \hat{=} & kr_0 \\
\langle Ruvt \rangle^\phi k & \hat{=} & \langle v \rangle^\phi (\lambda x \cdot R(\lambda k' \cdot \langle u \rangle^\phi k')(\lambda j \cdot \lambda y, k'' \cdot xj(\lambda x' \cdot y(\lambda y' \cdot x'y'k'')))tk)
\end{array}$$

Exercice 60 *Noter que la seule différence entre la définition de* $\langle u \rangle^\circ k$ *et celle de* $\langle u \rangle^\phi k$ *est un* ∇ *en plus dans le cas* $\mathcal{C}u$ *de la dernière: pourquoi ? Vérifier que pour tout* $u : F$ *(via une preuve spéciale* η*-longue, cf. exercice 56), pour tout* $k : \neg_\phi F^\circ$*,* $\langle u \rangle^\phi k$ *est de type* ϕ*.*

Exercice 61 *Étendre la traduction* $u, k \mapsto \langle u \rangle^\phi k$ *aux cas des conjonctions et des disjonctions, de sorte que* $\langle u \rangle^\phi k$ *soit toujours de type* ϕ *pour tout* $u : F$ *(via une preuve spéciale* η*-longue, cf. exercice 56) et pour tout* $k : \neg_\phi F^\circ$*.*

La version relativisée du lemme 19 est:

Lemme 22 *Pour toute formule* Δ^0_0 F*,* F *est* ϕ-décidable*:* $F \vee \neg_\phi F$ *est démontrable en* $\mathbf{HA}_1$*.*

Preuve : Par le lemme 19, d_F est un terme clos de type $F \vee \neg F$, donc ${}^\phi d_F \hat{=} \text{case } d_F \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1 x_1 \\ \iota_2 x_2 \mapsto \iota_2(\lambda y \cdot \nabla(x_2 y)) \end{array} \right\}$ est un terme clos de type $F \vee \neg_\phi F$. $\diamondsuit$

De même, on relativise le lemme 17:

Lemme 23 *Toute formule F ϕ-décidable est ϕ-*classique*: $\neg_\phi \neg_\phi F \Rightarrow F \vee \phi$ est prouvable en $\mathbf{HA}_1$.*

Preuve : Soit u un terme clos de type $F \vee \neg_\phi F$. Alors ${}^\phi cl_F(u) \hat{=} \lambda x \cdot \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1 x_1 \\ \iota_2 x_2 \mapsto \iota_2(x x_2) \end{array} \right\}$ est de type $\neg_\phi \neg_\phi F \Rightarrow F \vee \phi$. $\diamondsuit$

On peut alors démontrer la version relativisée du lemme 20 — c'est le lemme 25 plus bas —, mais ceci demande de nombreuses preuves auxiliaires:

Lemme 24 *Si F est ϕ-décidable, alors $\neg_\phi(\forall i \leq t \cdot F) \Rightarrow \exists i \leq t \cdot \neg_\phi F$ est démontrable en $\mathbf{HA}_1$.*

Preuve : En reprenant les termes de preuve fabriqués dans la preuve du lemme 19, et en écrivant $F(i)$ à la place de F, posons:

