THÈSE
de
L'UNIVERSITÉ PARIS VII

Discipline : **Mathématiques**

Spécialité : **Logique et Fondement de l'Informatique**

par

Christophe RAFFALLI

L'ARITHMÉTIQUE FONCTIONNELLE DU SECOND ORDRE AVEC POINTS FIXES

Soutenance le 9 Février 1994 devant le jury suivant :

Thierry Coquand	(rapporteur)
Gerard Huet	
Jean-Louis Krivine	(directeur de la thèse)
Daniel Leivant	(rapporteur)
Michel Parigot	
Gordon Plotkin	

préparée au sein de
l'Équipe de Logique, CNRS UA 753
UFR de Mathématiques de Paris VII

Remerciements

Cette thèse, fruit de quatre années de travail, ne serait rien sans la présence d'un bon nombre de personnes à qui je dois beaucoup. Je dois remercier en tout premier lieu mes parents, sans qui je n'aurais jamais vu le jour, ainsi que toutes les personnes ayant participées aux prémices de mon éduction, et tout particulièrement Madame Fouillot. Je dois aussi citer les enseignants du lycée Condorcet, de l'école normale supérieure de Cachan et de la maîtrise de mathématique de l'université d'Orsay qui ont tour à tour participé à l'éveil de ma culture mathématique.

Viennent ensuite René Cori et Daniel Lascar qui m'ont initié à la logique dans le cadre du D.E.A. de "logique et fondement de l'informatique" de l'université Paris VII, ainsi que Jean-Louis Krivine et Michel Parigot, responsables de ma spécialisation en λ-calcul typé, et qui ont bien voulu diriger mes recherches. Je tiens aussi à remercier tous les autres membres de l'équipe qui savent si bien faire régner une ambiance particulièrement propice à la recherche au sein de ce laboratoire. Il me faut aussi citer Paul Rozière, Pascal Manouri, Gilles Amiot, Vincent Danos, Laurent Regnier et Marianne Simonot qui furent plus proches de moi et durent donc subir souvent de trop longues discutions.

Je ne dois pas non plus oublier Gordon Plotkin et George Cleland qui m'ont accueilli au L.F.C.S., dans le cadre de mon service national, ainsi que Renaud Marlet, qui m'a supporté dans son bureau pendant près d'un an.

Enfin, je remercie mes deux rapporteurs : Daniel Leivant et Thierry Coquant qui ont dû se donner beaucoup de peine pour ingurgiter ma thèse, ainsi que tous les autres membres du jury qui ont accepté de m'honorer de leur présence : Gérard Huet, Gordon Plotkin, Jean Louis Krivine et Michel Parigot.

À Rébecca

Résumé

Le paradigme “preuves comme programmes” permet de considérer les preuves en déduction naturelle du second ordre comme des programmes en lambda-calcul. Toutefois, sa pratique dans le cadre de L’Arithmétique Fonctionnelle du Second Ordre, due à Leivant et Krivine, produit des programmes peu réalistes. Afin de combler en partie ce défaut, Parigot introduit un connecteur de plus petit point fixe, utilisé pour construire des types inductifs. Cette thèse étudie une extension simultanée du système par des connecteurs de plus petit et de plus grand points fixes. Ce dernier autorise l’utilisation de données infinies tels les “streams”.

L’approche que nous présentons est originale, car les données infinies sont représentées par des termes non-normalisables. En contre-partie, la normalisation est remplacée par la notion de solvabilité héréditaire, qui est vérifiée par une très large partie du système.

Nous montrons aussi qu’avec cette approche, chaque élément d’un type de données (infinies ou non) trouve une unique représentation en lambda-calcul, ce qui permet d’en déduire la correction des programmes extraits.

Le reste de la thèse est consacré à quelques exemples ainsi qu’à l’étude des problèmes posés par la définition de l’égalité sur les types de données infinies.

Abstract

The “proofs as programs” paradigm identifies proofs in natural deduction with programs in pure lambda-calculus. However, its usage within Second Order Functional Arithmetic, due to Leivant and Krivine, doesn’t produce realistic programs. To partially solve this problem, Parigot introduced a least fixpoint connective used to construct inductive types. This thesis studies a simultaneous extension with least and greatest fixpoints. The latter is useful for defining infinite data structures like “streams”.

Our approach is original because infinite data structures are represented by non-normalizable terms. Nevertheless, for a large part of the system, we can replace normalization by the notion of hereditary solvability.

We also show that each element of a data type (finite or infinite) has a unique representation in pure lambda-calculus. We use this to prove that extracted programs are correct.

The remainder of the thesis deals with some examples as well as the problem of the equality on infinite data types.

Table des matières.

Chapitre 1

Introduction.

1.1 AF_2 et les systèmes de types.

Depuis les travaux de Curry, beaucoup de systèmes de types ont été créés (Automath de De Brujn, [6], le system F de Girard [7], la théorie des types de Martin-Löf [18], le Calcul des Constructions de Coquand et Huet [5] , . . .). Le dénominateur commun à tous ces systèmes est l'extraction de programmes à partir de preuves, grâce à l'isomorphisme de Curry-Howard [8], qui établit une correspondance entre les programmes et les preuves de leurs spécifications.

L'un d'eux est l'arithmétique fonctionnelle du second ordre (AF_2), due à Leivant et Krivine [11, 12, 14]. Ce système est sous bien des aspects un peu à part. Nous allons commencer cette introduction en rappelant les principes qui font sa spécificité.

Tous les systèmes de types exploitent les relations entre une logique donnée, équipée d'un système de déduction, et un langage de programmation théorique, la plupart du temps dérivé du λ-calcul de Church [3] (ces deux notions sont parfois fortement imbriquées voire totalement confondues).

Le système AF_2 ne déroge pas à cette règle. La logique utilisée est la logique intuitionniste du second ordre : les formules sont construites à partir des termes du premier ordre, des variables de prédicats et des connecteurs d'implication et de quantification universelle du premier et du second ordre.

Première spécificité d'AF_2 : le choix de la logique du second ordre. La proximité de cette logique avec la pratique mathématique courante est une des motivations de ce choix. C'est pourquoi l'un des principes de base est de ne pas étendre la logique sous-jacente au système afin de rester sur les bases mathématiques de la logique du second ordre (intuitionniste ou classique). En conséquence, toutes les extensions que nous proposons sont conservatives. Elles ont pour unique but l'optimisation des programmes extraits.

La possibilité de définir de manière interne les "types de données" représente un autre avantage de la logique du second ordre. Il suffit pour cela d'étendre le langage en ajoutant les constructeurs du nouveau type. Il n'est donc pas nécessaire d'ajouter d'axiomes. Par exemple, pour définir l'ensemble des entiers, il suffit d'ajouter les constantes $\mathbf{0}$ (zéro) et $\mathbf{s}$ (successeur) au langage et de définir l'ensemble des entiers comme le plus petit prédicat contenant zéro et clos pour la

fonction successeur. Cela conduit à la formule suivante :

$$N[x] = \forall X \left(X\mathbf{0}, \forall y\,(Xy \to X\mathbf{s}y) \to Xx \right)$$

Seconde particularité d'AF_2 : l'utilisation d'axiomes sous forme d'équations comme spécifications de fonctions. Les programmes sont alors extraits de la preuve de totalité de ces fonctions. Le système **ProPre** de Manoury et Simonot [17, 20], qui permet de construire automatiquement la preuve de totalité d'un grand nombre de fonctions, et d'assurer que le programme extrait suit l'algorithme décrit par les équations, montre l'intérêt de cette approche.

Ainsi, pour un programme calculant l'addition, on ajoute au langage une constante de fonction **add**. L'addition est spécifiée par les équations usuelles :

$$\mathbf{add}\,(\mathbf{0}, y) = y \quad \text{et} \quad \mathbf{add}\,(\mathbf{s}x, y) = \mathbf{s}\ \mathbf{add}\,(x, y)$$

Le programme est alors extrait d'une preuve de la formule affirmant que l'addition est totale :

$$\forall x \forall y \left(N[x], N[y] \to N[\mathbf{add}\,(x, y)] \right)$$

Les preuves sont réalisées en déduction naturelle. De plus, pour extraire le programme, on ne garde trace que des axiomes et des règles d'introduction et d'élimination de l'implication. Ainsi, on extrait des termes du λ-calcul pur (les variables provenant des axiomes, l'application de l'élimination de l'implication et l'abstraction de son introduction).

Troisième particularité d'AF_2 : l'utilisation du λ-calcul pur. Ce choix est justifié, car le λ-calcul est suffisamment puissant pour modéliser la plupart des aspects des langages de programmation, tout en étant extrêmement simple, facilitant ainsi raisonnements et preuves.

La correction des programmes ainsi extraits repose sur l'unicité de la représentation des données extraite de la définition du type (cf [11, 12, 14]). Par exemple, dans le cas des entiers, il y a au plus un terme en forme normale extrait d'une preuve de la formule $N[t]$. Ainsi, l'ensemble des termes de type $N[t]$ est une représentation des entiers en λ-calcul pur. Dans le cas particulier de cette formule, on obtient exactement les entiers de Church, l'entier n étant représenté par le terme $\lambda\mathtt{x}\lambda\mathtt{f}(\mathtt{f}^{\mathtt{n}}\mathtt{x})$. Ainsi, tout terme extrait d'une preuve de $N[\mathbf{add}(\mathbf{ss0}, \mathbf{s0})]$ se réduit à l'entier de Church 3.

Pour tous les types de données (listes, arbres ...), on a des propriétés similaires, qui permettent d'assurer la correction du programme vis à vis des spécifications équationnelles. De plus, on peut extraire des programmes pour toutes les fonctions prouvablement totales dans l'arithmétique du second ordre. Cela suffit amplement ! Ainsi, se limiter à la logique intuitionniste ne réduit pas l'ensemble des fonctions programmables [11].

1.2 Les types inductifs.

Malheureusement, les types définis comme les entiers de Church ont de mauvaises propriétés. Essentiellement, chaque pas d'une récurrence est linéaire, au lieu d'être en temps constant. Ainsi, il n'existe pas de prédécesseur en temps constant sur les entiers de Church [9, 24].

Or, si l'on veut continuer d'utiliser le λ-calcul pur comme base de notre théorie, il est nécessaire de pallier à ce défaut afin d'avoir à notre disposition une théorie plus réaliste. Pour cela, Parigot introduit des types récursifs [22]. Par exemple, pour les entiers, on pourrait écrire la formule récursive suivante :

$$N[x] = \forall X \left(X\mathbf{0}, \forall y \left(N[y] \to X\mathbf{s}y \right) \to Xx \right)$$

Afin d'éviter la gestion de formules récursives, nous ajoutons un connecteur à la logique permettant d'écrire des formules de la forme :

$$\mu K \lambda x_1 \dots x_n \, F\langle t_1, \dots, t_n \rangle$$

Dans cette formule, K est une variable de prédicat d'arité n n'ayant que des occurrences positives dans F. Cette formule est alors interprétée comme $K_0(t_1, \dots, t_n)$, K_0 étant le plus petit prédicat vérifiant $\forall x_1 \dots x_n (K_0(x_1, \dots, x_n) \leftrightarrow F[K \Leftarrow K_0])$. Ceci revient à dire que K_0 est le plus petit point fixe de la fonction qui associe le prédicat $\lambda x_1 \dots x_n \, F$ au prédicat K (ce point fixe existe car la positivité de K dans F assure que cette fonction est croissante).

Ce connecteur utilise trois nouvelles règles : deux pour l'équivalence ci-dessus, et une troisième pour exprimer qu'il s'agit d'un plus petit point fixe.

En utilisant ce nouveau connecteur, la définition inductive des entiers s'écrit :

$$N[x] = \mu K \lambda x' \forall X \left(X\mathbf{0}, \forall y \, (Ky \to X\mathbf{s}y) \to Xx' \right) \langle x \rangle$$

Parigot montre que tous les résultats de AF_2 peuvent être étendus, et que les programmes que l'on obtient ont des performances plus proches de celles des programmes de tous les jours.

1.3 Les types coïnductifs.

Cette théorie est toujours loin de couvrir toutes les possibilités des langages de programmation réels. L'une des lacunes est l'absence de type de données infinies tel que les "streams" (canaux d'entrée/sortie). Une solution simple pour ajouter ces notions est de considérer le connecteur ν, interprété comme un plus grand point fixe, dual du connecteur μ précédemment introduit.

Nous ajoutons donc à la logique des formules de la forme :

$$\nu K \lambda x_1 \dots x_n \, F\langle t_1, \dots, t_n \rangle$$

où K est une variable de prédicat d'arité n n'ayant que des occurrences positives dans F. Cette formule est interprétée comme $K_0(t_1, \dots, t_n)$, K_0 étant le plus grand prédicat vérifiant $\forall x_1 \dots x_n (K_0(x_1, \dots, x_n) \leftrightarrow F[K \Leftarrow K_0])$. Nous ajoutons alors trois règles au système, analogues à celles du connecteur μ..

Le propos de cette thèse est d'étudier les propriétés et les possibilités du système $AF_2^{\mu\nu}$ (arithmétique fonctionnelle du second ordre avec plus petit et plus grand points fixes) ainsi défini.

Ces notions de types inductifs et coïnductifs ont été étudiées dans le cadre d'autres systèmes de types [19, 23, 25, 4]. Ce travail est néanmoins nouveau, car les problèmes sont abordés sous un jour différent du fait des spécificités d'AF_2.

Ces différences proviennent essentiellement de deux sources :

- Définition des types de données par des formules logiques (plutôt que par ajout externe, comme dans la plupart des autres systèmes de types).
- Utilisation du λ-calcul pur.

Ainsi le type des "streams" est extrêmement proche du type des listes. Les deux types utiliseront un constructeur d'arité deux (le type liste utilisant en plus un constructeur initial d'arité zéro pour la liste vide). La seule différence importante est l'utilisation du connecteur ν à la place de μ.

Les types de données infinies ne sont donc pas définis par la donnée des destructeurs du type, mais bien par celle des constructeurs du type. Cette approche analogue des types de données finies et infinies permet, en utilisant les deux connecteurs μ et ν, de définir de nouveaux types, tels que des listes infinies de 0 et de 1 n'utilisant qu'un nombre fini de 1.

La représentation des types de données infinies, en λ-calcul pur, par des termes non normalisables est une nouvelle caractéristique de cette approche. Nous montrons qu'ainsi la notion de type de données de Krivine peut être étendue aux types infinis, en obtenant l'unicité de la représentation de chaque élément du type de données (pour la ω-équivalence, ou égalité des arbres de Böhm).

En contrepartie de cette approche, on perd toute propriété de normalisation forte ou faible. Par contre, nous étudions des propriétés telles que la résolubilité ou la ω-résolubilité (absence de $\perp$ dans l'arbre de Böhm) des termes typés.

Nous nous intéressons aussi à des exemples pratiques, tel le classique crible d'Eratosthène. Nous montrons aussi comment une représentation non-standard des entiers permet de typer toutes les fonctions récursives partielles.

De plus, en essayant de résoudre les problèmes posés par l'égalité sur ces types de données infinies, nous développons une définition originale de l'égalité sur les types de données aussi bien finies que infinies. Cela débouche sur un concept nouveau de type de données avec égalité, où l'on définit le type par une formule binaire $R[x, y]$ affirmant que x est un élément du type et qu'il est égal à y. Nous montrons alors que cette notion permet de définir des types quotients.

1.4 Autres travaux sur les types inductifs et coïnductifs.

Beaucoup de travaux ont été publiés traitant du problème des définitions inductives à l'intérieur de différents formalismes. De plus, quelques-uns d'entre eux traitent aussi des types coïnductifs. Nous allons maintenant comparer certaines de ces approches avec la notre.

Le travail le plus proche est celui de Michel Parigot puisque notre système est une extension directe du sien. Ainsi, si on restreint $AF_2^{\mu\nu}$ pour n'utiliser que le plus petit point fixe, on obtient quasiment le système TTR (Théorie des Types Récursifs) présenté dans [22]. La seule différence est la présence dans TTR d'une règle utilisant l'inclusion qui peut se substituer à la règle de plus

petit point fixe (aussi présente dans TTR). Celle-ci permet de dériver des règles d'induction sur les types de données n'utilisant pas de points fixes dans les termes extraits, avec comme conséquence un résultat de normalisation forte. Toutefois, cette approche, basée beaucoup plus sur la notion d'inclusion, ne semble pas pouvoir s'étendre aux types coïnductifs. C'est pourquoi nous avons préféré étudier uniquement les règles utilisant un point fixe. On reprend cependant la notion d'inclusion de Parigot dans le chapitre 6, sous une forme très simplifiée et uniquement pour optimiser les programmes extraits.

Les travaux de Leivant sont aussi assez proches. Cela n'a rien d'étonnant, puisqu'il est à l'origine du système AF_2 [14]. Dans [16], Leivant montre comment utiliser des définitions inductives en logique du second ordre et en extraire des programmes. L'approche est similaire à celle de Parigot, avec toutefois un point de vue légèrement différent. En effet, Leivant présente différents morphismes associant des programmes aux preuves, tandis que Parigot utilise un formalisme plus proche des sytèmes de types, où le terme extrait est fabriqué conjointement avec la preuve, laissant moins de liberté pour comparer différentes possibilités d'extraction. Toutefois, cette différence n'est pas essentielle et les deux systèmes sont proches (par exemple, on obtient les mêmes représentations en λ-calcul pur pour les définitions inductives des types de données).

Ces travaux ont en commun un système très proche et ne traitent pas des types coïnductifs. Toutefois, ceux-ci on été introduits dans d'autres systèmes de type. Par exemple, dans sa thèse [19], Mendler propose une extension du système $NuPrl$ (proche du système de Martin-Löf) avec des types inductifs et coïnductifs. Bien que du point de vue de la logique nos deux systèmes diffèrent beaucoup, les définitions des deux points fixes restent intuitivement identiques. Ainsi, les connecteurs μ et ν de Mendler trouve une interprétation et des règles similaires : par exemple, les règles

$$\frac{\Gamma \vdash b = b' \in B[\left(\mu x : U_j\right) B/x] \quad \Gamma \vdash \left(\mu x : U_j\right) B \in U_j}{\Gamma \vdash b = b' \in \left(\mu x : U_j\right) B}$$

$$\frac{\Gamma \vdash b = b' \in \left(\nu x : U_j\right) B}{\Gamma \vdash out\,(b) = out\,(b') \in B[\left(\nu x : U_j\right) B/x]}$$

sont respectivement analogues aux règles de factorisation du plus petit point fixe et de développement du plus grand point fixe, tandis que les règles μ_{ind} et ν_{ind} sont les analogues de nos règles de plus petit et de plus grand points fixes avec inclusion.

Le point de divergence le plus notable semble provenir du fait que, dans la théorie des types de Martin-Löf, on confond les programmes et les termes dont on étudie les propriétés, tandis que dans le système AF_2 les deux notions restent séparées. En conséquence, lorsqu'on enrichit le langage des objets afin de prouver les propriétés de nouvelles structures, il est aussi nécessaire d'ajouter des règles de réduction pour les nouvelles constantes. Ce faisant, on doit préserver la normalisation forte, car la réduction des objets à une forme canonique est un principe fondamental de ce genre de système.

Ainsi, dans le cas du plus petit point fixe, la règle μ_{ind} introduit un terme de la forme $\mu_{ind}(b; \lambda y \lambda z\, d)$, qui se réduit en $d[\mu_{ind}(b; \lambda z \lambda y\, d), b/z, y]$. Ce terme se réduit donc comme $(!\lambda z \lambda y\; d\; b)$ (où ! serait un combinateur de point fixe), ce qui le rend similaire à un terme extrait de la règle de plus petit point fixe que l'on utilise. La seule différence est que le point fixe n'est réduit qu'en présence de deux arguments, ce qui assure la normalisation forte car le second argument est en un certain sens bien fondé en tant qu'élément d'un plus petit point fixe.

Dans le cas du plus grand point fixe, la règle ν_{ind} introduit un terme de la forme $\nu_{ind}(b; \lambda y \lambda z\, d)$. Cet object étant un élément d'un type de la forme $(\nu x : U_j)B$, il est potentiellement infini. Il ne doit donc pas être réduit comme $(!\lambda y \lambda z\, d\ b)$. En fait, c'est l'introduction de la constante out, associée au développement du plus grand point fixe, qui bloque la réduction grâce à la règle suivante : $out(\nu_{ind}(b; \lambda z \lambda y\, d)) \geq d[\nu_{ind}(b; \lambda z \lambda y\, d), b/z, y]$.

On voit ici apparaître ce qui semble être la différence majeure entre les deux approches : le fait de bloquer la réduction des objets potentiellement infinis en associant un contenu algorithmique aux régles de développement ou de factorisation du plus grand point fixe. Ce blocage de la réduction revient dans d'autres travaux sur le sujet comme ceux de Leclerc et Paulin-Mohring [13] qui ont etudié comment ajouter les types coinductifs au calcul des constructions. Toutefois, des réalisations plus récentes comme les travaux de Tatsuta [25] ou de Coquand [4] abandonnent aussi la normalisation sans cependant proposer de résultats similaires à ceux des chapitres 4 et 5.

1.5 Organisation de la thèse.

La lecture de cette thèse suppose peu de pré-requis. Avant tout, une bonne connaissance du λ-calcul pur est nécessaire. Le lecteur trouvera dans l'appendice A les définitions de la ω-équivalence et de la ω-résolubilité, deux notions peu communes que nous utilisons dans les chapitres 4 et 5. Il est aussi indéniable qu'une pratique des systèmes de déduction et du calcul des prédicats facilite la compréhension.

Le chapitre 2 est consacré à la présentation du système. Il commence par la définition du système AF_2 avant de l'étendre pour définir $AF_2^{\mu\nu}$. Ensuite, nous démontrons le lemme de réduction qui assure la conservation du type pendant la réduction.

Le chapitre 3 traite de sémantique. Nous y définissons la notion d'interprétation et y démontrons le lemme de conservation qui affirme que l'ensemble des termes de type A est inclus dans l'interprétation du type A. Cette notion est le principal outil utilisé dans les chapitres suivants.

Le chapitre 4 s'intéresse aux propriétés de la normalisation dans ce système étendu. Nous montrons deux résultats permettant de conclure à la ω-résolubilité des termes typés. Le premier résultat utilise le système complet et donne une condition sémantique sur les formules afin que tous les termes typés soient héréditairement résolubles. Le deuxième donne une condition syntaxique simple pour le système avec le plus grand point fixe uniquement.

Le chapitre 5 étudie les types de données. Après présentation d'un certain nombre d'exemples, nous reprenons la définition de la notion de type de données due à Krivine [11, 12], qui est suffisante pour justifier la méthode de programmation. Nous exposons ensuite une méthode de construction d'une classe de type de données plus large que ce qui était déjà connu (puisqu'elle contient des types tels les streams), mais qui reste incluse dans la notion de Krivine. Les exemples de types de données infinies intervenant dans ce chapitre sont les streams et quelques variations sur les suites infinies de 0 et 1 obtenues en mélangeant un plus petit et un plus grand point fixe.

Le chapitre 6 introduit un certain nombre de règles dérivées, ainsi que les optimisations qui permettent d'en extraire des termes plus simples.

Le chapitre 7 contient deux exemples d'applications : le crible d'Eratosthène (qui n'est pas sans surprise), et l'utilisation d'un type de données non-standard pour les entiers qui permet de typer toutes les fonctions récursives.

Le chapitre 8 est consacré aux problèmes posés par la définition de l'égalité. On y définit l'égalité sur les types de données infinies ainsi que la notion de type de données avec égalité qui permet de définir des types quotients.

La thèse se termine par des appendices, où l'on trouve la définition de la ω-équivalence et de la ω-résolubilité (appendice A), la justification logique des règles de points fixes qui montrent que la logique sous-jacente à $AF_2^{\mu\nu}$ est une extension conservative de la logique intuitionniste du second ordre (appendice B), et la définition de nouveaux connecteurs permettant de simplifier les termes extraits en omettant le contenu algorithmique d'une partie de la preuve (appendice C). Dans ces appendices, nous définissons des notions et prouvons des résultats qui sont par ailleurs utile pour le reste de la thèse, mais qui ne sont pas nouveaux.

L'appendice D est un index des notations et conventions utilisées tout au long de ce travail. Certaines notations ne sont introduites que dans cet appendice, le lecteur est donc invité à s'y reporter fréquemment.

Chapitre 2

Définition du système.

2.1 Définition du système AF_2.

Rappelons brièvement la syntaxe et les règles de typage du *système* AF_2.

La première chose à faire est de définir l'*alphabet* utilisé pour construire les formules du système. Pour cela, donnons-nous les ensembles infinis et dénombrables suivants :

- $\mathcal{V}$ (*variables du premier ordre*).
- $\mathcal{V}_n$ pour tout entier n (*variables de prédicat* d'arité n).
- $\mathcal{C}_n$ pour tout entier n (*constantes de fonction* d'arité n).

L'*alphabet* $\mathcal{A}$ de AF_2 est alors

$$\mathcal{A} = \left(\bigcup_{n\in N} \mathcal{V}_n\right) \cup \left(\bigcup_{n\in N} \mathcal{C}_n\right) \cup \mathcal{V} \cup \left\{\rightarrow, \forall, (,)\right\}.$$

L'étape suivante est la définition de l'ensemble $\mathcal{T}$ des *termes logiques* (ou *termes du premier ordre*) et de l'ensemble $\mathcal{F}$ des *formules*, comme les plus petits sous-ensembles de mots sur l'alphabet $\mathcal{A}$ vérifiant :

$x \in \mathcal{T}$	si $x \in \mathcal{V}$
$c \in \mathcal{T}$	si $c \in \mathcal{C}_0$
$f(t_1, \ldots, t_n) \in \mathcal{T}$	si $(t_1, \ldots, t_n) \in \mathcal{T}^n$ et $f \in \mathcal{C}_n$
$X \in \mathcal{F}$	si $X \in \mathcal{V}_0$
$X(t_1, \ldots, t_n) \in \mathcal{F}$	si $(t_1, \ldots, t_n) \in \mathcal{T}^n$ et $X \in \mathcal{V}_n$
$(F \rightarrow G) \in \mathcal{F}$	si $(F, G) \in \mathcal{F}^2$
$(\forall x\, F) \in \mathcal{F}$	si $F \in \mathcal{F}$ et $x \in \mathcal{V}$
$(\forall X\, F) \in \mathcal{F}$	si $F \in \mathcal{F}$ et $X \in \mathcal{V}_n$

On définit[1] alors les notions habituelles de variables libres ou liées par un quantificateur (ici uniquement $\forall$), puis la α*-équivalence* qui identifie deux formules égales aux noms des variables

[1] Ces définitions ne seront pas données en détail ici.

liées près, et enfin la *substitution* des variables du premier ordre par des termes logiques, ainsi que la *substitution* des variables du second ordre par des formules (en renommant les variables liées afin d'éviter les captures). Ces substitutions sont notées respectivement :

$$F[x \Leftarrow t] \text{ et } F[X \Leftarrow \lambda x_1 \dots \lambda x_n\, G],\ X \text{ étant d'arité } n.$$

On considérera aussi la substitution *physique*, qui substitue une formule sans renommer les variables liées, et donc qui autorise les captures. On la notera

$$F\langle X \Leftarrow \lambda x_1 \dots \lambda x_n\, G\rangle,\ X \text{ étant d'arité } n.$$

Cette substitution est utilisée pour désigner les sous-formules d'une formule. En effet, G est une sous-formule de F si et seulement si F s'écrit $F'\langle X \Leftarrow G\rangle$.

Le lecteur est invité à lire dès maintenant l'index des notations où il trouvera toutes les autres notations, conventions ou abréviations relatives à la syntaxe du système.

Une caractéristique importante du système AF_2 est d'utiliser des équations pour spécifier les fonctions que l'on veut programmer. Pour cela, on se donne un *système d'équations* $\mathcal{E}$ sur les termes logiques. Si, en utilisant le raisonnement équationnel, on déduit $t = u$ du système $\mathcal{E}$, on écrira :

$$\mathcal{E} \vdash t = u.$$

Enfin, on définit les *séquents* :

$$\Gamma \vdash \mathtt{t} : F$$

où $\mathtt{t}$ est un terme du λ-calcul, F une formule, et Γ un *contexte* de la forme $\mathtt{x_1} : F_1, \dots, \mathtt{x_n} : F_n$ ($\mathtt{x_1}, \dots, \mathtt{x_n}$ étant des variables du λ-calcul et $F_1, \dots, F_n$ étant des formules). De plus, on suppose qu'aucune variable du λ-calcul n'est déclarée plus d'une fois dans le contexte Γ. Notons aussi que le contexte est géré comme un ensemble de couples. L'ordre de déclaration n'importe donc pas.

On dira que l'on prouve un tel séquent s'il a été construit en appliquant les *règles de déductions* suivantes :

- axiome :
$$\frac{}{\mathtt{x_1} : F_1, \dots, \mathtt{x_n} : F_n \vdash \mathtt{x_i} : F_i \text{ pour } 1 \leq i \leq n}\ Ax$$
- règle équationnelle :
$$\frac{\Gamma \vdash \mathtt{t} : F[x \Leftarrow u] \qquad \mathcal{E} \vdash u = v}{\Gamma \vdash \mathtt{t} : F[x \Leftarrow v]}\ Eq$$
- introduction de l'implication :
$$\frac{\mathtt{x} : F, \Gamma \vdash \mathtt{t} : G}{\Gamma \vdash \lambda \mathtt{x}\, \mathtt{t} : F \to G}\ \to_i$$
- élimination de l'implication :
$$\frac{\Gamma \vdash \mathtt{t} : F \to G \qquad \Gamma \vdash \mathtt{u} : F}{\Gamma \vdash (\mathtt{t}\ \mathtt{u}) : G}\ \to_e$$

- introduction de l'abstraction d'ordre 1 :

$$\frac{\Gamma \vdash \mathtt{t} : F \text{ et } x \text{ non libre dans } \Gamma}{\Gamma \vdash \mathtt{t} : \forall x F} \forall_i^1$$

- élimination de l'abstraction d'ordre 1 :

$$\frac{\Gamma \vdash \mathtt{t} : \forall x F}{\Gamma \vdash \mathtt{t} : F[x \Leftarrow u]} \forall_e^1$$

- introduction de l'abstraction d'ordre 2 :

$$\frac{\Gamma \vdash \mathtt{t} : F \text{ et } X \text{ non libre dans } \Gamma}{\Gamma \vdash \mathtt{t} : \forall X F} \forall_i^2$$

- élimination de l'abstraction d'ordre 2 :

$$\frac{\Gamma \vdash \mathtt{t} : \forall X F}{\Gamma \vdash \mathtt{t} : F[X \Leftarrow \lambda \overline{x} G]} \forall_e^2$$

On ne s'intéressera pas en détail aux propriétés de ce système qui sont décrites dans [11, 12]. On va directement proposer l'extension qui fait l'objet de cette thèse.

2.2 Extension du système.

Comme nous l'avons vu dans l'introduction, les programmes obtenus dans le système AF_2 pur ne peuvent prétendre être réalistes. L'extension que nous proposons comble en partie ce défaut.

Nous allons ajouter à la logique deux connecteurs qui vont permettre de définir des types *inductifs* autorisant une représentation plus efficace des types de données usuels, et des types *coïnductifs* permettant de définir des types de données infinies.

On commence par ajouter $\{\lambda, \langle, \rangle, \mu, \nu\}$ à l'alphabet d'AF_2 qui devient :

$$\mathcal{A} = \left(\bigcup_{n \in N} \mathcal{V}_n\right) \cup \left(\bigcup_{n \in N} \mathcal{C}_n\right) \cup \mathcal{V} \cup \left\{\rightarrow, \forall, (,), \lambda, \langle, \rangle, \mu, \nu\right\}.$$

On ne peut pas définir directement l'ensemble des formules, car nos deux nouveaux connecteurs (μ et ν) nécessitent une condition de positivité des occurrences de la variable de prédicat qu'ils lient. On définit donc d'abord l'ensemble $\mathcal{F}'$ des *pseudo-formules* en ajoutant simplement nos deux connecteurs. L'ensemble des *pseudo-formules* est donc le plus petit ensemble de mots sur l'alphabet $\mathcal{A}$ vérifiant :

$x \in \mathcal{T}$	si $x \in \mathcal{V}$
$c \in \mathcal{T}$	si $c \in \mathcal{C}_0$
$f(t_1,\ldots,t_n) \in \mathcal{T}$	si $(t_1,\ldots,t_n) \in \mathcal{T}^n$ et $f \in \mathcal{C}_n$
$X \in \mathcal{F}'$	si $X \in \mathcal{V}_0$
$X(t_1,\ldots,t_n) \in \mathcal{F}'$	si $(t_1,\ldots,t_n) \in \mathcal{T}^n$ et $X \in \mathcal{V}_n$
$(F \to G) \in \mathcal{F}'$	si $(F,G) \in \mathcal{F}'^2$
$(\forall x\, F) \in \mathcal{F}'$	si $F \in \mathcal{F}'$ et $x \in \mathcal{V}$
$(\forall X\, F) \in \mathcal{F}'$	si $F \in \mathcal{F}'$ et $X \in \mathcal{V}_n$
$\mu X \lambda x_1 \ldots \lambda x_n\, F\langle t_1,\ldots,t_n\rangle \in \mathcal{F}'$	si $X \in \mathcal{V}_n$,$\overline{x} \in \mathcal{V}^n$, $F \in \mathcal{F}'$ et $\overline{t} \in T^n$
$\nu X \lambda x_1 \ldots \lambda x_n\, F\langle t_1,\ldots,t_n\rangle \in \mathcal{F}'$	si $X \in \mathcal{V}_n$,$\overline{x} \in \mathcal{V}^n$, $F \in \mathcal{F}'$ et $\overline{t} \in T^n$

Notons que, dans $\mu X \lambda x_1 \ldots \lambda x_n\, F\langle t_1,\ldots,t_n\rangle$ et $\nu X \lambda x_1 \ldots \lambda x_n\, F\langle t_1,\ldots,t_n\rangle$, les occurrences des variables X,$x_1,\ldots,x_n$ apparaissant dans F sont liées, tandis que les occurrences de $x_1,\ldots,x_n$ dans $t_1,\ldots,t_n$ restent libres.

Afin d'introduire cette condition de positivité, on associe alors à chaque pseudo-formule deux ensembles $V^+(F)$ et $V^-(F)$ qui sont les variables de prédicat ayant des occurrences respectivement *positives* ou *négatives* dans F. Ces ensembles sont définis par induction sur la construction de la formule :

$V^+(X(t_1,\ldots,t_n))$	$=$	$\{X\}$	$V^-(X(t_1,\ldots,t_n))$	$=$	$\emptyset$
$V^+(F \to G)$	$=$	$V^+(G) \cup V^-(F)$	$V^-(F \to G)$	$=$	$V^-(G) \cup V^+(F)$
$V^+(\forall x\, F)$	$=$	$V^+(F)$	$V^-(\forall x\, F)$	$=$	$V^-(F)$
$V^+(\forall X\, F)$	$=$	$V^+(F) - \{X\}$	$V^-(\forall X\, F)$	$=$	$V^-(F) - \{X\}$
$V^+(\mu X \lambda \overline{x}\, F\langle \overline{t}\rangle)$	$=$	$V^+(F) - \{X\}$	$V^-(\mu X \lambda \overline{x}\, F\langle \overline{t}\rangle)$	$=$	$V^-(F) - \{X\}$
$V^+(\mu X \lambda \overline{x}\, F\langle \overline{t}\rangle)$	$=$	$V^+(F) - \{X\}$	$V^-(\mu X \lambda \overline{x}\, F\langle \overline{t}\rangle)$	$=$	$V^-(F) - \{X\}$

On peut maintenant définir l'ensemble $\mathcal{F}$ des formules du nouveau système comme l'ensemble des pseudo-formules qui ont $\mu X \lambda \overline{x}\, F\langle \overline{t}\rangle$ ou $\nu X \lambda \overline{x}\, F\langle \overline{t}\rangle$ comme sous-formule seulement si $X \notin V^-(F)$ (c'est à dire que X n'a que des occurrences positives dans F).

On interprète la formule $\mu X \lambda x_1 \ldots \lambda x_n\, F\langle t_1,\ldots,t_n\rangle$, (resp. ν) comme $X(t_1,\ldots,t_n)$ où X est le plus petit (resp. le plus grand) prédicat vérifiant $\forall x_1 \ldots \forall x_n (X(x_1,\ldots,x_n) \leftrightarrow F)$. On peut aussi l'interpréter comme $X(t_1,\ldots,t_n)$, X étant le plus petit (resp. le plus grand) point fixe de la fonction qui associe au prédicat X le prédicat $\lambda \overline{x}\, F$. La condition de positivité assure la croissance de cette fonction et donc l'existence du point fixe. On notera que ces prédicats sont définissables dans AF_2 (cf annexe B), mais ne permettent pas d'extraire les programmes voulus.

On utilisera très souvent des substitutions de la forme $F[X \Leftarrow \lambda \overline{x} \mu X \lambda \overline{x}\, G\langle \overline{x}\rangle]$. Or, l'expression $\lambda \overline{x} \mu X \lambda \overline{x}\, G\langle \overline{x}\rangle$ est, en quelque sorte, η-équivalente à $\mu X \lambda \overline{x}\, G$. Donc pour simplifier, on notera cette substitution $F[X \Leftarrow \mu X \lambda \overline{x}\, G]$.

On définit aussi les notions de variables *visiblement positives* et *visiblement négatives* dans une formule F. On définit ces notions par induction sur la hauteur de la formule F :

- X est visiblement positive dans F si et seulement si on est dans l'un des cas suivants :
  - $F = X(\overline{t})$.
  - $F = G \to H$, avec X visiblement positive dans H, et sans d'occurrence dans G.

  - $F = \forall \chi H$, avec X visiblement positive dans H et $\chi \neq X$, χ désignant une variable quelconque (premier ou second ordre).

- X est visiblement négative dans F si et seulement si on est dans l'un des cas suivants :
  - $F = X(\overline{t}) \to H$, avec X sans occurrence dans H.
  - $F = G \to H$, avec X visiblement négative dans H et sans occurrence dans G.
  - $F = \forall \chi H$, avec X visiblement négative dans H et $\chi \neq X$.

On remarquera que X est visiblement positive dans F si et seulement X est la variable la plus à droite dans la formule F. De plus, si X est visiblement positive ou négative dans F alors X a exactement une occurrence dans F.

En fait, la formule $\forall \overline{x}(F \to X(\overline{t}))$ (resp. $\forall \overline{x}(X(\overline{t}) \to F)$) est le cas typique d'une formule où X est visiblement positive (resp. négative).

Ces conditions impliquent que si X est interprété par le prédicat toujours vrai (resp. toujours faux) et si X est visiblement positive dans F (resp. négative), alors F est vrai. La justification de ces notions est plus compliquée et n'interviendra qu'au travers du lemme 3.13 du chapitre 3 et dans l'annexe B.

2.3 Les règles du nouveau système.

On ajoute six règles à celles du sytème AF_2. Pour justifier informellement les premières de ces règles, considérons les deux formules suivantes :

$$\begin{array}{rcl} G_0 & = & \mu X \lambda \overline{x}\, F\langle \overline{t} \rangle \\ G_1 & = & F[\overline{x} \Leftarrow \overline{t}][X \Leftarrow \mu X \lambda \overline{x}\, F] \end{array}$$

L'interprétation de G_0, implique que G_0 et G_1 sont équivalentes. En effet, $G_0 \leftrightarrow K(\overline{t})$, K étant le plus petit prédicat satisfaisant $\forall \overline{x}(K(\overline{x}) \leftrightarrow F[X \Leftarrow K])$. On en déduit donc $G_0 \leftrightarrow K(\overline{t}) \leftrightarrow F[\overline{x} \Leftarrow \overline{t}][X \Leftarrow K] \leftrightarrow G_1$. Les règles μ_d et μ_f vont affirmer cette équivalence et son absence de contenu algorithmique. Un terme sera de type G_0 si et seulement s'il est de type G_1.

En fait, on va être un peu plus général que cela. Ces règles vont exprimer que pour toute formule H, $H\langle Y \Leftarrow G_0 \rangle$ est équivalente $H\langle Y \Leftarrow G_1 \rangle$. Cette équivalence est dérivable à partir de l'équivalence de G_0 et G_1. Mais cela introduit une η-expansion au niveau du terme qui se répercute sur le lemme de réduction. En fait, travailler avec ces règles revient à travailler sur l'ensemble des formules quotientées par l'équivalence de G_0 et G_1.

Voici donc ces règles (les règles ν_d et ν_f sont obtenues en remplaçant μ par ν) :

- développement d'un point fixe (valable aussi avec ν) :

$$\frac{\Gamma \vdash \mathtt{t} : H\Big\langle Y \Leftarrow \mu X \lambda \overline{x}\, F\langle \overline{t} \rangle \Big\rangle}{\Gamma \vdash \mathtt{t} : H\Big\langle Y \Leftarrow F\Big[\overline{x} \Leftarrow \overline{t}\Big][X \Leftarrow \mu X \lambda \overline{x}\, F]\Big\rangle}\ \mu_d$$

- factorisation d'un point fixe (valable aussi avec ν) :

$$\frac{\Gamma \vdash \mathtt{t} : H\Big\langle Y \Leftarrow F\big[\overline{x} \Leftarrow \overline{t}\big][X \Leftarrow \mu X \lambda \overline{x}\, F]\Big\rangle}{\Gamma \vdash \mathtt{t} : H\Big\langle Y \Leftarrow \mu X \lambda \overline{x}\, F \langle \overline{t} \rangle\Big\rangle}\ \mu_f$$

Pour l'instant, nos deux connecteurs sont traités de la même manière. Afin de les distinguer, on introduit deux règles qui expriment le fait que ces connecteurs dénotent respectivement un plus petit et un plus grand point fixe :

- règle du plus petit point fixe :

$$\frac{\Gamma \vdash \mathtt{t} : H \to \Big(H[X \Leftarrow \lambda \overline{x}\, F]\Big)}{\Gamma \vdash !\mathtt{t} : H[X \Leftarrow \mu X \lambda \overline{x}\, F]}\ \mu_i \quad \star$$

- règle du plus grand point fixe :

$$\frac{\Gamma \vdash \mathtt{t} : H \to \Big(H[X \Leftarrow \lambda \overline{x}\, F]\Big)}{\Gamma \vdash !\mathtt{t} : H[X \Leftarrow \nu X \lambda \overline{x}\, F]}\ \nu_i \quad \star$$

$\star$: la règle μ_i (resp. ν_i) n'étant applicable que si X n'est pas libre dans Γ, n'a pas d'occurrence négative dans F, et est visiblement négative (resp. visiblement positive) dans H.

La notation "!" dans les termes du λ-calcul dénote un opérateur de point-fixe. Il peut être aussi bien défini que considéré comme une constante. On conviendra simplement que le terme $!\mathtt{t}$ se réduit en une seule étape de β-réduction au terme $(\mathtt{t})!\mathtt{t}$.

Dans la pratique, on utilise ces règles surtout avec H de la forme $\forall \overline{x}(X(\overline{t}) \to A)$ pour la règle μ_i et de la forme $\forall \overline{x}(A \to X(\overline{t}))$ dans le cas de la règle ν_i.

Ces règles se justifient informellement en considérant nos nouveaux connecteurs comme des points fixes. Par exemple, le plus grand peut être construit par induction ordinale, en posant que K_0 est le prédicat toujours vrai et que K_α est l'intersection pour tout ordinal $\beta < \alpha$ des prédicats $K'_\beta = \lambda \overline{x}\, F[X \Leftarrow \lambda \overline{x}\, K_\beta(\overline{x})]$. La prémisse de la règle assure que $H[X \Leftarrow K_\alpha]$ implique $H[X \Leftarrow K_{\alpha+1}]$, la condition affirmant que X est visiblement positif dans H justifiant le cas de 0 et celui des ordinaux limites (cf chapitre 3).

2.4 Quelques exemples de typages.

Afin de rendre les choses un peu plus concrètes, on va introduire deux exemples simples, l'un utilisant le plus petit point fixe, pour définir une représentation unaire des entiers naturels, l'autre utilisant le plus grand point fixe pour définir les *streams*.

Le lecteur peut aussi se reporter aux chapitres 6 et 7 pour des exemples plus complets (la lecture des chapitres précédents n'étant pas nécessaire). Ces exemples seront aussi repris dans le chapitre 5 sur les types de données.

2.4.1 Les entiers récursifs.

On se donne deux constantes **0** et **s** respectivement d'arité 0 et 1. On considère alors la formule $\mathrm{N}[x]$ suivante définissant les entiers naturels :

$$\mathrm{N}[x] = \mu K \lambda y \forall X \left(X\mathbf{0}, \forall z \left(Kz \to X\mathbf{s}z \right) \to Xy \right) \langle x \rangle$$

Pour comprendre cette définition, il suffit de considérer la formule

$$F[K, y] = \forall X \left(X\mathbf{0}, \forall z \left(Kz \to X\mathbf{s}z \right) \to Xy \right).$$

Cette formule associe au prédicat K un prédicat $\lambda y\, F[K, y]$, qui est le plus petit prédicat contenant **0** et les successeurs d'élément de K. Il est donc clair que le plus petit point fixe de la fonction qui associe $\lambda y\, F[K, y]$ à K est l'ensemble des entiers naturels.

On extrait alors les programmes suivants pour le zéro et le successeur:

$$\begin{array}{lcl} \vdash \mathtt{zero} = \lambda \mathtt{x} \lambda \mathtt{f}\, \mathtt{x} & : & \mathrm{N}[\mathbf{0}] \\ \vdash \mathtt{succ} = \lambda \mathtt{n} \lambda \mathtt{x} \lambda \mathtt{f}\, (\mathtt{f}\ \mathtt{n}) & : & \forall x \left(\mathrm{N}[x] \to \mathrm{N}[\mathbf{s}x] \right) \end{array}$$

Cette définition permet d'obtenir des programmes sur les entiers naturels en représentation unaire qui ont les propriétés informatiques usuelles. On peut par exemple obtenir un prédécesseur en temps constant, un programme qui calcule le plus petit de deux entiers en un temps proportionnel au résultat, etc (cf [22]).

2.4.2 Typage des streams.

On va maintenant donner un premier exemple d'utilisation du plus grand point fixe : le typage des "streams" (ou listes infinies). On se donne un constructeur d'arité 2 noté **cs** et on considère la formule $\mathrm{S}_A[x]$ caractérisant les streams d'objets de type A :

$$\mathrm{S}_A[x] = \nu K \lambda y \forall X \left(\forall a \forall s \left(A[a], Ks \to X\mathbf{cs}\,(a, s) \right) \to Xy \right) \langle x \rangle$$

De la même manière que pour les entiers, considérons la formule

$$F[K, y] = \forall X \left(\forall a \forall s \left(A[a], Ks \to X\mathbf{cs}\,(a, s) \right) \to Xy \right).$$

Cette formule définit l'ensemble de tous les termes de la forme $\mathbf{cs}(a, s)$, où a appartient à A et s appartient à K. Le plus grand point fixe de la fonction qui associe $\lambda y\, F[K, y]$ à K correspond donc bien à l'ensemble des streams d'objets de type A.

Comme première application, nous allons typer le stream $\mathbf{sn}(n)$ des entiers supérieurs ou égaux à n, défini par l'équation

$$\mathbf{sn}\,(n) = \mathbf{cs} \left(n, \mathbf{sn}\,(\mathbf{s}n) \right).$$

Il suffit de prouver

$$\forall x \left(\mathrm{N}[x] \to \mathrm{S}_{\mathrm{N}}[\mathbf{sn}\,(x)] \right)$$

Pour cela, il suffit d'appliquer la règle du plus grand point fixe sur cette formule :

$$\forall x \left(\mathrm{N}[x] \to K\,\mathbf{sn}\,(x)\right) \to \forall x \left(\mathrm{N}[x] \to \forall X \left(\forall a \forall s \left(\mathrm{N}[a], Ks \to X\,\mathbf{cs}\,(a,s)\right) \to X\,\mathbf{sn}\,(x)\right)\right)$$

Pour prouver cette dernière formule, supposons que

$$\begin{array}{lcl} \mathtt{r} & : & \forall x(\mathrm{N}[x] \to K\,\mathbf{sn}(x)) \\ \mathtt{n} & : & \mathrm{N}[x] \text{ et} \\ \mathtt{f} & : & \forall a \forall s(\mathrm{N}[a], Ks \to X\,\mathbf{cs}(a,s)) \end{array}$$

On obtient après quelques éliminations :

$$\left(\mathtt{f\ n}\ \left(\mathtt{r}\ \left(\mathtt{succ\ n}\right)\right)\right) : X\,\mathbf{cs}\left(x, \mathbf{sn}\,(\mathbf{s}x)\right)$$

et donc, en utilisant l'équation $\mathbf{cs}(x, \mathbf{sn}(\mathbf{s}x)) = \mathbf{sn}(x)$, l'introduction de l'implication (3 fois) et la règle du plus grand point fixe, on obtient :

$$\vdash \mathtt{sn} = !\lambda\mathtt{r}\lambda\mathtt{n}\lambda\mathtt{f}\ \left(\mathtt{f\ n}\ \left(\mathtt{r}\ \left(\mathtt{succ\ n}\right)\right)\right) : \forall x \left(\mathrm{N}[x] \to \mathrm{S_N}[\mathbf{sn}\,(x)]\right)$$

On peut bien entendu aussi typer les fonctions **car** et **cdr** définis par les équations

$$\begin{array}{rcl} \mathbf{car}(\mathbf{cs}(a,s)) & = & a \\ \mathbf{cdr}(\mathbf{cs}(a,s)) & = & s \end{array}$$

La preuve est immédiate et on obtient

$$\begin{array}{lcl} \vdash \mathtt{car} = \lambda\mathtt{s}(\mathtt{s}\ \lambda\mathtt{x}\lambda\mathtt{y}\,\mathtt{x}) & : & \forall x(\mathrm{S}_A[x] \to A[\mathbf{car}(x)]) \\ \vdash \mathtt{cdr} = \lambda\mathtt{s}(\mathtt{s}\ \lambda\mathtt{x}\lambda\mathtt{y}\,\mathtt{y}) & : & \forall x(\mathrm{S}_A[x] \to \mathrm{S}_A[\mathbf{cdr}(x)]) \end{array}$$

2.5 Lemme de réduction.

L'objet de cette section est de démontrer le lemme syntaxique fondamental de tous les systèmes de type : la compatibilité avec la notion de réduction (ici la β-réduction). Tantôt appelé "préservation du type", tantôt "subject reduction", il s'énonce ainsi :

Proposition 2.1 (subject reduction). *Si* $\mathtt{t}$ *et* $\mathtt{t}'$ *sont deux termes tel que* $\mathtt{t} \triangleright_\beta \mathtt{t}'$ *et* $\Gamma \vdash \mathtt{t} : F$*, alors*

$$\Gamma \vdash \mathtt{t}' : F$$

Afin de prouver cette proposition, on va d'abord prouver trois lemmes techniques.

Lemme 2.2. *Si l'on prouve* $\Gamma \vdash \mathtt{t} : F$*, et si* Φ *est une expression substituable à la variable* χ *(variable du premier ou du second ordre) alors on a aussi* $\Gamma[\chi \Leftarrow \Phi] \vdash \mathtt{t} : F[\chi \Leftarrow \Phi]$.

(Si $\Gamma = \mathtt{x_1} : F_1, \ldots, \mathtt{x_n} : F_n$, alors $\Gamma[\chi \Leftarrow \Phi]$ dénote $\mathtt{x_1} : F_1[\chi \Leftarrow \Phi], \ldots, \mathtt{x_n} : F_n[\chi \Leftarrow \Phi]$).

Démonstration : On démontre ce résultat par induction sur la hauteur des preuves. Étant donnée une preuve de $\Gamma \vdash t : F$, on distingue les cas suivants, en regardant la dernière règle appliquée :

- s'il s'agit de l'axiome (Ax) (preuve de hauteur nulle), on a $\Gamma = \mathtt{x_1} : F_1, \ldots, \mathtt{x_n} : F_n$, $\mathtt{t} = \mathtt{x_i}$ et $F = F_i$. On a donc sans problème $\Gamma[\chi \Leftarrow \Phi] \vdash \mathtt{t} : F[\chi \Leftarrow \Phi]$.
- s'il s'agit des règles Eq, $\rightarrow_i$, $\rightarrow_e$, $\forall^1_e$, $\forall^2_e$, μ_d, ν_d, μ_f ou ν_f, il suffit d'appliquer l'hypothèse d'induction aux prémisses avec la même substitution, et d'appliquer à nouveau la règle (dans le cas de la règle équationnelle (Eq), il faut remarquer que si $\mathcal{E} \vdash u = v$ alors $\mathcal{E} \vdash u[x \Leftarrow t] = v[x \Leftarrow t]$).
- s'il s'agit des règles $\forall^1_i$, $\forall^2_i$, μ_i ou ν_i (c'est-à-dire les règles introduisant un quantificateur), alors il peut être nécessaire de renommer la variable quantifiée (si elle est libre dans l'expression substituée). Pour cela, il suffit d'appliquer deux fois l'hypothèse d'induction à la prémisse, une fois pour renommer la variable, une seconde fois pour substituer Φ à χ. On peut alors appliquer la même règle.

■

On remarque que la preuve ainsi obtenue a la même structure que la preuve initiale. Elle utilise exactement les mêmes règles, dans le même ordre (on a simplement effectué des substitutions dans les formules de la preuve). Cette remarque sera utilisée dans une preuve ultérieure.

Lemme 2.3. *Si l'on prouve $\Gamma, \mathtt{x} : F \vdash \mathtt{t} : G$ et $\Gamma \vdash \mathtt{u} : F$, alors on peut prouver*

$$\Gamma \vdash \mathtt{t}[\mathtt{x} \Leftarrow \mathtt{u}] : G$$

Démonstration : On démontre ce résultat par induction sur la hauteur de la preuve de $\Gamma, \mathtt{x} : F \vdash \mathtt{t} : G$. Le cas de l'axiome est immédiat, la preuve de $\Gamma \vdash \mathtt{u} : F$ donne le résultat (on a $F = G$ et $\mathtt{t} = \mathtt{x}$). Pour toutes les autres règles, l'hypothèse d'induction appliquée aux prémisses donne le résultat. ■

Lemme 2.4. *On peut toujours faire passer les règles Eq, μ_d, ν_d, μ_f et ν_f (équation, développement et factorisation d'un point fixe) après les règles $\forall^1_i$, $\forall^1_e$, $\forall^2_i$ et $\forall^2_e$ (règle de quantification universelle). De même, si l'une des règles Eq, μ_d, ν_d, μ_f et ν_f est avant la prémisse gauche de la règle $\rightarrow_e$ (élimination de l'implication), on peut la faire passer après, en modifiant éventuellement la preuve de la prémisse droite.*

Démonstration : Aucune de ces permutations ne pose de problème. Examinons par exemple les cas de μ_d avec $\rightarrow_e$ et $\forall^2_e$:

$$\dfrac{\dfrac{\Gamma \vdash \mathtt{t} : (\forall \chi F)\langle X \Leftarrow \lambda \overline{z} G_0\rangle}{\Gamma \vdash \mathtt{t} : (\forall \chi F)\langle X \Leftarrow \lambda \overline{z} G_1\rangle}\,\mu_d}{\Gamma \vdash \mathtt{t} : F\langle X \Leftarrow \lambda \overline{z} G_1\rangle[\chi \Leftarrow \Phi]}\,\forall^2_e \quad \text{donne} \quad \dfrac{\dfrac{\Gamma \vdash \mathtt{t} : (\forall \chi F)[X \Leftarrow \lambda \overline{z} G_0]}{\Gamma \vdash \mathtt{t} : F[\chi \Leftarrow \Phi]\Big\langle X \Leftarrow \lambda \overline{z} G_0[\chi \Leftarrow \Phi]\Big\rangle}\,\forall^2_e}{\Gamma \vdash \mathtt{t} : F[\chi \Leftarrow \Phi]\Big\langle X \Leftarrow \lambda \overline{z} G_1[\chi \Leftarrow \Phi]\Big\rangle}\,\mu_d$$

(Φ étant une expression substituable à X, et G_0 et G_1 étant de la forme souhaitée pour appliquer la règle μ_d. Il faut remarquer qu'alors $G_0[\chi \Leftarrow \Phi]$ et $G_1[\chi \Leftarrow \Phi]$ sont aussi de la forme souhaitée.)

$$\dfrac{\dfrac{\Gamma \vdash \mathtt{t} : (F \to G)\langle X \Leftarrow \lambda \overline{z} G_0\rangle}{\Gamma \vdash \mathtt{t} : (F \to G)\langle X \Leftarrow \lambda \overline{z} G_1\rangle}\,\mu_d \quad \Gamma \vdash \mathtt{u} : F\langle X \Leftarrow \lambda \overline{z} G_1\rangle}{\Gamma \vdash (\mathtt{t})\,\mathtt{u} : G\langle X \Leftarrow \lambda \overline{z} G_1\rangle}\,\rightarrow_e$$

donne

$$\dfrac{\dfrac{\Gamma \vdash \mathtt{t} : (F \to G)\,\langle X \Leftarrow \lambda\overline{z}G_0\rangle \quad \dfrac{\Gamma \vdash \mathtt{t} : F\langle X \Leftarrow \lambda\overline{z}G_1\rangle}{\Gamma \vdash \mathtt{u} : F\langle X \Leftarrow \lambda\overline{z}G_0\rangle}\,\mu_f}{\Gamma \vdash (\mathtt{t}\ \mathtt{u}) : G\langle X \Leftarrow \lambda\overline{z}G_0\rangle}\,\to_e}{\Gamma \vdash (\mathtt{t}\ \mathtt{u}) : G\langle X \Leftarrow \lambda\overline{z}G_1\rangle}\,\mu_d$$

■

On peut maintenant prouver le lemme de réduction :

Proposition. *(2.1) Si* $\mathtt{t}$ *et* $\mathtt{t}'$ *sont deux termes tel que* $\mathtt{t} \triangleright_\beta \mathtt{t}'$ *et* $\Gamma \vdash \mathtt{t} : F$*, alors on a*

$$\Gamma \vdash \mathtt{t}' : F$$

Démonstration : On peut considérer que $\mathtt{t}$ se réduit à $\mathtt{t}'$ en une étape, on obtient ensuite le résultat par induction sur la longueur de la réduction. Considérons donc deux termes $\mathtt{t}$ et $\mathtt{t}'$ tel que $\mathtt{t}$ se réduise en une étape à $\mathtt{t}'$, et tel que $\Gamma \vdash \mathtt{t} : F$. On construit alors une preuve de $\Gamma \vdash \mathtt{t}' : F$ par induction sur la hauteur de la preuve de départ. Distinguons les cas suivants, en regardant la dernière règle appliquée :

- La dernière règle ne peut être l'axiome, car une variable ne se réduit pas.
- Si la dernière règle est parmi Eq, $\forall_i^1$, $\forall_e^1$, $\forall_i^2$, $\forall_e^2$, μ_d, ν_d, μ_f ou ν_f (règles sans contenu algorithmique), il suffit d'appliquer l'hypothèse d'induction à la prémisse.
- S'il s'agit de l'introduction de l'implication ($\to_e$), on a $\mathtt{t} = \lambda\mathtt{x}\,\mathtt{u}$ et $\mathtt{t}' = \lambda\mathtt{x}\,\mathtt{u}'$ et $\mathtt{u}$ se réduit en une étape à $\mathtt{u}'$. Il suffit donc d'appliquer à nouveau l'hypothèse d'induction à la prémisse.
- S'il s'agit de la règle du plus petit point fixe (μ_i) (resp. du plus grand (ν_i)), on a $\mathtt{t} = !\mathtt{u}$. On peut alors distinguer deux cas :
  - $\mathtt{t}' = !\mathtt{u}'$ avec $\mathtt{u}$ qui se réduit en une étape à $\mathtt{u}'$. Dans ce cas il suffit d'appliquer à nouveau l'hypothèse d'induction à la prémisse.
  - $\mathtt{t}' = (\mathtt{u}\ !\mathtt{u})$. On a alors une preuve Π de $\Gamma \vdash \mathtt{u} : F \to F[X \Leftarrow \lambda\overline{x}\,G]$, avec X non libre dans Γ n'ayant pas d'occurrence négative dans G, et étant visiblement négative (resp. visiblement positive) dans F. On doit trouver une preuve de $\Gamma \vdash (\mathtt{u}\ !\mathtt{u}) : F[X \Leftarrow \lambda\overline{x}\mu X\lambda\overline{x}\,G\langle\overline{x}\rangle]$.

    Posons les notations suivantes :

$$\begin{array}{rcl} G_0 & = & \mu X\lambda\overline{x}G\langle\overline{x}\rangle \\ G_1 & = & G[X \Leftarrow \lambda\overline{x}G_0] \\ F_0 & = & F[X \Leftarrow \lambda\overline{x}G_0] \\ F_1 & = & F[X \Leftarrow \lambda\overline{x}G_1] \end{array}$$

On peut considérer la preuve suivante (en remplaçant μ par ν dans le cas de la règle du plus grand point fixe) :

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\,\Pi}{\Gamma \vdash \mathtt{u} : F \to F[X \Leftarrow \lambda\overline{x}\,G]}}{\Gamma \vdash \mathtt{u} : \forall X\left(F \to F[X \Leftarrow \lambda\overline{x}\,G]\right)}\,\forall_e^2}{\Gamma \vdash \mathtt{u} : F_0 \to F_1}\,\forall_e^2 \qquad \dfrac{\dfrac{\vdots\,\Pi}{\Gamma \vdash \mathtt{u} : F \to F[X \Leftarrow \lambda\overline{x}\,G]}}{\Gamma \vdash !\mathtt{u} : F_0}\,\mu_i}{\dfrac{\Gamma \vdash (\mathtt{u}\ !\mathtt{u}) : F[X \Leftarrow \lambda\overline{x}G_1]}{\Gamma \vdash (\mathtt{u}\ !\mathtt{u}) : F[X \Leftarrow \lambda\overline{x}G_0]}\,\mu_f}\,\to_e$$

- S'il s'agit de l'élimination d'une implication, on a $\mathtt{t} = (\mathtt{u}\ \mathtt{v})$. Distinguons les cas suivants :
  - Si $\mathtt{t}' = (\mathtt{u}'\ \mathtt{v})$, $\mathtt{u}$ se réduisant en une étape à $\mathtt{u}'$. Il suffit d'appliquer l'hypothèse d'induction à la prémisse gauche de la règle.
  - Si $\mathtt{t}' = (\mathtt{u}\ \mathtt{v}')$, $\mathtt{v}$ se réduisant en une étape à $\mathtt{v}'$. Il suffit d'appliquer l'hypothèse d'induction à la prémisse droite de la règle.
  - Si $\mathtt{u} = \lambda\mathtt{x}\,\mathtt{w}$ et $\mathtt{u}' = \mathtt{w}[\mathtt{x} \Leftarrow \mathtt{v}]$, la preuve a nécessairement la forme suivante :

$$\dfrac{\dfrac{\dfrac{\vdots\,\Pi}{\Gamma, \mathtt{x} : G \vdash \mathtt{w} : F}}{\Gamma \vdash \mathtt{u} : G \to F}\,\to_i \qquad \dfrac{\vdots\,\Pi_1}{\Gamma \vdash \mathtt{v} : G_0}}{\Gamma \vdash \mathtt{t} : F_0}\,\to_e$$

Dans le morceau de preuve noté Π_0, il ne peut y avoir que des règles sans contenu algorithmique (Eq, $\forall_i^1$, $\forall_e^1$, $\forall_i^2$, $\forall_e^2$, μ_d, ν_d, μ_f ou ν_f). On peut éliminer les règles Eq, μ_d, ν_d, μ_f ou ν_f. Il suffit de faire descendre une à une ces règles sous les autres règles de Π_0 ($\forall_i^1$, $\forall_e^1$, $\forall_i^2$, $\forall_e^2$), puis sous l'élimination de l'implication ($\to_e$) grâce au lemme 2.4. On obtient ainsi une preuve de la forme suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\,\Pi}{\Gamma, \mathtt{x} : G \vdash \mathtt{w} : F}}{\Gamma \vdash \mathtt{u} : G \to F}\,\to_i \qquad \dfrac{\vdots\,\Pi_1'}{\Gamma \vdash \mathtt{v} : G_0'}}{\Gamma \vdash \mathtt{t} : F_0'}\,\to_e}{\Gamma \vdash \mathtt{t} : F_0}\,\Pi_2$$

où Π_0' n'utilise donc que les règles $\forall_i^1$, $\forall_e^1$, $\forall_i^2$ et $\forall_e^2$ (règles de quantification universelle). On va montrer que l'on peut se ramener au cas ou Π_0' est vide. On remarque que

Π'_0 commence nécessairement par l'introduction d'un quantificateur et se termine par l'élimination d'un quantificateur. Considérons la première règle d'élimination dans Π'_0. Π'_0 contient donc la séquence suivante :

$$\dfrac{\dfrac{\dfrac{\vdots\ \Pi_p}{\Gamma \vdash \mathtt{u} : H}}{\Gamma \vdash \mathtt{u} : \forall\chi\, H}\ \forall_i^1 ou \forall_i^2}{\Gamma \vdash \mathtt{u} : H[\chi \Leftarrow \theta]}\ \forall_e^1 ou \forall_e^2$$
$$\vdots\ \Pi_s$$

Or χ n'apparaît pas dans Γ, donc, en appliquant le lemme 2.2, on peut trouver une preuve Π'_p de $\Gamma \vdash \mathtt{u} : H[\chi \Leftarrow \theta]$ ayant la même structure que Π. C'est-à-dire qu'elle utilise exactement les mêmes règles dans le même ordre. Ainsi, on a fait diminuer la longueur de Π'_0 de deux. Et, en continuant, on trouve une preuve de cette forme :

$$\dfrac{\dfrac{\dfrac{\vdots\ \Pi'}{\Gamma, \mathtt{x} : G'_0 \vdash \mathtt{w} : F'_0}}{\Gamma \vdash \mathtt{u} : G'_0 \to F'_0}\ \to_i \qquad \dfrac{\vdots\ \Pi'_1}{\Gamma \vdash \mathtt{v} : G'_0}}{\Gamma \vdash \mathtt{t} : F'_0}\ \to_e$$
$$\vdots\ \Pi_2$$
$$\Gamma \vdash \mathtt{t} : F_0$$

Ensuite, en appliquant le lemme 2.3 aux preuves Π' et Π'_1, on obtient une preuve Π'' donnant :

$$\vdots\ \Pi''$$
$$\Gamma \vdash \mathtt{w}[\mathtt{x} \Leftarrow \mathtt{v}] : F'_0$$
$$\vdots\ \Pi_2$$
$$\Gamma \vdash \mathtt{w}[\mathtt{x} \Leftarrow \mathtt{v}] : F_0$$

■

On peut faire la remarque suivante sur cette preuve : si Π est une preuve de $\Gamma \vdash \mathtt{t} : F$, si $\mathtt{t}$ se réduit en $\mathtt{t}'$, alors toutes les sous-formules apparaissant dans la preuve Π' de $\Gamma \vdash \mathtt{t}' : F$, construite ci-dessus, peuvent être obtenues par des substitutions successives de sous-formules ou de termes logiques apparaissant dans Π. En effet, lors de cette construction, on a ajouté de nouvelles formules à la preuve uniquement dans le cas du point fixe (F_1 et G_1, sont justement obtenues par des substitutions de formules déjà présentes), et dans le cas de l'élimination de l'implication, par l'application des lemmes 2.2 et 2.4 qui eux aussi ne font apparaître de nouvelles formules que par substitutions de formules déjà présentes (dans le cas du lemme 2.2, cette remarque n'est pas vrai en général, mais la formule Φ à laquelle on l'applique dans cette construction est bien déjà présente).

Cette remarque sera utilisée dans une preuve du chapitre 4.

Chapitre 3

Sémantique.

3.1 Définition et premières propriétés.

Le principe de la notion d'*interprétation* est d'associer à chaque formule F un ensemble de termes du λ-calcul. On démontrera que cet ensemble contient l'ensemble des termes de type F. Ainsi, on aura à notre disposition un puissant outil *sémantique* pour prouver les propriétés des termes typés.

La puissance de cette notion provient de la variété des interprétations possibles pour le quantificateur du second ordre. Ces différentes sortes d'interprétations sont déterminées par la donnée d'un sous-ensemble $\mathcal{C}$ de $\mathcal{P}(\Lambda)$, qui est destiné à être le domaine dans lequel on choisit les variables du second ordre. Λ désigne l'ensemble des termes du λ-calcul, quotienté bar $\sim_\beta$. Il n'est en effet pas nécessaire de travailler avec des conditions moins fortes que la β-équivalence, puisque toutes les propriétés que l'on étudie sont compatibles avec la β-équivalence.

Définition 3.1. Donnons-nous pour l'instant un sous-ensemble $\mathcal{C}$ de $\mathcal{P}(\Lambda)$ vérifiant les conditions suivantes :

- $\mathcal{C}$ est stable pour l'implication.
- $\mathcal{C}$ est stable pour l'intersection et la réunion.
- $\mathcal{C}$ contient $\emptyset$ et Λ

Cet ensemble est le domaine de variation de l'interprétation des variables de prédicat d'arité nulle. Pour tout entier n, on définit aussi le domaine de variation des variables de prédicat d'arité n par :

$$\mathcal{C}_n = \Lambda^n \to \mathcal{C} \qquad \text{(on a donc } \mathcal{C}_n \subset \Lambda^n \to \mathcal{P}(\Lambda))$$

Définition 3.2. On dira qu'un élément Φ de $\Lambda^n \to P(\Lambda)$ est adéquat, si $\Phi \in \mathcal{C}_n$.

Définition 3.3. Une interprétation σ est définie par la donnée de :

- $|x|^\sigma \in \Lambda$ pour chaque constante ou variable d'individu x.
- $|\mathbf{f}|^\sigma \in \Lambda^n \to \Lambda$ pour chaque constante de fonction $\mathbf{f}$ d'arité n.

- $|X|^\sigma \in \mathcal{C}_n$ pour chaque variable de prédicat X d'arité n.

Notation : On notera $\sigma[\chi \Leftarrow \Phi]$ (χ étant une variable du premier ou du second ordre et Φ étant une interprétation possible pour χ) l'interprétation définie par :

$$|\chi|^{\sigma[\chi\Leftarrow\Phi]} = \Phi \text{ et } |\chi'|^{\sigma[\chi\Leftarrow\Phi]} = |\chi'|^\sigma \text{ si } \chi' \neq \chi$$

Notation : Pour Φ et Φ' deux ensembles de λ-termes, on notera :

$$\Phi \Rightarrow \Phi' = \left\{ \mathtt{t} \in \Lambda \setminus \text{ pour tout } \mathtt{u} \in \Phi, (\mathtt{t}\ \mathtt{u}) \in \Phi' \right\}$$

L'hypothèse $\mathcal{C}$ stable pour l'implication signifie que si Φ et Φ' appartiennent à $\mathcal{C}$, alors $\Phi \Rightarrow \Phi'$ aussi.

Définition 3.4. On étend, par induction, la définition d'une interprétation σ à tous les termes logiques et à toutes les formules en posant :

- $|\mathbf{f}(t_1, \ldots, t_n)|^\sigma = |\mathbf{f}|^\sigma(|t_1|^\sigma, \ldots, |t_n|^\sigma)$
- $|X(t_1, \ldots, t_n)|^\sigma = |X|^\sigma(|t_1|^\sigma, \ldots, |t_n|^\sigma)$
- $|F \to G|^\sigma = |F|^\sigma \Rightarrow |G|^\sigma$
- $|\forall x\, F|^\sigma = \bigcap\{|F|^{\sigma[x\Leftarrow\mathtt{v}]} \setminus \mathtt{v} \in \Lambda\}$
- $|\forall X\, F|^\sigma = \bigcap\{|F|^{\sigma[X\Leftarrow\Phi]} \setminus \Phi \in \mathcal{C}_n\}$ (X étant d'arité n)
- $|\mu X \lambda\overline{x}\, F\langle\overline{t}\rangle|^\sigma = \bigcap\{\Phi(|\overline{t}|^\sigma) \setminus \Phi \in \mathcal{C}_n, \forall\overline{\mathtt{v}} \in \Lambda^n\ |F|^{\sigma[X\Leftarrow\Phi][\overline{x}\Leftarrow\overline{\mathtt{v}}]} \subseteq \Phi(\overline{\mathtt{v}})\}$
- $|\nu X \lambda\overline{x}\, F\langle\overline{t}\rangle|^\sigma = \bigcup\{\Phi(|\overline{t}|^\sigma) \setminus \Phi \in \mathcal{C}_n, \forall\overline{\mathtt{v}} \in \Lambda^n\ \Phi(\overline{\mathtt{v}}) \subseteq |F|^{\sigma[X\Leftarrow\Phi][\overline{x}\Leftarrow\overline{\mathtt{v}}]}\}$

De plus, on ne considérera que les $\mathcal{C}$-interprétations σ satisfaisant les équations $\mathcal{E}$ du système, c'est-à-dire que $\mathcal{E} \vdash u = v$ implique $|u|^\sigma = |v|^\sigma$ (Cette condition ne sera utilisée que pour le lemme de conservation 3.14).

L'interprétation des connecteurs μ et ν trouve sont origine dans l'expression usuelle du plus petit et du plus grand point fixe d'une fonction croissante f. En effet, le plus petit (resp. le plus grand) point fixe de f peut être défini comme l'intersection (resp. la réunion) de tous les x vérifiant $f(x) \leq x$ (resp. $x \leq f(x)$). On a besoin d'une telle définition car elle est fondée même si f n'est pas croissante, ce que l'on ne peut pas montrer avant d'avoir défini f !

Comme on va beaucoup manipuler nos interprétations de "manière fonctionnelle", on introduit la notation suivante :

Notation : on note $|\lambda\chi_1 \ldots \chi_n\, F|^\sigma$ la fonction qui à $\Phi_1, \ldots, \Phi_n$ associe $|F|^{\sigma[\overline{\chi}\Leftarrow\overline{\Phi}]}$, où pour chaque i on a, ou bien χ_i est une variable du premier ordre et $\Phi_i \in \Lambda$, ou bien χ_i est une variable de prédicat et $\Phi_i \in \mathcal{C}_n$.

On notera aussi $\Phi \leq \Phi'$ l'ordre canonique sur $\mathcal{C}_n$, défini par $\Phi \leq \Phi'$ si et seulement si $\Phi(\overline{\mathtt{t}}) \subseteq \Phi'(\overline{\mathtt{t}})$ pour tout $\overline{\mathtt{t}} \in \Lambda^n$.

Par exemple : $|\lambda X \lambda x_1 \ldots x_p\, F|^\sigma \in \mathcal{C}_n \to (\Lambda^p \to \mathcal{P}(\Lambda))$ (X étant d'arité n).

Avec cette notation, on peut simplifier la définition de l'interprétation des points fixes en posant :

- $|\mu X \lambda \overline{x}\, F\langle \overline{t} \rangle|^\sigma = \bigcap\{\Phi(|\overline{t}|^\sigma) \setminus \Phi \in \mathcal{C}_n, |\lambda X \lambda \overline{x}\, F|^\sigma(\Phi) \leq \Phi\}$
- $|\nu X \lambda \overline{x}\, F\langle \overline{t} \rangle|^\sigma = \bigcup\{\Phi(|\overline{t}|^\sigma) \setminus \Phi \in \mathcal{C}_n, \Phi \leq |\lambda X \lambda \overline{x}\, F|^\sigma(\Phi)\}$

Proposition 3.5. *Pour toute formule F et toute $\mathcal{C}$-interprétation σ, $|F|^\sigma$ est adéquate (c'est-à-dire $|F|^\sigma \in \mathcal{C}$).*

Remarque: Avec cette proposition, la fonction $|\lambda X \lambda x_1 \ldots x_p\, F|^\sigma$ de l'exemple précédent appartient à $\mathcal{C}_n \to \mathcal{C}_p$.

Démonstration : Démontrons ce résultat par induction sur la hauteur de la formule F.

- Si $F = X(\overline{t})$, alors par définition de σ $|F|^\sigma = |X|^\sigma(|\overline{t}|^\sigma)$ qui est adéquat.
- Si $F = G \to H$, par hypothèse d'induction, $|G|^\sigma$ et $|H|^\sigma$ sont adéquats. Donc $\mathcal{C}$ étant stable pour l'implication, $|F|^\sigma$ est bien adéquat.
- Si $F = \forall x G$. Dans ce cas, pour tout terme $\mathtt{t}$, $|G|^{\sigma[x \Leftarrow \mathtt{t}]}$ est adéquat par hypothèse d'induction, et l'ensemble $\mathcal{C}$ étant stable pour l'intersection, on trouve que $|F|^\sigma$ est bien adéquat.
- Si $F = \forall X G$. Dans ce cas, pour tout Φ adéquat ($\Phi \in \mathcal{C}_n$ si X est d'arité n), $|G|^{\sigma[X \Leftarrow \Phi]}$ est adéquat par hypothèse d'induction, et l'ensemble $\mathcal{C}$ étant stable pour l'intersection, on trouve que $|F|^\sigma$ est bien adéquat.
- Si $F = \mu X \lambda \overline{x}\, G\langle \overline{t} \rangle$ (resp. $F = \nu X \lambda \overline{x}\, G\langle \overline{t} \rangle$), comme précédemment, l'interprétation de F, qui est construite comme une intersection (resp. réunion) d'éléments de $\mathcal{C}_n$, est adéquate.

■

Les deux propositions que l'on va énoncer maintenant affirment la compatibilité de la notion d'interprétation avec la substitution.

Proposition 3.6. *Pour tous termes logiques t, u, toute formule F et toute variable x,*

$$\Big|t[x \Leftarrow u]\Big|^\sigma = |t|^{\sigma[x \Leftarrow |u|^\sigma]} \text{ et } \Big|F[x \Leftarrow u]\Big|^\sigma = |F|^{\sigma[x \Leftarrow |u|^\sigma]}$$

Démonstration : Par induction sur les termes logiques, et étendue sans difficulté aux formules. ■

Proposition 3.7. *Pour toutes formules F et G, et pour toute variable de prédicat X d'arité n,*

$$\Big|F[X \Leftarrow \lambda x_1 \ldots x_n\, G]\Big|^\sigma = |F|^{\sigma[X \Leftarrow |\lambda x_1 \ldots x_n\, G|^\sigma]}$$

Démonstration : Par induction sur la construction de la formule et découlant directement de la définition de l'interprétation (il faut remarque que $\sigma[X \Leftarrow |\lambda x_1 \ldots x_n\, G|^\sigma]$ est bien une $\mathcal{C}$-interprétation par application du lemme 3.5). ■

Ce dernier lemme est faux pour la substitution physique. Toutefois, on peut montrer le résultat suivant :

Proposition 3.8. *Si pour toute interprétation σ on a $|G|^\sigma = |G'|^\sigma$, alors pour toute interprétation σ, toute formule F et toute variable de prédicat X d'arité 0, on a :*

$$\left|F\langle X \Leftarrow G\rangle\right|^\sigma = \left|F\langle X \Leftarrow G'\rangle\right|^\sigma$$

Démonstration : Par induction sur la construction de la formule F ■

3.2 Lemme de croissance.

On va s'intéresser, dans cette section, aux propriétés de l'interprétation liées aux occurrences *positives* ou *négatives* de variables de prédicats. On en déduira que les connecteurs μ et ν correspondent bien respectivement à un plus petit et à un plus grand point fixe.

Lemme 3.9 (lemme de croissance). *Si F est une formule où X variable du second ordre d'arité n n'a que des occurrences positives (resp. négatives), alors pour toute interprétation σ, la fonction $|\lambda X\ F|^\sigma \in \mathcal{C}_n \to \mathcal{C}$ est croissante (resp. décroissante).*

Démonstration : On démontre le résultat par induction sur la construction de la formule F. Pendant toute la démonstration, on considère Φ et $\Phi' \in \mathcal{C}_n$ tel que $\Phi \leq \Phi'$ (resp. $\geq$). Il faut donc montrer que pour toute interprétation σ, on a $|\lambda X\ F|^\sigma(\Phi) \subseteq |\lambda X\ F|^\sigma(\Phi')$.

On note $\Psi_F^\sigma = |\lambda X\ F|^\sigma$ (on omettra σ quand il n'y aura pas d'ambiguïté).

La démonstration par induction nous amène à considérer les cas suivants :

- $F = X(\overline{t})$ (resp. impossible). On a alors

$$\Psi_F(\Phi) = \Phi\left(\left|\overline{t}\right|^\sigma\right) \subseteq \Phi'\left(\left|\overline{t}\right|^\sigma\right) = \Psi_F(\Phi')$$

- $F = Y(t_1, \ldots, t_p)$ et $X \neq Y$. Ψ_F est constante donc croissante et décroissante.

- $F = G \to H$, et X n'a que des occurrences positives dans F (resp. négatives). X n'a donc que des occurrences négatives dans G (resp. positives) et positives dans H (resp. négatives). Donc par hypothèse d'induction, Ψ_G est décroissante (resp. croissante) et Ψ_H est croissante (resp. décroissante).

  Soit $\mathtt{v} \in \Psi_F(\Phi)$ il faut montrer que $\mathtt{v} \in \Psi_F(\Phi')$. Soit $\mathtt{w} \in \Psi_G(\Phi')$ quelconque. On a $\mathtt{w} \in \Psi_G(\Phi)$ (car Ψ_G est décroissante et $\Phi \leq \Phi'$ (resp. croissante et $\Phi \geq \Phi'$)). D'où $(\mathtt{v}\ \mathtt{w}) \in \Psi_H(\Phi)$ et donc $(\mathtt{v}\ \mathtt{w}) \in \Psi_H(\Phi')$ (car Ψ_H est croissante et $\Phi \leq \Phi'$ (resp. décroissante et $\Phi \geq \Phi'$)). On a donc bien $\Psi_F(\Phi) \leq \Psi_F(\Phi')$.

- $F = \forall y\, G$, et X n'a que des occurrences positives dans F (resp. négatives). X n'a donc que des occurrences positives dans G (resp. négatives). Par hypothèse d'induction, pour tout $\mathtt{v} \in \Lambda$, $\Psi_G^{\sigma[y \Leftarrow \mathtt{v}]}$ est croissante (resp. décroissante). Il découle donc directement de la définition de l'interprétation que Ψ_F est croissante (resp. décroissante), car $\mathtt{v} \in \Psi_F(\Phi)$ si et seulement si pour tout $\mathtt{w} \in \Lambda$, $\mathtt{v} \in \Psi_G^{\sigma[y \Leftarrow \mathtt{w}]}(\Phi)$.

- $F = \forall Y\, G$. La démonstration est similaire à celle du cas précédent.