$$\begin{array}{rcll}
ae_1 & \hat{=} & \lambda x \cdot \iota \mathtt{0} \langle l0, \lambda y \cdot x(\lambda i \cdot \lambda z \cdot rep0S(i) y z) \rangle & : \neg_\phi(\forall i \leq \mathtt{0} \cdot F(i)) \Rightarrow \exists i \leq \mathtt{0} \cdot \neg_\phi F(i) \\
allS(t) & \hat{=} & \lambda x, y \cdot \lambda i \cdot \lambda z \cdot & \\
 & & \quad \text{case } lch(i,t) z \left\{ \begin{array}{l} \iota_1 z_1 \mapsto x i z_1 \\ \iota_2 z_2 \mapsto reps_{k \cdot F(k)} i(\mathtt{S}(t)) z_2 y \end{array} \right\} & : (\forall i \leq t \cdot F(i)) \Rightarrow F(\mathtt{S}(t)) \Rightarrow \forall i \leq \mathtt{S}(t) \cdot F(i) \\
ll0(s) & \hat{=} & \iota s(refl\ s) & : \mathtt{0} \leq s \\
ae_2(t) & \hat{=} & \lambda x, y \cdot \text{case } {}^\phi d_{\forall i \leq t \cdot F(i)} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } {}^\phi d_{F(\mathtt{S}(t))} \left\{ \begin{array}{l} \iota_1 y_1 \mapsto \iota \mathtt{0} \langle ll0(\mathtt{S}(t)), \lambda z \cdot x(allS(t) x_1 y_1) \rangle \\ \iota_2 y_2 \mapsto \iota(\mathtt{S}(t)) \langle lrefl(\mathtt{S}(t)), y_2 \rangle \end{array} \right\} \\ \iota_2 x_2 \mapsto \text{case } x x_2 \{ \iota i z \mapsto \iota i \langle lS(i,t)(\pi_1 z), \pi_2 z \rangle \} \end{array} \right\} & \\
 & & : (\neg_\phi(\forall i \leq t \cdot F(i)) \Rightarrow \exists i \leq t \cdot \neg_\phi F(i)) \Rightarrow \neg_\phi(\forall i \leq \mathtt{S}(t) \cdot F(i)) \Rightarrow \exists i \leq \mathtt{S}(t) \cdot \neg_\phi F(i) & \\
ae(t) & \hat{=} & R ae_1(\lambda j \cdot ae_2(j)) t & : \neg_\phi(\forall i \leq t \cdot F(i)) \Rightarrow \exists i \leq t \cdot \neg_\phi F(i)
\end{array}$$

$\diamondsuit$

Lemme 25 *En $\mathbf{HA}_1$, pour toute formule F dans Δ^0_0, les formules $F \vee \phi$, ${}^\phi F^{\neg\neg}$ et ${}^\phi F^\circ$ sont prouvablement équivalentes.*

Preuve : Pour toute formule G dans Δ^0_0, notons ${}^\phi c_G \hat{=} {}^\phi cl_G({}^\phi d_G)$ de type $\neg_\phi \neg_\phi G \Rightarrow G \vee \phi$, où ${}^\phi cl_G$ est défini au lemme 23 et ${}^\phi d_G$ est défini au lemme 22. Construisons par récurrence structurelle sur F des termes clos ${}^\phi u^\circ_F : F \vee \phi \Rightarrow {}^\phi F^\circ$ et $v_F : {}^\phi F^{\neg\neg} \Rightarrow F \vee \phi$. Le lemme se déduira de ces termes, et du terme ${}^\phi w_F \hat{=} \lambda x, k \cdot k x : {}^\phi F^\circ \Rightarrow {}^\phi F^{\neg\neg}$.

Si $k : \neg_\phi F$ et $u : F \vee \phi$, on pose $[k]u \hat{=} \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto k x_1 \\ \iota_2 x_2 \mapsto x_2 \end{array} \right\}$ de type ϕ.

Ensuite, pour tout opérateur f (π_1, π_2, ι_1, ι_2) tel que $fu : G$ dès que $u : F$ on pose ${}^\phi f v \hat{=} \text{case } v \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1(f x_1) \\ \iota_2 x_2 \mapsto \iota_2 x_2 \end{array} \right\}$: dès que $v : F \vee \phi$, ${}^\phi f v$ est de type $G \vee \phi$. De même, on pose ${}^\phi\langle u, v \rangle \hat{=} \text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } v \left\{ \begin{array}{l} \iota_1 y_1 \mapsto \iota_1 \langle x_1, y_1 \rangle \\ \iota_2 y_2 \mapsto \iota_2 y_2 \end{array} \right\} \\ \iota_2 x_2 \mapsto \iota_2 x_2 \end{array} \right\}$ de type $(F \wedge G) \vee \phi$ dès que $u : F \vee \phi$ et $v : G \vee \phi$.