- $F = \mu Y\, \lambda \overline{y} G\langle \overline{t} \rangle$, et X n'a que des occurrences positives dans F (resp. négatives). X n'a donc que des occurrences positives dans G (resp. négatives). Soit $\mathtt{v} \in \Psi_F(\Phi)$, on doit montrer $\mathtt{v} \in \Psi_F(\Phi')$. Étant donnée $\Theta \in \mathcal{C}_n$ tel que $|\lambda Y\, \lambda \overline{y}\, G|^{\sigma[X \Leftarrow \Phi']}(\Theta) \leq \Theta$, il suffit de montrer $\mathtt{v} \in \Theta(|\overline{t}|^{\sigma})$. Or, pour tout $\overline{\mathtt{w}} \in \Lambda^n$, on a

$$\begin{array}{rcl} |\lambda Y\, \lambda \overline{y}\, G|^{\sigma[X \Leftarrow \Phi]}(\Theta)(\overline{\mathtt{w}}) & = & |G|^{\sigma[Y \Leftarrow \Theta][\overline{y} \Leftarrow \overline{\mathtt{w}}][X \Leftarrow \Phi]} \\ & = & |\lambda X\, G|^{\sigma[Y \Leftarrow \Theta][\overline{y} \Leftarrow \overline{\mathtt{w}}]}(\Phi) \\ & \subseteq & |\lambda X\, G|^{\sigma[Y \Leftarrow \Theta][\overline{y} \Leftarrow \overline{\mathtt{w}}]}(\Phi') \text{ (hyp. d'induction)} \\ & = & |G|^{\sigma[Y \Leftarrow \Theta][\overline{y} \Leftarrow \overline{\mathtt{w}}][X \Leftarrow \Phi']} \\ & = & |\lambda Y\, \lambda \overline{y}\, G|^{\sigma[X \Leftarrow \Phi']}(\Theta)(\overline{\mathtt{w}}) \end{array}$$

  Donc on a

$$\begin{array}{rcl} |\lambda Y\, \lambda \overline{y}\, G|^{\sigma[X \Leftarrow \Phi]}(\Theta) & \leq & |\lambda Y\, \lambda \overline{y}\, G|^{\sigma[X \Leftarrow \Phi']}(\Theta) \\ & \leq & \Theta \end{array}$$

  Or comme par hypothèse on a $\mathtt{v} \in \Psi_F(\Phi)$, on en déduit $\mathtt{v} \in \Theta(|\overline{t}|^{\sigma})$.

- $F = \nu Y\, \lambda \overline{y} G\langle \overline{t} \rangle$, et X n'a que des occurrences positives dans F (resp. négatives). X n'a donc que des occurrences positives dans G (resp. négatives). Soit $\mathtt{v} \in \Psi_F(\Phi)$, on doit montrer $\mathtt{v} \in \Psi_F(\Phi')$. Par définition, on trouve Θ tel que $\mathtt{u} \in \Theta(|\overline{t}|^{\sigma})$ et $\Theta \leq |\lambda Y\, \lambda \overline{y}\, G|^{\sigma[X \Leftarrow \Phi]}(\Theta)$. Il suffit donc de montrer $\Theta \leq |\lambda Y\, \lambda \overline{y}\, G|^{\sigma[X \Leftarrow \Phi']}$, que l'on obtient par un raisonnement analogue au précédent. ■

Corollaire 3.10. *Si F est une formule où X variable du second ordre d'arité n n'a que des occurrences positives (resp. négatives), alors pour toute interprétation σ, la fonction*

$$|\lambda X\, \lambda x_1 \ldots \lambda x_n\, F|^{\sigma} \in \mathcal{C}_n \to \mathcal{C}_n$$

est croissante (resp. décroissante)

Démonstration : Ce corollaire découle immédiatement du lemme de croissance. ■

Corollaire 3.11. *On déduit du lemme précédent que si $\Pi \in \mathcal{C}_n$ est le plus petit (resp. le plus grand) point fixe de $\Psi_F = |\lambda X\, \lambda x_1 \ldots x_n\, F|^{\sigma}$ (X, d'arité n, n'ayant que des occurrences positives dans F) alors :*

$$\left|\mu X\, \lambda \overline{x}\, F\langle \overline{t} \rangle\right|^{\sigma} = \Pi\left(\left|\overline{t}\right|^{\sigma}\right) \text{ (resp.} |\nu X\, \lambda \overline{x}\, F\langle \overline{t} \rangle|^{\sigma} = \Pi(|\overline{t}|^{\sigma})\text{)}$$

Démonstration : Prenons comme définition de Π

$$\Pi = \left|\lambda \overline{y}\, \mu X\, \lambda \overline{x}\, F\langle \overline{y} \rangle\right|^{\sigma} \text{ (resp. } \Pi = |\lambda \overline{y}\, \nu X\, \lambda \overline{x}\, F\langle \overline{y} \rangle|^{\sigma}\text{)}$$

On a bien le résultat voulu. Reste à montrer que Π est bien le point fixe recherché.

- Cas du plus petit point fixe: si $\mathtt{u} \in \Pi(\overline{\mathtt{v}})$ on a, par définition de l'interprétation, $\mathtt{u} \in \Phi(\overline{\mathtt{v}})$ pour tout Φ vérifiant $\Psi_F(\Phi) \leq \Phi$. Donc $\Pi \leq \Phi$ pour tout Φ vérifiant $\Psi_F(\Phi) \leq \Phi$. D'où $\Pi \leq \Phi$ si Φ est un point fixe de Ψ_F.

  Reste à montrer que Π est un point fixe de Ψ_F. D'après ce qui précède, et comme Ψ_F est croissante, pour tout Φ vérifiant $\Psi_F(\Phi) \leq \Phi$ on a $\Psi_F(\Pi) \leq \Psi_F(\Phi) \leq \Phi$. Soit

$\mathtt{u} \in \Psi_F(\Pi)(\overline{\mathtt{v}})$. Pour tout Φ tel que $\Psi_F(\Phi) \leq \Phi$ on a $\mathtt{u} \in \Phi(\overline{\mathtt{v}})$. Donc $\mathtt{u} \in \Pi(\overline{\mathtt{v}})$. On vient de montrer :

$$\Psi_F(\Pi) \leq \Pi$$

De plus comme Ψ_F est croissante, on en déduit que $\Psi_F(\Psi_F(\Pi)) \leq \Psi_F(\Pi)$. Or $\Pi \leq \Phi$ pour tout Φ vérifiant $\Psi_F(\Phi) \leq \Phi$, d'où l'inclusion réciproque :

$$\Pi \leq \Psi_F(\Pi)$$

- Cas du plus grand point fixe: par définition de l'interprétation, $\mathtt{u} \in \Pi(\overline{\mathtt{v}})$ si et seulement s'il existe Φ vérifiant $\Phi \leq \Psi_F(\Phi)$ et $\mathtt{u} \in \Phi(\overline{\mathtt{v}})$. Donc, en particulier, si Φ est un point fixe de Ψ_F, on a $\Phi \leq \Pi$.

  Reste à montrer que Π est un point fixe de Ψ_F. Or, d'après ce qui précède, et comme Ψ_F est croissante, pour tout Φ vérifiant $\Phi \leq \Psi_F(\Phi)$ on a $\Phi \leq \Psi_F(\Phi) \leq \Psi_F(\Pi)$. Étant donné $\mathtt{u} \in \Pi(\overline{\mathtt{v}})$, il existe Φ tel que $\Phi \leq \Psi_F(\Phi)$ et $\mathtt{u} \in \Phi(\overline{\mathtt{v}})$. D'où $\mathtt{u} \in \Psi_F(\Pi)(\overline{\mathtt{v}})$. On vient donc de montrer :

  $$\Pi \leq \Psi_F(\Pi)$$

  De plus, comme Ψ_F est croissante, on en déduit que $\Psi_F(\Pi) \leq \Psi_F(\Psi_F(\Pi))$, et puisque $\Phi \leq \Pi$ pour tout Φ vérifiant $\Phi \leq \Psi_F(\Phi)$ on a :

  $$\Psi_F(\Pi) \leq \Pi$$

■

3.3 Autres propriétés liées aux occurrences.

On va étudier maintenant les propriétés liées à la présence de variables *visiblement positives* ou *visiblement négatives*. La première de ces propositions est utilisée pour le cas 0 de l'induction ordinale (règles μ_i et ν_i) dans la preuve du lemme de conservation.

Proposition 3.12. *Si F est une formule où X est visiblement positive (resp. visiblement négative), alors, pour toute interprétation σ, on a $|F|^\sigma = \Lambda$ si σ vérifie $|X|^\sigma(\overline{\mathtt{t}}) = \Lambda$ (resp. $|X|^\sigma(\overline{\mathtt{t}}) = \emptyset$) pour tout $\overline{\mathtt{t}} \in \Lambda^n$.*

Démonstration : On montre ce résultat par induction sur la construction de la formule F. Si X est visiblement positive (resp. visiblement négative) dans F, on est alors dans l'un des cas suivants :

- $F = X(\overline{t})$ (resp. cas impossible). On a bien $|F|^\sigma = |X|^\sigma(|\overline{t}|^\sigma) = \Lambda$, par hypothèse sur l'interprétation σ.
- $F = G \to H$ où X est visiblement positive dans H (resp. ou bien $G = X(\overline{t})$ et X est sans occurrence dans H ou bien X est visiblement négative dans H et sans occurrence dans G). Alors l'hypothèse d'induction donne $|H|^\sigma = \Lambda$, (resp. l'hypothèse d'induction donne ou bien $|H|^\sigma = \Lambda$, ou bien $|G|^\sigma = \emptyset$), d'où le résultat.
- Si $F = \forall x\, G$ ou $F = \forall X\, G$, l'interprétation de F étant définie comme une intersection, l'hypothèse d'induction donne le résultat voulu.

■

La propriété suivante est nécessaire pour le cas des ordinaux limites dans la même preuve.

Proposition 3.13. *Considérons deux familles $\{\Phi^+_\beta\}_{\beta\in\alpha}$ et $\{\Phi^-_\beta\}_{\beta\in\alpha}$ indexées sur un ordinal α, à valeur dans $\mathcal{C}_n$, la première étant croissante, la seconde étant décroissante. Notons Φ^+ et Φ^- les deux éléments de $\mathcal{C}_n$ suivants :*

$$\begin{array}{rcl}\Phi^+(\mathtt{u}_1,\ldots,\mathtt{u}_n) & = & \bigcup_{\beta<\alpha}\Phi^+_\beta(\mathtt{u}_1,\ldots,\mathtt{u}_n)\\ \Phi^-(\mathtt{u}_1,\ldots,\mathtt{u}_n) & = & \bigcap_{\beta<\alpha}\Phi^-_\beta(\mathtt{u}_1,\ldots,\mathtt{u}_n)\end{array}$$

Considérons aussi une formule F, une variable de prédicat X d'arité n, et une interprétation σ. On note Ψ_F pour $|\lambda X\ F|^\sigma$. On a alors les deux propositions suivantes :

1. *Si X est visiblement négative dans F, on a :*
$$\Psi_F\left(\Phi^+\right) = \bigcap_{\beta<\alpha}\Psi_F\left(\Phi^+_\beta\right)$$
2. *Si X est visiblement positive dans F, on a :*
$$\Psi_F\left(\Phi^-\right) = \bigcap_{\beta<\alpha}\Psi_F\left(\Phi^-_\beta\right)$$

Démonstration : On a $\Phi^+ \geq \Phi^+_\beta$ et $\Phi^- \leq \Phi^-_\beta$ pour tout ordinal $\beta<\alpha$. Or, l'unique occurrence de X dans F est négative (resp. positive) si X est visiblement négative (resp. positive) dans F. D'où, par le lemme de croissance, $\Psi_F(\Phi^+) \subseteq \Psi_F(\Phi^+_\beta)$ (resp. $\Psi_F(\Phi^-) \subseteq \Psi_F(\Phi^-_\beta)$) pour tout ordinal $\beta<\alpha$. On en déduit les inclusions suivantes :

1. Si X est visiblement négative dans F, on a :
$$\Psi_F\left(\Phi^+\right) \subseteq \bigcap_{\beta<\alpha}\Psi_F\left(\Phi^+_\beta\right)$$
2. Si X est visiblement positive dans F, on a :
$$\Psi_F\left(\Phi^-\right) \subseteq \bigcap_{\beta<\alpha}\Psi_F\left(\Phi^-_\beta\right)$$

On montre l'inclusion réciproque (c'est en fait elle dont on aura besoin) par induction sur la construction de la formule F.

Premier cas : on suppose que X est visiblement négative dans F. On doit donc montrer :

$$\Psi_F\left(\Phi^+\right) \supseteq \bigcap_{\beta<\alpha}\Psi_F\left(\Phi^+_\beta\right)$$

Or X étant visiblement négative dans F, on est dans l'un des cas suivants :

- $F = H \to G$, avec X sans occurrence dans H et visiblement négative dans G.

  Supposons que $\mathtt{t} \in \bigcap_{\beta<\alpha}\Psi_F(\Phi^+_\beta)$. Il faut montrer que $\mathtt{t} \in \Psi_F(\Phi^+)$. C'est-à-dire que pour tout $\mathtt{u} \in \Psi_H(\Phi^+)$, on a $(\mathtt{t}\ \mathtt{u}) \in \Psi_G(\Phi^+)$. Soit $\mathtt{u} \in \Psi_H(\Phi^+)$ quelconque, X étant sans occurrence dans H, on obtient pour tout $\beta<\alpha$, $\Psi_H(\Phi^+) = \Psi_F(\Phi^+_\beta)$. Donc $(\mathtt{t}\ \mathtt{u}) \in \Psi_G(\Phi^+_\beta)$ pour tout $\beta<\alpha$, ce qui implique $(\mathtt{t}\ \mathtt{u}) \in \bigcap_{\beta<\alpha}\Psi_G(\Phi^+_\beta)$. D'où le résultat par hypothèse d'induction.

- $F = X(\overline{t}) \to G$ avec X sans occurrence dans G

  Supposons que $\mathtt{t} \in \bigcap_{\beta<\alpha} \Psi_F(\Phi^+_\beta)$. Il faut montrer que $\mathtt{t} \in \Psi_F(\Phi^+)$. C'est-à-dire que pour tout $\mathtt{u} \in \Psi_{X(\overline{t})}(\Phi^+)$, on a $(\mathtt{t}\ \mathtt{u}) \in \Psi_G(\Phi^+)$. Soit $\mathtt{u} \in \Psi_{X(\overline{t})}(\Phi^+)$ quelconque, on a :

  $$\Psi_{X(\overline{t})}\left(\Phi^+\right) = \Phi^+\left(\left|\overline{t}\right|^\sigma\right) = \bigcup_{\beta<\alpha} \Phi^+_\beta\left(\left|\overline{t}\right|^\sigma\right) = \bigcup_{\beta<\alpha} \Psi_{X(\overline{t})}\left(\Phi^+_\beta\right)$$

  Il existe donc $\beta < \alpha$ tel que $\mathtt{u} \in \Psi_{X(\overline{t})}(\Phi^+_\beta)$. De $\mathtt{t} \in \Psi_F(\Phi^+_\beta)$, on déduit $(\mathtt{t}\ \mathtt{u}) \in \Psi_G(\Phi^+_\beta)$, Or X étant sans occurrence dans G, on a $\Psi_G(\Phi^+_\beta) = \Psi_G(\Phi^+)$. D'où le résultat.

- $F = \forall Y\, G$ ou $F = \forall x\, G$, X étant visiblement négative dans G. L'hypothèse d'induction donne :

  $$\Psi_G\left(\Phi^+\right) \supseteq \bigcap_{\beta<\alpha} \Psi_G\left(\Phi^+_\beta\right)$$

  Or, l'interprétation de F étant définie comme une intersection, celle-ci permute avec l'intersection ci-dessus et donne le résultat voulu.

Deuxième cas : X est visiblement positif dans F. On doit donc montrer :

$$\Psi_F\left(\Phi^-\right) \supseteq \bigcap_{\beta<\alpha} \Psi_F\left(\Phi^-_\beta\right)$$

X étant visiblement positif dans F, on est dans l'un des cas suivants :

- $F = X(\overline{t})$. On a :

  $$\begin{array}{rcl} \Psi_F(\Phi^-) & = & \Phi^-(|\overline{t}|^\sigma) \\ & = & \bigcap_{\beta<\alpha} \Phi^-_\beta(|\overline{t}|^\sigma) \\ & = & \bigcap_{\beta<\alpha} \Psi_F(\Phi^-_\beta) \end{array}$$

- $F = H \to G$, X est visiblement positif dans G et sans occurrence dans H. Soit $\mathtt{t} \in \bigcap_{\beta<\alpha} \Psi_F(\Phi^-_\beta)$. Il faut montrer que $\mathtt{t} \in \Psi_F(\Phi^-)$. C'est-à-dire que pour tout $\mathtt{u} \in \Psi_H(\Phi^-)$, on a $(\mathtt{t}\ \mathtt{u}) \in \Psi_G(\Phi^-)$. Soit $\mathtt{u} \in \Psi_H(\Phi^-)$ quelconque. Ψ_H étant constante, on a

  $$\Psi_H\left(\Phi^-\right) = \bigcap_{\beta<\alpha} \Psi_H\left(\Phi^-_\beta\right)$$

  Donc pour tout $\beta < \alpha$, on a $\mathtt{u} \in \Psi_H(\Phi^-_\beta)$. Or, comme $\mathtt{t} \in \Psi_F(\Phi^-_\beta)$, on en déduit que $(\mathtt{t}\ \mathtt{u}) \in \Psi_G(\Phi^-_\beta)$ pour tout $\beta < \alpha$. En appliquant l'hypothèse d'induction, on en déduit $(\mathtt{t}\ \mathtt{u}) \in \Psi_G(\Phi^-)$.

- $F = \forall Y\, G$ ou $F = \forall x\, G$, X étant visiblement positif dans G. L'hypothèse d'induction donne :

  $$\Psi_G\left(\Phi^-\right) \supseteq \bigcap_{\beta<\alpha} \Psi_G\left(\Phi^-_\beta\right)$$

  Or, l'interprétation de F étant définie comme une intersection, celle-ci permute avec l'intersection ci-dessus et donne le résultat voulu.

■

3.4 Lemme de conservation.

On va montrer maintenant le lemme fondamental à propos de l'interprétation :

Lemme 3.14. *(*lemme de conservation*) Pour toute formule F, tout terme $\mathtt{t}$ et toute $\mathcal{C}$-interprétation σ*

$$\textit{si } \vdash \mathtt{t} : F \textit{ alors } \mathtt{t} \in |F|^{\sigma}$$

Démonstration : Afin de démontrer ce lemme, on prouve par induction sur la construction de la preuve le résultat plus général suivant :

$$\text{si } \mathtt{x_1} : A_1, \ldots, \mathtt{x_n} : A_n \vdash \mathtt{t} : F \text{ et } \mathtt{u_1} \in |A_1|^{\sigma}, \ldots, \mathtt{u_n} \in |A_n|^{\sigma} \text{ alors } \mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F|^{\sigma}$$

l'induction se fait en regardant la dernière règle appliquée. Dans toute la preuve, on notera Γ le contexte $\mathtt{x_1} : A_1, \ldots, \mathtt{x_n} : A_n$.

- Si la dernière règle appliquée est l'axiome (Ax) :

  $$\frac{}{\Gamma \vdash \mathtt{x_i} : A_i}$$

  Le résultat est immédiat car $\mathtt{x_i}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] = \mathtt{u_i} \in |A_i|^{\sigma}$.

- Si la dernière règle appliquée est la règle équationnelle (Eq) :

  $$\frac{\Gamma \vdash \mathtt{t} : F[x \Leftarrow u] \qquad \mathcal{E} \vdash u = v}{\Gamma \vdash \mathtt{t} : F[x \Leftarrow v]}$$

  Par hypothèse, si $\mathcal{E} \vdash u = v$, alors $|u|^{\sigma} = |v|^{\sigma}$. Donc $|F[x \Leftarrow u]|^{\sigma} = |F|^{\sigma[x \Leftarrow |u|^{\sigma}]} = |F|^{\sigma[x \Leftarrow |v|^{\sigma}]} = |F[x \Leftarrow v]|^{\sigma}$.

- Si la dernière règle appliquée est l'introduction d'une implication $(\rightarrow_i)$:

  $$\frac{\mathtt{y} : F, \Gamma \vdash \mathtt{t} : G}{\Gamma \vdash \lambda \mathtt{y}\, \mathtt{t} : F \rightarrow G}$$

  Alors, par hypothèse d'induction, si l'on choisit $\mathtt{u_1} \in |A_1|^{\sigma}, \ldots, \mathtt{u_n} \in |A_n|^{\sigma}$, pour tout $\mathtt{v} \in |F|^{\sigma}$, on a $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}][\mathtt{y} \Leftarrow \mathtt{v}] \in |G|^{\sigma}$. Donc, par définition de l'interprétation, on trouve $(\lambda \mathtt{y}\, \mathtt{t})[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F \rightarrow G|^{\sigma}$, d'où le résultat.

- Si la dernière règle appliquée est l'élimination d'une implication $(\rightarrow_e)$:

  $$\frac{\Gamma \vdash \mathtt{t} : F \rightarrow G \qquad \Gamma \vdash \mathtt{v} : F}{\Gamma \vdash (\mathtt{t}\ \mathtt{v}) : G}$$

  Alors, par hypothèse d'induction, si l'on choisit $\mathtt{u_1} \in |A_1|^{\sigma}, \ldots, \mathtt{u_n} \in |A_n|^{\sigma}$, on a $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F \rightarrow G|^{\sigma}$ et $\mathtt{v}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F|^{\sigma}$. D'où le résultat, car par définition de l'interprétation, $(\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}]\ \mathtt{v}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}]) = (\mathtt{t}\ \mathtt{v})[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |G|^{\sigma}$.

- Si la dernière règle appliquée est l'introduction d'un quantificateur (du premier ou du second ordre) $(\forall_i^1$ ou $\forall_i^2)$:

  $$\frac{\Gamma \vdash \mathtt{t} : F \text{ et } \chi \text{ non libre dans } \Gamma}{\Gamma \vdash \mathtt{t} : \forall \chi F}$$

χ n'étant pas libre dans Γ, on a pour toute interprétation σ et pour toute interprétation Φ possible pour χ, $|A_i|^{\sigma[\chi \Leftarrow \Phi]} = |A_i|^\sigma$. Donc, pour tout $\mathtt{u_1} \in |A_1|^\sigma, \ldots, \mathtt{u_n} \in |A_n|^\sigma$, on a $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F|^{\sigma[\chi \Leftarrow \Phi]}$. D'où $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |\forall \chi F|^\sigma$.

- Si la dernière règle appliquée est l'élimination d'un quantificateur (du premier ou du second ordre) ($\forall_e^1$ ou $\forall_e^2$) :

$$\frac{\Gamma \vdash \mathtt{t} : \forall \chi F}{\Gamma \vdash \mathtt{t} : F[\chi \Leftarrow \Phi]}$$

Alors, par hypothèse d'induction, si l'on choisit $\mathtt{u_1} \in |A_1|^\sigma, \ldots, \mathtt{u_n} \in |A_n|^\sigma$, on a $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |\forall \chi F|^\sigma$. Donc, en appliquant la définition de l'interprétation et le lemme 3.6 ou 3.7, on obtient $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F|^{\sigma[\chi \Leftarrow |\Phi|^\sigma]} = |F[\chi \Leftarrow \Phi]|^\sigma$, d'où le résultat.

- Si la dernière règle est le développement ou la factorisation d'un point fixe (aussi bien avec μ qu'avec ν) (μ_d, ν_d, μ_f ou ν_f), le résultat vient de l'égalité suivante :

$$\left| H\left\langle Y \Leftarrow \mu X \lambda \overline{x}\, F\langle \overline{t} \rangle \right\rangle \right|^{\sigma'} = \left| H\left\langle Y \Leftarrow F\left[\overline{x} \Leftarrow \overline{t}\right]\left[X \Leftarrow \lambda \overline{y}\, \mu X \lambda \overline{x}\, F\langle \overline{y} \rangle\right] \right\rangle \right|^{\sigma'}$$

Ce résultat étant valable aussi bien avec ν. En effet, pour toute interprétation σ', si on note :

$$\begin{array}{rcl} \Psi_F & = & |\lambda X \lambda \overline{x}\, F|^{\sigma'} \\ \Pi & = & |\lambda \overline{y}\, \mu X \lambda \overline{x}\, F\langle \overline{y} \rangle|^{\sigma'} \end{array}$$

on a :

$$\begin{array}{rcl} |\mu X \lambda \overline{x}\, F\langle \overline{t} \rangle|^{\sigma'} & = & \Pi(|\overline{t}|^{\sigma'}) \\ |F[\overline{x} \Leftarrow \overline{t}][X \Leftarrow \lambda \overline{y}\, \mu X \lambda \overline{x}\, F\langle \overline{y} \rangle]|^{\sigma'} & = & \Psi_F(\Pi)(|\overline{t}|^{\sigma'}) \end{array}$$

Or, d'après le corollaire 3.11, Π est le plus petit point fixe de Ψ_F (le plus grand dans le cas de ν). D'où $\Psi_F(\Pi) = \Pi$.

On en déduit alors que la prémisse et la conclusion de la règle ont même interprétation (ce qui donne le résultat cherché), en utilisant le lemme de substitution 3.8.

- Si la dernière règle appliquée est celle du plus petit point fixe (μ_i) (resp du plus grand point fixe (ν_i)) :

$$\frac{\Gamma \vdash \mathtt{t} : H \to H[X \Leftarrow \lambda \overline{x}\, F]}{\Gamma \vdash !\mathtt{t} : H\left[X \Leftarrow \lambda \overline{y}\, \mu X \lambda \overline{x}\, F\langle \overline{y} \rangle\right]}\star$$

$\star$: la règle μ_i (resp. ν_i) n'étant applicable que si X n'est pas libre dans Γ, n'a pas d'occurrence négative dans F et est visiblement négative (resp. visiblement positive) dans H.

On se donne une fois pour toutes $\mathtt{u}_1 \in |A_1|^\sigma, \ldots, \mathtt{u}_n \in |A_n|^\sigma$ et on note :

$$\begin{array}{rcl} \mathtt{t}' & = & \mathtt{t}[\overline{\mathtt{u}} \Leftarrow \overline{\mathtt{a}}] \\ \Psi_H & = & |\lambda X\, H|^\sigma \\ \Psi_F & = & |\lambda X \lambda \overline{x}\, F|^\sigma \\ \Pi & = & |\lambda \overline{y}\, \mu X \lambda \overline{x}\, F\langle \overline{y} \rangle|^\sigma \end{array}$$

Par hypothèse d'induction et X n'étant pas libre dans le contexte, on a pour tout $\Phi \in \mathcal{C}_n$, $\mathtt{t}' \in |H \to H[X \Leftarrow \lambda \overline{x}\, F]|^{\sigma[X \Leftarrow \Phi]}$. C'est-à-dire que :

$$\text{pour tout } \Phi \in \mathcal{C}_n \text{ et tout } \mathtt{v} \in \Psi_H(\Phi), \text{ on a } (\mathtt{t}'\ \mathtt{v}) \in \Psi_H\left(\Psi_F(\Phi)\right) \quad \text{(i)}$$

De plus, on sait que Π est le plus petit (resp. le plus grand) point fixe de Ψ_G, donc il peut être défini par induction sur les ordinaux en posant :

$$\begin{array}{rcl}\Phi_0(\overline{\mathtt{t}}) & = & \emptyset \text{ (resp } \Lambda) \\ \Phi_\alpha(\overline{\mathtt{t}}) & = & \bigcup_{\beta<\alpha} \Psi_F(\Phi_\beta)(\overline{\mathtt{t}}) \text{ (resp } \bigcap)\end{array}$$

Ψ_F étant croissante, il existe un ordinal α_0 à partir duquel la collection précédente est constante. On a alors :

$$\Pi = \Phi_{\alpha_0}$$

Or, on veut montrer que $!\mathtt{t}' \in |H[X \Leftarrow \lambda\overline{y}\,\mu X \lambda\overline{x}\, F\langle\overline{y}\rangle]|^\sigma$, c'est à dire que $!\mathtt{t}' \in \Psi_H(\Pi)$. Pour cela, on va prouver par induction que pour tout ordinal α, $!\mathtt{t}' \in \Psi_H(\Phi_\alpha)$.

- Cas 0 : on a bien $!\mathtt{t}' \in \Psi_H(\Phi_0)$ car comme X est visiblement négative (resp. positive) dans H on a $\Psi_H(\Phi_0) = \Lambda$ (d'après la proposition 3.12).
- Cas d'induction : on suppose que pour tout ordinal $\beta < \alpha$, on a $!\mathtt{t}' \in \Psi_H(\Phi_\beta)$. On déduit de (i) $(\mathtt{t}'\ !\mathtt{t}') \in \Psi_H(\Psi_F(\Phi_\beta))$. Or $!\mathtt{t}' \triangleright_\beta (\mathtt{t}'\ !\mathtt{t}')$ donc $!\mathtt{t}' \in \Psi_H(\Psi_F(\Phi_\beta))$. De plus, d'après la proposition 3.13, on a $\Psi_H(\Phi_\alpha) = \bigcap_{\beta<\alpha} \Psi_H(\Psi_F(\Phi_\beta))$. Donc, on a bien $!\mathtt{t}' \in \Psi_F(\Phi_\alpha)$.

■

3.5 Interprétation pleine, interprétation classique.

Définition 3.15. On obtient l'interprétation usuelle, que l'on appellera *pleine*, pour :

$$\mathcal{C} = \mathcal{P}(\Lambda)$$

Définition 3.16. De la même manière, on obtient l'interprétation dite *classique* en prenant :

$$\mathcal{C} = \left\{\emptyset, \Lambda\right\}$$

Proposition 3.17. *Il est immédiat que ces deux sous-ensembles de $\mathcal{P}(\Lambda)$ vérifient les conditions de stabilité requises.*

Notation : Dans le cas de l'interprétation classique, on notera :

$$\mathcal{M} \models F \text{ pour } |F|^{\mathcal{M}} = \Lambda$$

On va montrer que cette notion d'interprétation classique correspond exactement aux *modèles pleins* de la logique classique du second ordre utilisant le λ-calcul pour interpréter les termes du premier ordre. Pour cela, on associe à chaque interprétation classique $\mathcal{M}$, un modèle classique plein (les variables du second ordre sont interprétées dans $\mathcal{P}(\Lambda^n)$) que l'on notera $\mathcal{M}^c$ et, que l'on définit par :

- $|t|^{\mathcal{M}^c} = |t|^{\mathcal{M}}$ pour tout terme logique t.

- $|X|^{\mathcal{M}^c} = \{(\mathtt{t_1}, \ldots, \mathtt{t_n}) \in \Lambda^n, |X|^{\mathcal{M}}(\mathtt{t_1}, \ldots, \mathtt{t_n}) = \Lambda\}$ pour toute variable de prédicat X d'arité n [1].

On définit alors la *satisfaction classique*, que l'on note $\mathcal{M}^c \models_c F$, par induction sur la formule F. Afin d'alléger les écritures, on notera $Sat^{x_1 \cdots x_n}_{\mathcal{M}^c}(F)$ l'ensemble des n-uplets de termes $(\mathtt{t_1}, \ldots, \mathtt{t_n})$ vérifiant $\mathcal{M}^c[\overline{x} \Leftarrow \overline{\mathtt{t}}] \models_c F$. La satisfaction classique est alors définie ainsi :

- $\mathcal{M}^c \models_c X(\overline{t})$ ssi $\overline{t} \in |X|^{\mathcal{M}^c}$.
- $\mathcal{M}^c \models_c F \to G$ ssi $\mathcal{M}^c \models_c F$ entraîne $\mathcal{M}^c \models_c G$.
- $\mathcal{M}^c \models_c \forall x F$ ssi pour tout $\mathtt{u} \in \Lambda$, $\mathcal{M}^c[x \Leftarrow \mathtt{u}] \models_c F$
- $\mathcal{M}^c \models_c \forall X F$ ssi pour tout $\Phi \in \mathcal{P}(\Lambda^n)$, $\mathcal{M}^c[X \Leftarrow \Phi] \models_c F$
- $\mathcal{M}^c \models_c \mu X \lambda \overline{x}\, F\langle \overline{t} \rangle$ ssi $|\overline{t}|^{\mathcal{M}^c} \in \bigcap\{\Phi \in \mathcal{P}(\Lambda^n) \setminus Sat^{\overline{x}}_{\mathcal{M}^c[X \Leftarrow \Phi]}(F) \subseteq \Phi\}$.
- $\mathcal{M}^c \models_c \nu X \lambda \overline{x}\, F\langle \overline{t} \rangle$ ssi $|\overline{t}|^{\mathcal{M}^c} \in \bigcup\{\Phi \in \mathcal{P}(\Lambda^n) \setminus \Phi \subseteq Sat^{\overline{x}}_{\mathcal{M}^c[X \Leftarrow \Phi]}(F)\}$.

Bien que cette définition s'en rapproche, on ne peut pas dire qu'elle correspond à un modèle classique usuel, car les connecteurs μ et ν ne sont pas usuels ! Pour vraiment se ramener à une sémantique classique, il suffit de remarquer que l'on peut traduire les points fixes en formules usuelles en utilisant la propriété suivante :

Proposition 3.18. *On a les deux équivalences suivantes :*

- $\mathcal{M}^c \models_c \mu X \lambda \overline{x}\, F\langle \overline{t} \rangle$ *ssi* $\mathcal{M}^c \models_c \forall X(\forall \overline{x}(F \to X\overline{x}) \to X\overline{t})$
- $\mathcal{M}^c \models_c \nu X \lambda \overline{x}\, F\langle \overline{t} \rangle$ *ssi* $\mathcal{M}^c \models_c \forall K(\forall X(\forall \overline{x}(X\overline{x} \to F), X\overline{t} \to K) \to K)$

(Ces deux équivalences seront montrées syntaxiquement dans l'appendice B).

Démonstration :

- Pour le plus petit point fixe, par définition $\mathcal{M}^c \models_c \forall X(\forall \overline{x}(F \to X\overline{x}) \to X\overline{t})$ est équivalent à : pour tout $\Phi \in \mathcal{P}(\Lambda^n)$, $\mathcal{M}^c[X \Leftarrow \Phi] \models_c \forall \overline{x}(F \to X\overline{x})$ entraîne $|\overline{t}|^{\mathcal{M}^c} \in \Phi$. Ceci s'écrit aussi $|\overline{t}|^{\mathcal{M}^c} \in \bigcap\{\Phi \in \mathcal{P}(\Lambda^n) \setminus \mathcal{M}^c[X \Leftarrow \Phi] \models_c \forall \overline{x}(F \to X\overline{x})\}$.

  Or, la condition $\mathcal{M}^c[X \Leftarrow \Phi] \models_c \forall \overline{x}(F \to X\overline{x})$ est par définition équivalente à : pour tout $\overline{\mathtt{v}} \in \Lambda^n$, $\mathcal{M}^c[X \Leftarrow \Phi][\overline{x} \Leftarrow \overline{\mathtt{v}}] \models_c F$ entraîne $\overline{\mathtt{v}} \in \Phi$. Cette condition peut donc s'écrire $Sat^{\overline{x}}_{\mathcal{M}^c[X \Leftarrow \Phi]}(F) \subseteq \Phi$

  Au total, on a obtenu : $\mathcal{M}^c \models_c \forall X(\forall \overline{x}(F \to X\overline{x}) \to X\overline{t})$ si et seulement si $|\overline{t}|^{\mathcal{M}^c} \in \bigcap\{\Phi \in \mathcal{P}(\Lambda^n) \setminus Sat^{\overline{x}}_{\mathcal{M}^c[X \Leftarrow \Phi]}(F) \subseteq \Phi\}$.Ce qui est exactement la définition de $\mathcal{M}^c \models_c \mu X \lambda \overline{x}\, F\langle \overline{t} \rangle$.

- Pour le grand point fixe, par définition $\mathcal{M}^c \models_c \forall K(\forall X(\forall \overline{x}(X\overline{x} \to F), X\overline{t} \to K) \to K)$ si et seulement si $\mathcal{M}^c[K \Leftarrow \bot] \not\models_c \forall X(\forall \overline{x}(X\overline{x} \to F), X\overline{t} \to K)$. Ce qui est équivalent

[1] Dans le cas des variables de prédicats d'arité nulle, cette définition convient encore. Il suffit de remarquer que l'ensemble des 0-uples est un singleton. Alors, si l'on note l'unique 0-uple "()", l'interprétation d'une variable d'arité nulle est soit $\emptyset$, soit $\{()\}$. Et l'on obtient $\mathcal{M}^c \models_c X$ si et seulement si $() \in |X|^{\mathcal{M}^c}$. Donc si l'on note $\bot = \emptyset$ et $\top = \{()\}$, l'interprétation d'une variable de prédicat d'arité nulle est soit $\top$, soit $\bot$, et l'on a $\mathcal{M}^c \models_c X$ si et seulement si $|X|^{\mathcal{M}^c} = \top$.

à il existe $\Phi \in \mathcal{P}(\Lambda^n)$ tel que $\mathcal{M}^c[X \Leftarrow \Phi] \models_c \forall \overline{x}(X\overline{x} \to F)$ et $|\overline{t}|^{\mathcal{M}^c} \in \Phi$, et peut s'écrire $|\overline{t}|^{\mathcal{M}^c} \in \bigcup\{\Phi \in \mathcal{P}(\Lambda^n) \setminus \mathcal{M}^c[X \Leftarrow \Phi] \models_c \forall \overline{x}(X\overline{x} \to F)\}$.

Comme précédemment, on obtient le résultat voulu car la condition $\mathcal{M}^c[X \Leftarrow \Phi] \models_c \forall \overline{x}(X\overline{x} \to F)$ est équivalente à $\Phi \subseteq Sat^{\overline{x}}_{\mathcal{M}^c[X \Leftarrow \Phi]}(F)$.

■

Définition 3.19. Définissons par induction la traduction suivante sur les formules, qui envoie les formules de notre système dans des formules de la logique du second ordre usuelle (sans μ et ν) :

- $X(\overline{t})^\circ = X(\overline{t})$
- $(F \to G)^\circ = F^\circ \to G^\circ$
- $(\forall x F)^\circ = \forall x F^\circ$
- $(\forall X F)^\circ = \forall X F^\circ$
- $(\mu X \lambda \overline{x}\, F\langle \overline{t} \rangle)^\circ = \forall X(\forall \overline{x}(F^\circ \to X\overline{x}) \to X\overline{t})$
- $(\nu X \lambda \overline{x}\, F\langle \overline{t} \rangle)^\circ = \forall K(\forall X(\forall \overline{x}(X\overline{x} \to F^\circ), X\overline{t} \to K) \to K)$

En utilisant cette traduction, on montre la proposition suivante, qui relie vraiment l'interprétation classique et les modèles plein :

Proposition 3.20. *Pour toute interprétation classique $\mathcal{M}$ et toute formule F, on a :*

$$\mathcal{M} \models F \text{ ssi } \mathcal{M}^c \models_c F^\circ$$

Démonstration : On vérifie d'abord $\mathcal{M}^c \models_c F$ ssi $\mathcal{M}^c \models_c F^\circ$ par induction sur la formule F. En effet, les seuls cas non triviaux sont les cas des points fixes, qui découlent de la proposition 3.18.

Reste à prouver $\mathcal{M} \models F$ ssi $\mathcal{M}^c \models_c F$, par induction sur la construction de la formule F. On utilisera la cohérence de l'interprétation classique ($|F|^{\mathcal{M}} \in \{\emptyset, \Lambda\}$) à travers l'équivalence suivante : pour tout terme $\mathtt{u}$, $\mathtt{u} \in |F|^\sigma$ si et seulement si $|F|^\sigma = \Lambda$.

De plus, on définit une bijection canonique $\Phi \mapsto \Phi^c$ de $\Lambda^n \to \{\emptyset, \Lambda\}$ dans $\mathcal{P}(\Lambda^n)$ par :

$$\Phi^c = \left\{(\mathtt{t_1}, \ldots, \mathtt{t_n}) \in \Lambda^n \text{ tq } \Phi\,(\mathtt{t_1}, \ldots, \mathtt{t_n}) = \Lambda\right\}$$

- $\mathcal{M} \models X(\overline{t})$ est équivalent à $|X|^{\mathcal{M}}(\overline{t}) = \Lambda$ qui par définition de $\mathcal{M}^c$, est équivalent à $\overline{t} \in |X|^{\mathcal{M}^c}(\overline{t})$. D'où le résultat.
- $\mathcal{M} \models F \to G$ ssi pour tout $\mathtt{u} \in \Lambda$, pour tout $\mathtt{v} \in |F|^{\mathcal{M}}$, on a $(\mathtt{u}\ \mathtt{v}) \in |G|^{\mathcal{M}}$. Or, l'interprétation classique étant cohérente, ceci est équivalent à $|F|^{\mathcal{M}} = \emptyset$ ou $|G|^{\mathcal{M}} = \Lambda$. Ce qui par hypothèse d'induction est équivalent à $\mathcal{M}^c \models_c F$ entraîne $\mathcal{M}^c \models_c G$.
- $\mathcal{M} \models \forall x F$ si et seulement si pour tout $\mathtt{v} \in \Lambda$, $|F|^{\mathcal{M}[x \Leftarrow \mathtt{v}]} = \Lambda$. On a alors le résultat voulu par hypothèse d'induction.
- $\mathcal{M} \models \forall X F$ si et seulement si pour tout $\Phi \in \Lambda^n \to \{\emptyset, \Lambda\}$, $|F|^{\mathcal{M}[X \Leftarrow \Phi]} = \Lambda$. Ce qui par hypothèse d'induction est équivalent à pour tout $\Phi \in \Lambda^n \to \{\emptyset, \Lambda\}$, $\mathcal{M}^c[X \Leftarrow \Phi^c] \models F$. On obtient le résultat voulu car l'application $\Phi \mapsto \Phi^c$ est une bijection.