Si $v_1 : F \vee \phi \Rightarrow H$ et $v_2 : G \vee \phi \Rightarrow H$, et $u : (F \vee G) \vee \phi$, on pose ${}^\phi\text{case } u \left\{ \begin{array}{l} v_1 \\ v_2 \end{array} \right\}$ le terme $\text{case } u \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } x_1 \left\{ \begin{array}{l} \iota_1 y_1 \mapsto v_1(\iota_1 y_1) \\ \iota_2 y_2 \mapsto v_2(\iota_1 y_2) \end{array} \right\} \\ \iota_2 x_2 \mapsto \iota_2 x_2 \end{array} \right\}$ de type H.

Soit A n'importe quelle égalité ou $\perp$, on définit:

$$
\begin{array}{rcl}
{}^\phi u^\circ_A & \hat{=} & \lambda x \cdot x \\
u^\circ_{F\Rightarrow G} & \hat{=} & \lambda x, y, k \cdot k(\\
& & \quad {}^\phi u^\circ_G(x@({}^\phi v_F(\lambda k' \cdot k' y)))) \\
{}^\phi u^\circ_{F\wedge G} & \hat{=} & \lambda x \cdot \langle {}^\phi u^\circ_F({}^\phi \pi_1 x), {}^\phi u^\circ_G({}^\phi \pi_2 x)\rangle \\
{}^\phi u^\circ_{F\vee G} & \hat{=} & \lambda x \cdot {}^\phi\text{case } x \left\{ \begin{array}{l} \lambda x_1 \cdot \iota_1({}^\phi u^\circ_F x_1) \\ \lambda x_2 \cdot \iota_2({}^\phi u^\circ_G x_2) \end{array} \right\}
\end{array}
$$

$$
\begin{array}{rcl}
{}^\phi v_A & \hat{=} & \lambda x \cdot \text{case } {}^\phi c_A x \left\{ \begin{array}{l} \iota_1 x_1 \mapsto x_1 \\ \iota_2 x_2 \mapsto \iota_2 x_2 \end{array} \right\} \\
{}^\phi v_{F\Rightarrow G} & \hat{=} & \lambda x \cdot {}^\phi c_{F\Rightarrow G}(\lambda k \cdot x(\lambda y \cdot \\
& & \quad {}^\phi a_{F,G} k(\lambda z \cdot {}^\phi v_G(y({}^\phi u^\circ_F z))))) \\
{}^\phi v_{F\wedge G} & \hat{=} & \lambda x \cdot {}^\phi c_{F\wedge G}(\lambda k \cdot x(\lambda z \cdot \\
& & \quad [k]^\phi \langle {}^\phi v_F(\lambda k' \cdot k'(\pi_1 z)), {}^\phi v_G(\lambda k' \cdot k'(\pi_2 z))\rangle))) \\
{}^\phi v_{F\vee G} & \hat{=} & \lambda x \cdot {}^\phi c_{F\vee G}(\lambda k \cdot x(\lambda y \cdot \\
& & \quad \text{case } y \left\{ \begin{array}{l} \iota_1 y_1 \mapsto [k]({}^\phi \iota_1({}^\phi v_F(\lambda k' \cdot k' y_1))) \\ \iota_2 y_2 \mapsto [k]({}^\phi \iota_2({}^\phi v_G(\lambda k' \cdot k' y_2))) \end{array} \right\}))
\end{array}
$$

$$
\begin{array}{rcl}
{}^\phi u^\circ_{\forall i \leq t \cdot F} & \hat{=} & \lambda x \cdot \lambda i \cdot \lambda k \cdot k(\lambda y, k' \cdot k'({}^\phi u^\circ_F(x@i@y))) \\
{}^\phi v_{\forall i \leq t \cdot F} & \hat{=} & \lambda x \cdot \text{case } {}^\phi d_{\forall i \leq t \cdot F} \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \iota_1 x_1 \\ \iota_2 x_2 \mapsto \text{case } ae(t) \left\{ \begin{array}{l} \iota i y \mapsto x(\lambda x' \cdot x' i(\lambda z \cdot z(\iota_1(\pi_1 y))(\lambda z' \cdot \\ \quad \text{case } {}^\phi v_F(\lambda k \cdot k z') \left\{ \begin{array}{l} \iota_1 y_1 \mapsto \pi_2 y y_1 \\ \iota_2 y_2 \mapsto y_2 \end{array} \right\}))) \end{array} \right\} \end{array} \right\} \\
{}^\phi u^\circ_{\exists i \leq t \cdot F} & \hat{=} & \lambda x \cdot \text{case } x \left\{ \begin{array}{l} \iota_1 x_1 \mapsto \text{case } x_1 \left\{ \iota i y \mapsto \iota i \langle \pi_1 y, {}^\phi u^\circ_F(\iota_1(\pi_2 y))\rangle \right\} \\ \iota_2 x_2 \mapsto \iota 0 \langle ll0(t), {}^\phi u^\circ_{F[i:=0]}(\iota_2 x_2)\rangle \end{array} \right\} \\
{}^\phi v_{\exists i \leq t \cdot F} & \hat{=} & \lambda x \cdot {}^\phi c_{\exists i \leq t \cdot F}(\lambda k \cdot x(\lambda x' \cdot \text{case } x' \left\{ \iota i y \mapsto \text{case } {}^\phi v_F(\lambda k' \cdot k'(\pi_2 y)) \left\{ \begin{array}{l} \iota_1 y_1 \mapsto k(\iota i \langle \pi_1 y, y_1\rangle) \\ \iota_2 y_2 \mapsto y_2 \end{array} \right\} \right\}))
\end{array}
$$

$\diamondsuit$

Si F est une formule Σ^0_1 prouvable en $\mathbf{PA}_1$, par l'exercice 60 ${}^\phi F^{\neg\neg} = \neg_\phi \neg_\phi {}^\phi F^\circ$ est prouvable en $\mathbf{HA}_1$, par une preuve $\lambda k \cdot \langle u \rangle^\phi k$. Ceci est vrai pour tout ϕ, en particulier pour $\phi = F$. Donc $({}^\phi F^\circ \Rightarrow F) \Rightarrow F$ est prouvable en $\mathbf{PA}_1$ pour $\phi = F$. Mais par le lemme 25, il existe un terme de preuve en $\mathbf{HA}_1$ de ${}^\phi F^\circ \Rightarrow F$, à savoir la continuation $k_F \hat{=} \lambda x \cdot \text{case } {}^\phi v_F(\lambda k \cdot kx) \left\{ \begin{array}{l} \iota_1 x_1 \mapsto x_1 \\ \iota_2 x_2 \mapsto x_2 \end{array} \right\}$. Donc $\langle u \rangle^\phi k_F$ est une preuve de F en $\mathbf{PA}_1$. On en déduit:

Théorème 12 (Kreisel-Friedman) *Pour toute formule Π^0_2 de l'arithmétique F, si F est prouvable en $\mathbf{PA}_1$, alors F est prouvable en $\mathbf{HA}_1$.*

Preuve : Écrivons $F \hat{=} \forall i_1, \ldots, i_m \cdot G$ où G est une formule Σ^0_1. En utilisant $(\forall E)$, si F est prouvable en $\mathbf{PA}_1$, alors G aussi. Comme G est Σ^0_1, G est prouvable en $\mathbf{HA}_1$ par la remarque ci-dessus. En utilisant $(\forall I)$, on en déduit que F est elle aussi prouvable en $\mathbf{HA}_1$. $\diamondsuit$

Il se trouve que ce résultat ne peut pas être étendu au-delà de Π^0_2, et il existe des formules Σ^0_2 prouvables en $\mathbf{PA}_1$ mais pas en $\mathbf{HA}_1$. Ceci sort cependant du cadre de ce cours.