- $\mathcal{M} \models \mu X \lambda \overline{x}\, F\langle \overline{t} \rangle$ (resp. ν) si et seulement si :

$$\bigcap \left\{ \Phi \left(\left| \overline{t} \right|^{\mathcal{M}} \right) \setminus \Phi \in \Lambda^n \to \mathcal{P}(\Lambda), |\lambda X \lambda \overline{x}\, F|^{\sigma}(\Phi) \leq \Phi \right\} = \Lambda$$

(resp. $\bigcup$ et $\geq$). Or, ceci est équivalent à :

$$\left| \overline{t} \right|^{\mathcal{M}} \in \bigcap \left\{ \Phi^c \setminus \Phi \in \Lambda^n \to \mathcal{P}(\Lambda), |\lambda X \lambda \overline{x}\, F|^{\sigma}(\Phi) \leq \Phi \right\}$$

(resp. $\bigcup$ et $\geq$). La condition $|\lambda X \lambda \overline{x}\, F|^{\sigma}(\Phi) \leq \Phi$ (resp. $\geq$) est équivalente, par hypothèse d'induction, à $Sat^{\overline{x}}_{\mathcal{M}^c[X \Leftarrow \Phi]}(F) \subseteq \Phi^c$ (resp. $\supseteq$). D'où le résultat.

■

Chapitre 4

Théorème d'ω-résolubilité.

On va maintenant s'intéresser aux propriétés des termes typés relatives à la normalisation. On ne peut pas espérer avoir de propriétés générales de normalisation forte, ni même de normalisation faible, car l'on type des termes de la forme $!\mathtt{t}$ qui ne sont pas normalisables.

On peut par contre s'intéresser à la résolubilité des termes typés, et même à la ω-résolubilité (pas de $\perp$ dans l'arbre de Böhm du terme, cf appendice A).

Cette propriété est indépendante de la partie premier ordre du système. On va donc considérer le système propositionnel sous-jacent (qui est un sous-système du système complet). De plus, le lemme de réduction (cf chapitre 1), aussi bien que le lemme de conservation (cf chapitre 2) restent valables dans le système propositionnel. On remarque aussi que si un terme est typable dans le système complet, alors il est typable dans le système propositionnel. Pour cela, il suffit de remplacer, dans la preuve, toutes les formules atomiques $X(\overline{t})$ par une variable propositionnel $X^\star$ (l'application $X \mapsto X^\star$ étant une bijection de l'ensemble de toutes les variables de prédicats dans l'ensemble des variables de prédicat d'arité nulle).

4.1 Une autre notion d'interprétation !

On va construire une autre notion d'interprétation, et par un choix adéquat de $\mathcal{C}$, on l'utilisera pour montrer sous quelles conditions les termes typés sont résolubles ou ω-résolubles.

Définition 4.1. On définit les deux ensembles de termes suivants:

- $\mathcal{R} = \{\mathtt{t} \in \Lambda \setminus \ \mathtt{t} \text{ résoluble } \}$
- $\mathcal{R}_0 = \{\mathtt{t} \setminus \mathtt{t} \sim_\beta (\mathtt{x}\ \mathtt{t_1} \ldots \mathtt{t_n}),\ \text{avec pour tout } i,\ \mathtt{t_i} \in \Lambda\}$

Lemme 4.2. *On a alors les inclusions suivantes :*

$$\mathcal{R}_0 \subset (\Lambda \Rightarrow \mathcal{R}_0) \subset (\mathcal{R} \Rightarrow \mathcal{R}_0) \subset (\mathcal{R}_0 \Rightarrow \mathcal{R}) \subset \mathcal{R}$$

Démonstration : En effet, si $\mathtt{t} \in \mathcal{R}_0$, alors $\mathtt{t} \sim_\beta (\mathtt{x}\ \mathtt{t_1} \ldots \mathtt{t_n})$. Ainsi pour tout terme $\mathtt{u}$ (résoluble ou non), on a $(\mathtt{t}\ \mathtt{u}) \sim_\beta (\mathtt{x}\ \mathtt{t_1} \ldots \mathtt{t_n}\ \mathtt{u}) \in \mathcal{R}_0$, donc $\mathtt{t} \in \Lambda \Rightarrow \mathcal{R}_0$. D'où la première inclusion.

Les deux inclusions suivantes sont immédiates, elles découlent de $\mathcal{R}_0 \subset \mathcal{R} \subset \Lambda$

Pour la dernière, prenons $\mathtt{t} \in \mathcal{R}_0 \Rightarrow \mathcal{R}$, et prenons $\mathtt{u} \in \mathcal{R}_0$. $(\mathtt{t}\ \mathtt{u})$ est résoluble, donc $\mathtt{t}$ aussi, d'où le résultat.

■

Définition 4.3. On définit alors les ensembles suivants :

- $C' = \{\Phi \in \mathcal{P}(\Lambda); \mathcal{R}_0 \subseteq \Phi \subseteq \mathcal{R}\}$
- $C = C' \cup \{\Lambda, \emptyset\}$

Définition 4.4. On appellera *$\mathcal{R}$-interprétation* l' interprétation définie en prenant l'ensemble $\mathcal{C}$, défini ci-dessus, pour limiter l'interprétation de la quantification du second ordre.

Proposition 4.5. *La $\mathcal{R}$-interprétation vérifie les conditions requises par la définition 3.1.*

Démonstration : $\mathcal{C}$ vérifie bien la stabilité pour l'intersection et la réunion, et contient bien $\emptyset$ et Λ. Par construction, tout élément de $\mathcal{C}$ est clos pour la β-équivalence. Pour démontrer la stabilité de l'implication, considérons deux éléments A et B de $\mathcal{C}$ et distinguons alors les cas suivants :

- Si $A = \emptyset$, on a $A \Rightarrow B = \Lambda$, qui est adéquat.
- Si $A \neq \emptyset$ et $B = \emptyset$, on a $A \Rightarrow B = \emptyset$, qui est adéquat.
- Si $A \neq \emptyset$ et $B = \Lambda$, on a $A \Rightarrow B = \Lambda$, qui est adéquat.
- Si $A \neq \emptyset$ et $\mathcal{R}_0 \subseteq B \subseteq \mathcal{R}$, on a les inclusions suivantes (d'après le lemme 4.2) :

$$\mathcal{R}_0 \subseteq \Lambda \Rightarrow \mathcal{R}_0 \subseteq A \Rightarrow B \subseteq \mathcal{R}_0 \Rightarrow \mathcal{R} \subseteq \mathcal{R}.$$

■

4.2 Lemme du quotient.

***Proposition 4.6* (lemme du quotient).** *Si $\mathcal{C}$ fournit une notion cohérente d'interprétation, si $\sim_{\mathcal{C}}$ est une relation d'équivalence sur $\mathcal{C}$ qui est compatible avec l'intersection, la réunion et l'implication, alors la notion de $\mathcal{C}$-interprétation est compatible avec $\sim_{\mathcal{C}}$.*

Pour être plus précis, les hypothèses du lemme du quotient affirment que l'on a une équivalence $\sim_{\mathcal{C}}$ vérifiant les conditions suivantes :

- Pour toutes familles $\{A_j\}_{j\in J}$ et $\{B_j\}_{j\in J}$ d'éléments de $\mathcal{C}$, si pour tout $j \in J$ on a $A_j \sim_{\mathcal{C}} B_j$, alors on a :

$$\bigcap_{j\in J} A_j \sim_{\mathcal{C}} \bigcap_{j\in J} B_j \text{ et } \bigcup_{j\in J} A_j \sim_{\mathcal{C}} \bigcup_{j\in J} B_j$$

- Pour toutes parties A_1, A_2, B_1 et B_2 de $\mathcal{C}$, si $A_1 \sim_{\mathcal{C}} B_1$ et $A_2 \sim_{\mathcal{C}} B_2$, alors on a :

$$A_1 \to A_2 \sim_{\mathcal{C}} B_1 \to B_2$$

La conclusion du lemme est que sous ces conditions, si l'on se donne deux $\mathcal{C}$-interprétations σ et σ' vérifiant pour toute variable X, $|X|^{\sigma} \sim_{\mathcal{C}} |X|^{\sigma'}$, alors pour toute formule F, on a :

$$|F|^{\sigma} \sim_{\mathcal{C}} |F|^{\sigma'}$$

Démonstration : On démontre ce résultat par induction sur la construction de la formule F :

- Si $F = X$, le résultat est immédiat, car par hypothèse $|X|^{\sigma} \sim_{\mathcal{C}} |X|^{\sigma'}$.
- Si $F = G \to H$, on déduit de l'hypothèse d'induction que $|G|^{\sigma} \sim_{\mathcal{C}} |G|^{\sigma'}$ et $|H|^{\sigma} \sim_{\mathcal{C}} |H|^{\sigma'}$. Ainsi, la compatibilité de l'implication donne bien $|F|^{\sigma} \sim_{\mathcal{C}} |F|^{\sigma'}$.
- Si $F = \forall X G$, l'hypothèse d'induction donne pour tout élément $\Phi \in \mathcal{C}$, $|G|^{\sigma[X \Leftarrow \Phi]} \sim_{\mathcal{C}} |G|^{\sigma'[X \Leftarrow \Phi]}$. Ainsi, La compatibilité de l'intersection donne bien $|F|^{\sigma} \sim_{\mathcal{C}} |F|^{\sigma'}$.
- Si $F = \mu Y G$ (resp. ν), ce point fixe peut être construit par induction sur les ordinaux, en posant :

  $$\Phi_0 = \emptyset \,(\text{resp.}\Lambda) \;\text{ et }\; \Phi_\alpha = \cup_{\beta<\alpha} |G|^{\sigma\left[Y \Leftarrow \Phi_\beta\right]} \,(\text{resp.}\cap)\,.$$

  et

  $$\Phi'_0 = \emptyset \,(\text{resp.}\Lambda) \;\text{ et }\; \Phi'_\alpha = \cup_{\beta<\alpha} |G|^{\sigma'\left[Y \Leftarrow \Phi'_\beta\right]} \,(\text{resp.}\cap)\,.$$

  Il suffit alors de montrer que pour tout ordinal α, on a $\Phi_\alpha \sim_{\mathcal{C}} \Phi'_\alpha$. Or, on a bien $\Phi_0 \sim_{\mathcal{C}} \Phi'_0$ et si l'on a $\Phi_\beta \sim_{\mathcal{C}} \Phi'_\beta$ pour tout ordinal inférieur à α, on déduit de l'hypothèse d'induction que $|G|^{\sigma[Y \Leftarrow \Phi_\beta]} \sim_{\mathcal{C}} |G|^{\sigma'[Y \Leftarrow \Phi'_\beta]}$, on déduit alors de la compatibilité de la réunion (resp. de l'intersection) que $\Phi_\alpha \sim_{\mathcal{C}} \Phi'_\alpha$.

■

Dans ces conditions, on peut parler d'une interprétation *quotient* associant à chaque variable une classe d'équivalence. On peut appliquer ce résultat à la $\mathcal{R}$-interprétation définie à la section précédente. En effet, on peut montrer la proposition suivante :

Proposition 4.7. *Les trois ensembles suivants : $C_{-1} = \{\emptyset\}$, $C_0 = \mathcal{C}'$ et $C_1 = \{\Lambda\}$, constituent une partition compatible avec la notion de $\mathcal{R}$-interprétation.*

Démonstration : La compatibilité de l'intersection (resp. de la réunion) découle du fait que tout élément de $\mathcal{C}'$ contient $\mathcal{R}_0$ (resp. est inclus dans $\mathcal{R}$). Donc une intersection est vide si et seulement si l'un des éléments est vide (resp. une réunion est pleine si et seulement si l'un des élément est Λ).

La compatibilité de l'implication résulte du lemme 4.2. ■

Définition 4.8. Une interprétation quotient Δ peut être définie comme une valuation des variables sur l'ensemble $\{-1, 0, 1\}$, la valuation $|F|^{\Delta}$ d'une formule étant définie par induction sur la construction de la formule F :

- $|G \to H|^{\Delta}$ est calculé en utilisant la table suivante :

		$\lvert H\rvert^{\Delta}$		
		-1	0	1
$\lvert G\rvert^{\Delta}$	-1	1	1	1
	0	-1	0	1
	1	-1	0	1

- $|\forall X\, F|^{\Delta} = \inf\{|F|^{\Delta[X \Leftarrow -1]}, |F|^{\Delta[X \Leftarrow 0]}, |F|^{\Delta[X \Leftarrow 1]}\}$.
- $|\mu X F|^{\Delta} = \inf\{\Phi \in \{-1, 0, 1\} \setminus |F|^{\Delta[X \Leftarrow \Phi]} \leq \Phi\}$
- $|\nu X F|^{\Delta} = \sup\{\Phi \in \{-1, 0, 1\} \setminus \Phi \leq |F|^{\Delta[X \Leftarrow \Phi]}\}$

Notation : On note Δ_σ l'interprétation quotient associée à σ, en posant $|X|^{\Delta_\sigma} = \Phi \in \{-1, 0, 1\}$ si et seulement si $|X|^{\sigma} \in \mathcal{C}_\Phi$.

On a alors la propriété suivante :

Proposition 4.9. *Pour toute formule F et toute $\mathcal{R}$-interprétation σ, on a :*

- $|F|^{\Delta_\sigma} = -1$ *si et seulement si* $|F|^{\sigma} = \emptyset$
- $|F|^{\Delta_\sigma} = 1$ *si et seulement si* $|F|^{\sigma} = \Lambda$
- $|F|^{\Delta_\sigma} = 0$ *si et seulement si* $\mathcal{R}_0 \subseteq |F|^{\sigma} \subseteq \mathcal{R}$

Démonstration : C'est une conséquence directe du lemme 4.6 et de la proposition 4.7. En fait, il suffit de vérifier que nos règles de calcul correspondent bien au calcul sur le quotient, ce qui ne pose aucun problème. ■

Cette sémantique à trois valeurs n'est pas seulement technique, on peut lui donner un "sens". En considérant la partition qui réunirait C_0 et C_1, on obtiendrait l'interprétation classique. Ainsi, dans cette valuation, -1 représente le faux classique, tandis que 0 et 1 sont deux valeurs pour le vrai classique. En fait, ces deux valeurs distinguent respectivement les formules "intelligemment vraies" du type "$X \to X$" et les "négations du faux" comme "$\forall X\, X \to Y$". On remarque aussi que le calcul de Δ_σ est décidable avec une complexité du même type que la décision du calcul propositionnel classique (3^N, où N est le nombre de variables distinctes (libres ou liées), dans le pire des cas).

4.3 Condition de résolubilité.

Afin d'examiner si un terme typé est *résoluble*, on va étudier sous quelles conditions l'interprétation d'une formule est $\emptyset$ ou Λ. Pour cela, il suffit d'utiliser l'interprétation quotient définie précédemment.

Définition 4.10. On définit le *degré propre* d'une formule F notée $\Delta_0(F)$ de la manière suivante : soit Δ_0 l'interprétation quotient vérifiant $|X|^{\Delta_0} = 0$ pour toute variable X, on pose alors :

$$\Delta_0\,(F) = |F|^{\Delta_0}$$

Définition 4.11. On dira qu'un séquent $\mathtt{x_1} : A_1, \ldots, \mathtt{x_n} : A_n \vdash \mathtt{t} : A$ est *strict* si $\Delta_0(A) = 0$ et si $\Delta_0(A_i) \neq -1$ pour $1 \leq i \leq n$.

Théorème 4.12. *Si l'on prouve un séquent strict $\Gamma \vdash \mathtt{t} : A$, alors t est résoluble.*

Démonstration : Considérons σ l'interprétation vérifiant $|X|^{\sigma} = R$ pour toute variable X. Le séquent étant strict, on a pour tout i, $\Delta_0(A_i) \neq -1$. Donc par définition du degré propre et d'après

la proposition 4.9 ci-dessus, on a $|A_i|^\sigma \neq \emptyset$. D'où $\mathtt{x_i} \in R_0 \subseteq |A_i|^\sigma$. Le lemme de conservation donne alors $\mathtt{t} \in |A|^\sigma$. Le séquent étant strict, on déduit de $\Delta_0(A) = 0$ que $R_0 \subseteq |A|^\sigma \subseteq R$, d'où $t \in R$. ■

4.4 Condition d'ω-résolubilité.

Considérons les définitions suivantes :

Définition 4.13. On dira qu'une formule F est *héréditairement stricte* si et seulement si pour toute sous-formule G de F, on a $\Delta_0(G) = 0$.

Lemme 4.14. *Si F et G sont deux formules héréditairement strictes, alors $F[X \Leftarrow G]$ est aussi héréditairement stricte.*

Démonstration : Soit $H = F[X \Leftarrow G]$, alors toute sous-formule H' de H est, ou bien une sous-formule de G et dans ce cas on a $\Delta_0(H') = 0$, ou bien de la forme $H' = F'[X \Leftarrow G]$ où F' est une sous-formule de F. Par définition, $\Delta_0(H') = 0$ si et seulement si $|H'|^{\Delta_0} = 0$ (Δ_0 étant l'interprétation quotient définie par $|X|^{\Delta_0} = 0$ pour toute variable X). Or, $|H'|^{\Delta_0} = |F'|^{\Delta_0[X \Leftarrow |G|^{\Delta_0}]}$. G étant héréditairement stricte, on a $|G|^{\Delta_0} = 0$. D'où $|F'|^{\Delta_0[X \Leftarrow |G|^{\Delta_0}]} = |F'|^{\Delta_0}$. Or, F' étant une sous-formule de F, on a $|F'|^{\Delta_0} = 0$. Il en résulte $\Delta_0(H') = 0$. ■

Théorème 4.15. *Si l'on prouve $\Gamma \vdash \mathtt{t} : F$, et si toutes les formules apparaissant dans la preuve sont héréditairement strictes, alors t est ω-résoluble (cf appendice A).*

Démonstration : Afin de prouver que $\mathtt{t}$ est ω-résoluble, il suffit de démontrer que $\mathtt{t} \in \Omega_n$ pour tout entier n (cf appendice A). Montrons cela par récurrence.

Notons $P(n)$ la propriété suivante : il existe une dérivation de $\Gamma \vdash \mathtt{t} : F$ dont toutes les formules sont héréditairement strictes, implique $\mathtt{t} \in \Omega_n$. On a bien $P(0)$ car $\Omega_0 = \Lambda$. Montrons que $P(n)$ implique $P(n+1)$. On considère une dérivation de $\Gamma \vdash \mathtt{t} : F$ dont toutes les formules sont héréditairement stricte.

On déduit de cette hypothèse que le séquent conclusion de la preuve est strict. Donc $\mathtt{t}$ est résoluble (cf 4.12). $\mathtt{t}$ se réduit à $\mathtt{t}' = \lambda \mathtt{x_1} \ldots \lambda \mathtt{x_p}(\mathtt{x}\ \mathtt{t_1} \ldots \mathtt{t_n})$. Par le théorème de réduction, on prouve $\Gamma \vdash \mathtt{t}' : F$. De plus, en utilisant la remarque de la fin du chapitre 1, on trouve que dans cette preuve toutes les formules sont des substitutions de formules apparaissant dans la preuve originale. Toutes les formules apparaissant dans cette preuve sont donc héréditairement strictes. De surcroît, au cours de cette preuve, on a nécessairement typé les termes $\mathtt{t_1}, \ldots, \mathtt{t_n}$. On peut donc appliquer l'hypothèse d'induction, et on trouve que $\mathtt{t_i} \in \Omega_n$ pour $1 \leq i \leq n$. Ainsi on a bien $\mathtt{t} \in \Omega_{n+1}$, par définition de Ω_{n+1}. ■

4.5 Cas du plus grand point fixe seul.

Dans cette section, on va considérer le système sans le plus petit point fixe (μ). On restreint donc l'ensemble des formules, ainsi que les règles du système. Tous les résultats du chapitre 2 restent valables. Pour le chapitre 3, il en est de même en simplifiant un peu les conditions requises par

la définition 3.1. En effet, il devient inutile de supposer que $\emptyset \in \mathcal{C}$, car l'ensemble vide n'était utilisé que pour la construction par induction du plus petit point fixe. On va constater que cette petite différence a de grandes conséquences sur les conditions de résolubilité et d'ω-résolubilité.

On peut définir une notion d'interprétation cohérente similaire à celle du début de ce chapitre en prenant :

- $\mathcal{C}' = \{\Phi; \mathcal{R}_0 \subseteq \Phi \subseteq \mathcal{R}\}$
- $\mathcal{C} = \mathcal{C}' \cup \{\Lambda\}$

On peut alors démontrer la proposition suivante :

Proposition 4.16. *Pour toute formule F et toute interprétation σ, on a $|F|^\sigma = \Lambda$ si et seulement si l'une des deux conditions suivantes est vérifiée :*

1. *la variable X la plus à droite dans F est libre et $|X|^\sigma = \Lambda$*
2. *F peut s'écrire $F'[X \Leftarrow \nu Y G]$, X et Y étant respectivement les variables les plus à droite dans F' et G.*

Démonstration : On démontre ce résultat par induction sur la construction de la formule F :

- Si $F = X$, on a $|F|^\sigma = \Lambda$ si et seulement si $|X|^\sigma = \Lambda$. D'où le résultat.
- Si $F = G \to H$, on a $|F|^\sigma = \Lambda$ si et seulement si $|H|^\sigma = \Lambda$ (en effet, si $|H|^\sigma \neq \Lambda$, alors on a $R_0 \subseteq |H|^\sigma \subseteq R$ et $|G|^\sigma \neq \emptyset$, d'où $|F|^\sigma \subseteq R$). On peut donc appliquer l'hypothèse d'induction à H, ce qui donne le résultat car la variable la plus à droite dans F est la variable la plus à droite dans H.
- Si $F = \forall X G$, on a $|F|^\sigma = \Lambda$ si et seulement si pour tout $\Phi \in \mathcal{C}$, $|G|^{\sigma[X \Leftarrow \Phi]} = \Lambda$. On peut alors appliquer l'hypothèse d'induction pour trouver le résultat cherché.
- Si $F = \nu X G$, on a $|F|^\sigma = \Lambda$ si et seulement si $|G|^{\sigma[X \Leftarrow \Lambda]} = \Lambda$. On peut distinguer deux cas : si X est la variable la plus à droite dans G, alors F vérifie la deuxième condition; sinon l'hypothèse d'induction appliquée à G donne le résultat voulu.

■

Corollaire 4.17. *Si l'on prouve $\Gamma \vdash \mathtt{t} : F$, et si F ne vérifie pas la condition 2 de la proposition 4.16, alors $\mathtt{t}$ est résoluble.*

Démonstration : Considérons l'interprétation σ définie par $|X|^\sigma = R$ pour toute variable X. Par hypothèse, F ne peut vérifier aucune des conditions de la proposition 4.16. On trouve donc $|F|^\sigma \subseteq R$. Comme pour le théorème 4.12, on trouve par le lemme de conservation $\mathtt{t} \in |F|^\sigma$. D'où $\mathtt{t} \in R$. ■

Théorème 4.18. *Si l'on prouve $\Gamma \vdash \mathtt{t} : F$, aucune formule dans la preuve ne contenant une sous-formule de la forme $\nu X G$ où X est la variable la plus à droite dans G, alors $\mathtt{t}$ est ω-résoluble.*

Démonstration : La preuve est analogue à celle du théorème 4.15 en remplaçant la propriété d'être héréditairement stricte par celle de n'avoir aucune sous-formule de la forme $\nu X G$

où X est la variable la plus à droite dans G, et en utilisant le corollaire précédent à la place du théorème 4.12. ■

La condition interdisant les formules de la forme $\nu X G$ où X est la variable la plus à droite dans G est nécessaire. En effet, on peut extraire un terme non-résoluble de la preuve suivante :

$$\dfrac{\dfrac{\dfrac{}{\mathtt{x} : X \vdash \mathtt{x} : X}}{\vdash \lambda \mathtt{x}\, \mathtt{x} : X \to X}\,{\scriptstyle \to_i}}{\vdash\, !\lambda \mathtt{x}\, \mathtt{x} : \nu X X}\,{\scriptstyle \nu_i}$$

On remarque une nette différence entre le plus petit et le plus grand point fixe du point de vue de la résolubilité des termes typés. Pour le plus grand seul, il suffit de vérifier une condition syntaxique extrêmement simple, alors que pour le plus petit, la condition est équivalente en complexité à la décision du calcul propositionnel classique. En fait, on aurait les mêmes problèmes avec le plus grand point fixe si l'on avait ajouté un quantificateur existentiel du second ordre "$\exists X F$" à la logique

Voici un exemple de typage d'un terme non-résoluble qui montre que pour le plus petit point fixe une restriction du même type que pour le plus grand ne suffit pas :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash \mathtt{z} : \forall X X}{\Gamma \vdash \mathtt{z} : X}\,{\scriptstyle \forall^2_e} \quad \dfrac{}{\Gamma \vdash \mathtt{r} : X \to C}}{\Gamma \vdash (\mathtt{r}\ \mathtt{z}) : C}\,{\scriptstyle \to_e}}{\mathtt{z} : \forall X X, \mathtt{r} : X \to C \vdash \lambda \mathtt{x}\, (\mathtt{r}\ \mathtt{z}) : F \to C}\,{\scriptstyle \to_i}}{\mathtt{z} : \forall X X \vdash \lambda \mathtt{r} \lambda \mathtt{x}\, (\mathtt{r}\ \mathtt{z}) : (X \to C) \to (F \to C)}\,{\scriptstyle \to_i}}{\mathtt{z} : \forall X X \vdash\, !\lambda \mathtt{r} \lambda \mathtt{x}\, (\mathtt{r}\ \mathtt{z}) : \mu X F \to C}\,{\scriptstyle \mu_i}}{\vdash \lambda \mathtt{z} ! \lambda \mathtt{r} \lambda \mathtt{x}\, (\mathtt{r}\ \mathtt{z}) : \forall X X, \mu X F \to C}\,{\scriptstyle \to_i}$$

Avec $\Gamma = \mathtt{z} : \forall X X, \mathtt{x} : F, \mathtt{r} : X \to C$, X est sans occurrence dans C et n'a pas d'occurrence négative dans F.

Chapitre 5

Types de données.

5.1 Exemples et construction de types de données.

La construction d'un type de données repose sur le choix d'un ensemble de symboles de fonction que l'on appellera les *constructeurs* du type (**0** et **s** pour les entiers, ou **nil** et **cons** pour les listes). Ces constructeurs sont "typés", c'est à dire que pour chaque argument de chaque constructeur, on choisit un type de données déjà défini, ou bien on décide qu'il s'agit d'un paramètre inductif (du type que l'on est en train de définir). Dans le cas des listes d'entiers, le premier argument de **cons** est un entier (type déjà défini) tandis que le second est inductif. On obtient ainsi ce que l'on appelle une algèbre de termes multi-sortes.

Les types de données ainsi construits couvrent la plupart des besoins en programmation. On peut facilement les contruire dans AF_2, en définissant l'ensemble des éléments du type comme le plus petit ensemble clos par ses constructeurs. Jean-Louis Krivine a pu définir une notion de type de données sémantique qui couvre totalement les algèbres de termes multi-sortes.

Cette notion de type de données sémantique est très importante, car c'est elle qui au travers des résultats du chapitre 3, assure la correction des programmes extraits, indépendamment des résultats de normalisation.

Ce que nous apportons de nouveau avec l'utilisation du plus grand et du plus petit point fixe conjugué, c'est une classe de type de données plus large, permettant d'utiliser des données infinies, voir des notions plus complexes telles que les suites binaires infinies n'utilisant qu'un nombre fini de "1". De plus, ces types de données vérifient encore la définition sémantique de type de données (légèrement modifiée), assurant ainsi la correction des programmes.

Nous allons commencer ce chapitre avec des exemples illustrant comment petit, à petit, on peut élargir la notion de type de données. Nous montrerons ensuite comment définir et utiliser la notion de type de données dans AF_2 pour justifier la correction des programmes. Ensuite, nous traiterons le cas des streams, qui met bien en évidence le mécanisme de la preuve. Enfin, nous définirons une très large classe de type de données vérifiant la définition sémantique.

5.1.1 Des types itératifs aux types récursifs.

- Les entiers : On choisit deux constructeurs : **0** d'arité nulle et **s** d'arité 1, l'argument de **s** étant inductif (c'est à dire du type entier, que l'on est en train de définir). La formule qui définit les entiers peut alors être :

$$N[x] = \forall X \left(X\mathbf{0}, \forall y \left(Xy \rightarrow X\mathbf{s}y \right) \rightarrow Xx \right)$$

  Cette formule définit bien les entiers en tant que plus petit ensemble clos par les fonctions **0** et **s**.

- Les listes d'entiers : On se donne deux constructeurs, **nil** d'arité nulle et **cons** d'arité 2, le premier argument de **cons** étant de type entier tandis que le second est inductif. On peut alors définir ce type par la formule suivante :

$$L_N[x] = \forall X \left(X\mathbf{nil}, \forall a \forall y \left(N[a], Xy \rightarrow X\mathbf{cons}\,(a,y) \right) \rightarrow Xx \right)$$

Le problème de ces définitions, c'est qu'elles conduisent aux types *itératifs* tels les *entiers de Church*. Chaque pas de récurrence sur l'un de ces types est alors en temps linéaire, avec entre autre comme conséquence, l'absence de prédécesseur en temps constant ! Une solution, proposée par Michel Parigot [22], est d'utiliser des types *inductifs*. On obtient alors une définition du même ensemble, mais comme le plus petit point fixe de la fonction qui clot un ensemble par les constructeurs du type.

Dans le cas des deux exemples précédents, on obtient :

- Pour les entiers :

$$N[x] = \mu K \lambda z \forall X \left(X\mathbf{0}, \forall y \left(Ky \rightarrow X\mathbf{s}y \right) \rightarrow Xz \right) \langle x \rangle$$

- Pour les listes d'entiers:

$$L_N[x] = \mu K \lambda z \forall X \left(X\mathbf{nil}, \forall a \forall y \left(N[a], Ky \rightarrow X\mathbf{cons}\,(a,y) \right) \rightarrow Xz \right) \langle x \rangle$$

Revenons sur le sens de cette définition dans le cas des listes. Considérons la formule à paramètres suivante :

$$F[K,z] = \forall X \left(X\mathbf{nil}, \forall a \forall y \left(N[a], Ky \rightarrow X\mathbf{cons}\,(a,y) \right) \rightarrow Xz \right)$$

Si l'on fixe un prédicat K d'arité 1, le prédicat "$\lambda z F[K,z]$" est le plus petit prédicat qui contient **nil**, et qui pour tout élément a de type N et pour tout élément y de K contient $\mathbf{cons}(a,y)$.

La formule $L_N[x]$ (qui s'écrit $\mu K \lambda z F[K,z]\langle x \rangle$) peut alors être vue comme le plus petit prédicat K tel que $\forall x (Kx \leftrightarrow F[K,x])$, ou bien encore comme le plus petit point fixe de la fonction qui associe à un prédicat K d'arité 1 le prédicat $\lambda z F[K,z]$.

Il est alors facile de voir que la formule $L_N[x]$ est bien l'ensemble des listes d'entiers. Il suffit en effet d'itérer la fonction précédente à partir de l'ensemble vide pour s'en convaincre.

5.1.2 Vers des données infinies.

Maintenant, que se passe-t-il si l'on utilise le plus grand point fixe ? Une courte réflexion montre que l'on obtient les mêmes listes plus les "listes infinies" (communément appelées *streams*). Pour obtenir réellement les streams, il suffit de supprimer le constructeur **nil**. On obtient alors la formule suivante pour les streams d'entiers :

$$S_N[x] = \nu K \lambda z \forall X \left(\forall a \forall y \left(N[a], Ky \to X \mathbf{cs}\,(a, y) \right) \to Xz \right) \langle x \rangle$$

De la même manière que pour les listes, si l'on considère la formule :

$$F[K, z] = \forall X \left(\forall a \forall y \left(N[a], Ky \to X \mathbf{cs}\,(a, y) \right) \to Xz \right)$$

La formule $S_N[x]$ (qui s'écrit $\nu K \lambda z F[K, z]\langle x \rangle$) peut alors être vue comme le plus grand prédicat K tel que $\forall x (Kx \leftrightarrow F[K, x])$, ou bien encore comme le plus grand point fixe de la fonction qui associe à un prédicat K d'arité 1 le prédicat $\lambda z F[K, z]$. Il s'agit bien là de l'ensemble des streams d'entiers.

Un autre exemple consiste en une représentation des listes infinies de 0 et de 1. On se donne deux constructeurs unaires inductifs : $\mathbf{s}_0$ et $\mathbf{s}_1$, qui ajoutent respectivement un 0 ou un 1 au début d'un stream. Par analogie, la formule suivante définit les suites infinies de 0 et de 1 :

$$B[x] = \nu K \lambda z \forall X \left(\forall a \forall y \left(Ky \to X \mathbf{s}_0 y \right), \forall a \forall y \left(Ky \to X \mathbf{s}_1 y \right) \to Xz \right) \langle x \rangle$$

Une question s'impose : sachant que dans cette formule il y à deux paramètres inductifs, que se passe-t-il si on utilise deux points fixes ?

Si on utilise deux plus grands points fixes, on obtient le même type. Si on utilise deux plus petits points fixes, on obtient (comme avec un seul) le type vide (du fait de l'absence d'un constructeur sans paramètre inductif). Mais que se passe-t-il si on alterne un plus petit et un plus grand point fixe ?

En fait, on a quatre possibilités qui sont semblables deux à deux en échangeant le rôle des constructeurs. Deux possibilités réellement différentes sont donc :

- $B'[x] = \mu K' \lambda y \nu K \lambda z F[K, K', z]\langle y \rangle \langle x \rangle$
- $B''[x] = \nu K \lambda y \mu K' \lambda z F[K, K', z]\langle y \rangle \langle x \rangle$

en notant :

$$F[K, K', z] = \forall X \left(\forall y \left(Ky \to X \mathbf{s}_0\,(y) \right), \forall y \left(K'y \to X \mathbf{s}_1\,(y) \right) \to Xz \right)$$

Que définissent ces types ?

Étant donnés deux prédicats K et K', le prédicat "$\lambda z F(K, K', z)$" définit l'ensemble des termes qui s'écrivent $\mathbf{s}_0(a)$, $a \in K$ ou $\mathbf{s}_1(a)$, $a \in K'$.

Si on prend le plus grand point fixe en K, en laissant K' constant, on obtient l'ensemble des termes s'écrivant $\mathbf{s}_0^n(\mathbf{s}_1(a))$, $a \in K'$ et $n \in \mathbb{N}$ ou s'écrivant comme une suite infinie de $\mathbf{s}_0$, terme que l'on notera $!\mathbf{s}_0$.

Puis, si on prend le plus petit point fixe en K' de cet ensemble, on obtient les termes qui s'écrivent comme une suite finie d'application de termes de la forme $\lambda x\, \mathbf{s}_0^n(\mathbf{s}_1(x))$, $n \in \mathbb{N}$ au terme $!\mathbf{s}_0$. Ceci correspond à tous les termes qui ne contiennent pas une infinité de $\mathbf{s}_1$. Ainsi, B' est le type des suites binaires infinies n'utilisant qu'un nombre fini de 1.

Inversement, si on prend le plus petit point fixe en K', en laissant K constant, on obtient l'ensemble des termes s'écrivant $\mathbf{s}_1^n(\mathbf{s}_0(a))$, $a \in K$ et $n \in \mathbb{N}$.

Ensuite, si on prend le plus grand point fixe en K, on obtient les termes qui s'écrivent comme une suite infinie d'application de termes de la forme $\lambda x\, \mathbf{s}_1^n(\mathbf{s}_0(x))$, $n \in \mathbb{N}$. Donc B'' est le types des suites binaires infinies qui utilisent une infinité de 0 (qui ne se terminent pas par une infinité de 1).

Le même genre de démarche sur les arbres conduit aux types des arbres infinis dont tous les chemins vont un nombre fini de fois à droite, ou bien ceux dont les chemins vont un nombre infini de fois à droite, etc.

5.2 Types de données et programmation.

Dans cette section, on va donner une définition sémantique de la notion de type de données et on va montrer que cette définition suffit pour assurer la correction des programmes. On montrera dans les sections suivantes que les formules définies en utilisant la méthode présentée informellement ci-dessus satisfont cette définition.

Afin de formaliser la notion d'*éléments d'un type de données* D[1], on va utiliser l'interprétation classique (cf chapitre 3). Rappelons que l'interprétation classique correspond aux modèles pleins de la logique classique du second ordre qui utilisent le λ-calcul pour interpréter les termes. Du fait de cette analogie, on appellera souvent modèle des interprétations classiques.

Définition 5.1. Si l'on se donne une interprétation classique $\mathcal{M}$ et une formule $D[x]$ à une variable libre définissant un type de donnée, l'ensemble des éléments du type dans le modèle $\mathcal{M}$, que l'on notera $D^{\mathcal{M}}$, sera l'ensemble des termes du λ-calcul $\mathtt{t}$ vérifiant :

$$\mathcal{M}[x \Leftarrow \mathtt{t}] \models D[x]$$

Autrement dit, on a $D^{\mathcal{M}} = Sat^x_{\mathcal{M}}(D[x])$

La notion de type de donnée sera donc liée à une interprétation classique $\mathcal{M}$ (Les formules définissant les types n'ayant qu'une variable libre, la notion de type de donnée ne dépendra en fait que de l'interprétation des constructeurs du type).

Afin de justifier la correction des programmes, la notion de type de donnée doit imposer une contrainte sur les termes du type D. On va imposer cette contrainte par une propriété de l'interprétation pleine. On est conduit à proposer la définition suivante, due à Jean-Louis Krivine :

[1] Il ne faut pas confondre les termes de type D (que l'on obtient par une preuve) et les éléments du type D. Dans le cas des types de données infinies nous verrons qu'il y a une différence.

Définition 5.2. On dira que le modèle $\mathcal{M}$ est *intentionnel* pour le type de données D défini par la formule $D[x]$ à une variable libre x (du premier ordre) si et seulement si pour toute interprétation pleine σ, coïncidant avec $\mathcal{M}$ sur les termes du premier ordre, on a :

$$\mathtt{u} \in \left|D[x]\right|^{\sigma[x \Leftarrow \mathtt{t}]} \text{ si et seulement si } \mathtt{u} \sim \mathtt{t} \text{ et } \mathtt{t} \in D^{\mathcal{M}}$$

La définition ci-dessus est relative à l'équivalence $\sim$. Dans la définition originale, la β-équivalence était utilisée. Cela ne suffit pas pour les types de données infinies. Nous utiliserons la ω-équivalence (ce qui pour les types de données finies ne change rien, puisque les deux équivalences coïncident sur les termes normalisables). On pourrait aussi parler de type de données à $\omega\eta$-équivalence près. (cf annexe A pour la définition de ces équivalences).

Montrons maintenant comment cette définition justifie la méthode de programmation :

Proposition 5.3. *Considérons un langage comprenant un symbole de fonction n-aire* $\mathbf{f}$ *satisfaisant un ensemble d'équation* $\mathcal{E}$*. Considérons un terme* $\mathtt{t}$ *tel que :*

$$\vdash \mathtt{t} : \forall x_1 \ldots \forall x_n \left(D_1[x_1], \ldots, D_n[x_n] \rightarrow E[\mathbf{f}\,(x_1, \ldots, x_n)]\right)$$

Si l'on considère un modèle $\mathcal{M}$ *intentionnel pour* $D_1, \ldots, D_n$ *et* E*, et si l'on considère* n *termes* $\mathtt{u}_1, \ldots, \mathtt{u}_n$ *éléments respectifs des types* $D_1, \ldots, D_n$*, alors* $(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n)$ *est un élément du type* E *et* $(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n) \sim |f|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_n)$ *(Ceci signifie que* $\mathtt{t}$ *est bien un terme calculant la fonction* $|f|^{\mathcal{M}}$*).*

Démonstration : Considérons une interprétation pleine σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. Par le lemme de conservation (3.14), on trouve :

$$\mathtt{t} \in \left|\forall x_1 \ldots \forall x_n \left(D_1[x_1], \ldots, D_n[x_n] \rightarrow E[\mathbf{f}\,(x_1, \ldots, x_n)]\right)\right|^{\sigma}$$

Considérons n termes $\mathtt{u}_1, \ldots, \mathtt{u}_n$ éléments respectif des types $D_1, \ldots, D_n$. Par définition de l'interprétation, on trouve :

$$(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n) \in \left|E[\mathbf{f}\,(x_1, \ldots, x_n)]\right|^{\sigma[\overline{x} \Leftarrow \overline{\mathtt{u}}]}$$

Ce qui donne :

$$(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n) \in \left|E[x]\right|^{\sigma\left[x \Leftarrow |f|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_n)\right]}$$

Or, $\mathcal{M}$ étant intentionnel pour E, on trouve :

$$(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n) \sim |f|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_n) \text{ et } \mathcal{M}\left[x \Leftarrow (\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n)\right] \models E[x]$$

■

Cela ne suffit pas encore tout à fait. En effet, supposons que l'on veuille programmer un ensemble de fonctions $\mathbf{f}_1, \ldots, \mathbf{f}_n$ définies sur des types de données $D_1, \ldots, D_p$ (chaque type étant un ensemble de termes). Supposons aussi que pour chacune des fonctions $\mathbf{f}_i$, on ait obtenu un terme $\mathtt{t}_i$ du type voulu en utilisant les équations $\mathcal{E}$ satisfaites par les fonctions $\mathbf{f}_1, \ldots, \mathbf{f}_n$. La proposition précédente assure que le terme $\mathtt{t}_i$ calcule bien la fonction $\mathbf{f}_i$ à condition d'avoir pu construire un modèle $\mathcal{M}$, intentionnel pour chacun des types et satisfaisant les équations $\mathcal{E}$.