La signification calculatoire de ce résultat, qui est contenue dans le cas où F est Σ^0_1, est qu'on peut toujours extraire d'une preuve classique u de F une preuve intuitionniste $\langle u \rangle^\phi k_F$ de F, en calculant la traduction en CPS de u et en l'appliquant à une continuation k_F bien choisie.

Une application informatique est la suivante. Considérons pour simplifier une formule de la forme $\forall i \cdot \exists j \cdot P(i, j)$ sans variable libre. Une preuve intuitionniste d'une telle formule, écrite en forme normale, décrit nécessairement pour tout i une valeur de j tel que $P(i, j)$ soit vrai. Autrement dit, une formule de cette forme est une *spécification* d'une fonction qui envoie tout entier i vers un entier j tel que $P(i, j)$. Prouver $\forall i \cdot \exists j \cdot P(i, j)$ en $\mathbf{HA}_1$, c'est donc essentiellement trouver une fonction de i vers j satisfaisant la spécification. En effet, considérons un terme de preuve normal u tel que $\vdash u : \forall i \cdot \exists j \cdot P(i, j)$. Pour tout entier m, posons $[m]$ le terme $\mathtt{S}(\mathtt{S}(\ldots(\mathtt{S}(\mathtt{0}))\ldots))$, où il y a m symboles $\mathtt{S}$. Alors on a $\vdash u[m] : \exists j \cdot P([m], j)$. Soit v une forme normale de $u[m]$ (par une stratégie de réduction donnée, disons; en fait, la réduction de preuves en $\mathbf{HA}_1$ est confluente, et il n'y a donc qu'une forme normale). Alors v est tel que $\vdash v : \exists j \cdot P([m], j)$. Comme v est normal et clos, v ne peut qu'être de la forme $\iota t w$, pour un certain terme clos normal t et un certain terme de preuve w. Or un terme clos normal t est nécessairement de la forme $[n]$ pour un certain entier n; et w

est une preuve de $P([m],[n])$: la preuve u définit donc bien une fonction totale des m vers les n tels que $P([m],[n])$ soit vraie. De plus, cette fonction est *calculable*: il existe un programme informatique qui calcule n en fonction de m; en effet, il suffit de construire $u[m]$, de normaliser et d'extraire $[n]$ de la forme normale. On dit que $\mathbf{HA}_1$ est une logique *constructive*, car d'une preuve d'une spécification de la forme $\forall i \cdot \exists j \cdot P(i,j)$ on peut toujours extraire un *programme* qui calcule j en fonction de i. (Attention, le terme "constructif" a de nombreux autres sens.)

Exercice 62 *Montrer cette affirmation formellement. L'étendre au cas des formules $\forall i_1, \ldots, i_p \cdot \exists j_1, \ldots, j_q \cdot P(i_1, \ldots, i_p, j_1, \ldots, j_q)$.*

En revanche, si $\forall i \cdot \exists j \cdot P(i,j)$ est prouvable en $\mathbf{PA}_1$, l'argument ci-dessus ne fonctionne pas: le terme v de type $\exists j \cdot P([m],j)$ peut être de la forme $\iota t w$ comme ci-dessus, ou bien de la forme $\mathcal{C}v'$; dans ce dernier cas, on ne peut pas extraire de v une valeur de v telle que $P([m],j)$ soit prouvable. La fonction qui aux i associe des j tels que $P(i,j)$ est donc en général partielle dans le cas de $\mathbf{PA}_1$, alors que $\mathbf{HA}_1$ permet de produire une fonction totale. Ce que le théorème 12 montrer ici est que lorsque $\forall i \cdot \exists j \cdot P(i,j)$ est Π^0_2, autrement dit quand $P(i,j)$ est décidable, alors il y a toujours moyen de transformer cette fonction partielle en une fonction totale calculable qui satisfait la propriété $P(i,j)$: $\mathbf{PA}_1$ est donc aussi une logique constructive, si on la restreint aux formules Π^0_2.

On notera que le programme extrait de la preuve est automatiquement correct, au sens où il calcule nécessairement ce que la spécification $P(i,j)$ prescrit. De plus, on n'a pas eu à l'écrire, il suffit de l'extraire de la preuve. Cette idée s'étend naturellement — dans un cadre intuitionniste — à l'utilitaire d'extraction de programme à partir d'une preuve dans Coq [BBC$^+$99].

Exercice 63 *Étendre le théorème 12 au cas des formules de la forme de la forme $F \hat{=} \forall X_1, \ldots, X_m, x_1, \ldots, x_p \cdot \exists Y_1, \ldots, Y_n, y_1, \ldots, y_q \cdot G$, où G est sans quantificateur, $X_1, \ldots, X_m$, $Y_1, \ldots, Y_n$ sont des variables du second ordre et $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ sont des variables du premier ordre: autrement dit, montrer que si F est prouvable en $\mathbf{PA}_2$, alors F est prouvable en $\mathbf{HA}_2$. (On considérera les versions de $\mathbf{PA}_2$ et $\mathbf{HA}_2$ avec les connecteurs $\Rightarrow$, $\bot$, $\wedge$, $\vee$, les quantifications universelles $\forall$ et existentielles $\exists$ tant du premier que du second ordre. Les règles de déduction sur le $\exists$ du second ordre sont celles de l'exercice 36. On posera $(\forall P \cdot F)^\circ \hat{=} \forall P \cdot F^{\neg\neg}$, $(\exists P \cdot F)^\circ \hat{=} \exists P \cdot F^\circ$ et on montrera que $F^\circ[P(k_1, \ldots, k_n) := G^\circ] = (F[P(k_1, \ldots, k_n) := G])^\circ$.)*

$\mathbf{PA}_1$ et $\mathbf{HA}_1$ sont expressives; notamment on peut définir toute fonction calculable en $\mathbf{PA}_1$ [Joh87], mais c'est difficile et peu naturel à montrer. On peut améliorer ceci en enrichissant le langage de calcul à l'intérieur des formules. L'exercice suivant donne un exemple. De nombreuses autres solutions sont envisageables, et l'on peut (presque) utiliser le langage de programmation que l'on souhaite à l'intérieur des formules.

Exercice 64 *Le langage des termes de $\mathbf{PA}_1^\omega$ et de $\mathbf{HA}_1^\omega$ est similaire à celui de $\mathbf{HA}_\omega$: il est formé des λ-termes simplement typés construits à partir des trois constantes $\mathtt{0} : \mathbb{N}$, $\mathtt{S} : \mathbb{N} \to \mathbb{N}$ et $R_\tau : \tau \to (\mathbb{N} \to \tau \to \tau) \to \mathbb{N} \to \tau$ (le récurseur de type τ), les types étant donnés par $\tau ::= \mathbb{N} \mid \tau \to \tau$. $\mathbf{PA}_1^\omega$ et $\mathbf{HA}_1^\omega$ (d'ordre 1 mais portant sur des termes d'ordre supérieur) sont définies comme $\mathbf{PA}_1$ et $\mathbf{HA}_1$ respectivement, mais avec le langage de termes ci-dessus, et la réduction suivante sur les formules au lieu de $\to_\mathbb{N}$:*

$$
\begin{array}{rrcl}
(\mathtt{0} \approx \mathtt{S}) & \mathtt{0} \approx \mathtt{S}t & \to_+ & \bot \\
(\mathtt{S} \approx \mathtt{S}) & \mathtt{S}s \approx \mathtt{S}t & \to_+ & s \approx t \\
(\mathtt{R0}) & R_\tau s_1 s_2 \mathtt{0} & \to_+ & s_1 \\
(\mathtt{RS}) & R_\tau s_1 s_2 (\mathtt{S}s_3) & \to_+ & s_2 s_3 (R_\tau s_1 s_2 s_3) \\
(\beta) & (\lambda i \cdot s)t & \to_+ & s[i := t]
\end{array}
$$