En fait, on montre facilement que si l'on a un modèle de ces équations, au sens usuel du terme (l'interprétation d'un symbole de fonction n'est définit que sur les type de données), alors on peut toujours l'étendre à l'ensemble de tous les λ-termes, à condition qu'aucun des types ne soit réduit à un seul élément (un tel type satisfait l'équation $\forall x \forall y (x = y)$ qui ne peut être étendue à Λ) ([11] p153-155).

Il faut faire deux remarques qui illustrent la différence entre les types infinis et finis :

- Dans un type données infinies, il n'y a que des structures récursives, puisqu'elles s'expriment avec des termes du λ-calcul. Par exemple, les éléments du type "stream d'entiers" correspondent uniquement aux listes infinies et récursives d'entiers.
- Par définition, il y a bijection entre les éléments d'un type de données D et les termes appartenant à l'interprétation de D, mais l'ensemble des termes de type D (ceux que l'on peut extraire d'une preuve) est seulement inclus dans les éléments du type. Par exemple, pour définir le moindre stream il faut ajouter un constructeur et des équations (pour le stream des entiers à partir de n, on prend un constructeur $\mathbf{sn}$ vérifiant $\mathbf{sn}(n) = \mathbf{cs}(n, \mathbf{sn}(\mathbf{s}n))$). Ainsi, il est impossible de typer tous les éléments d'un type de données infinies.

  Cela reste vrai quelque soit l'ensemble d'équations que l'on se donne. En effet, on peut typer les deux termes qui expriment la bijection entre les streams d'entiers et les fonctions totales des entiers dans les entiers (cette bijection associe à un stream la fonction qui associe à un entier n l'entier qui se trouve à la position n dans le stream). Or, les fonctions typables sont exactement les fonctions prouvablement totales dans l'arithmétique du second ordre [11]. Les streams typables sont donc exactement ceux qui correspondent à ces fonctions, tandis que les éléments du type stream sont tous les streams récursifs.

5.3 Les streams.

Avant d'attaquer le cas général qui est assez complexe, nous allons caractériser les modèles intentionnels pour la définition des streams donnée dans le chapitre 2. Cet exemple permettra de mieux comprendre le mécanisme de la preuve.

On rappelle que les streams d'objets de type A sont définis par la formule suivante :

$$\mathrm{S}_A[x] = \nu K \lambda y \forall X \left(\forall a \forall s \left(A[a], Ks \to X \mathbf{cs}\,(a, s) \right) \to Xy \right) \langle x \rangle$$

Proposition 5.4. *Si $\mathcal{M}$ est intentionnel pour le type A, Les deux propositions suivantes sont équivalentes :*

1. *$\mathcal{M}$ est intentionnel pour $\mathrm{S}_A[x]$*
2. *Pour tous termes $\mathtt{u}$ et $\mathtt{v}$ tel que $\mathtt{u} \in A^{\mathcal{M}}$ et $\mathtt{v} \in \mathrm{S}_A^{\mathcal{M}}$, on a $|\mathbf{cs}|^{\sigma}(\mathtt{u}, \mathtt{v}) \sim_{\omega} \lambda \mathtt{f}\,(\mathtt{f}\ \mathtt{u}\ \mathtt{v})$.*

Démonstration : Commençons par prouver (1) implique (2). Supposons le modèle $\mathcal{M}$ intentionnel pour $\mathrm{S}_A[x]$ et A. Choisissons deux termes $\mathtt{u}$ et $\mathtt{v}$ tel que $\mathtt{u} \in A^{\mathcal{M}}$ et $\mathtt{v} \in \mathrm{S}_A^{\mathcal{M}}$. Choisissons une interprétation σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. On a alors $\mathtt{u} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{u}]}$ et $\mathtt{v} \in |\mathrm{S}_A[y]|^{\sigma[y \Leftarrow \mathtt{v}]}$. D'où, par définition de l'interprétation de la formule

$\mathrm{S}_A[y]$, $\lambda \mathtt{f}(\mathtt{f}\ \mathtt{u}\ \mathtt{v}) \in |\mathrm{S}_A[\mathbf{cs}(x,y)]|^{\sigma[x \Leftarrow \mathtt{u}][y \Leftarrow \mathtt{v}]}$. Or, $\mathcal{M}$ étant intentionnel pour $\mathrm{S}_A[x]$, on en déduit $\lambda \mathtt{f}(\mathtt{f}\ \mathtt{u}\ \mathtt{v}) \sim_\omega |\mathbf{cs}|^\sigma(\mathtt{u},\mathtt{v})$.

Montrons maintenant (2) implique (1). Choisissons une interprétation σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre, et vérifiant (2). Définissons $\Delta_0 \in \Lambda \to \mathcal{P}(\Lambda)$ par induction :

- $\mathtt{t} \in \Delta_0^0(\mathtt{u})$ pour tous λ-termes $\mathtt{t}$ et $\mathtt{u}$.
- $\mathtt{t} \in \Delta_0^\alpha(\mathtt{u})$ ssi pour tout ordinal $\gamma < \alpha$, il existe $\mathtt{a}, \mathtt{b}, \mathtt{t}', \mathtt{u}' \in \Lambda$ tel que $\mathtt{t} \sim_\beta \lambda \mathtt{f}(\mathtt{f}\ \mathtt{a}\ \mathtt{t}')$, $\mathtt{u} \sim_\beta |\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}')$, $\mathtt{a} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{t}' \in \Delta_0^\gamma(\mathtt{u}')$.
- $\mathtt{t} \in \Delta_0(\mathtt{u})$ ssi $\mathtt{t} \in \Delta_0^\alpha(\mathtt{u})$ pour tout ordinal α.

Montrons que $\mathtt{t} \in |S[x]|^{\sigma[x \Leftarrow \mathtt{u}]}$ ssi $\mathtt{t} \in \Delta_0(\mathtt{u})$. Notons $\Psi(\Phi) = |\lambda x \forall X (\forall n \forall y (A[n] \to Ky \to X\mathbf{cs}(n,y)) \to Xx)|^{\sigma[K \Leftarrow \Phi]}$, et commençons par prouver :

$\mathtt{t} \in \Psi(\Phi)(\mathtt{u})$ ssi $\mathtt{t} \sim_\beta \lambda \mathtt{f}(\mathtt{f}\ \mathtt{a}\ \mathtt{t}')$ et $\mathtt{u} \sim_\beta |\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}')$ avec $\mathtt{a} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{t}' \in \Phi(\mathtt{u}')$.

- Pour l'implication gauche droite, supposons que $\mathtt{t} \in \Psi(\Phi)(\mathtt{u})$, et choisissons une λ-variable $\mathtt{f}$ n'apparaissant pas libre dans $\mathtt{t}$. Définissons Θ par $\mathtt{t} \in \Theta(\mathtt{u})$ ssi $\mathtt{t} \sim_\beta (\mathtt{f}\ \mathtt{a}\ \mathtt{t}')$, $\mathtt{u} \sim_\beta |\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}')$, $\mathtt{a} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{t}' \in \Phi(\mathtt{u}')$. Notons $\sigma' = \sigma[x \Leftarrow \mathtt{u}][K \Leftarrow \Phi][X \Leftarrow \Theta]$. Pour tout $\mathtt{a} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{t}' \in \Phi(\mathtt{u}')$, on a $(\mathtt{f}\ \mathtt{a}\ \mathtt{t}') \in \Theta(|\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}'))$. Donc $\mathtt{f} \in |\forall n \forall y (A[n] \to Ky \to X\mathbf{cs}(n,y))|^{\sigma'}$. D'où $(\mathtt{t}\ \mathtt{f}) \in |Xx|^{\sigma'}$. Par définition de Θ, on obtient alors $(\mathtt{t}\ \mathtt{f}) \sim_\beta (\mathtt{f}\ \mathtt{a}\ \mathtt{t}')$ et $\mathtt{u} \sim_\beta |\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}')$ avec $\mathtt{a} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{t}' \in \Phi(\mathtt{u}')$. Or, $\mathtt{f}$ n'apparaissant pas libre dans $\mathtt{t}$, de $(\mathtt{t}\ \mathtt{f}) \sim_\beta (\mathtt{f}\ \mathtt{a}\ \mathtt{t}')$ on déduit $\mathtt{t} \sim_\beta \lambda \mathtt{f}(\mathtt{f}\ \mathtt{a}\ \mathtt{t}')$. D'où le résultat attendu.
- Pour le sens droite-gauche, supposons $\mathtt{t} \sim_\beta \lambda \mathtt{f}(\mathtt{f}\ \mathtt{a}\ \mathtt{t}')$, $\mathtt{u} \sim_\beta |\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}')$, $\mathtt{a} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{t}' \in \Phi(\mathtt{u}')$. Choisissons $\Theta \in \Lambda \to \mathcal{P}(\Lambda)$ et notons $\sigma' = \sigma[x \Leftarrow \mathtt{u}][K \Leftarrow \Phi][X \Leftarrow \Theta]$. Soit $\mathtt{v} \in |\forall n \forall y (A[n] \to Ky \to X\mathbf{cs}(n,y))|^{\sigma'}$. On doit montrer que $(\mathtt{t}\ \mathtt{v}) \in |Xx|^{\sigma'}$. Mais $(\mathtt{t}\ \mathtt{v}) \sim_\beta (\mathtt{v}\ \mathtt{a}\ \mathtt{t}')$ avec $\mathtt{a} \in |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{t}' \in \Phi(\mathtt{u}')$. Ainsi, on obtient $(\mathtt{v}\ \mathtt{a}\ \mathtt{t}') \in |X\mathbf{cs}(n,y)|^{\sigma'[n \Leftarrow \mathtt{b}][y \Leftarrow \mathtt{u}']}$, qui implique $(\mathtt{v}\ \mathtt{a}\ \mathtt{t}') \in \Theta(|\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}'))$. On obtient alors le résultat voulu car $|\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}') \sim_\beta \mathtt{u}$.

On en déduit donc $\Delta_0^{\alpha+1} = \Psi(\Delta_0^\alpha)$. Ainsi, la définition de Δ_0 correspond à celle du plus grand fixe par induction. On obtient donc bien $\mathtt{t} \in |S[x]|^{\sigma[x \Leftarrow \mathtt{u}]}$ ssi $\mathtt{t} \in \Delta_0(\mathtt{u})$.

De la même manière, on peut définir $\Theta_0 \in \mathcal{P}(\Lambda)$ par induction :

- $\Theta_0^0 = \Lambda$.
- $\mathtt{u} \in \Theta_0^\alpha$ ssi pour tout ordinal $\gamma < \alpha$, il existe $\mathtt{b}, \mathtt{u}'$ tel que $\mathtt{u} \sim_\beta |\mathbf{cs}|^\sigma(\mathtt{b},\mathtt{u}')$ avec $\mathcal{M} \models |A[x]|^{\sigma[x \Leftarrow \mathtt{b}]}$ et $\mathtt{u}' \in \Theta_0^\gamma$.
- $\mathtt{u} \in \Theta_0$ ssi $\mathtt{u} \in \Theta_0^\alpha$ pour tout ordinal α.

Par une preuve similaire, on obtient $\mathcal{M}[x \Leftarrow \mathtt{u}] \models S[x]$ ssi $\mathtt{u} \in \Theta_0$.

Maintenant, pour prouver la proposition, il suffit de montrer que $\mathtt{t} \in \Delta_0(\mathtt{u})$ ssi $\mathtt{t} \sim_\omega \mathtt{u}$ et $\mathtt{u} \in \Theta_0$. On obtient ce résultat par la succession des lemmes suivants, dont on trouvera la preuve avec celle du cas général :

1. (lemme 5.12) On montre $\Delta_0(\mathtt{u}) \neq \emptyset$ si et seulement si $\mathtt{u} \in \Theta_0$.

2. (lemme 5.14) On en déduit $\mathtt{t} \in \Delta_0(\mathtt{u})$ implique $\mathtt{t} \sim_\omega \mathtt{u}$. Cela découle de la définition de $\sim_\omega$, il suffit de montrer par reccurence que $\mathtt{t} \in \Delta_0(\mathtt{u})$ implique $\mathtt{t} \sim_{\omega_n} \mathtt{u}$. On utilise (2) uniquement pour montrer ce lemme.

3. (lemme 5.15) On prouve alors que $\Delta_0(\mathtt{u})$ est clos pour la ω-équivalence.

4. (lemme 5.16) On déduit immédiatement des deux lemmes précédents que $\mathtt{t} \in \Delta_0(\mathtt{u})$ si et seulement si $\mathtt{t} \sim_\omega \mathtt{u}$ et $\Delta_0(\mathtt{u}) \neq \emptyset$, d'où le résultat.

■

5.4 Construction formelle des types de données.

Dans cette section, on va exposer formellement la méthode utilisée au début de ce chapitre pour construire un type de données.

On suppose que l'on a déjà démontré que le modèle $\mathcal{M}$ était intentionnel pour les types $D_1, \ldots, D_m$ (cf définition 5.1). À partir de ces types, on va définir un nouveau type de données D.

Comme on l'a vu dans les exemples précédents, on peut définir ce type de données en utilisant le plus petit point fixe, le plus grand, ou même un mélange des deux.

Pour formaliser cela, on se donne le rang r du type qui est le nombre de points fixes utilisés. On se donne aussi $\epsilon_1, \ldots, \epsilon_r \in \{-, +\}$ pour indiquer respectivement si le i-ème point fixe est un plus petit point fixe ($\epsilon_i = -$) ou un plus grand point fixe ($\epsilon_i = +$). Dans cette numérotation des points fixes, on conviendra que l'indice 1 correspond au dernier point fixe à calculer (le plus à gauche dans la formule).

On doit donner aussi les constructeurs du type : $\mathbf{c}_1, \ldots, \mathbf{c}_n$. Pour chaque constructeur $\mathbf{c}_i$, on doit donner son arité n_i, et si elle est non nulle, on doit donner le type de chacun des arguments : $T_{i,j}$ pour $0 < j \leq n_i$. Bien sur, $T_{i,j}$ est soit D, soit l'un des types de données D_k déjà définis. De plus, si $T_{i,j} = D$, on se donne un entier $r_{i,j} \in \{1, \ldots, r\}$ (pour indiquer à quel niveau ce paramètre inductif sera lié par un point fixe).

On peut maintenant écrire la formule qui définit ce type. Pour cela, on se donne $r+1$ variables de prédicats d'arité 1 : $X, K_1, \ldots, K_r$. Pour chaque constructeur $\mathbf{c}_i$ d'arité n_i, on définit la formule C_i de la manière suivante :

$$C_i = \forall a_1 \ldots \forall a_{n_i} \left(A_1 a_1, \ldots, A_{n_i} a_{n_i} \rightarrow X \, \mathbf{c}_i \left(a_1, \ldots, a_{n_i} \right) \right)$$

avec $A_j = D_k$ si $T_{i,j} = D_k$ et $A_j = K_{r_{i,j}}$ si $T_{i,j} = D$.

On définit alors les formules suivantes par récurrence :

- On pose :
$$F_r[K_1, \ldots, K_r, x] = \forall X \left(C_1, \ldots, C_n \rightarrow X \left(x \right) \right)$$

- Pour $0 \leq j < r$ avec $\epsilon_{j+1} = -$, on pose :
$$F_j[K_1, \ldots, K_j, x] = \mu K_{j+1} \lambda x F_{j+1}[K_1, \ldots, K_{j+1}, x] \langle x \rangle$$

- Pour $0 \leq j < r$ avec $\epsilon_{j+1} = +$, on pose :

$$F_j[K_1, \ldots, K_j, x] = \nu K_{j+1} \lambda x F_{j+1}[K_1, \ldots, K_{j+1}, x]\langle x \rangle$$

La formule $D[x]$ qui définit le type D est alors :

$$D[x] = F_0[x]$$

Afin de bien mettre en place les notations, appliquons-les à deux types définis au début de ce chapitre :

- Pour les streams d'entiers S_N :

$$S_N[x] = \nu K \lambda z \forall X \left(\forall a \forall y \left(Na, Ky \rightarrow X \mathbf{cs}\,(a, y) \right) \rightarrow Xz \right) \langle x \rangle$$

  - $r = 1$: un seul point fixe.
  - $\epsilon_1 = +$: c'est un plus grand point fixe.
  - $\mathbf{c}_1 = \mathbf{cs}$, $n_1 = 2$: un seul constructeur d'arité 2.
  - $T_{1,1} = N$: le premier argument de **cs** est un entier.
  - $T_{1,2} = S_N$: le second argument est inductif, c'est à dire du type que l'on est en train de définir.
  - $r_{1,2} = 1$: ce paramètre inductif est lié par le premier et le seul point fixe intervenant dans la construction du type.

- Pour le type des suites binaires n'utilisant qu'un nombre fini de 1 (B') :

$$B'[x] = \nu K \lambda y \mu K' \lambda z F\,(K, K', z)\,\langle y \rangle \langle x \rangle$$

  avec :

$$F[K, K', z] = \forall X \left(\forall y \left(Ky \rightarrow X \mathbf{s}_0\,(y) \right), \forall y \left(K'y \rightarrow X \mathbf{s}_1\,(y) \right) \rightarrow Xz \right)$$

  - $r = 2$: deux points fixes.
  - $\epsilon_1 = +$ et $\epsilon_2 = -$: un plus grand puis un plus petit point fixe.
  - $\mathbf{c}_1 = \mathbf{s}_0$, $n_1 = 1$ et $\mathbf{c}_2 = \mathbf{s}_1$, $n_2 = 1$: deux constructeurs d'arité 1.
  - $T_{1,1} = T_{2,1} = B'$: les arguments de ces deux constructeurs sont inductifs.
  - $r_{1,1} = 1$ et $r_{2,1} = 2$: le premier paramètre inductif est lié par le premier point fixe, le second étant lié par le second point fixe.

On va maintenant calculer les éléments du type de données D, dans une interprétation classique $\mathcal{M}$. Pour cela, construisons les ensembles suivants :

- $\mathcal{C}_i(\Phi_1, \ldots, \Phi_r) = \{ |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}) \setminus \mathtt{t}_j \in \Phi_{r_{i,j}} \text{ si } T_{i,j} = D, \mathtt{t}_j \in D_k^{\mathcal{M}} \text{ si } T_{i,j} = D_k \}$

- $\Theta_r(\Phi_1, \ldots, \Phi_r) = \bigcup_{0<i\leq n} \mathcal{C}_i(\Phi_1, \ldots, \Phi_r)$
- $\Theta_j(\Phi_1, \ldots, \Phi_j) = \bigcap\{\Phi \setminus \Theta_{j+1}(\Phi_1, \ldots, \Phi_j, \Phi) \subseteq \Phi\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = -$
- $\Theta_j(\Phi_1, \ldots, \Phi_j) = \bigcup\{\Phi \setminus \Phi \subseteq \Theta_{j+1}(\Phi_1, \ldots, \Phi_j, \Phi)\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = +$

On montre alors le lemme et la proposition suivantes :

Lemme 5.5. *Pour tout $\Psi \in \mathcal{P}(\Lambda)$, pour tout $\overline{\Phi} \in \mathcal{P}(\Lambda)^n$ et pour tout modèle $\mathcal{M}$, on a :*

$$\mathcal{M}\left[\overline{K} \Leftarrow \overline{\Phi}\right][X \Leftarrow \Psi] \models \mathcal{C}_i \text{ est équivalent à } \mathcal{C}_i\left(\Phi_1, \ldots, \Phi_r\right) \subseteq \Psi.$$

Démonstration : On rappelle que :

$$C_i = \forall a_1 \ldots \forall a_{n_i} \left(A_1 a_1, \ldots, A_{n_i} a_{n_i} \rightarrow X \mathbf{c}_i \left(a_1, \ldots, a_{n_i}\right)\right)$$

avec $A_j = D_k$ si $T_{i,j} = D_k$ et $A_j = K_{r_{i,j}}$ si $T_{i,j} = D$.

Donc, par définition de l'interprétation classique, $\mathcal{M}[\overline{K} \Leftarrow \overline{\Phi}][X \Leftarrow \Psi] \models \mathcal{C}_i$ si et seulement si pour tous termes $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$, $\mathtt{t}_j \in Sat^{a_j}_{\mathcal{M}[\overline{K}\Leftarrow\overline{\Phi}][X\Leftarrow\Psi]}(A_j a_j)$ pour tout j, entraîne $|\mathbf{c}_i|^{\mathcal{M}}(\overline{\mathtt{t}}) \in \Psi$. Ce qui donne bien le résultat voulu, vue la définition des formules A_j. ■

Proposition 5.6. *En utilisant les notations introduites ci-dessus, l'ensemble $D^{\mathcal{M}}$ des éléments du type D est égal à Θ_0.*

Démonstration : On réalise cette preuve par induction sur la construction de Θ_0. On commence par montrer que :

$$\Theta_r\left(\Phi_1, \ldots, \Phi_r\right) = Sat^x_{\mathcal{M}[\overline{K}\Leftarrow\overline{\Phi}]}\left(F_r[K_1, \ldots, K_r, x]\right)$$

Par définition de l'interprétation classique, $\mathtt{t} \in Sat^x_{\mathcal{M}[\overline{K}\Leftarrow\overline{\Phi}]}(F_r[K_1, \ldots, K_r, x])$ est équivalent à, pour tout $\Psi \in \mathcal{P}(\Lambda)$,

$$\mathcal{M}\left[\overline{K} \Leftarrow \overline{\Phi}\right][X \Leftarrow \Psi][x \Leftarrow \mathtt{t}] \models C_i \text{ pour tout } i, \text{ entraine } \mathtt{t} \in \Psi$$

Donc, en utilisant le lemme 5.5, ceci est équivalent à : pour tout $\Psi \in \mathcal{P}(\Lambda)$, pour tout i $\mathcal{C}_i(\overline{\Phi}) \subseteq \Psi$, entraîne $\mathtt{t} \in \Psi$. Ce qui est équivalent à $\mathtt{t}$ appartient à la réunion des $\mathcal{C}_i(\Phi_1, \ldots, \Phi_r)$, qui est exactement la définition de $\Theta_r(\Phi_1, \ldots, \Phi_r)$.

Reste à montrer que si :

$$\Theta_{j+1}\left(\Phi_1, \ldots, \Phi_{j+1}\right) = Sat^x_{\mathcal{M}[\overline{K}\Leftarrow\overline{\Phi}][x\Leftarrow u]}\left(F_{j+1}[K_1, \ldots, K_{j+1}, x]\right)$$

alors la même propriété est vrai pour j. Ceci découle immédiatement des définitions de Θ_j et F_j, car F_j est construit comme un plus petit ou un plus grand point fixe de F_{j+1}, et la définition de Θ_j est exactement l'interprétation de ce point fixe. ■

Corollaire 5.7. *Pour $0 \leq j < r$, on peut aussi définir Θ_j à partir de Θ_{j+1} par induction sur les ordinaux, de la manière suivante :*

- *Si $\epsilon_j = -$, on pose*

$$\begin{aligned}\Theta_j^0(\Phi_1, \ldots, \Phi_j) &= \emptyset \\ \Theta_j^\alpha(\Phi_1, \ldots, \Phi_j) &= \bigcup_{\gamma<\alpha} \Theta_{j+1}\left(\Phi_1, \ldots, \Phi_j, \Theta_j^\gamma\left(\Phi_1, \ldots, \Phi_j\right)\right)\end{aligned}$$

  Cette suite est croissante, donc constante à partir d'un ordinal α_0, et on a $\Theta_j = \Theta_j^{\alpha_0}$.

- *Si $\epsilon_j = +$, on pose*

$$\begin{aligned}\Theta_j^0(\Phi_1, \ldots, \Phi_j) &= \Lambda \\ \Theta_j^\alpha(\Phi_1, \ldots, \Phi_j) &= \bigcap_{\gamma<\alpha} \Theta_{j+1}\left(\Phi_1, \ldots, \Phi_j, \Theta_j^\gamma\left(\Phi_1, \ldots, \Phi_j\right)\right)\end{aligned}$$

  Cette suite est décroissante, donc constante à partir d'un ordinal α_0, et on a $\Theta_j = \Theta_j^{\alpha_0}$.

Démonstration : Dans la preuve de la proposition précédente, on a vu que $\Theta_j(\Phi_1, \ldots, \Phi_j)$ correspondait à l'interprétation d'une formule F_j, les Φ_k étant les interprétations de variables positives. Ceci montre que Θ_j est une fonction croissante (cf lemme de croissance 3.9). Cela justifie les constructions ci-dessus (définition par induction des points fixes). ■

Corollaire 5.8. *Si $\mathtt{u}$ est un élément du type D dans le modèle $\mathcal{M}$, alors il existe un constructeur $\mathbf{c}_i$ et des termes $\mathtt{u_1}, \ldots, \mathtt{u_{n_i}}$ tel que $\mathtt{u_j} \in T_{i,j}^{\mathcal{M}}$ et tel que $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u_1}, \ldots, \mathtt{u_{n_i}})$.*

Démonstration : Par définition Θ_r, si $\mathtt{u} \in \Theta_r(\Phi_1, \ldots, \Phi_r)$, il existe un constructeur $\mathbf{c}_i$ et des termes $\mathtt{u_1}, \ldots, \mathtt{u_{n_i}}$ tel que $\mathtt{u_j} \in D_k^{\mathcal{M}}$ si $T_{i,j} = D_k$ et $\mathtt{u_j} \in \Phi_{r_{i,j}}$ si $T_{i,j} = D$.

On montre alors par induction que si $\mathtt{u} \in \Theta_j(\Phi_1, \ldots, \Phi_j)$, il existe un constructeur $\mathbf{c}_i$ et des termes $\mathtt{u_1}, \ldots, \mathtt{u_{n_i}}$ tel que :

- $\mathtt{u_j} \in D_k^{\mathcal{M}}$ si $T_{i,j} = D_k$
- $\mathtt{u_j} \in \Phi_{r_{i,j}}$ si $T_{i,j} = D$ avec $r_{i,j} \leq j$.
- $\mathtt{u_j} \in \Theta_j(\Phi_1, \ldots, \Phi_j)$ si $T_{i,j} = D$ avec $r_{i,j} > j$

On vient de montrer le cas $j = r$. Supposons cette propriété vrai pour Θ_{j+1} et montrons qu'elle reste vrai pour Θ_j. Pour cela, il suffit d'utiliser le fait que Θ_j est un point fixe de Θ_{j+1} :

$$\Theta_j\left(\Phi_1, \ldots, \Phi_j\right) = \Theta_{j+1}\left(\Phi_1, \ldots, \Phi_j, \Theta_j\left(\Phi_1, \ldots, \Phi_j\right)\right)$$

Le cas $j = 0$ donne alors le résultat voulu car $D^{\mathcal{M}} = \Theta_0$. ■

On vient de montrer que la construction de Θ_0 correspond bien au calcul des éléments du type D. De plus, les lemmes et corollaires précédants montrent que la construction de Θ_0 correspond bien à la construction intuitive réalisée sur des exemples dans la première section de ce chapitre.

5.5 Adéquation de la construction.

Dans cette section, nous montrons qu'il est facile de caractériser les modèles intentionnels pour le type de données D ainsi construit. Pour cela, on définit la notion de modèle syntaxiquement intentionnel pour le type D :

Définition 5.9. On dira que le modèle $\mathcal{M}$ est syntaxiquement intentionnel pour le type D si et seulement si pour tout constructeur $\mathbf{c}_i$ du type D et pour tous termes $\mathtt{u}_1, \ldots, \mathtt{u}_{\mathtt{n_i}}$ tel que $\mathtt{u_j} \in T_{i,j}^{\mathcal{M}}$ (ce qui signifie que $\mathtt{u}_1, \ldots, \mathtt{u}_{\mathtt{n_i}}$ sont des termes du bon type pour le constructeur $\mathbf{c}_i$), on a

$$|\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_k) \sim_\omega \lambda \mathtt{f_i} \ldots \lambda \mathtt{f_n}\,(\mathtt{f_i}\ \mathtt{u}_1 \ldots \mathtt{u_k}).$$

Tout au long de cette section, on va développer un certain nombre de lemmes et de définitions afin de finalement démontrer les théorèmes 5.17 et 5.18 qui affirment qu'un modèle $\mathcal{M}$ est syntaxiquement intentionnel pour le type D si et seulement s'il est intentionnel pour le type D (sous reserve que $\mathcal{M}$ soit aussi intentionnel pour les types $D_1, \ldots, D_m$ intervenant dans la définition de D).

Pour cela, on commence par se donner un modèle $\mathcal{M}$ quelconque et une interprétation pleine σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. Puis, on définit les fonctions $\Delta_j \in (\Lambda \to \mathcal{P}(\Lambda))^j \to (\Lambda \to \mathcal{P}(\Lambda))$ suivantes :

- $\mathtt{t} \in \Delta_r(\Psi_1, \ldots, \Psi_r)(\mathtt{u})$ ssi il existe un entier i et des termes $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}, \mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$ tel que :
  - $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_{n_i})$
  - $\mathtt{t} \sim_\beta \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f_n}(\mathtt{f_i}\ \mathtt{t}_1 \ldots \mathtt{t_{n_i}})$ ($\mathtt{f}_1, \ldots, \mathtt{f}_n$ n'étant pas libre dans $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$)
  - Pour tout j, si $T_{i,j} = D_k$ (type déjà défini), $\mathtt{t}_j \in |D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$
  - Pour tout j, si $T_{i,j} = D$ (paramètre inductif), $\mathtt{t}_j \in \Psi_{r_{i,j}}(\mathtt{u}_j)$
- $\Delta_j(\mathtt{u}) = \bigcap\{\Psi(\mathtt{u}) \setminus \Delta_{j+1}(\Psi_1, \ldots, \Psi_j, \Psi) \leq \Psi\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = -$
- $\Delta_j(\mathtt{u}) = \bigcup\{\Psi(\mathtt{u}) \setminus \Psi \leq \Delta_{j+1}(\Psi_1, \ldots, \Psi_j, \Psi)\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = +$

On montre alors les lemmes suivants :

Lemme 5.10. *Pour tous termes* $\mathtt{t}$ *et* $\mathtt{u}$ *on a :*

$$\mathtt{t} \in \left|D[x]\right|^{\sigma[x \Leftarrow \mathtt{u}]} \text{ ssi } \mathtt{t} \in \Delta_0(\mathtt{u})$$

Démonstration : Commençons par montrer que

$$\mathtt{t} \in \left|F_r[K_1, \ldots, K_r, x]\right|^{\sigma\left[\overline{K} \Leftarrow \overline{\Psi}\right][x \Leftarrow \mathtt{u}]} \text{ ssi } \mathtt{t} \in \Delta_r(\Psi_1, \ldots, \Psi_r)(\mathtt{u})$$

Montrons d'abord le sens droite-gauche de l'équivalence :

Supposons $\mathtt{t} \in \Delta_r(\Psi_1, \ldots, \Psi_r)(\mathtt{u})$, on doit montrer $\mathtt{t} \in F_r[K_1, \ldots, K_r, x]^{\sigma[\overline{K} \Leftarrow \overline{\Psi}][x \Leftarrow \mathtt{u}]}$. Soit $\Psi \in \Lambda \to \mathcal{P}(\Lambda)$ et $\mathtt{v}_1, \ldots, \mathtt{v}_n$ vérifiant $\mathtt{v}_k \in |C_k|^{\sigma[\overline{K} \Leftarrow \overline{\Psi}][X \Leftarrow \Psi][x \Leftarrow \mathtt{u}]}$ pour tout k.

Par définition de l'interprétation, on doit prouver $(\mathtt{t}\ \mathtt{v_1} \dots \mathtt{v_n}) \in \Psi(\mathtt{u})$. Par définition de Δ_r, on trouve des termes tel que $\mathtt{t} \sim_\beta \lambda \mathtt{f_1} \dots \lambda \mathtt{f_n}(\mathtt{f_i}\ \mathtt{t_1} \dots \mathtt{t_{n_i}})$, $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_{n_i})$, $\mathtt{t}_j \in |A_j(a_j)|^{\sigma[\overline{K} \Leftarrow \overline{\Psi}][a_j \Leftarrow \mathtt{u}_j]}$ avec $A_j = D_k$ si $T_{i,j} = D_k$ et $A_j = K_{r_{i,j}}$ si $T_{i,j} = D$. On en déduit, par définition de C_i, $(\mathtt{t}\ \mathtt{v_1} \dots \mathtt{v_n}) \sim_\beta (\mathtt{v_i}\ \mathtt{t_1} \dots \mathtt{t_{n_i}}) \in \Psi(|\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_{n_i}))$. Or, on a, $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_{n_i})$. D'où le résultat.

Montrons maintenant le sens gauche-droite de l'équivalence :

On suppose $\mathtt{t} \in F_r[K_1, \dots, K_r, x]^{\sigma[\overline{K} \Leftarrow \overline{\Psi}][x \Leftarrow \mathtt{u}]}$. Choisissons $\Psi \in \Lambda \to \mathcal{P}(\Lambda)$ de la manière suivante : soit n variables $\mathtt{f}_1, \dots, \mathtt{f}_n$ n'apparaissant pas libres dans $\mathtt{t}$ et $\mathtt{u}$. On pose $\mathtt{v} \in \Psi(\mathtt{w})$ si et seulement s'il existe un entier i et des termes $\mathtt{t}_1, \dots, \mathtt{t}_{n_i}, \mathtt{u}_1, \dots, \mathtt{u}_{n_i}$ tel que :

- $\mathtt{w} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_{n_i})$
- $\mathtt{v} \sim_\beta (\mathtt{f_i}\ \mathtt{t_1} \dots \mathtt{t_{n_i}})$ ($\mathtt{f}_1, \dots, \mathtt{f}_n$ n'étant pas libre dans $\mathtt{t}_1, \dots, \mathtt{t}_{n_i}$)
- Pour tout j, si $T_{i,j} = D_k$ (type déjà défini), $\mathtt{t}_j \in |D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$
- Pour tout j, si $T_{i,j} = D$ (paramètre inductif), $\mathtt{t}_j \in \Psi_{r_{i,j}}(\mathtt{u}_j)$

Par définition de l'interprétation, on a $(\mathtt{t}\ \mathtt{v_1} \dots \mathtt{v_n}) \in \Psi(\mathtt{u})$ pour tous termes $\mathtt{v}_1, \dots, \mathtt{v}_n$, vérifiant pour tout k, $\mathtt{v}_k \in |C_k|^{\sigma[\overline{K} \Leftarrow \overline{\Psi}][X \Leftarrow \Psi][x \Leftarrow \mathtt{u}]}$. Or, pour tous termes $\mathtt{t}_1, \dots, \mathtt{t}_{n_k}, \mathtt{u}_1, \dots, \mathtt{u}_{n_k}$, vérifiant $\mathtt{t}_j \in |A_j a_j|^{\sigma[\overline{K} \Leftarrow \overline{\Psi}][a_j \Leftarrow \mathtt{u}_j]}$ pour tout j, on a $(\mathtt{f_k}\ \mathtt{t_1} \dots \mathtt{t_{n_i}}) \in \Psi(|\mathbf{c}_k|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_{n_k}))$. Donc, Par définition de $\mathcal{C}_k$, $\mathtt{f}_k \in |C_k|^{\sigma[\overline{K} \Leftarrow \overline{\Psi}][X \Leftarrow \Psi][x \Leftarrow \mathtt{u}]}$. Donc, on obtient $(\mathtt{t}\ \mathtt{f_1}, \dots, \mathtt{f_n}) \in \Psi(u)$. D'où il existe un entier i et des termes $\mathtt{t}_1, \dots, \mathtt{t}_{n_i}, \mathtt{u}_1, \dots, \mathtt{u}_{n_i}$ tel que :

- $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_{n_i})$
- $(\mathtt{t}\ \mathtt{f_1} \dots \mathtt{f_n}) \sim_\beta (\mathtt{f_i}\ \mathtt{t_1} \dots \mathtt{t_{n_i}})$ ($\mathtt{f}_1, \dots, \mathtt{f}_n$ n'étant pas libres dans $\mathtt{t}_1, \dots, \mathtt{t}_{n_i}$)
- Pour tout j, si $T_{i,j} = D_k$ (type déjà défini), $\mathtt{t}_j \in |D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$
- Pour tout j, si $T_{i,j} = D$ (paramètre inductif), $\mathtt{t}_j \in \Psi_{r_{i,j}}(\mathtt{u}_j)$

Or, $(\mathtt{t}\ \mathtt{f_1} \dots \mathtt{f_n}) \sim_\beta (\mathtt{f_i}\ \mathtt{t_1} \dots \mathtt{t_{n_i}})$ et $\mathtt{f}_1, \dots, \mathtt{f}_n$ non libres dans $\mathtt{t}_1, \dots, \mathtt{t}_{n_i}$ impliquent $\mathtt{t} \sim_\beta \lambda \mathtt{f_1} \dots \mathtt{f_n}\,(\mathtt{f_i}\ \mathtt{t_1} \dots \mathtt{t_{n_i}})$. Donc les conditions ci-dessus implique bien $\mathtt{t} \in \Delta_r(\Psi_1, \dots, \Psi_r)(\mathtt{u})$.

Il reste à montrer qui si

$$\mathtt{t} \in F_{j+1}[K_1, \dots, K_{j+1}, x]^{\sigma\left[\overline{K} \Leftarrow \overline{\Psi}\right][x \Leftarrow \mathtt{u}]} \text{ ssi } \mathtt{t} \in \Delta_{j+1}\left(\Psi_1, \dots, \Psi_{j+1}\right)(\mathtt{u})$$

Alors la même propriété est vrai pour j.

Or, ceci est immédiat car la formule F_j est un point fixe, et car la définition de Δ_j à partir de Δ_{j+1} correspond exactement à l'interprétation de ce point fixe. ■

Corollaire 5.11. *Pour $0 \leq j < r$, on peut aussi définir Δ_j à partir de Δ_{j+1}, par induction sur les ordinaux de la manière suivante :*

- *Si* $\epsilon_j = -$*, on pose pour tout* $\mathtt{u} \in \Lambda$ *:*

$$\begin{aligned}\Delta_j^0(\Psi_1,\ldots,\Psi_j)(\mathtt{u}) &= \emptyset \\ \Delta_j^\alpha(\Psi_1,\ldots,\Psi_j)(\mathtt{u}) &= \bigcup_{\gamma<\alpha} \Delta_{j+1}\left(\Psi_1,\ldots,\Psi_j,\Delta_j^\gamma\left(\Psi_1,\ldots,\Psi_j\right)\right)(\mathtt{u})\end{aligned}$$

  Cette suite est croissante, donc constante à partir d'un ordinal α_0 *et* $\Delta_j = \Delta_j^{\alpha_0}$*.*

- *Si* $\epsilon_j = +$*, on pose pour tout* $\mathtt{u} \in \Lambda$ *:*

$$\begin{aligned}\Delta_j^0(\Psi_1,\ldots,\Psi_j)(\mathtt{u}) &= \Lambda \\ \Delta_j^\alpha(\Psi_1,\ldots,\Psi_j)(\mathtt{u}) &= \bigcap_{\gamma<\alpha} \Delta_{j+1}\left(\Psi_1,\ldots,\Psi_j,\Delta_j^\gamma\left(\Psi_1,\ldots,\Psi_j\right)\right)(\mathtt{u})\end{aligned}$$

  Cette suite est décroissante, donc constante à partir d'un ordinal α_0 *et* $\Delta_j = \Delta_j^{\alpha_0}$*.*

Démonstration : cette définition correspond à la définition par induction des points fixes. ■

Lemme 5.12. *Pour tout terme* $\mathtt{u}$ *on a :*

$$\Delta_0(\mathtt{u}) \neq \emptyset \text{ ssi } \mathtt{u} \in \Theta_0$$

Démonstration : On montre ce résultat par induction sur la construction de Δ_0 et Θ_0. On commence par prouver que si l'on prend $\Psi_1,\ldots,\Psi_r \in \Lambda \to \mathcal{P}(\Lambda)$ et $\Phi_1,\ldots,\Phi_r \in \mathcal{P}(\Lambda)$ vérifiant pour tout i :

$$\Psi_i(\mathtt{u}) \neq \emptyset \text{ ssi } \mathtt{u} \in \Phi_i \qquad (i)$$

alors on a :

$$\Delta_r(\Psi_1,\ldots,\Psi_r)(\mathtt{u}) \neq \emptyset \text{ ssi } \mathtt{u} \in \Theta_r(\Phi_1,\ldots,\Phi_r)$$

Ceci découle de la définition de Δ_r qui implique : $\Delta_r(\Psi_1,\ldots,\Psi_r)(\mathtt{u}) \neq \emptyset$ si et seulement si il existe un entier i et n_i terme $\mathtt{u}_1,\ldots,\mathtt{u}_{n_i}$ tel que :

- $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1,\ldots,\mathtt{u}_{n_i})$
- Pour tout j, si $T_{i,j} = D_k$, $|D_k(x)|^{\sigma[x\Leftarrow \mathtt{u}_j]} \neq \emptyset$
- Pour tout j, si $T_{i,j} = D$, $\Psi_{r_{i,j}}(\mathtt{u}_j) \neq \emptyset$

Or, D_k étant un type de données et vues les conditions sur les Ψ_i et les Φ_i, ceci est équivalent à il existe un entier i et n_i termes $\mathtt{u}_1,\ldots,\mathtt{u}_{n_i}$ tel que

- $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1,\ldots,\mathtt{u}_{n_i})$
- Pour tout j, si $T_{i,j} = D_k$, $\mathtt{u}_i \in D_k^{\mathcal{M}}$
- Pour tout j, si $T_{i,j} = D$, $\mathtt{u}_j \in \Phi(\mathtt{u}_j)$

Ce qui correspond exactement à la définition de $\mathtt{u} \in \Theta_r$.

Reste à démontrer par induction que si sous les conditions (i) on a :

$$\Delta_{j+1}\left(\Psi_1,\ldots,\Psi_{j+1}\right)(\mathtt{u}) \neq \emptyset \text{ ssi } \mathtt{u} \in \Theta_{j+1}\left(\Phi_1,\ldots,\Phi_{j+1}\right)$$

alors la même propriété est vrai pour j. Pour le démontrer, on utilise les corollaires 5.7 et 5.11, qui affirment que :

- Si $\epsilon_j = -$ (resp. $+$), on peut poser pour tout $\mathtt{u} \in \Lambda$:

$$\begin{array}{rcl} \Delta_j^0(\Psi_1, \ldots, \Psi_j)(\mathtt{u}) & = & \emptyset(\text{resp.}\Lambda) \\ \Delta_j^\alpha(\Psi_1, \ldots, \Psi_j)(\mathtt{u}) & = & \bigcup_{\gamma<\alpha} \Delta_{j+1}\left(\Psi_1, \ldots, \Psi_j, \Delta_j^\gamma\left(\Psi_1, \ldots, \Psi_j\right)\right)(\mathtt{u})\,(\text{resp.}\cap) \end{array}$$

  Cette suite est alors croissante (resp. décroissante), donc constante à partir d'un ordinal α_0 et $\Delta_j = \Delta_j^{\alpha_0}$.

- Si $\epsilon_j = -$ (resp. $+$), on peut poser :

$$\begin{array}{rcl} \Theta_j^0(\Phi_1, \ldots, \Phi_j) & = & \emptyset(\text{resp.}\Lambda) \\ \Theta_j^\alpha(\Phi_1, \ldots, \Phi_j) & = & \bigcup_{\gamma<\alpha} \Theta_{j+1}\left(\Phi_1, \ldots, \Phi_j, \Theta_j^\gamma\left(\Phi_1, \ldots, \Phi_j\right)\right)\,(\text{resp.}\cap) \end{array}$$

  Cette suite est alors croissante (resp. décroissante), donc constante à partir d'un ordinal α_0' et $\Theta_j = \Theta_j^{\alpha_0'}$.

Il suffit alors de montrer par induction, que pour tout ordinal α on a

$$\Delta_j^\alpha\left(\Psi_1, \ldots, \Psi_{j+1}\right)(\mathtt{u}) \neq \emptyset \text{ ssi } \mathtt{u} \in \Theta_j^\alpha\left(\Phi_1, \ldots, \Phi_{j+1}\right)$$

Or, le cas 0 est trivial et le cas d'induction découle de l'hypothèse sur Δ_{j+1} et Θ_{j+1}. ■

Lemme 5.13. *Étant donnée $\Psi_1, \ldots, \Psi_k \in \Lambda \to \mathcal{P}(\Lambda)$, pour tous termes $\mathtt{t}$ et $\mathtt{u}$, on a $\mathtt{t} \in \Delta_k(\Psi_1, \ldots, \Psi_k)(\mathtt{u})$ si et seulement s'il existe un entier i et des termes $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}, \mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$ tel que :*

- $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_{n_i})$
- $\mathtt{t} \sim_\beta \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_n(\mathtt{f}_i\ \mathtt{t}_1 \ldots \mathtt{t}_{n_i})$ *($\mathtt{f}_1, \ldots, \mathtt{f}_n$ n'étant pas libre dans $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$)*
- *Pour tout j, si $T_{i,j} = D_k$ (type déjà défini), $\mathtt{t}_j \in |D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$*
- *Pour tout j, si $T_{i,j} = D$ et $r_{i,j} \leq k$ (paramètre inductif), $\mathtt{t}_j \in \Psi_{r_{i,j}}(\mathtt{u}_j)$*
- *Pour tout j, si $T_{i,j} = D$ et $r_{i,j} > k$ (paramètre inductif), $\mathtt{t}_j \in \Delta_k(\Psi_1, \ldots, \Psi_k)(\mathtt{u}_j)$*

Démonstration : Pour $k = r$, cette propriété correspond à la définition de Δ_r puisque on n'a jamais $r_{i,j} > k$. Supposons maintenant que cette propriété est vrai pour Δ_{k+1} et montrons qu'elle est vrai pour Δ_k. Δ_k étant défini comme un point fixe de Δ_{k+1} (aussi bien pour $\epsilon_k = -$ que pour $\epsilon_k = +$.), on a $\Delta_k(\Psi_1, \ldots, \Psi_k) = \Delta_{k+1}(\Psi_1, \ldots, \Psi_k, \Delta_k(\Psi_1, \ldots, \Psi_k))$. Donc, en appliquant l'hypothèse sur Δ_{k+1}, on trouve $\mathtt{t} \in \Delta_k(\Psi_1, \ldots, \Psi_k)$ si et seulement s'il existe un entier i et des termes $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}, \mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$ tel que :

- $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_{n_i})$
- $\mathtt{t} \sim_\beta \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_n(\mathtt{f}_i\ \mathtt{t}_1 \ldots \mathtt{t}_{n_i})$ ($\mathtt{f}_1, \ldots, \mathtt{f}_n$ n'étant pas libre dans $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$)
- Pour tout j, si $T_{i,j} = D_k$ (type déjà défini), $\mathtt{t}_j \in |D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$
- Pour tout j, si $T_{i,j} = D$ et $r_{i,j} \leq k$ (paramètre inductif), $\mathtt{t}_j \in \Psi_{r_{i,j}}(\mathtt{u}_j)$
- Pour tout j, si $T_{i,j} = D$ et $r_{i,j} = k+1$, $\mathtt{t}_j \in \Delta_k(\Psi_1, \ldots, \Psi_k)(\mathtt{u}_j)$

- Pour tout j, si $T_{i,j} = D$ et $r_{i,j} > k+1$, $\mathtt{t}_j \in \Delta_{k+1}(\Psi_1, \ldots, \Psi_k, \Delta_k(\Psi_1, \ldots, \Psi_k))(\mathtt{u}_j)$

Ce qui donne bien le résultat voulu. ■

Lemme 5.14. *Si le modèle $\mathcal{M}$ est intentionnel pour les types $D_1, \ldots, D_m$ intervenant dans la définition de D, et, si $\mathcal{M}$ est syntaxiquement intentionel pour le type D, alors :*

Pour tous termes $\mathtt{t}$ *et* $\mathtt{u}, \mathtt{t} \in \Delta_0(\mathtt{u})$ *entraine* $\mathtt{t} \sim_\omega \mathtt{u}$ *et* $\Delta_0(\mathtt{u}) \neq \emptyset$

Démonstration : Supposons le modèle $\mathcal{M}$ intentionnel pour les types $D_1, \ldots, D_m$ et syntaxiquement intentionnel pour le type D. Montrons par récurrence que pour tout entier n et pour tous termes $\mathtt{t}$ et $\mathtt{u}$, $\mathtt{t} \in \Delta_0(\mathtt{u})$ entraîne $\mathtt{t} \sim_{\omega_n} \mathtt{u}$ (ce qui correspond à la définition de $\mathtt{t} \sim_\omega \mathtt{u}$). Le cas $n = 0$ découle du fait que la ω_0-équivalence est l'équivalence triviale. Supposons maintenant que le cas n est vérifié. En appliquant le lemme 5.13 à Δ_0, on trouve $\mathtt{t} \in \Delta_0(\mathtt{u})$ si et seulement s'il existe un entier i et des termes $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}, \mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$ tel que :

- $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_{n_i})$
- $\mathtt{t} \sim_\beta \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_\mathtt{n}(\mathtt{f}_\mathtt{i}\ \mathtt{t}_1 \ldots \mathtt{t}_{\mathtt{n_i}})$ ($\mathtt{f}_1, \ldots, \mathtt{f}_n$ n'étant pas libre dans $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$)
- Pour tout j, si $T_{i,j} = D_k$, $\mathtt{t}_j \in |D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$
- Pour tout j, si $T_{i,j} = D$, $\mathtt{t}_j \in \Delta_0(\mathtt{u}_j)$

Les D_k étant des types de données (hypothèse que $\mathcal{M}$ est intentionnel pour ces types) et par hypothèse d'induction, on trouve $\mathtt{t}_j \sim_{\omega_n} \mathtt{u}_j$ pour tout j. Or, par hypothèse, on a $\mathtt{u}_j \in T_{i,j}^{\mathcal{M}}$ (en utilisant le lemme 5.12 si $T_{i,j} = D$ ou bien le fait que $\mathcal{M}$ soit intentionnel pour D_k si $T_{i,j} = D_k$). Donc, $\mathcal{M}$ étant syntaxiquement intentionnel pour le type D, on trouve $\mathtt{u} \sim_\omega \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_\mathtt{n}(\mathtt{f}_\mathtt{i}\ \mathtt{u}_1 \ldots \mathtt{u}_{\mathtt{n_i}})$. Comme on a déjà $\mathtt{t} \sim_\beta \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_\mathtt{n}(\mathtt{f}_\mathtt{i}\ \mathtt{t}_1 \ldots \mathtt{t}_{\mathtt{n_i}})$, on trouve bien $\mathtt{t} \sim_{\omega_{n+1}} \mathtt{u}$. ■

Lemme 5.15. *Pour tout terme* $\mathtt{u}$, $\Delta_0(\mathtt{u})$ *est clos pour la ω-équivalence.*

Démonstration : Pour démontrer cela, on va montrer pour tout entier k la propriété $P(k)$ suivante : Pour tous $\Psi_1, \ldots, \Psi_k \in \Lambda \to \mathcal{P}(\Lambda)$ si $\Psi_j(\mathtt{u})$ est clos pour la ω-équivalence pour tout j et tout terme $\mathtt{u}$, alors $\Delta_k(\Psi_1, \ldots, \Psi_k)(\mathtt{u})$ est clos pour la ω-équivalence pour tout terme $\mathtt{u}$.

On veut donc démontrer $P(0)$. Montrons d'abord $P(r)$. Soit $\Psi_1, \ldots, \Psi_r \in \Lambda \to \mathcal{P}(\Lambda)$ vérifiant $\Psi_j(u)$ clos pour la ω-équivalence pour tout j et tout terme $\mathtt{u}$. Soit trois termes $\mathtt{t}, \mathtt{t}'$ et $\mathtt{u}$ vérifiant $\mathtt{t} \in \Delta_r(\Psi_1, \ldots, \Psi_r)(\mathtt{u})$ et $\mathtt{t} \sim_\omega \mathtt{t}'$. Par définition, il existe un entier i et des termes $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}, \mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$ tel que :

- $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_{n_i})$
- $\mathtt{t} \sim_\beta \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_\mathtt{n}(\mathtt{f}_\mathtt{i}\ \mathtt{t}_1 \ldots \mathtt{t}_{\mathtt{n_i}})$ ($\mathtt{f}_1, \ldots, \mathtt{f}_n$ n'étant pas libre dans $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$)
- Pour tout j, si $T_{i,j} = D_k$ (type déjà défini), $\mathtt{t}_j \in |D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$
- Pour tout j, si $T_{i,j} = D$ (paramètre inductif), $\mathtt{t}_j \in \Psi_{r_{i,j}}(\mathtt{u}_j)$

De plus, par définition de la ω-équivalence, $\mathtt{t}' = \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_\mathtt{n}(\mathtt{f}_\mathtt{i}\ \mathtt{t}'_1 \ldots \mathtt{t}'_{\mathtt{n_i}})$ avec pour tout j, $\mathtt{t}'_j \sim_\omega \mathtt{t}_j$. Or $|D_k(x)|^{\sigma[x \Leftarrow \mathtt{u}_j]}$ ainsi que $\Psi_{r_{i,j}}(\mathtt{u}_j)$ sont clos pour la ω-équivalence. Donc on a bien $\mathtt{t}' \in \Delta_r(\Psi_1, \ldots, \Psi_r)(\mathtt{u})$

Il suffit de montrer maintenant que $P(k+1)$ implique $P(k)$. Supposons donc $P(k+1)$ et choisissons $\Psi_1, \ldots, \Psi_k \in \Lambda \to \mathcal{P}(\Lambda)$ vérifiant $\Psi_j(\mathtt{u})$ clos pour la ω-équivalence pour tout j et tout terme $\mathtt{u}$. En appliquant le corollaire 5.11, on trouve :

- Si $\epsilon_k = -$, on pose pour tout $\mathtt{u} \in \Lambda$:

$$\begin{array}{rcl} \Delta_k^0(\Psi_1, \ldots, \Psi_k)(\mathtt{u}) & = & \emptyset \\ \Delta_k^\alpha(\Psi_1, \ldots, \Psi_k)(\mathtt{u}) & = & \bigcup\limits_{\gamma<\alpha} \Delta_{k+1}\left(\Psi_1, \ldots, \Psi_k, \Delta_k^\gamma(\Psi_1, \ldots, \Psi_k)\right)(\mathtt{u}) \end{array}$$

  Cette suite est croissante donc constante à partir d'un ordinal α_0, et $\Delta_k = \Delta_k^{\alpha_0}$.

- Si $\epsilon_k = +$, on pose pour tout $\mathtt{u} \in \Lambda$:

$$\begin{array}{rcl} \Delta_k^0(\Psi_1, \ldots, \Psi_k)(\mathtt{u}) & = & \Lambda \\ \Delta_k^\alpha(\Psi_1, \ldots, \Psi_k)(\mathtt{u}) & = & \bigcap\limits_{\gamma<\alpha} \Delta_{k+1}\left(\Psi_1, \ldots, \Psi_k, \Delta_k^\gamma(\Psi_1, \ldots, \Psi_k)\right)(\mathtt{u}) \end{array}$$

  Cette suite est décroissante donc constante à partir d'un ordinal α_0, et $\Delta_k = \Delta_k^{\alpha_0}$.

Or $\emptyset$ et Λ sont clos pour la ω-équivalence, donc l'hypothèse sur les Ψ_j et $P(k+1)$ donne $\Delta_k^\alpha(\Psi_1, \ldots, \Psi_k)(\mathtt{u})$ clos pour la ω-équivalence, pour tout ordinal α et pour tout terme $\mathtt{u}$. D'où le résultat.

■

Lemme 5.16. *Pour tous termes* $\mathtt{t}$ *et* $\mathtt{u}$*, si le modèle* $\mathcal{M}$ *est syntaxiquement intentionnel pour le type* D *et s'il est intentionnel pour les types* $D_1, \ldots, D_m$ *intervenant dans la définition de* D*, on a :*

$$\mathtt{t} \in \Delta_0(\mathtt{u}) \text{ ssi } \mathtt{t} \sim_\omega \mathtt{u} \text{ et } \Delta_0(\mathtt{u}) \neq \emptyset$$

Démonstration : le sens gauche-droite de l'équivalence découle directement du lemme 5.14. Pour le sens droite-gauche, supposons $\mathtt{t} \sim_\omega \mathtt{u}$ et $\Delta_0(\mathtt{u}) \neq \emptyset$. On trouve un terme $\mathtt{t}' \in \Delta_0(\mathtt{u})$ et par le lemme 5.14 on obtient $\mathtt{t}' \sim_\omega \mathtt{u}$ et ainsi $\mathtt{t} \sim_\omega \mathtt{t}'$. Or, par le lemme 5.15, $\Delta_0(\mathtt{u})$ est clos pour la ω-équivalence. Donc on a bien $\mathtt{t} \in \Delta_0(\mathtt{u})$. ■

Ces lemmes permettent de démontrer le théorème suivant :

Théorème 5.17. *Si le modèle* $\mathcal{M}$ *est intentionnel pour tous les types* D_k *intervenant dans la construction de* D *et s'il est syntaxiquement intentionnel pour* D*, alors il est intentionnel pour* D*, c'est à dire que pour toute interprétation* σ *coïncidant avec* $\mathcal{M}$ *sur les termes du premier ordre, on a*

$$\mathtt{t} \in \left|D(x)\right|^{\sigma[x \Leftarrow \mathtt{u}]} \text{ ssi } \mathtt{u} \in \mathcal{D}^M \text{ et } \mathtt{t} \sim_\omega \mathtt{u}$$

Démonstration : Ce théorème se démontre simplement par la suite d'équivalences obtenue par les lemmes 5.10,5.16,5.12 et la proposition 5.6 : On choisit un modèle $\mathcal{M}$ vérifiant les hypothèses du théorème et une interprétation σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. On obtient alors :

$\mathtt{t} \in \lvert D[x]\rvert^{\sigma[x \Leftarrow \mathtt{u}]}$	équivalent à	$\mathtt{t} \in \Delta_0(\mathtt{u})$	(par 5.10)
	équivalent à	$\mathtt{t} \sim_\omega \mathtt{u}$ et $\Delta_0(\mathtt{u}) \neq \emptyset$	(par 5.16)
	équivalent à	$\mathtt{t} \sim_\omega \mathtt{u}$ et $\mathtt{u} \in \Theta_0$	(par 5.12)
	équivalent à	$\mathtt{t} \sim_\omega \mathtt{u}$ et $\mathtt{u} \in D^{\mathcal{M}}$	(par 5.6)

■

Théorème 5.18. *On peut démontrer la réciproque du théorème précédent : Si le modèle $\mathcal{M}$ est intentionnel pour D ainsi que pour tous les type D_k intervenant dans la construction de D, alors il est syntaxiquement intentionnel pour D.*

Démonstration : Soit un modèle $\mathcal{M}$ intentionnel pour tous les types D_k intervenant dans la construction de D ainsi que pour D. Soit $\mathbf{c}_i$ un constructeur du type D, et $\mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$ des termes tel que $\mathtt{u}_j \in T_{i,j}^{\mathcal{M}}$ ($\mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$ sont des termes du bon type pour le constructeur $\mathbf{c}_i$). Soit $\mathtt{u} = |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_{n_i}) \in D^{\mathcal{M}}$, et $\mathtt{t} = \lambda \mathtt{f_1} \ldots \lambda \mathtt{f_n}(\mathtt{f_i}\, \mathtt{u_1}\, \ldots\, \mathtt{u_{n_i}})$. De $\mathtt{u}_j \in T_{i,j}^{\mathcal{M}}$ on déduit $\mathtt{t} \in |D[x]|^{\sigma[x \Leftarrow \mathtt{u}]}$ pour toute interprétation σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. Or $\mathcal{M}$ est intentionnel pour D, on en déduit donc $\mathtt{t} \sim_\omega \mathtt{u}$. ■

Chapitre 6

Règles dérivées et optimisations.

En utilisant notre système, on s'aperçoit rapidement que l'on a besoin d'utiliser des règles dérivées pour alléger les preuves. On remarque aussi que les programmes associés à ces règles dérivées ne sont pas ceux attendus et peuvent souvent être améliorés.

Dans ce chapitre, on va étudier un certain nombre de ces règles dérivées et pour toutes celles qui le nécessitent, on donnera un contenu algorithmique optimisé à la règle. Il faudra alors montrer que l'on préserve le lemme de réduction (lemme 2.1) et le lemme de conservation (lemme 3.14) dont découlent tous les autres résultats.

6.1 La récurrence mutuelle.

Proposition 6.1. *Soit K une variable de prédicat. Soit n formules F_i où la variable K est visiblement positive (resp. visiblement négative). Soit $\lambda\overline{x}\,G$ une expression substituable à K où K n'a que des occurrences positives. Soit Γ un contexte où K n'est pas libre.*

Si pour chaque $i \in \{1, \ldots, n\}$ on trouve un terme $\mathtt{u_i}$ tel que :

$$\Gamma \vdash \mathtt{u_i} : F_1, \ldots, F_n \to F_i[K \Leftarrow \lambda\overline{x}G]$$

Alors pour chaque i, on trouve un terme $\mathtt{t_i}$ tel que :

$$\Gamma \vdash \mathtt{t_i} : F_i\Big[K \Leftarrow \lambda\overline{x}\left(\nu K\,\lambda\overline{x}\,G\langle\overline{x}\rangle\right)\Big] \text{ (resp } \mu) \text{ .}$$

Démonstration : Afin de démontrer cette règle dérivée, notons Φ la formule suivante (où l'on a choisit les variables $\overline{y}$ non libres dans les formules $F_1, \ldots F_n$) :

$$\Phi = \forall\overline{y}\left(\forall X\left(F_1[K \Leftarrow X], \ldots, F_n[K \Leftarrow X] \to X\,(\overline{y})\right) \to K\,(\overline{y})\right)$$

On montre alors que pour tout i, on trouve un terme $\mathtt{p_i}$ tel que :

$$\vdash \mathtt{p_i} : \forall K\,(\Phi \to F_i)$$

On trouve $\mathtt{p}_i$ par induction sur la longueur de la branche droite de la formule F_i. Si K est visiblement positive dans F_i, on est dans l'un des cas suivants :

- $F_i = K(\overline{t})$. Dans ce cas, on trouve facilement $\mathtt{p_i} = \lambda \mathtt{f}(\mathtt{f}\ \lambda \overline{\mathtt{x}}\, \mathtt{x_i})$.
- $F_i = A \to F'_i$. Par hypothèse d'induction, on trouve $\mathtt{p'_i} : \forall K(\Phi \to F'_i)$ Il suffit alors de prendre $\mathtt{p_i} = \lambda \mathtt{f} \lambda \mathtt{a}(\mathtt{p'_i}\ \lambda \mathtt{c}(\mathtt{f}\ \lambda \overline{\mathtt{x}}(\mathtt{c}\ \mathtt{x_1} \ldots \mathtt{x_{i-1}}\ (\mathtt{x_i}\ \mathtt{a})\mathtt{x_{i+1}} \ldots \mathtt{x_n})))$.
- $F_i = \forall \chi F'_i$. De la même manière, l'hypothèse d'induction donne p'_i pour la formule F'_i. Il suffit alors de prendre $\mathtt{p_i} = \lambda \mathtt{f}(\mathtt{p'_i}\ \lambda \mathtt{c}(\mathtt{f}\ \lambda \overline{\mathtt{x}}(\mathtt{c}\ \mathtt{x_1} \ldots \mathtt{x_i} \ldots \mathtt{x_n})))$.

Donc, si l'on note $\mathtt{v_i} = (\mathtt{u_i}\ (\mathtt{p_1}\,\mathtt{f}) \ldots (\mathtt{p_n}\,\mathtt{f}))$, on obtient :

$$\Gamma, \mathtt{f} : \Phi \vdash \mathtt{v_i} : F_i[K \Leftarrow \lambda \overline{x} G]$$

D'où :

$$\Gamma, \mathtt{f} : \Phi, \mathtt{c} : \forall X \left(F_1[K \Leftarrow X], \ldots, F_n[K \Leftarrow X] \to X\,(\overline{y})\right) \vdash (\mathtt{c}\ \mathtt{v_1} \ldots \mathtt{v_n}) : G[\overline{x} \Leftarrow \overline{y}]$$

Ainsi :

$$\Gamma \vdash\ !\lambda \mathtt{f} \lambda \mathtt{c}\,(\mathtt{c}\ \mathtt{v_1} \ldots \mathtt{v_n}) : \Phi\Big[K \Leftarrow \lambda \overline{x} \nu K \lambda \overline{x}\, G\langle \overline{x} \rangle\Big]$$

D'où enfin :

$$\Gamma \vdash \Big(\mathtt{p_i}\ !\lambda \mathtt{f} \lambda \mathtt{c}\,(\mathtt{c}\ \mathtt{v_1} \ldots \mathtt{v_n})\Big) : F_i\Big[K \Leftarrow \lambda \overline{x} \nu K \lambda \overline{x}\, G\langle \overline{x} \rangle\Big]$$

Respectivement, si K est visiblement négative dans F_1, ..., F_n, on obtient la même preuve simplement en changeant la définition de Φ par :

$$\Phi = \forall Y \forall \overline{y} \left(\forall X \left(F_1[K \Leftarrow X], \ldots, F_n[K \Leftarrow X], X\,(\overline{y}) \to Y\right), K\,(\overline{y}) \to Y\right)$$

■

6.2 Optimisation de la récurrence mutuelle.

En regardant un peu la réduction des termes extraits de la preuve précédente, on s'aperçoit, qu'à η-équivalence prés, on a :

$$\mathtt{t_i} \rhd_\beta (\mathtt{u_i}\ \mathtt{t_1} \ldots \mathtt{t_n})$$

Donc, pour optimiser la règle précédente, il suffit de se donner des points fixes de récurrence mutuelle que l'on notera $!^{\mathtt{n}}_{\mathtt{i}}$, obéissants à la règle de réduction suivante :

$$\left(!^{\mathtt{n}}_{\mathtt{i}}\,\mathtt{u_1} \ldots \mathtt{u_n}\right) \rhd_\beta \left(\mathtt{u_i}\ \ \left(!^{\mathtt{n}}_{\mathtt{1}}\,\mathtt{u_1} \ldots \mathtt{u_n}\right) \ldots \left(!^{\mathtt{n}}_{\mathtt{n}}\,\mathtt{u_1} \ldots \mathtt{u_n}\right)\right)$$

La règle de la récurrence mutuelle peut alors s'écrire :

$$\frac{\Gamma \vdash \mathtt{u_1} : F_1, \ldots, F_n \to F_1[K \Leftarrow \lambda \overline{x} G] \quad \ldots \quad \Gamma \vdash \mathtt{u_n} : F_1, \ldots, F_n \to F_n[K \Leftarrow \lambda \overline{x} G]}{\Gamma \vdash \left(!^{\mathtt{n}}_{\mathtt{i}} \mathtt{u_1} \ldots \mathtt{u_n}\right) : F_i\Big[K \Leftarrow \lambda \overline{x}\left(\nu K \lambda \overline{x}\, G\langle \overline{x} \rangle\right)\Big] \text{ (resp. } \mu)}\ \star$$

$\star$ Avec :

- K une variable de prédicat,
- F_1, ..., F_n, n formules où la variable K est visiblement positive (resp. visiblement négative).
- $\lambda\overline{x}\,G$ une expression substituable à K où K n'a que des occurrences positives.
- Γ un contexte où K n'est pas libre.

L'extension de la preuve du lemme de réduction et du lemme de conservation ne pose aucun problème pour cette règle, toutes les preuves restent très similaires.

6.3 Règle des points fixes simultanés.

L'utilisation de la règle précédente correspond à l'utilisation d'une seule induction pour montrer plusieurs propriétés, ce qui correspond au niveau des programmes à des fonctions mutuellement récursives.

Dualement, on peut aussi introduire plusieurs point fixes en même temps. C'est à dire que l'on peut faire deux récurrences en même temps, par exemple pour prouver une propriété $P(x,y)$ à partir de $\forall y\, P(\mathbf{0},y)$ et $\forall x \forall y\, P(x,y) \to P(\mathbf{s}x,\mathbf{s}y)$. Cela correspond à la règle dérivée suivante :

Proposition 6.2. *Soit K_1, ... K_n des variables de prédicat. Soit une formule F où chaque variable K_i est ou bien visiblement négative, ou bien visiblement positive. Soit $\lambda\overline{x_1}\,G_1$, ..., $\lambda\overline{x_n}\,G_n$ des expressions respectivement substituables à K_1, ... K_n. De plus, on suppose que K_i n'a que des occurrences positives dans G_i, et aucune occurrence dans G_j si $j \neq i$. Soit Γ un contexte où K_1, ... K_n ne sont pas libres. Soit H_1, ..., H_n les formules définies par $H_i = \mu X\, \lambda\overline{x_i}\, G_i\langle\overline{x_i}\rangle$ si K_i est visiblement négative dans F, ou bien $H_i = \nu X\, \lambda\overline{x_i}\, G_i\langle\overline{x_i}\rangle$ si K_i est visiblement positive dans F.*

Si l'on trouve un terme $\mathtt{u}$ *tel que :*

$$\Gamma \vdash \mathtt{u} : F \to F[K_1 \Leftarrow \lambda\overline{x_1}\,G_1]\ldots[K_n \Leftarrow \lambda\overline{x_n}\,G_n]$$

Alors, on a :

$$\Gamma \vdash !\mathtt{u} : F[K_1 \Leftarrow \lambda\overline{x_1}\,H_1]\ldots[K_n \Leftarrow \lambda\overline{x_n}\,H_n]$$

Démonstration : Cette règle dérivée est en fait beaucoup plus facile à prouver qu'à énoncer. Notons :

$$\begin{array}{rcl} G_i' & = & G_i[K_i \Leftarrow \lambda\overline{x_i}\,H_i] \\ F' & = & F_i[K_2 \Leftarrow \lambda\overline{x_2}\,H_2]\ldots[K_n \Leftarrow \lambda\overline{x_n}\,H_n] \end{array}$$

De l'hypothèse

$$\Gamma \vdash \mathtt{u} : F \to F[K_1 \Leftarrow \lambda\overline{x_1}\,G_1]\ldots[K_n \Leftarrow \lambda\overline{x_n}\,G_n]$$

on déduit (par substitution de H_i à K_i pour $2 \leq i \leq n$) :

$$\Gamma \vdash \mathtt{u} : F' \to F[K_1 \Leftarrow \lambda\overline{x_1}\,G_1][K_2 \Leftarrow \lambda\overline{x_2}\,G_2']\ldots[K_n \Leftarrow \lambda\overline{x_n}\,G_n']$$

en utilisant $n-1$ fois la factorisation du point fixe, on obtient

$$\Gamma \vdash \mathtt{u} : F' \to F'[K_1 \Leftarrow \lambda\overline{x_1}\, G_1]$$

Ce qui donne alors le résultat voulu en appliquant la règle du point fixe (sur la variable K_1). ■

Cette règle ne nécessite pas d'optimisation et on déduit donc directement de la proposition précédente la règle suivante :

$$\frac{\Gamma \vdash \mathtt{u} : F \to F[K_1 \Leftarrow \lambda\overline{x_1}\, G_1] \ldots [K_n \Leftarrow \lambda\overline{x_n}\, G_n]}{\Gamma \vdash !\mathtt{u} : F[K_1 \Leftarrow \lambda\overline{x_1}\, H_1] \ldots [K_n \Leftarrow \lambda\overline{x_n}\, H_n]}\ \star$$

$\star$ Avec :

- $K_1, \ldots K_n$ des variables de prédicat.
- F une formule où $K_1, \ldots, K_n$ sont visiblement négatives ou visiblement positives.
- $\lambda\overline{x_1}\, G_1, \ldots, \lambda\overline{x_n}\, G_n$ des expressions respectivement substituables à $K_1, \ldots K_n$.
- K_i n'a que des occurrences positives dans G_i et aucune occurrence dans G_j si $j \neq i$.
- Γ un contexte où $K_1, \ldots K_n$ ne sont pas libres.
- $H_1, \ldots, H_n$, n formules définies par $H_i = \mu X\, \lambda\overline{x_i}\, G_i\langle\overline{x_i}\rangle$ si K_i est visiblement négative dans F ou $H_i = \nu X\, \lambda\overline{x_i}\, G_i\langle\overline{x_i}\rangle$ si K_i est visiblement positive dans F.

6.4 Plus grand point fixe avec "inclusion".

On déduit de la règle de la récurrence mutuelle la règle dérivée suivante :

Proposition 6.3. *Soit K une variable de prédicat d'arité n. Soit une formule F où la variable K est visiblement positive (resp. négative). Soit $\lambda\overline{x}G$ une expression substituable à K où K n'a que des occurrences positives. Et soit Γ un contexte où K n'est pas libre.*

Si l'on prouve :

$$\Gamma \vdash \mathtt{u} : F, \forall\overline{x}\left(\nu K\, \lambda\overline{x}\, G\langle\overline{x}\rangle \to K\,(\overline{x})\right) \to F[K \Leftarrow \lambda\overline{x}G] \text{ (resp. } \forall\overline{x}(K\,(\overline{x}) \to \mu K\, \lambda\overline{x}\, G\langle\overline{x}\rangle)\text{)}$$

Alors on trouve un terme $\mathtt{t}$ *tel que :*

$$\Gamma \vdash \mathtt{t} : F\left[K \Leftarrow \lambda\overline{x}\left(\nu K\, \lambda\overline{x}\, G\langle\overline{x}\rangle\right)\right] \text{ (resp. } \mu\text{)}$$

Démonstration : Posons $F_1 = F$ et $F_2 = \forall\overline{x}(\nu K\, \lambda\overline{x}\, G\langle\overline{x}\rangle \to K(\overline{x}))$. On a par hypothèse (en prenant $\mathtt{u}_1 = \mathtt{u}$)

$$\Gamma \vdash \mathtt{u}_1 : F_1, F_2 \to F_1[K \Leftarrow \lambda\overline{x}G]$$

Pour appliquer la règle de la récurrence mutuelle, il suffit de trouver un terme $\mathtt{u}_2$ tel que :

$$\Gamma \vdash \mathtt{u}_2 : F_1, F_2 \to \forall\overline{x}\left(\nu K\, \lambda\overline{x}\, G\langle\overline{x}\rangle \to G\right)$$

Or, K n'ayant que des occurrences positives dans G, si l'on prouve $\Gamma \vdash \mathtt{u} : A \to B$, on trouve $\mathtt{v}$ tel que $\Gamma \vdash \mathtt{v} : G[K \Leftarrow \lambda\overline{x}A] \to G[K \Leftarrow \lambda\overline{x}B]$ (la preuve de cette règle dérivée classique est laissée au soin du lecteur). On a (en renommant éventuellement les variables $\overline{x}$ si elles sont libres dans Γ ou F_1) :

$$\Gamma, \mathtt{x} : F_1, \mathtt{y} : F_2 \vdash \mathtt{y} : \nu K \lambda\overline{x}\, G\langle\overline{x}\rangle \to K\,(\overline{x})\,.$$

Donc, on trouve $\mathtt{v}$ tel que :

$$\Gamma, \mathtt{x} : F_1, \mathtt{y} : F_2 \vdash \mathtt{v} : G\Big[K \Leftarrow \lambda\overline{x}\nu K \lambda\overline{x}\, G\langle\overline{x}\rangle\Big] \to G.$$

Puis, par la règle de factorisation du point fixe et par deux introductions d'implication, on trouve :

$$\Gamma \vdash \lambda\mathtt{x}\lambda\mathtt{y}\,\mathtt{v} : F_1, F_2 \to \forall\overline{x}\left(\nu K \lambda\overline{x}\, G\langle\overline{x}\rangle \to G\right).$$

■

Dans le cas du plus petit point fixe, la preuve est similaire.

6.5 Optimisation des règles de points fixes avec inclusion.

Cette règle fait encore intervenir une η-équivalence : le terme $\lambda\mathtt{y}\,\mathtt{v}$ est un η-équivalent de l'identité. Le terme extrait au total de la règle est $(!^2_1\mathtt{u}_1\lambda\mathtt{x}\lambda\mathtt{y}\,\mathtt{v})$.

Pour optimiser cette règle et se débarrasser de cette η-équivalence inutile, il suffit d'introduire une notion d'inclusion et de l'utiliser pour remplacer l'hypothèse $\forall\overline{x}(\nu K \lambda\overline{x}\, G\langle\overline{x}\rangle \to K(\overline{x}))$, ce qui permet, en fait, de considérer que $\lambda\mathtt{y}\,\mathtt{y}$ est de ce type.

Pour utiliser l'inclusion, on doit étendre un peu le système :

6.5.1 Extension du système avec l'inclusion.

On autorise l'utilisation d'inclusions de la forme $\forall\overline{x}(A \subseteq B)$ dans un contexte. Aucun terme du λ-calcul ne sera associé à ces inclusions. En pratique, on utilisera ces expressions que lorsque l'une des deux formules A ou B sera atomique.

Les trois règles manipulant ces expressions sont :

- Règle de l'inclusion ($\subseteq$).

$$\frac{\forall\overline{x}\,(A \subseteq B)\,, \Gamma \vdash \mathtt{t} : A[\overline{x}/\overline{t}]}{\forall\overline{x}\,(A \subseteq B)\,, \Gamma \vdash \mathtt{t} : B[\overline{x}/\overline{t}]}\ \subseteq$$

- règle du plus petit point fixe avec inclusion ($\mu_i{}'$)

$$\frac{\forall\overline{y}\left(X\,(\overline{y}) \subseteq \mu X \lambda\overline{x}\, G\langle\overline{y}\rangle\right), \Gamma \vdash \mathtt{t} : F \to \left(F[X/\lambda\overline{x}\, G]\right)}{\Gamma \vdash !\mathtt{t} : F[X/\lambda\overline{y}\left(\mu X \lambda\overline{x}\, G\langle\overline{y}\rangle\right)]}\ \mu_i{}'$$

- règle du plus grand point fixe avec inclusion ($\nu_i{}'$)

$$\frac{\forall \overline{y}\left(\nu X\,\lambda \overline{x}\,G\langle \overline{y}\rangle \subseteq X\,(\overline{y})\right), \Gamma \vdash \mathtt{t} : F \to \left(F[X/\lambda \overline{x}\,G]\right)}{\Gamma \vdash\, !\mathtt{t} : F[X/\lambda \overline{y}\left(\,\nu X\,\lambda \overline{x}\,G\langle \overline{y}\rangle\right)]}\ \nu_i{}'$$

Les deux dernières règles étant soumises aux mêmes contraintes que μ_i et ν_i.

6.5.2 Justifications des règles.

Pour étendre le lemme de conservation du chapitre 3, il suffit de considérer uniquement les interprétations σ qui vérifient les inclusions présentes dans le contexte. C'est à dire que si $\forall \overline{x}(A \subseteq B)$ est présente de le contexte, on ne considérera que les interprétations σ qui vérifient pour tous termes $\overline{\mathtt{t}} \in \Lambda^n$

$$|A|^{\sigma[\overline{x}/\overline{\mathtt{t}}]} \subseteq |B|^{\sigma[\overline{x}/\overline{\mathtt{t}}]}$$

Ceci permet de justifier la règle $\subseteq$ (c'est à dire de prouver le cas de $\subseteq$ dans la preuve du lemme, par induction sur la dérivation du séquent). Pour justifier $\mu_i{}'$ et $\nu_i{}'$, Il n'y a en fait pas de modification à apporter à la preuve. En effet, le point fixe est construit par induction ordinal. Or, il est immédiat que tout au long de cette construction l'inclusion hypothèse reste vérifiée. (par exemple pour le plus petit point fixe, on part de l'ensemble vide, et on approche le point fixe en croissant, donc on vérifie toujours l'inclusion).

Une seconde chose à vérifier est le lemme de réduction, et en tout premier lieu la réduction du redex. Il suffit de remarquer qu'aucune règle d'inclusion ne peut séparer l'introduction et l'élimination de l'implication. En effet, le morceau de preuve séparant ces deux règles commence par une règle dont la formule principale contient une implication et se termine par une conclusion qui contient une implication. Or, une règle d'inclusion ferait nécessairement apparaître une formule atomique, et l'on ne pourrait jamais faire réapparaître une implication. On ne peut effectivement pas appliquer deux fois de suite la règle d'inclusion car une variable de prédicat ne peut apparaître qu'une fois dans une inclusion (à cause de la restriction sur le contexte dans les règles de points fixes).

Le second point à vérifier est la réduction du point fixe. La preuve est la même à un détail près : il faut remarquer que si l'on a une preuve de :

$$\Gamma, \forall \overline{x}\left(X\,(\overline{x}) \subseteq \mu X\,\lambda \overline{x}\,G\langle \overline{x}\rangle\right) \vdash \mathtt{u} : F \to F[X \Leftarrow \lambda \overline{x}\,G],$$

avec X non libre dans Γ n'ayant pas d'occurrence négative dans G, et étant visiblement négative (resp. visiblement positive) dans F. Alors on a une preuve de même hauteur de :

$$\Gamma \vdash \mathtt{u} : F[X \Leftarrow \lambda \overline{x} G_0] \to F[X \Leftarrow \lambda \overline{x} G_1]$$

où $G_0 = \mu X\,\lambda \overline{x} G\langle \overline{x}\rangle$ et $G_1 = G[X \Leftarrow \lambda \overline{x} G_0]$. Pour cela, il suffit d'utiliser les mêmes règles, dans le même ordre sauf pour les inclusions, qui sont remplacées par des développements du point fixe.

6.6 L'ultime règle.

On peut aussi utiliser l'inclusion avec les deux règles dérivées précédentes. De plus on peut fusionner ces deux règles pour n'en faire qu'un seule :

$$\frac{\begin{array}{c}\Gamma, I_1, \ldots, I_n \vdash \mathtt{u_1} : F_1, \ldots, F_p \to F_1[K_1 \Leftarrow \lambda\overline{x_1}\, G_1] \ldots [K_n \Leftarrow \lambda\overline{x_n}\, G_n] \\ \vdots \\ \Gamma, I_1, \ldots, I_n \vdash \mathtt{u_p} : F_1, \ldots, F_p \to F_p[K_1 \Leftarrow \lambda\overline{x_1}\, G_1] \ldots [K_n \Leftarrow \lambda\overline{x_n}\, G_n]\end{array}}{\Gamma \vdash \left(!^{\mathtt{p}}_{\mathtt{i}}\, \mathtt{u_1} \ldots \mathtt{u_p}\right) : F_i[K_1 \Leftarrow \lambda\overline{x_1}\, H_1] \ldots [K_n \Leftarrow \lambda\overline{x_n}\, H_n]}\ \star$$

$\star$ Avec :

- $K_1, \ldots, K_n$ des variables de prédicat.
- Pour tout $j \in \{1, \ldots, n\}$, on choisit $\epsilon_j \in \{+, -\}$.
- $F_1, \ldots, F_p$ des formules avec pour tout i au moins une des variable K_j libre dans F_i.
- Si $\epsilon_j = +$, alors pour tout i, K_j est visiblement positive ou sans occurrence dans F_i.
- Si $\epsilon_j = -$, alors pour tout i, K_j est visiblement négative ou sans occurrence dans F_i.
- $\lambda\overline{x_1}\, G_1, \ldots, \lambda\overline{x_n}\, G_n$ des expressions respectivement substituables à $K_1, \ldots K_n$.
- K_j n'a que des occurrences positives dans G_j, et aucune occurrence dans G_k si $j \neq k$.
- Γ un contexte où $K_1, \ldots K_n$ ne sont pas libres.
- $H_1, \ldots, H_n$, n formules définies par $H_j = \mu K_j \lambda\overline{x_j}\, G_j\langle\overline{x_j}\rangle$ (resp. ν) si $\epsilon_j = -$ (resp. $+$).
- $I_1, \ldots, I_n$, n inclusions définies par $I_j = \forall\overline{x_j}(K_j(\overline{x_j}) \subseteq H_j)$ si $\epsilon_j = -$ ou $I_j = \forall\overline{x_j}(H_j \subseteq K_j(\overline{x_j}))$ si $\epsilon_j = +$.