Montrer qu'on peut définir l'addition et la multiplication respectivement par $s+t \hat{=} R_\mathbb{N} s(\lambda i, j \cdot \mathtt{S}j)t$ et $s*t \hat{=} R_\mathbb{N}\mathtt{0}(\lambda i, j \cdot j+s)t$. Comment définiriez-vous t^s, $s!$, $s-t$ (répondant $s-t$ si $s \geq t$, 0 sinon), $\mathtt{if}\ s\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2$ (répondant t_1 si $s \neq 0$ et t_2 sinon), $s \operatorname{div} t$ (la partie entière de s/t, que vous prendrez égale à $\mathtt{0}$ quand $t \leftrightarrow^*_+ \mathtt{0}$), $s \bmod t$?

Exercice 65 *Montrer le théorème de normalisation forte de* $\mathbf{PA}_1^\omega$ *et* $\mathbf{HA}_1^\omega$*. En déduire que ces logiques sont cohérentes.*

Exercice 66 *Montrer le théorème de Kreisel-Friedman pour* $\mathbf{PA}_1^\omega$ *et* $\mathbf{HA}_1^\omega$*.*

Exercice 67 (☡) *En utilisant les définitions trouvées à l'exercice 64, montrer que* $s \approx t$ *et* $s - t \approx \mathtt{0} \wedge t - s \approx \mathtt{0}$ *sont prouvablement équivalentes en* $\mathbf{HA}_1^\omega$*. (Faites-le informellement, et produisez les termes de preuve si vous en avez le courage.) Montrer d'autre part que les paires de formules suivantes sont aussi prouvablement équivalentes en* $\mathbf{PA}_1^\omega$*:*

1. $s \approx \mathtt{0} \wedge t \approx \mathtt{0}$ *et* $s + t \approx \mathtt{0}$*;*
2. $s \approx \mathtt{0} \vee t \approx \mathtt{0}$ *et* $s * t \approx \mathtt{0}$*;*
3. $\neg s \approx \mathtt{0}$ *et* $\mathtt{if}\ s\ \mathtt{then}\ \mathtt{0}\ \mathtt{else}\ \mathtt{S0}$*;*
4. $s \approx \mathtt{0} \Rightarrow t \approx \mathtt{0}$ *et* $\neg s \approx \mathtt{0} \vee t \approx \mathtt{0}$*;*
5. $\forall i \leq t \cdot s(i) \approx \mathtt{0}$ *et* $R_{\mathbb{N}}(s(\mathtt{0}))(\lambda i, j \cdot s(\mathtt{S}i) + j)t \approx \mathtt{0}$*;*
6. $\exists i \leq t \cdot s(i) \approx \mathtt{0}$ *et* $R_{\mathbb{N}}(s(\mathtt{0}))(\lambda i, j \cdot s(\mathtt{S}i) * j)t \approx \mathtt{0}$*;*

En déduire que toute formule Δ_0^0 *de* $\mathbf{PA}_1^\omega$ *est prouvablement équivalente en* $\mathbf{HA}_1^\omega$ *à une formule de la forme* $s \approx \mathtt{0}$*. En conclure qu'on peut redémontrer le théorème de Kreisel-Friedman sans utiliser de CPS et en remplaçant le lemme 25 par un lemme plus simple à démontrer.*

Il existe de nombreux autres outils que l'on peut utiliser pour analyser les systèmes de preuve; consulter [Koh98] pour plus de détails.

References

[Acz99] Peter Aczel. On relating type theories and set theories. In *Types'98*. Springer Verlag, 1999.

[BBC+99] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, Gérard Huet, Henri Laulhère, Cesar Muñoz, Chetan Murthy, Catherine Parent-Vigouroux, Patrick Loiseleur, Christine Paulin-Mohring, Amokrane Saïbi, and Benjamin Werner. The Coq proof assistant — reference manual. Disponible en `http://coq.inria.fr/doc/main.html`., décembre 1999. Version 6.3.1.

[Coq94] Thierry Coquand. A new paradox in type theory. In D. Prawitz, B. Skyrms, and D. Westerståhl, editors, *Proceedings 9th International Congress of Logic, Methodology and Philosophy of Science, Uppsala, Suède, 7–14 août 1991*, volume 134 of *Studies in Logic and the Foundations of Mathematics*, pages 555–570. North-Holland, Amsterdam, avril 1994. Disponible en `http://hypatia.dcs.qmw.ac.uk/data/C/CoquandTFH/nyparadox.ps.Z`.

[DHK98] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. Technical Report RR-3400, Inria, Institut National de Recherche en Informatique et en Automatique, avril 1998. Disponible en `http://www.inria.fr/RRRT/RR-3400.html`.

[FFKD87] Matthias Felleisen, Daniel P. Friedman, Emil Kohlbecker, and Bruce Duba. A syntactic theory of sequential control. *Theoretical Computer Science*, 52(3):205–237, 1987.

[Fri78] Harvey Friedman. Classically and intuitionistically provably recursive functions. In D. S. Scott and G. H. Muller, editors, *Higher Set Theory*, volume 699 of *Lecture Notes in Mathematics*, pages 21–28. Springer Verlag, 1978.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[Gri90] Timothy G. Griffin. A formulas-as-types notion of control. In *Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages*, pages 47–58, San Francisco, California, janvier 1990.

[HHP87] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. In *Symposium on Logic in Computer Science, Ithaca, NY*, pages 194–204. IEEE, June 1987.

[HMT90] Robert Harper, Robin Milner, and Mads Tofte. *The Definition of Standard ML*. MIT Press, 1990.

[Joh87] Peter T. Johnstone. *Notes on Logic and Set Theory*. Cambridge University Press, 1987.

[Koh98] Ulrich Kohlenbach. Proof interpretations. Cours de doctorat, BRICS, Danemark. Disponible en `http://www.brics.dk/~kohlenb/file.ps`, 1998.

[MNPS91] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991. Disponible en `ftp://ftp.cis.upenn.edu/pub/papers/miller/apal91.dvi.Z`, errata en `ftp://ftp.cis.upenn.edu/pub/papers/miller/apal91-errata.dvi`.

[Par92] Michel Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *3rd International Conference on Logic Programming and Automated Reasoning*, volume 417 of *Lecture Notes in Computer Science*, Saint-Petersburg, USSR, juillet 1992. Springer Verlag.

[PH78] Jeff Paris and Leo Harrington. A mathematical incompleteness of Peano Arithmetic. In Jon Barwise, editor, *Handbook of Mathematical Logic*, chapter D.8, pages 1133–1142. North-Holland, Amsterdam, second edition, 1978.

[Plo76] Gordon Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1(1):125–159, 1976.

[Rég94] Laurent Régnier. Une équivalence sur les lambda-termes. *Theoretical Computer Science*, 126, 1994. Disponible en `http://hypatia.dcs.qmw.ac.uk/data/R/RegnierL/sigma.ps.gz`.

[Wer97] Benjamin Werner. Sets in types, types in sets. In Martin Abadi and Takahashi Ito, editors, *TACS'97*, volume 1281. LNCS, Springer-Verlag, 1997. Disponible en `http://pauillac.inria.fr/~werner/publis/zfc.ps.gz`, fichier Coq en `http://pauillac.inria.fr/~werner/ZFC.v`.