Chapitre 7

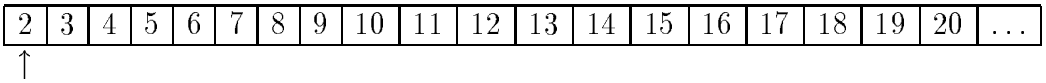
Applications.

7.1 Le crible d'Eratosthène.

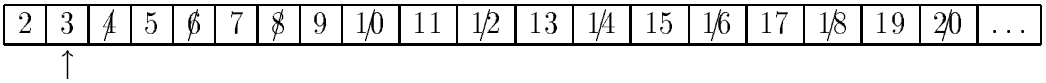
7.1.1 Définition de l'algorithme.

Dans sa version la plus dépouillée, le crible d'Eratosthène utilise un tableau infini représentant tous les entiers naturels supérieurs ou égaux à deux. On raye alors tous les entiers non premiers de la manière suivante : On démarre avec un *pointeur* sur le premier entier de la liste (2), et on raye tout ses multiples. Puis, on avance le *pointeur* jusqu'au premier entier non rayé, dont on raye tous les multiples, etc . En itérant ce processus à l'infini, les entiers qui ne sont pas rayés, sont tous les nombres premiers.

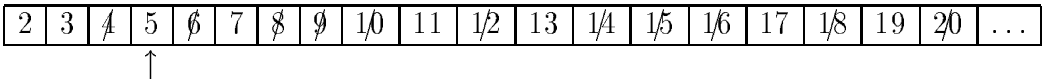
On part de la liste des entiers supérieurs à 2 :



On raye les multiples de 2 :



On raye les multiples de 3 :



On raye les multiples de 5 : . . .

7.1.2 Programmation du crible.

Pour programmer le crible d'Eratosthène, on va utiliser un stream de booléens, $\{s_n\}_{n\in\mathbb{N}}$, indiquant si le n-ième entier (c'est à dire $n+2$) a été rayé on non. On part donc du stream contenant uniquement le booléen vrai, et on termine avec un stream vérifiant $s_n = true$ si et seulement si

$n+2$ est premier, en appliquant l'algorithme précédent.

Les formules définissant les streams de booléens et les entiers sont (cf chapitre 5) :

$$\begin{array}{lcl} N[x] & = & \mu K \lambda x \forall X (X0, \forall y (Ky \to X\mathbf{s}y) \to Xx)\langle x\rangle \\ B[x] & = & \nu K \lambda x \forall X (\forall y (Ky \to X\mathbf{s}_0 y), \forall y (Ky \to X\mathbf{s}_1 y) \to Xx)\langle x\rangle \end{array}$$

On commence par programmer le stream initial $\mathbf{s_i}$ ne contenant que le booléen vrai. L'équation de $\mathbf{s_i}$ est :

$$\mathbf{s_i} = \mathbf{s}_1 (\mathbf{s_i})$$

Et par une preuve similaire à celle du stream des entiers supérieurs ou égaux à n (cf chapitre 2), on trouve :

$$\vdash\ !\lambda \mathtt{r} \lambda \mathtt{f} \lambda \mathtt{g}\ (\mathtt{g}\ \mathtt{r}) : B[\mathbf{s_i}]$$

On va alors écrire la fonction qui prend un entier n et un stream de booléens s et qui passe à faux un booléen sur n. C'est cette fonction qui va permettre de rayer les multiples de l'entier n. Pour cela, on va utiliser une fonction intermédiaire à trois variables, et ces deux fonctions obéiront aux équations suivantes :

$$\begin{array}{lcl} \mathbf{ray}_r(\mathbf{s}_0(s), n, 0) & = & \mathbf{s}_0(\mathbf{ray}_r(s, n, \mathbf{pred}(n))) \\ \mathbf{ray}_r(\mathbf{s}_0(s), n, \mathbf{s}m) & = & \mathbf{s}_0(\mathbf{ray}_r(s, n, m)) \\ \mathbf{ray}_r(\mathbf{s}_1(s), n, 0) & = & \mathbf{s}_0(\mathbf{ray}_r(s, n, \mathbf{pred}(n))) \\ \mathbf{ray}_r(\mathbf{s}_1(s), n, \mathbf{s}m) & = & \mathbf{s}_1(\mathbf{ray}_r(s, n, m)) \\ \mathbf{ray}(s, n) & = & \mathbf{ray}_r(s, n, \mathbf{pred}(n)) \end{array}$$

on suppose déjà extrait un programme pour la fonction **pred** calculant le prédécesseur d'un entier. On extrait alors un programme pour la fonction $\mathbf{ray}_r$ d'une preuve de :

$$\forall s \forall x \forall y \left(B[s], N[x], N[y] \to B[\mathbf{ray}_r\ (s, x, y)] \right)$$

Pour cela, on applique la règle du plus grand point fixe et quelques introductions d'implication, et sous les hypothèses suivantes :

$$\begin{array}{lcl} \mathtt{r} & : & \forall s \forall x \forall y (B[s], N[x], N[y] \to K\mathbf{ray}_r(s, x, y)) \\ \mathtt{s} & : & B[s] \\ \mathtt{x} & : & N[x] \\ \mathtt{y} & : & N[y] \end{array}$$

il reste à prouver la formule :

$$F[s, x, y] = \forall X \left(\forall y\, (Ky \to X\mathbf{s}_0 y)\, , \forall y\, (Ky \to X\mathbf{s}_1 y) \to X\mathbf{ray}_r\ (s, x, y) \right)$$

Or, par application de la règle de développement du point fixe et par élimination d'un quantificateur sur l'hypothèse $\mathtt{s} : B[s]$, on trouve :

$$\mathtt{s} : \forall s \left(B[s] \to F[\mathbf{s}_0 s, x, y] \right), \forall s \left(B[s] \to F[\mathbf{s}_1 s, x, y] \right) \to F\,(s, x, y)$$

En utilisant la règle d'élimination d'un quantificateur, il suffit donc de prouver :

$$\forall s \left(B[s] \to F[\mathbf{s}_0 s, x, y] \right) \text{ et } \forall s \left(B[s] \to F[\mathbf{s}_1 s, x, y] \right)$$

Posons :

$$\mathtt{s}' : B[s']$$

Il suffit de montrer $F[\mathbf{s}_\epsilon s', x, y]$ pour $\epsilon \in \{0, 1\}$. Or, en utilisant l'hypothèse $\mathtt{y} : N[y]$, après développement du point fixe et élimination du quantificateur, on trouve :

$$\mathtt{y} : F[\mathbf{s}_\epsilon s', x, 0], \forall y \left(N[y] \to F[\mathbf{s}_\epsilon s', x, \mathbf{s}y]\right) \to F[\mathbf{s}_\epsilon s', x, y]$$

Il suffit donc de montrer $F[\mathbf{s}_\epsilon s', x, 0]$ et $\forall y(N[y] \to F[\mathbf{s}_\epsilon s', x, \mathbf{s}y])$

- Montrons d'abord $F[\mathbf{s}_\epsilon s', x, 0]$, c'est-à-dire :

  $$\forall X \left(\forall s \left(Ks \to X\mathbf{s}_0 s\right), \forall s \left(Ks \to X\mathbf{s}_1 s\right) \to X\mathbf{ray}_r \left(\mathbf{s}_\epsilon s', x, 0\right)\right)$$

  Posons

  $$\begin{array}{rcl} \mathtt{f} & : & \forall s(Ks \to X\mathbf{s}_0 s) \\ \mathtt{g} & : & \forall s(Ks \to X\mathbf{s}_1 s) \end{array}$$

  en utilisant les équations, il reste à montrer :

  $$X\mathbf{s}_0\mathbf{ray}_r \left(s', x, \mathbf{pred}\ x\right)$$

  Or on a $\mathtt{s}' : B[s']$, $x : Nx$ et donc $(\mathtt{pred\ x}) : N[\mathbf{pred}\ x]$, donc en utilisant l'hypothèse

  $$\mathtt{r} : \forall s \forall x \forall y \left(B[s], N[x], N[y] \to K\mathbf{ray}_r \left(s, x, y\right)\right)$$

  on trouve $(\mathtt{r\ s'\ x\ (pred\ x)}) : K\mathbf{ray}_r(s', x, \mathbf{pred}\ x)$ puis

  $$\left(\mathtt{f\ \ \left(r\ s'\ x\ \left(pred\ x\right)\right)}\right) : X\mathbf{s}_0\mathbf{ray}_r \left(s', x, \mathbf{pred}\ x\right)$$

- Montrons maintenant $\forall y(N[y] \to F[\mathbf{s}_\epsilon s', x, \mathbf{s}y])$. En supposant :

  $$\begin{array}{rcl} \mathtt{y'} & : & N[y'] \\ \mathtt{f} & : & \forall s(Ks \to X\mathbf{s}_0 s) \\ \mathtt{g} & : & \forall s(Ks \to X\mathbf{s}_1 s) \end{array}$$

  et en utilisant les équations, il reste à montrer :

  $$X\mathbf{s}_\epsilon \left(\mathbf{ray}_r \left(s', x, y'\right)\right)$$

  Or, on a $\mathtt{s}' : B[s']$, $\mathtt{x} : N[x]$ et $\mathtt{y}' : N[y]$, donc en utilisant l'hypothèse :

  $$r : \forall s \forall x \forall y \left(B[s], N[x], N[y] \to K\mathbf{ray}_r \left(s, x, y\right)\right)$$

  on trouve $(\mathtt{r\ s'\ x\ y'}) : K\mathbf{ray}_r(s', x, y')$ puis :

  $$\begin{array}{rcl} (\mathtt{f\,(r\ s'\,x\,y')}) & : & X\mathbf{s}_0\mathbf{ray}_r(s', x, y) \\ (\mathtt{g\,(r\ s'\,x\,y')}) & : & X\mathbf{s}_1\mathbf{ray}_r(s', x, y) \end{array}$$

En résumant la preuve, on a obtenu :

$$\vdash \mathtt{ray_r} : \forall s \forall x \forall y \left(B[s], N[x], N[y] \rightarrow B[\mathbf{ray}_r\,(s,x,y)] \right) \text{ avec}$$

$$\begin{aligned}
\mathtt{ray_r} \;=\; & !\lambda \mathtt{r} \lambda \mathtt{s} \lambda \mathtt{x} \lambda \mathtt{y} (\mathtt{s} \\
& \quad \lambda \mathtt{s}'(\mathtt{y} \\
& \qquad \lambda \mathtt{f} \lambda \mathtt{g} (\mathtt{f}\ (\mathtt{r}\ \mathtt{s}'\ \mathtt{x}\ (\mathtt{pred}\ \mathtt{x}))) \\
& \qquad \lambda \mathtt{y}' \lambda \mathtt{f} \lambda \mathtt{g} (\mathtt{f}\ (\mathtt{r}\ \mathtt{s}'\ \mathtt{x}\ \mathtt{y}'))) \\
& \quad \lambda \mathtt{s}'(\mathtt{y} \\
& \qquad \lambda \mathtt{f} \lambda \mathtt{g} (\mathtt{f}\ (\mathtt{r}\ \mathtt{s}'\ \mathtt{x}\ (\mathtt{pred}\ \mathtt{x}))) \\
& \qquad \lambda \mathtt{y}' \lambda \mathtt{f} \lambda \mathtt{g} (\mathtt{g}\ (\mathtt{r}\ \mathtt{s}'\ \mathtt{x}\ \mathtt{y}'))))
\end{aligned}$$

On obtient alors très facilement :

$$\vdash \mathtt{ray} = \lambda \mathtt{s} \lambda \mathtt{x} \left(\mathtt{ray_r}\ \mathtt{s}\ \mathtt{x}\ (\mathtt{pred}\ \mathtt{x}) \right) : \forall s \forall x \left(B[s], N[x] \rightarrow B[\mathbf{ray}\,(s,x)] \right)$$

On peut maintenant se donner les équations du crible en appliquant à l'infini la fonction précédente :

$$\begin{aligned}
\mathbf{crible}_r(\mathbf{s}_0 s, n) &= \mathbf{s}_0 \mathbf{crible}_r(s, \mathbf{s}n) \\
\mathbf{crible}_r(\mathbf{s}_1 s, n) &= \mathbf{s}_1 \mathbf{crible}_r(\mathbf{ray}(s,n), \mathbf{s}n) \\
\mathbf{crible} &= \mathbf{crible}_r(s_0, \mathbf{ss0})
\end{aligned}$$

On obtient par une méthode identique :

$$\vdash \mathtt{crible_r} : \forall s \forall x \left(B[s], N[x] \rightarrow B[\mathbf{crible}_r\,(b,x)] \right) \text{ avec :}$$

$$\begin{aligned}
\mathtt{crible_r} \;=\; & !\lambda \mathtt{r} \lambda \mathtt{s} \lambda \mathtt{x} (\mathtt{s} \\
& \quad \lambda \mathtt{s}' \lambda \mathtt{f} \lambda \mathtt{g} (\mathtt{f}\ (\mathtt{r}\ (\mathtt{ray}\ \mathtt{s}'\ \mathtt{x})(\mathtt{succ}\ \mathtt{x}))) \\
& \quad \lambda \mathtt{s}' \lambda \mathtt{f} \lambda \mathtt{g} (\mathtt{f}\ (\mathtt{r}\ \mathtt{s}'\ (\mathtt{succ}\ \mathtt{x})))
\end{aligned}$$

et aussi :

$$\vdash \mathtt{crible} = \left(\mathtt{crible_r}\ \mathtt{s_i}\ \left(\mathtt{succ}\ (\mathtt{succ}\ \mathtt{zero}) \right) \right) : B[\mathbf{crible}]$$

7.1.3 Extraction du stream des nombres premiers.

Reste à extraire le stream de tous les nombres premiers. Ici, on rencontre un problème : on ne peut pas typer une fonction prenant en entrée un stream de booléens et rendant un stream d'entiers correspondant à la position des 1 dans le stream d'entrée. En effet, cette fonction appliquée au stream ne contenant que des 0, n'est pas définie, car par définition un stream est toujours infini.

En fait, si l'on type le stream des nombres premiers, la preuve doit contenir une preuve de l'existence d'une infinité de nombres premiers. En conséquence, une telle preuve ne peut être simple.

On va proposer deux méthodes pour résoudre ce problème. La première utilise l'un des exemples du chapitre 5 : les suites de booléens contenant une infinité de 1 définies par :

$$B'[x] = \nu K' \mu K\, \lambda z F[K, K', z]\langle x \rangle$$

en notant :

$$F[K,K',z] = \forall X\left(\forall y\left(Ky \to X\mathbf{s}_0\left(y\right)\right), \forall y\left(K'y \to X\mathbf{s}_1\left(y\right)\right) \to Xz\right)$$

L'existence d'une infinité de nombres premiers, nous assure que le programme `crible` : $B[\mathbf{crible}]$ que l'on a obtenu précédemment peut être considéré de type $B'[\mathbf{crible}]$. Il suffit alors de définir les fonctions suivantes :

$$\begin{array}{lcl} \mathbf{extr}(\mathbf{s}_1 s, n) & = & \mathbf{cs}(n, \mathbf{extr}(s, \mathbf{s}n)) \\ \mathbf{extr}(\mathbf{s}_0 s, n) & = & \mathbf{extr}(s, \mathbf{s}n) \\ \mathbf{primes} & = & \mathbf{extr}(\mathbf{crible}, \mathbf{ss0}) \end{array}$$

et d'extraire le stream des nombres premiers en trouvant deux programmes respectivement de type $\forall s \forall n\ (B'[s], N[n] \to S_N[\mathbf{extr}(s,n)])$ et $S_N[\mathbf{primes}]$.

Pour cela, prouvons $S_N[\mathbf{extr}(s)]$ sous les hypothèses suivantes :

$$\begin{array}{lcl} & & \forall s(B'[s] \subseteq Ks) \\ \mathtt{r} & : & \forall s \forall n(Ks, N[n] \to S_N[\mathbf{extr}(s)]) \\ \mathtt{s} & : & \forall X(\forall x(Kx \to X(\mathbf{s}_0 x)), \forall x(B'[x] \to X(\mathbf{s}_1 x)) \to X(s)) \\ \mathtt{n} & : & N[n] \end{array}$$

Sous ces hypothèses, on obtient :

$$\begin{array}{lcl} \lambda\mathtt{s}\left(\mathtt{r}\,\mathtt{n}\,\mathtt{s}\right) & : & \forall x(Kx \to S_N[\mathbf{extr}(\mathbf{s}_0 x)]) \\ \lambda\mathtt{s}\lambda\mathtt{f}\left(\mathtt{f}\,\mathtt{n}\left(\mathtt{r}\,\mathtt{n}\,\mathtt{s}\right)\right) & : & \forall x(Kx \to S_N[\mathbf{extr}(\mathbf{s}_1 x)]) \end{array}$$

Et donc :

$$\left(\mathtt{s}\ \lambda\mathtt{s}\ \left(\mathtt{r}\,\mathtt{s}\ \left(\mathtt{succ}\,\mathtt{n}\right)\right)\ \lambda\mathtt{s}\lambda\mathtt{f}\ \left(\mathtt{f}\,\mathtt{n}\ \left(\mathtt{r}\,\mathtt{s}\ \left(\mathtt{succ}\,\mathtt{n}\right)\right)\right)\right) : S_N[\mathbf{extr}\left(s\right)]$$

En appliquant trois introductions d'implication, la règle du plus petit point fixe avec inclusion et la factorisation du plus grand point fixe, on obtient :

$$\vdash \mathtt{extr} : \forall s \forall n\ \left(B'[s], N[n] \to S_N[\mathbf{extr}\left(s,n\right)]\right)\ \text{avec}$$

$$\mathtt{extr} = !\lambda\mathtt{r}\lambda\mathtt{s}\lambda\mathtt{n}\left(\mathtt{s}\ \lambda\mathtt{s}\ \left(\mathtt{r}\,\mathtt{s}\ \left(\mathtt{succ}\,\mathtt{n}\right)\right)\ \lambda\mathtt{s}\lambda\mathtt{f}\ \left(\mathtt{f}\,\mathtt{n}\ \left(\mathtt{r}\,\mathtt{s}\ \left(\mathtt{succ}\,\mathtt{n}\right)\right)\right)\right)$$

De là, on trouve sans problème :

$$\vdash \mathtt{primes} = \left(\mathtt{extr}\ \mathtt{crible}\ \ \left(\mathtt{succ}\ \left(\mathtt{succ}\,\mathtt{zero}\right)\right)\right) : S_N[\mathbf{primes}]$$

Cette méthode montre une utilisation intéressante des types utilisant plusieurs points fixes. Par contre, elle a le défaut d'être particulière aux streams. On va présenter une méthode beaucoup plus générale pour séparer l'extraction de programme de la preuve de totalité : la méthode du *constructeur muet*. Cette méthode consiste à ajouter au type un constructeur unaire $\boldsymbol{\tau}$, et à définir le type par un plus grand point fixe. Ce nouveau constructeur est muet, car on l'effacera pour lire la valeur du résultat.

On obtient ainsi le type de données S_N^τ suivant pour les streams d'entiers :

$$S_N^\tau[x] = \nu S \forall X \left(\forall n \forall s \left(N[a], Ss \to X \left(str\,(a,s) \right) \right), \forall s \left(Ss \to X \boldsymbol{\tau}\,(s) \right) \to Xx \right) \langle x \rangle$$

On peut alors extraire le stream voulu à partir d'un stream de booléens, à l'aide des équations suivantes :

$$\begin{array}{lcl} \mathbf{extr}(\mathbf{s}_1 s, n) & = & \mathbf{cs}(n, \mathbf{extr}(s, \mathbf{s}n)) \\ \mathbf{extr}(\mathbf{s}_0 s, n) & = & \boldsymbol{\tau}(\mathbf{extr}(s, \mathbf{s}n)) \\ \mathbf{primes} & = & \mathbf{extr}(\mathbf{crible}, \mathbf{ss0}) \end{array}$$

On obtient sans problème :

$$\vdash \texttt{extr} : \forall s \forall n \left(B[s], N[n] \to S_N^\tau[\mathbf{extr}\,(s,n)] \right) \text{ evec}$$

$$\texttt{extr} = !\lambda \texttt{r} \lambda \texttt{s} \lambda \texttt{n} \left(\texttt{s}\ \lambda \texttt{s}' \lambda \texttt{f} \lambda \texttt{t} \left(\texttt{t}\ \left(\texttt{r}\,\texttt{s}'\ (\texttt{succ}\,\texttt{n}) \right) \right)\ \lambda \texttt{s}' \lambda \texttt{f} \lambda \texttt{t} \left(\texttt{f}\ \texttt{n}\ \left(\texttt{r}\,\texttt{s}'\ (\texttt{succ}\,\texttt{n}) \right) \right) \right)$$

Et on trouve ainsi cette représentation du stream des nombres premiers :

$$\vdash \texttt{primes} = \left(\texttt{extr}\,\texttt{crible}\ \left(\texttt{succ}\ (\texttt{succ}\,\texttt{zero}) \right) \right) : S_N^\tau[\mathbf{primes}]$$

7.2 Typage des fonctions récursives.

7.2.1 Une autre représentation des entiers.

Afin de montrer à quel point la méthode du constructeur muet est générale, on va l'utiliser pour les entiers et prouver qu'ainsi on peut typer toutes les fonctions récursives partielles. Cela prouve que cette méthode sépare le typage de la preuve de totalité.

On suppose qu'il y a dans le langage une constante **0** et deux symboles de fonctions unaires **s** et $\boldsymbol{\tau}$. Considérons le type suivant pour les entiers :

$$N^\tau[x] = \nu K \forall X \left(X\mathbf{0}, \forall y\,(Xy \to X\mathbf{s}y), \forall y\,(Xy \to X\boldsymbol{\tau}y) \to Xx \right) \langle x \rangle$$

Cette représentation des entiers diffère de l'habitude par deux aspects :

- Il y a une infinité de représentations pour chaque entier (aussi bien au niveau des termes logiques que des termes du λ-calcul). Cela est inhérent à la méthode du constructeur muet.
- De plus, si l'on quotiente cet ensemble par l'équation $\boldsymbol{\tau}(x) = x$, on n'obtient pas l'ensemble des entiers, mais un ensemble qui contient tous les entiers standards $(\texttt{succ}^\texttt{n}\ \texttt{zero})$, plus une infinité d'entiers non-standards : $!\texttt{succ}$ ou $(\texttt{succ}^\texttt{n}\ !\tau)$. On reconnaît là les entiers paresseux (en identifiant $\bot$ et $!\tau$). Cela est inhérent à la méthode du constructeur muet appliquée à un type de données finies, car on remplace le plus petit point fixe par un plus grand.

7.2.2 Définition par équation des fonctions.

Afin de programmer les fonctions récursives, on doit les définir par des équations. Or, toute fonction récursive peut être construite par induction à partir de **s**, **0** et des projections, en utilisant la composition, la récurrence ou le schéma-μ.

Il suffit donc de montrer comment l'on peut définir chacun de ces schémas par des équations.

- La constante **s** ne nécessite aucune équation.
- La fonction constante égale à **0** est définie par l'équation

$$\mathbf{f}\left(x_1,\ldots,x_n\right)=\mathbf{0}$$

- Les projections sont définies par l'équation

$$\mathbf{f}\left(x_1,\ldots,x_n\right)=x_i$$

- Si **f** est définie par composition d'une fonction **g** d'arité p avec p fonction $\mathbf{h}_p$ d'arité n, on choisit :

$$\mathbf{f}\left(x_1,\ldots,x_n\right)=\mathbf{g}\left(\mathbf{h}_1\left(x_1,\ldots,x_n\right),\ldots,\mathbf{h}_p\left(x_1,\ldots,x_n\right)\right)$$

- Si **f** d'arité $n+1$ est définie par récurrence à partir de **h** (cas initial) et **g**, on ajoute un nouveau symbole de fonction $\mathbf{f}'$ d'arité $n+3$ et on choisit les équations suivantes (qui correspondent à une définition "terminale" de la récurrence) :

$$\begin{array}{rcl}
\mathbf{f}(x,x_1,\ldots,x_n) & = & \mathbf{f}'(x,\mathbf{0},\mathbf{h}(x_1,\ldots,x_n),x_1,\ldots,x_n)\\
\mathbf{f}'(\mathbf{0},x',y,x_1,\ldots,x_n) & = & \boldsymbol{\tau}(y)\\
\mathbf{f}'(\mathbf{s}x,x',y,x_1,\ldots,x_n) & = & \boldsymbol{\tau}(\mathbf{f}'(x,\mathbf{s}x',\mathbf{g}(x',y,x_1,\ldots,x_n),x_1,\ldots,x_n))\\
\mathbf{f}'(\boldsymbol{\tau}(x),x',y,x_1,\ldots,x_n) & = & \boldsymbol{\tau}(\mathbf{f}'(x,x',y,x_1,\ldots,x_n))
\end{array}$$

- Si **f** d'arité $n+1$ est définie par application du schéma-μ à **g**, on ajoute un nouveau symbole de fonction $\mathbf{f}'$ d'arité $n+2$ et on choisit les équations suivantes :

$$\begin{array}{rcl}
\mathbf{f}(y,x_1,\ldots,x_n) & = & \mathbf{f}'(\mathbf{g}(y,x_1,\ldots,x_n),y,x_1,\ldots,x_n)\\
\mathbf{f}'(\mathbf{0},y,x_1,\ldots,x_n) & = & \boldsymbol{\tau}(y)\\
\mathbf{f}'(\mathbf{s}x,y,x_1,\ldots,x_n) & = & \boldsymbol{\tau}(\mathbf{f}'(\mathbf{g}(\mathbf{s}y,x_1,\ldots,x_n),\mathbf{s}y,x_1,\ldots,x_n))\\
\mathbf{f}'(\boldsymbol{\tau}(x),y,x_1,\ldots,x_n) & = & \boldsymbol{\tau}(\mathbf{f}'(x,y,x_1,\ldots,x_n))
\end{array}$$

Il est facile de voir que, modulo l'équation $\boldsymbol{\tau}(x)=x$, ces schémas permettent bien de construire toutes les fonctions récursives partielles. Plus précisément à une fonction **f** définie par un ensemble d'équations $\mathcal{E}$ en utilisant les schémas précédents, on peut faire correspondre une fonction récursive $\overline{f}$ en posant $\overline{f}(p_1,\ldots,p_n)=q$ si et seulement si :

$$\mathcal{E},\boldsymbol{\tau}\left(x\right)=x\vdash\mathbf{f}\left(\mathbf{s}^{p_1}\left(\mathbf{0}\right),\ldots,\mathbf{s}^{p_n}\left(\mathbf{0}\right)\right)=\mathbf{s}^{q}\left(\mathbf{0}\right)$$

Cette définition est bien fondée car les équations que l'on a choisies sont compatibles avec la relation d'équivalence engendrée par l'équation $\boldsymbol{\tau}(x)=x$. De plus, nos schémas ont bien la

correspondance attendue, c'est à dire, que si **f** est définie en utilisant les équations proposées ci-dessus pour un certain schéma, alors $\overline{f}$ peut être définie par ce schéma. Ceci prouve bien que l'on peut définir toutes les fonctions récursives.

Le seul point non trivial est la récurrence, car comme on le verra à la fin de ce chapitre, on est obligé de prendre une forme de récurrence terminale pour les équations, ce qui ne correspond pas immédiatement à la définition usuelle de la récurrence.

On va maintenant montrer que l'on peut typer **f**. C'est-à-dire que **f** étant d'arité n, on peut trouver un terme de type :

$$\forall x_1, \ldots, x_n \left(N^\tau[x_1], \ldots, N^\tau[x_n] \to N^\tau[\mathbf{f}\,(x_1, \ldots, x_n)] \right)$$

Pour cela il suffit de typer chacun des schémas utilisés dans la définition de **f**. Les seuls schémas qui posent problème sont le schéma-μ et la récurrence.

7.2.3 Typage du schéma-μ.

Supposons que **f** est définie par schéma-μ à partir de **g** (voir équations page précédente), Supposons que l'on a déjà obtenu une preuve pour **g** :

$$\vdash \mathtt{g} : \forall x \forall x_1 \ldots \forall x_n (N^\tau[x], N^\tau[x_1], \ldots, N^\tau[x_n] \to N^\tau[\mathbf{g}(x, x_1, \ldots, x_n)])$$

Commençons par chercher un terme $\mathtt{f}'$ extrait d'une preuve de :

$$x_1 : N^\tau[x_1], \ldots, x_n : N^\tau[x_n] \vdash \mathtt{f}' : \forall x \forall y \left(N^\tau[x], N^\tau[y] \to N^\tau[\mathbf{f}'\,(x, y, x_1, \ldots, x_n)] \right)$$

Pour obtenir cette dernière preuve, il suffit d'utiliser la règle dérivée dite du plus grand point fixe avec inclusion (cf chapitre 6), c'est-à-dire de trouver un terme $\mathtt{u}$ tel que :

$$\Gamma, \forall x \left(N^\tau[x] \subseteq K\,x \right) \vdash \mathtt{u} : F \to F[K/\lambda x G] \text{ avec :}$$

$$\begin{array}{rcl} \Gamma & = & \mathtt{x_1} : N^\tau[x_1], \ldots, \mathtt{x_n} : N^\tau[x_n] \\ F & = & \forall x \forall y (N^\tau[x], N^\tau[y] \to K(\mathbf{f}'(x, y, x_1, \ldots, x_n))) \\ G & = & \forall X (X\mathbf{0}, \forall y (Ky \to X\mathbf{s}y), \forall y (Ky \to X\boldsymbol{\tau}y) \to Xx) \end{array}$$

Or, sous les hypothèses suivantes :

$$\begin{array}{rcl} & & \forall x (N^\tau[x] \subseteq K\,x) \\ \mathtt{x_1} & : & N^\tau[x_1], \ldots, \mathtt{x_n} : N^\tau[x_n] \\ \mathtt{r} & : & F = \forall x \forall y (N^\tau[y], N^\tau[x] \to K(\mathbf{f}'(y, x, x_1, \ldots, x_n))) \\ \mathtt{x} & : & N^\tau[x] \\ \mathtt{y} & : & N^\tau[y] \\ \mathtt{a} & : & X\mathbf{0} \\ \mathtt{f} & : & \forall y (Ky \to X\mathbf{s}y) \\ \mathtt{t} & : & \forall y (Ky \to X\boldsymbol{\tau}y) \end{array}$$

on obtient, en utilisant les équations définissant la récurrence :

$$\begin{array}{rcl} \mathtt{c_0} = (\mathtt{t\ y}) & : & X\mathbf{f}'(\mathbf{0}, y, x_1, \ldots, x_n) \\ \mathtt{c_s} = \lambda \mathtt{x}'(\mathtt{t}\ (\mathtt{r}\ (\mathtt{g}\,(\mathtt{succ\,y})\,\mathtt{x_1} \ldots \mathtt{x_n})\ (\mathtt{succ\,y}))) & : & \forall x' (N^\tau[x'] \to X\mathbf{f}'(\mathbf{s}x', y, x_1, \ldots, x_n)) \\ \mathtt{c}_\tau = \lambda \mathtt{x}'(\mathtt{t}\ (\mathtt{r\,x}'\,\mathtt{y})) & : & \forall x' (N^\tau[x'] \to X\mathbf{f}'(\boldsymbol{\tau}x', y, x_1, \ldots, x_n)) \end{array}$$

Et donc, sous ces hypothèse, on a :

$$(\mathtt{x}\ \mathtt{c}_0\ \mathtt{c}_s\ \mathtt{c}_\tau) : X\mathbf{f}'(x, y, x_1, \ldots, x_n)$$

D'où :

$$\vdash \mathtt{f} : \forall y \forall x_1 \ldots \forall x_n \left(N^\tau[y], N^\tau[x_1], \ldots, N^\tau[x_n] \to N^\tau[\mathbf{f}(y, x_1, \ldots, x_n)]\right) \text{ avec :}$$

$$\begin{array}{l}
\mathtt{f} = \lambda \mathtt{y} \lambda \mathtt{x_1} \ldots \lambda \mathtt{x_n} (\mathtt{f}'\ (\mathtt{g}\ \mathtt{y}\ \mathtt{x_1} \ldots \mathtt{x_n})\ \mathtt{y}) \\
\mathtt{f}' = !\lambda \mathtt{r} \lambda \mathtt{x} \lambda \mathtt{y} \lambda \mathtt{a} \lambda \mathtt{f} \lambda \mathtt{t} (\mathtt{x} \\
\qquad\qquad\qquad\qquad (\mathtt{t}\ \mathtt{y}) \\
\qquad\qquad\qquad\qquad \lambda \mathtt{x}'(\mathtt{t}\ (\mathtt{r}\ (\mathtt{g}\ (\mathtt{succ}\,\mathtt{y})\ \mathtt{x_1} \ldots \mathtt{x_n})\ (\mathtt{succ}\,\mathtt{y}))) \\
\qquad\qquad\qquad\qquad \lambda \mathtt{x}'(\mathtt{t}\ (\mathtt{r}\ \mathtt{x}'\ \mathtt{y})))
\end{array}$$

7.2.4 Typage de la récurrence.

Supposons que **f** est définie par schéma-μ à partir de **h** et **g**. Supposons que l'on a déjà obtenu deux preuves pour **h** et **g** :

$$\begin{array}{l}
\vdash \mathtt{g} : \forall x \forall y \forall x_1 \ldots \forall x_n (N^\tau[x], N^\tau[y], N^\tau[x_1], \ldots, N^\tau[x_n] \to N^\tau[\mathbf{g}(x, y, x_1, \ldots, x_n)]) \\
\vdash \mathtt{h} : \forall x_1 \ldots \forall x_n (N^\tau[x_1], \ldots, N^\tau[x_n] \to N^\tau[\mathbf{h}(x_1, \ldots, x_n)])
\end{array}$$

Commençons par chercher un terme $\mathtt{f}'$ extrait d'une preuve de :

$$\Gamma \vdash \mathtt{f}' : \forall x \forall x' \forall y \left(N^\tau[x], N^\tau[x'], N^\tau[y] \to N^\tau[\mathbf{f}'(x, y, x_1, \ldots, x_n)]\right) \text{ avec :}$$

$$\Gamma = x_1 : N^\tau[x_1], \ldots, x_n : N^\tau[x_n]$$

Pour obtenir cette dernière preuve, il suffit d'utiliser la règle dérivée dite du plus grand point fixe avec inclusion (cf chapitre 6), c'est-à-dire de trouver un terme $\mathtt{u}$ tel que :

$$\Gamma, \forall x \left(N^\tau[x] \subseteq Kx\right) \vdash \mathtt{u} : F \to F[K/\lambda x G] \text{ avec}$$

$$\begin{array}{rcl}
\Gamma & = & \mathtt{x_1} : N^\tau[x_1], \ldots, \mathtt{x_n} : N^\tau[x_n] \\
F & = & \forall x \forall y (N^\tau[x], N^\tau[x'], N^\tau[y] \to K(\mathbf{f}'(x, y, x_1, \ldots, x_n))) \\
G & = & \forall X (X\mathbf{0}, \forall y (Ky \to X\mathbf{s}y), \forall y (Ky \to X\boldsymbol{\tau}y) \to Xx)
\end{array}$$

Or, sous les hypothèses suivantes :

$$\begin{array}{rcl}
 & & \forall x (N^\tau[x] \subseteq Kx) \\
\mathtt{x_1} & : & N^\tau[x_1], \ldots, \mathtt{x_n} : N^\tau[x_n] \\
\mathtt{r} & : & F = \forall x \forall y (N^\tau[x], N^\tau[x'], N^\tau[y] \to K(\mathbf{f}'(x, y, x_1, \ldots, x_n))) \\
\mathtt{x} & : & N^\tau[x] \\
\mathtt{x}' & : & N^\tau[x'] \\
\mathtt{y} & : & N^\tau[y] \\
\mathtt{a} & : & X\mathbf{0} \\
\mathtt{f} & : & \forall y (Ky \to X\mathbf{s}y) \\
\mathtt{t} & : & \forall y (Ky \to X\boldsymbol{\tau}y)
\end{array}$$

on obtient, en utilisant les équations définissant la récurrence :

$$\begin{array}{rcl} \mathtt{c_0} = (\mathtt{t\ y}) & : & X\mathbf{f}'(\mathbf{0}, x', y, x_1, \ldots, x_n) \\ \mathtt{c_s} = \lambda\mathtt{z}(\mathtt{t}\ (\mathtt{r\ z}\ (\mathtt{succ\ x'})\ (\mathtt{g\ x'\ y\ x_1 \ldots x_n}))) & : & \forall z(N^\tau[z] \to X\mathbf{f}'(\mathbf{s}z, x', y, x_1, \ldots, x_n)) \\ \mathtt{c}_\tau = \lambda\mathtt{z}(\mathtt{t}\ (\mathtt{r\ z\ x'\ y})) & : & \forall z(N^\tau[z] \to X\mathbf{f}'(\boldsymbol{\tau}z, x', y, x_1, \ldots, x_n)) \end{array}$$

Donc, sous ces hypothèses, on a :

$$(\mathtt{x\ c_0\ c_s\ c}_\tau) : X\mathbf{f}'\,(x, x', y, x_1, \ldots, x_n)$$

D'où :

$$\vdash \mathtt{f} : \forall x \forall x_1 \ldots \forall x_n \left(N^\tau[x], N^\tau[x_1], \ldots, N^\tau[x_n] \to N^\tau[\mathbf{f}\,(x_1, \ldots, x_n)]\right) \text{ avec}$$

$$\begin{array}{l} \mathtt{f} = \lambda\mathtt{x}\lambda\mathtt{x_1} \ldots \lambda\mathtt{x_n}(\mathtt{f'\ x\ zero}\ (\mathtt{h\ x_1 \ldots x_n})) \\ \mathtt{f'} = !\lambda\mathtt{r}\lambda\mathtt{x}\lambda\mathtt{x'}\lambda\mathtt{y}\lambda\mathtt{a}\lambda\mathtt{f}\lambda\mathtt{t}(\mathtt{x} \\ \qquad\qquad\qquad\qquad (\mathtt{t\ y}) \\ \qquad\qquad\qquad\qquad \lambda\mathtt{z}(\mathtt{t}\ (\mathtt{r\ z}\ (\mathtt{succ\ x'})\ (\mathtt{g\ x'\ y\ x_1 \ldots x_n}))) \\ \qquad\qquad\qquad\qquad \lambda\mathtt{z}(\mathtt{t}\ (\mathtt{r\ z\ x'\ y}))) \end{array}$$

7.3 Le problème de la récurrence.

Pourquoi doit-t'on écrire le schéma de récurrence sous cette forme ? En fait, la réponse est simple. Dans sa forme naturelle, la définition de **f** par récurrence à partir de **g** et **h** s'écrirait :

$$\begin{array}{rcl} \mathbf{f}(\mathbf{0}, x_1, \ldots, x_n) & = & \boldsymbol{\tau}(\mathbf{h}(x_1, \ldots, x_n)) \\ \mathbf{f}(\mathbf{s}x, x_1, \ldots, x_n) & = & \boldsymbol{\tau}(\mathbf{g}(x, \mathbf{f}(x, x_1, \ldots, x_n), x_1, \ldots, x_n)) \\ \mathbf{f}(\boldsymbol{\tau}(x), x_1, \ldots, x_n) & = & \boldsymbol{\tau}(\mathbf{f}(x, x_1, \ldots, x_n)) \end{array}$$

Or, la deuxième équation peut conduire à une fonction non totale. En effet, définissons **g** de cette manière :

$$\begin{array}{rcl} \mathbf{g}(x, \boldsymbol{\tau}\boldsymbol{\tau}y, x_1, \ldots, x_n) & = & y \\ \mathbf{g}(x, y, x_1, \ldots, x_n) & = & y \text{ si } y \text{ est de la forme } \mathbf{0}, \mathbf{s}y', \boldsymbol{\tau}\mathbf{0} \text{ ou } \boldsymbol{\tau}\mathbf{s}y' \end{array}$$

La fonction **g** correspond à la deuxième projection, en effaçant si possible deux $\boldsymbol{\tau}$ au début du résultat. Or, si l'on utilise **g** dans une récurrence, **f** n'est pas une fonction totale. En effet, si l'on considère l'entier non-standard $\boldsymbol{\omega}$ défini par l'équation $\boldsymbol{\omega} = \mathbf{s}\boldsymbol{\omega}$, on a :

$$\mathbf{f}\,(\boldsymbol{\omega}, x_1, \ldots, x_n) = \boldsymbol{\tau}\left(\mathbf{g}\left(x, \mathbf{f}\,(\boldsymbol{\omega}, x_1, \ldots, x_n), x_1, \ldots, x_n\right)\right)$$

ce qui ne donne jamais un élément du type N^τ, car **f** ne produit qu'un seul $\boldsymbol{\tau}$ alors que **g** en nécessite deux.

Chapitre 8

Égalité et infini.

Un problème n'a pas été abordé tout au long de cette thèse : la définition de l'égalité. Nous allons montrer, dans cette section, quels problèmes posent les types de données infinies du point de vue de l'égalité, et on montrera une manière particulièrement bien adaptée à $AF_2^{\mu\nu}$ pour les résoudre. De plus, cela nous conduira à développer une nouvelle notion de type données avec égalité, qui est relié fortement à celle de type quotient.

8.1 L'égalité de Leibnitz.

Définition 8.1. L'égalité de Leibnitz est définie par

$$L[u,v] = \forall X\,(Xu \to Xv)$$

La proposition suivante et son corollaire montrent que l'égalité de Leibnitz est essentiellement reliée à la β-équivalence.

Proposition 8.2. *Pour toute interprétation classique $\mathcal{M}$, on a :*

$$|u|^{\mathcal{M}} \sim_\beta |v|^{\mathcal{M}} \text{ si et seulement si } \mathcal{M} \models L[u,v]$$

Corollaire 8.3. *Pour une interprétation σ, on a :*

$$|u|^\sigma \not\sim_\beta |v|^\sigma \text{ implique } \nvdash L[u,v]$$

Démonstration : La proposition découle du fait que l'on a défini toutes les notions d'interprétations sur le λ-calcul quotienté par la β-équivalence. Pour démontrer le corollaire, il suffit de montrer que pour toute interprétation σ, $\vdash L[u,v]$ implique $|u|^\sigma \sim_\beta |v|^\sigma$. Supposons $\vdash L[u,v]$. On choisit alors une interprétation classique $\mathcal{M}_\sigma$ qui coïncide avec σ pour l'interprétation des termes logiques. Par le lemme de conservation, on obtient $\mathcal{M}_\sigma \models L[u,v]$, d'où le résultat en appliquant la proposition. ■

Nous allons montrer que l'égalité de Leibnitz ne convient pas pour les types de données infinies.

Intuitivement, l'égalité naturelle sur les éléments d'un type de données est la ω-équivalence [1]. En effet, c'est en utilisant la ω-équivalence que l'on a prouvé tous les résultats sur les types de données (cf chapitre 5).

Plus précisément, considérons le type suivant :

$$W[x] = \nu K \lambda x \forall X \left(\forall y\,(Ky \to X\mathbf{s}y) \to Xx\right)\langle x\rangle$$

C'est le plus simple des types de données infinies. En appliquant les résultats du chapitre 5, on montre que pour toute interprétation σ tel que $|\mathbf{s}|^\sigma(x) = \lambda \mathtt{f}(\mathtt{f}\ |\mathtt{x}|^\sigma)$, on a $\vdash W[u]$ implique $|u|^\sigma \sim_\omega\ !\lambda \mathtt{r}\lambda \mathtt{f}(\mathtt{f}\ \mathtt{r})$. Donc, ce type ne contient qu'un seul élément, ce que l'égalité devrait permettre de démontrer.

Considérons deux constantes logiques $\boldsymbol{\omega}$ et $\boldsymbol{\omega}'$ vérifiant les équations $\boldsymbol{\omega} = \mathbf{s}\boldsymbol{\omega}$ et $\boldsymbol{\omega}' = \mathbf{ss}\boldsymbol{\omega}'$. On obtient sans problème $!\lambda \mathtt{r}\lambda \mathtt{f}(\mathtt{f}\ \mathtt{r}) : W[\boldsymbol{\omega}]$ et $!\lambda \mathtt{r}\lambda \mathtt{f}(\mathtt{f}\ \lambda \mathtt{f}(\mathtt{f}\ \mathtt{r})) : W[\boldsymbol{\omega}']$. Or, ces deux termes ne sont pas β-équivalents. On a donc :

$$\nvdash \forall x \forall y \left(W[x], W[y] \to L[x,y]\right)$$

On ne peut donc pas prouver que le type W ne contient qu'un seul élément pour l'égalité de Leibnitz, ce qui montre que l'égalité de Leibnitz ne convient pas pour les types de données infinies.

8.2 Problème de l'égalité sur les données infinies.

Nous allons montrer une proposition plus forte qui exprime l'impossibilité de définir une égalité "générale" qui corresponde à ce que l'on veut.

Une égalité est définie par une formule $F[x,y]$ dont les seules variables libres sont x et y. Le fait que l'égalité soit "générale" signifie qu'elle est indépendante de la définition des types de données. Ce que l'on peut exprimer en disant qu'elle n'utilise aucun constructeur des types de données.

Or, pour que cette égalité convienne, il faudrait pouvoir prouver, sans utiliser aucun axiome ou équation, que $\vdash \forall x \forall y (W[x], W[y] \to F[x,y])$. Or, on prouve la propriété suivante :

Proposition 8.4. *Si $F[x,y]$ est une formule où les seules variables libres sont x et y, et où la constante* $\mathbf{s}$ *n'apparaît pas, alors si l'on prouve $\vdash \forall x \forall y (W[x], W[y] \to F[x,y])$ sans utiliser aucun axiome ou équation, on prouve aussi $\vdash \forall x \forall y F[x,y]$*

Démonstration : Supposons $\vdash \forall x \forall y (W[x], W[y] \to F[x,y])$. En remplaçant tous les termes de la forme $\mathbf{s}t$ par t dans la preuve (cette substitution est possible car on n'utilise aucun axiome ou équation.), on obtient une preuve de $\vdash \forall x \forall y (W'[x], W'[y] \to F[x,y])$ avec $W'[x] = \nu K \lambda x \forall X (\forall y (Ky \to Xy) \to Xx)\langle x\rangle$. Or, en utilisant la règle du plus grand point fixe, on prouve sans problème $\vdash \forall x W'[x]$. D'où $\vdash \forall x \forall y F[x,y]$. ■

[1] Pour les types de données finies, cela ne pose aucun problème. En effet, les éléments du type sont alors des termes normalisables pour lesquels la ω-équivalence coïncide avec la β-équivalence.

Ceci prouve bien que l'on ne peut pas définir d'égalité générale qui permettent de démontrer que le type W n'a qu'un seul élément.

En fait l'égalité que l'on veut définir est caractérisée par la notion suivante :

Définition 8.5. Une relation $R[x, y]$ définit *l'égalité intentionnelle* pour le type D si et seulement si pour tout modèle $\mathcal{M}$ intentionnel pour D, on a pour tout élément $\mathtt{t}$ et $\mathtt{t}'$ du type D :

$$\mathcal{M}[x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models R[x, y] \text{ si et seulement si } \mathtt{t} \sim_\omega \mathtt{t}'$$

On vient de montrer que l'on ne pouvait pas définir une égalité intentionnelle pour tous les types de données infinies. Mais comment définir l'égalité intentionnelle pour un type de données infinies particulier ?

8.3 Le schéma de coïnduction.

La solution habituelle à ce problème est le schéma de coïnduction qui définit l'égalité comme la plus grande relation close pour les destructeurs du type. Pour le type W précédent, ce schéma s'écrit (avec **pred** vérifiant l'équation $\mathbf{pred}(\mathbf{s}x) = x$) :

$$C_W[x, y] = \exists R \left(R(x, y) \wedge \forall x \forall y \left(W[x], W[y], R(x, y) \rightarrow R(\mathbf{pred}\, x, \mathbf{pred}\, y) \right) \right)$$

Cette formule, qui est universellement vrai, peut alors être utilisée comme définition de l'égalité sur le type W.

Prenons un exemple plus concret : le schéma de coïnduction sur les streams d'objets de type A. Notons R_A l'égalité sur le type A. Nous utiliserons **car** et **cdr** vérifiant $\mathcal{E} \vdash \mathbf{car}(\mathbf{cs}(a, x)) = a$ et $\mathcal{E} \vdash \mathbf{cdr}(\mathbf{cs}(a, x)) = x$. On peut alors considérer la formule $C_{S_A}[x, y]$ égale à :

$$\exists R \left(R(x, y) \wedge \forall x \forall y \left(S_A[x], S_A[y], R(x, y) \rightarrow R_A(\mathbf{car}\, x, \mathbf{car}\, y) \wedge R(\mathbf{cdr}\, x, \mathbf{cdr}\, y) \right) \right)$$

Proposition 8.6. *Si C_A définit l'égalité intentionnelle pour le type A, alors le schéma de coïnduction C_{S_A} définit l'égalité intentionnelle sur le type S_A.*

Démonstration : Choisissons un modèle $\mathcal{M}$ intentionnel pour le type stream et deux termes $\mathtt{t}$ et $\mathtt{t}'$ éléments du type $\mathcal{D}$. Montrons d'abord que $\mathtt{t} \sim_\omega \mathtt{t}'$ entraîne $\mathcal{M}[x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models C_S[x, x']$. On choisit la ω-équivalence pour interpréter R. On trouve alors $\mathcal{M}[R \Leftarrow \sim_\omega][x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models R[x, x']$. Reste à montrer :

$$\mathcal{M}[R \Leftarrow \sim_\omega][x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models \forall x \forall y \left(R(x, y) \rightarrow R_A(\mathbf{car}\, x, \mathbf{car}\, y) \wedge R\left(\mathbf{cdr}(x), \mathbf{cdr}(y)\right) \right)$$

On choisit donc deux éléments $\mathtt{u}$ et $\mathtt{u}'$ du type S_A tel que $\mathtt{u} \sim_\omega \mathtt{u}'$, et il faut montrer $|\mathbf{car}|^{\mathcal{M}}(\mathtt{u}) \sim_\omega |\mathbf{car}|^{\mathcal{M}}(\mathtt{u}')$ et $|\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}) \sim_\omega |\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}')$. Or, par le corollaire 5.8 on trouve des termes $\mathtt{b}, \mathtt{b}', \mathtt{v}, \mathtt{v}'$ tel que $\mathtt{u} = |\mathbf{cs}|^{\mathcal{M}}(\mathtt{b}, \mathtt{v})$ et $\mathtt{u}' = |\mathbf{cs}|^{\mathcal{M}}(\mathtt{b}', \mathtt{v}')$. $\mathcal{M}$ étant intentionnel pour D, il est syntaxiquement intentionnel (théorème 5.18). On a donc $\mathtt{u} \sim_\omega \lambda \mathtt{f}(\mathtt{f}\, \mathtt{b}\, \mathtt{v})$ et $\lambda \mathtt{f}(\mathtt{f}\, \mathtt{b}'\, \mathtt{v}') \sim_\omega \mathtt{u}'$. On en déduit $\mathtt{b} \sim_\omega \mathtt{b}'$ et $\mathtt{v} \sim_\omega \mathtt{v}'$. Or $\mathcal{E} \vdash \mathbf{car}(\mathbf{cs}(a, x)) = a$ et $\mathcal{E} \vdash \mathbf{cdr}(\mathbf{cs}(a, x)) = a$, ce qui implique $|\mathbf{car}|^{\mathcal{M}}(\mathtt{u}) \sim_\beta \mathtt{b} \sim_\omega \mathtt{b}' \sim_\beta |\mathbf{car}|^{\mathcal{M}}(\mathtt{u}')$ et $|\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}) \sim_\beta \mathtt{v} \sim_\omega \mathtt{v}' \sim_\beta |\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}')$.

Reste à montrer la réciproque, c'est à dire que $\mathcal{M}[x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models C_S[x, x']$ entraîne $\mathtt{t} \sim_\omega \mathtt{t}'$. Supposons $\mathcal{M}[x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models C_S[x, x']$. On trouve donc une relation Φ tel que $\Phi(\mathtt{t}, \mathtt{t}')$ et tel que $\mathcal{M}[R \Leftarrow \Phi][x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models \forall x \forall y (R(x, y) \rightarrow R_A(\mathbf{car}\, x, \mathbf{car}\, y) \wedge R(\mathbf{cdr}(x), \mathbf{cdr}(y)))$. Il suffit alors de montre que pour tous termes $\mathtt{u}$ et $\mathtt{u}'$ élément du type S_A, on a $\Phi(\mathtt{u}, \mathtt{u}')$ implique $\mathtt{u} \sim_\omega \mathtt{u}'$. On montre cela par induction sur la définition de $\sim_\omega$ (cf appendice A). On a bien $\Phi(\mathtt{u}, \mathtt{u}')$ implique $\mathtt{u} \sim_{\omega_0} \mathtt{u}'$. Supposons que $\Phi(\mathtt{u}, \mathtt{u}')$ implique $\mathtt{u} \sim_{\omega_n} \mathtt{u}'$, choisissons deux termes $\mathtt{u}$ et $\mathtt{u}'$ éléments du type S_A. Comme précédemment par le corollaire 5.8, on trouve des termes $\mathtt{b}, \mathtt{b}', \mathtt{v}, \mathtt{v}'$ tel que $\mathtt{u} = |\mathbf{cs}|^{\mathcal{M}}(\mathtt{b}, \mathtt{v})$ et $\mathtt{u}' = |\mathbf{cs}|^{\mathcal{M}}(\mathtt{b}', \mathtt{v}')$. Or, par hypothèse sur Φ, on en déduit $\mathcal{M}[R \Leftarrow \Phi][x \Leftarrow \mathtt{u}][y \Leftarrow \mathtt{u}'] \models R_A(\mathbf{car}\, x, \mathbf{car}\, y)$ et $\Phi(|\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}), |\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}'))$. D'où $|\mathbf{car}|^{\mathcal{M}}(\mathtt{u}) \sim_\omega |\mathbf{car}|^{\mathcal{M}}(\mathtt{u}')$ (car et R_A définit l'égalité intentionnelle pour le type A) et $|\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}) \sim_{\omega_n} |\mathbf{cdr}|^{\mathcal{M}}(\mathtt{u}')$ (hypothèse d'induction). En utilisant les équations, on trouve alors $\mathtt{b} \sim_\omega \mathtt{b}'$ et $\mathtt{v} \sim_{\omega_n} \mathtt{v}'$. Finalement, on en déduit $\mathtt{u} \sim_{\omega_{n+1}} \mathtt{u}'$ en utilisant le fait que $\mathcal{M}$ est syntaxiquement intentionnel pour le type S_A (théorème 5.18). ■

On pourrait montrer un tel résultat pour tous les types de données définis par la méthode du chapitre 5. Le schéma de coïnduction est donc une solution admissible. En revanche, il présente certaines caractéristiques qui montrent qu'il n'est pas vraiment naturel dans notre système de type :

- Pour tout type utilisant plus d'un constructeur, on doit utiliser les destructeurs du type pour écrire le schéma de coïnduction. Ceci brise la similarité entre types de données finies et types de données infinies.
- Pour tout modèle $\mathcal{M}$, le schéma de coïnduction C_D sur le type D vérifie : $\mathcal{M} \models \forall x \forall y (\neg D[x] \rightarrow C_D[x, y])$. Ceci implique que la transitivité ne peut pas être prouvée sous la forme : $\forall x \forall y \forall z (C_D[x, z], C_D[z, y] \rightarrow C_D[x, y])$, mais doit être écrite

  $$\forall x \forall y \forall z \left(D[x], D[y], D[z], C_D[x, z], C_D[z, y] \rightarrow C_D[x, y] \right)$$

  (Ce problème n'apparaît pas dans les systèmes de type utilisant des quantificateurs bornés)

8.4 Une autre approche.

Nous allons présenter un autre type de définition de l'égalité utilisant des points fixes. L'idée de cette définition est la même que celle qui a conduit à la définition des types de données.

On peut par exemple partir de la définition de l'égalité sur les entiers comme la plus petite relation R tel que l'on ait $\mathrm{R}(0, 0)$ et $\mathrm{R}(x, y) \rightarrow \mathrm{R}(\mathbf{s}x, \mathbf{s}y)$. Cela peut s'écrire :

$$\mathrm{N}'[x, y] = \forall X \left(X\,(\mathbf{0}, \mathbf{0}), \forall z \forall z' \left(X\,(z, z') \rightarrow X\,(\mathbf{s}z, \mathbf{s}z') \right) \rightarrow X\,(y, y') \right)$$

De la même manière que pour les entiers, on peut écrire cette définition en utilisant un plus petit point fixe. On obtient alors :

$$\mathrm{N}'[x, x'] = \mu K \lambda y \lambda y' \forall X \left(X\,(\mathbf{0}, \mathbf{0}), \forall z \forall z' \left(K\,(z, z') \rightarrow X\,(\mathbf{s}z, \mathbf{s}z') \right) \rightarrow X\,(y, y') \right) \langle x, x' \rangle$$

En utilisant un plus grand point fixe, on obtient la formule suivante pour l'égalité des streams :

$$\mathrm{S}'_{N'}[x, x'] \quad = \quad \nu K \lambda y \lambda y' \forall X \left(F[K, X] \rightarrow X\,(y, y') \right) \langle x, x' \rangle \text{ où :}$$

$$F[K,X] \quad = \quad \forall a \forall s \forall a' \forall s' \left(N'[a,a'], K\,(s,s') \rightarrow X\left(\mathbf{cs}\,(a,s)\,,\mathbf{cs}\,(a',s')\right)\right)$$

On peut généraliser cette définition à tous les types de données construits dans le chapitre 5. La formule définissant l'égalité sur un type D s'obtient en remplaçant les variables de prédicat unaire par des variables de prédicat binaire dans la formule définissant le type D. En reprenant les notations de la section 5.4, et en supposant que l'on a déjà défini les égalités $D'_1, \ldots, D'_m$ sur les types $D_1 \ldots D_m$ intervenant dans la définition du type D, on définit l'égalité sur ce type de la manière suivante :

On se donne $r+1$ variables de prédicats d'arité 2 : $X, L_1, \ldots, L_r$. Pour chaque constructeur $\mathbf{c}_i$ d'arité n_i, on définit une formule C'_i de la manière suivante :

$$C'_i = \forall a_1 \ldots a_{n_i} a'_1 \ldots a'_{n_i} \left(A'_1\,(a_1,a'_1)\,,\ldots,A'_{n_i}(a_{n_i},a'_{n_i}) \rightarrow X\left(\mathbf{c}_i(a_1 ... a_{n_i}), \mathbf{c}_i(a'_1 ... a'_{n_i})\right)\right)$$

avec $A'_j = D'_k$ si $T_{i,j} = D_k$ et $A'_j = L_{r_{i,j}}$ si $T_{i,j} = D$.

On définit alors les formules suivantes par récurrence :

- On pose :
$$F'_r[L_1,\ldots,L_r,x,x'] = \forall X \left(C'_1,\ldots,C'_n \rightarrow X\,(x,x')\right)$$
- Pour $0 \leq j < r$ avec $\epsilon_{j+1} = -$, on pose :
$$F'_j[L_1,\ldots,L_j,x,x'] = \mu L_{j+1} \lambda x \lambda x' F'_{j+1}[L_1,\ldots,L_{j+1},x,x']\langle x,x'\rangle$$
- Pour $0 \leq j < r$ avec $\epsilon_{j+1} = +$, on pose :
$$F'_j[L_1,\ldots,L_j,x,x'] = \nu L_{j+1} \lambda x \lambda x' F'_{j+1}[L_1,\ldots,L_{j+1},x,x']\langle x,x'\rangle$$

La formule $D'[x,x']$ qui définit l'égalité sur le type D est alors :

$$D'[x,x'] = F'_0[x,x']$$

La formule $D'[x,y]$ définissant l'égalité et la formule $D[x]$ définissant le type lui-même sont reliées par les propriétés suivantes :

$$\vdash \forall x \forall y (D'[x,y] \rightarrow D[x]) \quad \vdash \forall x \forall y (D'[x,y] \rightarrow D[y]) \quad \vdash \forall x (D[x] \rightarrow D'[x,x])$$
$$\vdash \forall x \forall y (D'[x,y] \rightarrow D'[y,x]) \quad \vdash \forall x \forall y \forall z (D'[x,y], D'[y,z] \rightarrow D'[x,z])$$

Ces propriétés montrent que $D'[x,y]$ définit une relation d'équivalence sur les éléments du type D et sont laissées au lecteur. Nous allons plutôt nous intéresser aux propriétés sémantiques de ces formules. En fait, nous allons montrer que cette définition de l'égalité est non seulement intentionnelle, mais que l'on a aussi un très fort contrôle sur les termes extraits d'une preuve de l'égalité. On exprime cela au travers de la définition suivante :

Définition 8.7. Une relation $R[x,y]$ définit une *égalité autonome* sur le type D, si pour tout modèle intentionnel pour D et pour toute interprétation σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre, on a :

$$\mathtt{t} \in \left|R[x,y]\right|^{\sigma[x \Leftarrow \mathtt{v}][y \Leftarrow \mathtt{w}]} \quad \text{ssi} \quad \mathtt{t} \sim_\omega \mathtt{v} \sim_\omega \mathtt{w} \text{ et } \mathcal{M}[x \Leftarrow \mathtt{v}][y \Leftarrow \mathtt{w}] \models R[x,y]$$

Proposition 8.8. *Si les formules $D'_1, \ldots, D'_m$ définissent des égalités autonomes (resp. intentionnelles) pour les types $D_1, \ldots, D_m$, alors D' définit une égalité autonome (resp. intentionnelle) sur le type D.*

Démonstration : Cette preuve est quasiment identique à la preuve du théorème 5.17. Ceci est due à la similarité entre les formules D et D'. Définissons les ensembles suivants :

- $\mathcal{C}'_i(\Phi_1, \ldots, \Phi_r) = \{(|\mathbf{c}_i|^{\mathcal{M}}(\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}), |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{t'}_1, \ldots, \mathtt{t'}_{n_i})) \setminus (\mathtt{t}_j, \mathtt{t}'_j) \in \Phi_{r_{i,j}}$ si $T_{i,j} = D, \mathcal{M}[x \Leftarrow \mathtt{t}_j][x' \Leftarrow \mathtt{t'}_j] \models D'_k[x, x']$ si $T_{i,j} = D_k\}$
- $\Theta'_r(\Phi_1, \ldots, \Phi_r) = \bigcup_{0<i\leq n} \mathcal{C}'_i(\Phi_1, \ldots, \Phi_r)$
- $\Theta'_j(\Phi_1, \ldots, \Phi_j) = \bigcap\{\Phi \setminus \Theta'_{j+1}(\Phi_1, \ldots, \Phi_j, \Phi) \subseteq \Phi\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = -$
- $\Theta'_j(\Phi_1, \ldots, \Phi_j) = \bigcup\{\Phi \setminus \Phi \subseteq \Theta'_{j+1}(\Phi_1, \ldots, \Phi_j, \Phi)\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = +$

En utilisant une preuve analogue à celle du lemme 5.6, on obtient :

$$\mathcal{M}[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u'}] \models D'[x, y] \text{ ssi } (\mathtt{u}, \mathtt{u'}) \in \Phi'_0$$

De la même manière, on définit aussi :

- $\mathtt{t} \in \Delta'_r(\Psi_1, \ldots, \Psi_r)(\mathtt{u}, \mathtt{u'})$ ssi il existe un entier i et des termes $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$, $\mathtt{u}_1, \ldots, \mathtt{u}_{n_i}$, $\mathtt{u}'_1, \ldots, \mathtt{u}'_{n_i}$ tel que
  - $\mathtt{u} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}_1, \ldots, \mathtt{u}_{n_i})$
  - $\mathtt{u'} \sim_\beta |\mathbf{c}_i|^{\mathcal{M}}(\mathtt{u}'_1, \ldots, \mathtt{u}'_{n_i})$
  - $\mathtt{t} \sim_\beta \lambda \mathtt{f}_1 \ldots \lambda \mathtt{f}_n(\mathtt{f}_i\ \mathtt{t}_1 \ldots \mathtt{t}_{n_i})$ ($\mathtt{f}_1, \ldots, \mathtt{f}_n$ n'étant pas libre dans $\mathtt{t}_1, \ldots, \mathtt{t}_{n_i}$)
  - Pour tout j, si $T_{i,j} = D_k$ (type déjà défini), $\mathtt{t}_j \in |D'_k(x, x')|^{\sigma[x \Leftarrow \mathtt{u}_j][x' \Leftarrow \mathtt{u}'_j]}$
  - Pour tout j, si $T_{i,j} = D$ (paramètre inductif), $\mathtt{t}_j \in \Psi_{r_{i,j}}(\mathtt{u}_j, \mathtt{u}'_j)$
- $\Delta'_j(\mathtt{u}, \mathtt{u'}) = \bigcap\{\Psi(\mathtt{u}, \mathtt{u'}) \setminus \Delta_{j+1}(\Psi_1, \ldots, \Psi_j, \Psi) \leq \Psi\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = -$
- $\Delta'_j(\mathtt{u}, \mathtt{u'}) = \bigcup\{\Psi(\mathtt{u}, \mathtt{u'}) \setminus \Psi \leq \Delta_{j+1}(\Psi_1, \ldots, \Psi_j, \Psi)\}$ si $0 \leq j < r$ et $\epsilon_{j+1} = +$

Par une preuve analogue à celle du lemme 5.10, on prouve :

$$\mathtt{t} \in \left|D'[x, y]\right|^{\sigma[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u'}]} \text{ ssi } \mathtt{t} \in \Delta_0(\mathtt{u}, \mathtt{u'})$$

On prouve aussi un résultat analogue au lemme 5.12. c'est à dire :

$$\Delta_0(\mathtt{u}, \mathtt{u'}) \neq \emptyset \text{ ssi } (\mathtt{u}, \mathtt{u'}) \in \Theta_0$$

En utilisant l'hypothèse que les relations $D'_1, \ldots, D'_m$ définissent des égalités autonomes sur les types $D_1, \ldots, D_m$, on obtient l'analogue du lemme 5.16 :

$$\mathtt{t} \in \Delta_0(\mathtt{u}, \mathtt{u'}) \text{ ssi } \mathtt{t} \sim_\omega \mathtt{u} \sim_\omega \mathtt{u'} \text{ et } \Delta_0(\mathtt{u}, \mathtt{u'}) \neq \emptyset$$

En mettant bout à bout toutes ces équivalences, et en supposant que les relations $D'_1, \ldots, D'_m$ définissent des égalités autonomes sur les types $D_1, \ldots, D_m$, on obtient :

$$\mathtt{t} \in \left|D'[x, y]\right|^{\sigma[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u'}]} \text{ ssi } \mathtt{t} \sim_\omega \mathtt{u} \sim_\omega \mathtt{u'} \text{ et } \mathcal{M}[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u'}] \models D'[x, y]$$

Ce qui correspond à la définition de "$D'[x, y]$ définit une égalité autonome pour D".

On montre aussi de manière similaire que si $D'_1, \ldots, D'_m$ définissent des égalités intentionnelles sur les types $D_1, \ldots, D_m$, on obtient :

$$(\mathtt{u}, \mathtt{u}') \in \Theta_0 \text{ ssi } \mathtt{u} \in D^{\mathcal{M}}, \mathtt{u}' \in D^{\mathcal{M}} \text{ et } \mathtt{u} \sim_\omega \mathtt{u}'.$$

D'où l'on déduit que si $D'_1, \ldots, D'_m$ définissent des égalités intentionnelles sur les types $D_1, \ldots, D_m$, D' définit une égalité intentionnelle pour D. ■

8.5 Types de données avec égalité.

Cette notion d'égalité autonome est très voisine de la notion de type de données. En fait, une telle notion peut être utilisée directement pour extraire des programmes sans utiliser la définition du type elle-même. (On utilise la formule $D'[x, x]$ en lieu et place de $D[x]$). On va maintenant généraliser cette notion et l'utiliser pour programmer.

Définition 8.9. On dira qu'un modèle $\mathcal{M}$ est intentionnel pour un *type de donnée avec égalité* défini par une formule $D[x, x']$ à deux variables libres x, x' (du premier ordre) si et seulement si pour toute interprétation pleine σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre, on a :

$$\mathtt{u} \in \left|D[x, x']\right|^{\sigma[x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}']} \text{ si et seulement si } \mathtt{u} \sim_\omega \mathtt{t} \text{ et } \mathcal{M}[x \Leftarrow \mathtt{t}][x \Leftarrow \mathtt{t}] \models D[x, x']$$

Commençons par montrer les propriétés suivantes :

Proposition 8.10. *Si $\mathcal{M}$ est syntaxiquement intentionnel pour le type de données D, ainsi que pour les types de données avec égalité $D'_1, \ldots, D'_m$, alors $\mathcal{M}$ est intentionnel pour le type $D'[x, y]$ construit dans la section précédente.*

Démonstration : La preuve, comme la proposition, sont quasiment identiques à celle de la proposition 8.8. La seule différence est que, sous les hypothèses que $D'_1, \ldots, D'_m$ sont des types de données avec égalité dans le modèle $\mathcal{M}$, et que $\mathcal{M}$ est syntaxiquement intentionnel pour un type de données D, on a besoin de prouver seulement :

$$\mathtt{t} \in \Delta_0(\mathtt{u}, \mathtt{u}') \text{ ssi } \mathtt{t} \sim_\omega \mathtt{u} \text{ et } \Delta_0(\mathtt{u}, \mathtt{u}') \neq \emptyset$$

Ce qui ne pose pas de problème. ■

Proposition 8.11. *Si le modèle $\mathcal{M}$ est intentionnel pour le type de données avec égalité $D[x, y]$ alors il est intentionnel pour le type de données $D[x, x]$.*

Démonstration : Soit un modèle $\mathcal{M}$ intentionnel pour le type de données avec égalité $D[x, y]$. Choisissons une interprétation σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. Par définition, on a :

$$\mathtt{u} \in \left|D[x, x]\right|^{\sigma[x \Leftarrow \mathtt{t}]} \text{ si et seulement si } \mathtt{u} \sim_\omega \mathtt{t} \text{ et } \mathcal{M}[x \Leftarrow \mathtt{t}] \models D[x, x]$$

Ce qui correspond exactement à la définition 5.2. ■

La définition de type de données avec égalité n'impose pas que la relation $D[x,y]$ corresponde à la ω-équivalence. C'est à dire que $D[x,y]$ n'est pas nécessairement une égalité intentionnelle pour le type $D[x,x]$.

Le type $D[x,y]$ correspond donc plutôt au quotient du type $D[x,x]$ par la relation $D[x,y]$. En effet, considérons les définitions suivantes :

Définition 8.12. Si le modèle $\mathcal{M}$ est intentionnel pour le type de données avec égalité $D[x,y]$, on définit l'ensemble des éléments du type dans le modèle $\mathcal{M}$ que l'on note $D^{\mathcal{M}}$, comme l'ensemble des termes du λ-calcul $\mathtt{t}$ vérifiant :

$$\mathcal{M}[x \Leftarrow \mathtt{t}] \models D[x,x]$$

Autrement dit, on a $D^{\mathcal{M}} = Sat^{x}_{\mathcal{M}}(D[x,x])$.

Définition 8.13. Sous les mêmes conditions, on définit l'égalité sur le type D, notée $\sim^{\mathcal{M}}_{D}$ par :

$$t \sim^{\mathcal{M}}_{D} t' \text{ ssi } \mathcal{M}[x \Leftarrow \mathtt{t}][x' \Leftarrow \mathtt{t}'] \models D[x,x']$$

La proposition suivante montre alors que si l'on extrait les programmes non plus de la preuve de totalité de la fonction, mais de la preuve de sa compatibilité avec la relation, le type D correspond au quotient de $D^{\mathcal{M}}$ par la relation $\sim^{\mathcal{M}}_{D}$.

Proposition 8.14. *Considérons un langage comprenant un symbole de fonction n-aire* $\mathbf{f}$ *satisfaisant un ensemble d'équations* $\mathcal{E}$*. Considérons un terme* $\mathtt{t}$ *tel que :*

$$\vdash \mathtt{t} : \forall x_1 \forall x'_1 \ldots \forall x_n \forall x'_n \left(D_1[x_1,x'_1], \ldots, D_n[x_n,x'_n] \rightarrow E[\mathbf{f}(x_1,\ldots,x_n), \mathbf{f}(x'_1,\ldots,x'_n)] \right)$$

Alors, si l'on considère un modèle $\mathcal{M}$ *intentionnel pour les types de données avec égalité* $D_1,\ldots,D_n$ *et* E*, ainsi que* n *termes* $\mathtt{u}_1,\ldots,\mathtt{u}_n$ *éléments respectifs des types* $D_1,\ldots,D_n$*, alors* $(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n)$ *est un élément du type* E *et* $(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n) \sim_\omega |f|^{\mathcal{M}}(\mathtt{u}_1,\ldots,\mathtt{u}_n)$ *(Ceci signifie que* $\mathtt{t}$ *est bien un terme calculant la fonction* $|f|^{\mathcal{M}}$*).*

De plus, cette fonction f *est compatible avec les relations* $\sim^{\mathcal{M}}_{D_1}, \sim^{\mathcal{M}}_{D_n}$ *et* $\sim^{\mathcal{M}}_{E}$*. C'est à dire que l'on a :*

$$\mathtt{u}_1 \sim^{\mathcal{M}}_{D_1} \mathtt{u}'_1 \ldots \mathtt{u}_n \sim^{\mathcal{M}}_{D_n} \mathtt{u}'_n \textit{ implique } |f|^{\mathcal{M}}(\mathtt{u}_1,\ldots,\mathtt{u}_n) \sim^{\mathcal{M}}_{E} |f|^{\mathcal{M}}(\mathtt{u}'_1,\ldots,\mathtt{u}'_n)$$

Démonstration : Considérons une interprétation pleine σ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. Par le lemme de conservation (3.14), on trouve :

$$\mathtt{t} \in \left| \forall x_1 \ldots \forall x_n \forall x'_n \left(D_1[x_1,x'_1], \ldots, D_n[x_n,x'_n] \rightarrow E[\mathbf{f}(x_1,\ldots,x_n), \mathbf{f}(x'_1,\ldots,x'_n)] \right) \right|^{\sigma}$$

Considérons des termes $\mathtt{u}_1,\ldots,\mathtt{u}_n,\mathtt{u}_1{}',\ldots,\mathtt{u}_n{}'$ tel que pour $1 \leq i \leq n$, $\mathcal{M}[x \Leftarrow \mathtt{u}_i][x \Leftarrow \mathtt{u}'_i] \models D_i[x,x']$. Par définition de l'interprétation, on trouve :

$$(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n) \in \left| E[\mathbf{f}(x_1,\ldots,x_n), \mathbf{f}(x'_1,\ldots,x'_n)] \right|^{\sigma[\overrightarrow{x} \Leftarrow \overrightarrow{\mathtt{u}}][\overrightarrow{x'} \Leftarrow \overrightarrow{\mathtt{u}'}]}$$

Ce qui donne :

$$(\mathtt{t}\ \mathtt{u}_1 \ldots \mathtt{u}_n) \in \left| E[x,x'] \right|^{\sigma\left[x \Leftarrow |f|^{\mathcal{M}}(\mathtt{u}_1,\ldots,\mathtt{u}_n)\right]\left[x' \Leftarrow |f|^{\mathcal{M}}(\mathtt{u}_1{}',\ldots,\mathtt{u}_n{}')\right]}$$

Or, $\mathcal{M}$ étant intentionnel pour E, on trouve

$$(\mathtt{t}\ \mathtt{u}_1 \dots \mathtt{u}_n) \sim_\omega |f|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_n) \text{ et}$$
$$\mathcal{M}\left[x \Leftarrow |f|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_n)\right]\left[x' \Leftarrow |f|^{\mathcal{M}}(\mathtt{u}_1{}', \dots, \mathtt{u}_n{}')\right] \models E[x, x']$$

Ce qui implique $(\mathtt{t}\ \mathtt{u}_1 \dots \mathtt{u}_n) \sim_\omega |f|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_n)$ et $|f|^{\mathcal{M}}(\mathtt{u}_1, \dots, \mathtt{u}_n) \sim_E^{\mathcal{M}} |f|^{\mathcal{M}}(\mathtt{u}'_1, \dots, \mathtt{u}'_n)$ ■

Nous remarquons que l'on n,a pas eu besoin d'imposer à la relation $D'[x, y]$ de définir une relation d'équivalence partielle. Néanmoins, cela est nécessaire si l'on veut que la relation $\sim_D^{\mathcal{M}}$ soit une relation d'équivalence sur $D^{\mathcal{M}}$, bien que cela n'intervienne pas dans la preuve ci-dessus.

8.6 Définition des types quotients.

Nous avons vu que les types de données avec égalité correspondait à des quotients. Nous allons voir maintenant comment le connecteur restriction (cf appendice C) permet de définir n'importe quel quotient d'un type de données. On exprime cela par la proposition suivante :

Proposition 8.15. *Si $\mathcal{M}$ est un modèle intentionnel pour le type D, et si $R[x, y]$ est une formule sans variable de prédicat libre, alors $\mathcal{M}$ est intentionnel pour le type de données avec égalité $D_{/R}[x, y]$ défini par $D_{/R}[x, y] = D[x] \upharpoonright R[x, y]$.*

Démonstration : Choisissons un modèle intentionnel pour le type D et une $\mathcal{P}(\Lambda)^\star$-interprétation (interprétation standard pour la sémantique étendue) $(\sigma, \mathcal{M}_\sigma)$ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. Par définition de l'interprétation, on a $\mathtt{t} \in |D_{/R}[x, y]|^{\sigma[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}']}$ si et seulement si $\mathtt{t} \in |D[x]|^{\sigma[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}']}$ et $\mathcal{M}_\sigma[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}'] \models R[x, y]$. Or, F n'ayant pas de variable de prédicat libre, et $\mathcal{M}_\sigma$ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre, ceci est équivalent à $\mathcal{M}[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}'] \models R[x, y]$.

Donc, $\mathcal{M}$ étant intentionnel pour le type D, on trouve $\mathtt{t} \in |D_{/R}[x, y]|^{\sigma[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}']}$ si et seulement si $\mathtt{t} \sim_\omega \mathtt{u}$, $\mathcal{M}[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}'] \models D[x]$ et $\mathcal{M}[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}'] \models R[x, y]$. Or, par définition de l'interprétation classique, ceci est équivalent à $\mathtt{t} \sim_\omega \mathtt{u}$ et $\mathcal{M}[x \Leftarrow \mathtt{u}][x \Leftarrow \mathtt{u}'] \models D_{/R}[x, y]$. ■

Là encore, il faut remarquer que l'on ne définit réellement un type quotient que si $R[x, y]$ est une relation d'équivalence sur le type D, bien que l'on n'utilise pas cette hypothèse dans la preuve précédente. De plus, dans la pratique, on simplifie les preuves si $R[x, y]$ implique $D[y]$, hypothèse souvent nécessaire pour prouver la compatibilité d'une fonction.

On va maintenant examiner un ou deux exemples de types quotients. On peut définir les entiers relatifs comme un quotient sur les couples d'entiers naturels. Pour cela, on utilise le type $N[x]$ défini au chapitre 2 ainsi que le type produit d'objet de type A et B, définit par $A \times B[x] = \forall X(\forall a \forall b(A[a], A[b] \to X\,\mathbf{pair}(a, b)) \to X\,x)$ (**pair** étant une constante de fonction binaire). On utilise aussi les constantes de fonctions **add**,**fst** et **snd** tel que $\mathcal{E} \vdash \mathbf{add}(\mathbf{0}, y) = y$, $\mathcal{E} \vdash \mathbf{add}(\mathbf{s}\,x, y) = \mathbf{s}(\mathbf{add}(x, y))$, $\mathcal{E} \vdash \mathbf{fst}(\mathbf{pair}(a, b)) = a$ et $\mathcal{E} \vdash \mathbf{snd}(\mathbf{pair}(a, b)) = b$. On considère aussi l'égalité $x = y$ définie par l'égalité de Leibnitz. On peut alors définir le type des entiers relatifs par

$$Z[x, y] = N \times N[x] \upharpoonright \left(N \times N[y] \wedge \mathbf{add}\left(\mathbf{fst}\,(x), \mathbf{snd}\,(y)\right) = \mathbf{add}\left(\mathbf{fst}\,(y), \mathbf{snd}\,(x)\right)\right)$$

Supposons que l'on a déjà obtenu les termes suivants :

$$\vdash \mathtt{pair} = \lambda\mathtt{a}\lambda\mathtt{b}\lambda\mathtt{f}(\mathtt{f\ a\ b}) : \forall a \forall b (A[a], B[b] \to A \times B[\mathbf{pair}(a,b)])$$
$$\vdash \mathtt{fst} = \lambda\mathtt{c}(\mathtt{c}\ \lambda\mathtt{a}\lambda\mathtt{b}\,\mathtt{a}) : \forall c (A \times B[c] \to A[\mathbf{fst}(c)])$$
$$\vdash \mathtt{snd} = \lambda\mathtt{c}(\mathtt{c}\ \lambda\mathtt{a}\lambda\mathtt{b}\,\mathtt{b}) : \forall c (A \times B[c] \to B[\mathbf{snd}(c)])$$
$$\vdash \mathtt{add} = !\lambda\mathtt{r}\lambda\mathtt{n}\lambda\mathtt{m}(\mathtt{n\ m}\ \lambda\mathtt{p}(\mathtt{succ}\ (\mathtt{r\,p\,m}))) : \forall n \forall m (N[n], N[m] \to N[\mathbf{add}(n,m)])$$

Il est alors facile d'obtenir le programme suivant pour l'addition des entiers relatifs, définit par l'équation $\mathbf{add}(\mathbf{pair}(x,y), \mathbf{pair}(x',y')) = \mathbf{pair}(\mathbf{add}(x,x'), \mathbf{add}(y,y'))$:

$$\vdash \mathtt{addz} : \forall x \forall x' \forall y \forall y' \left(Z[x,x'], Z[y,y'] \to Z[\mathbf{add}\,(x,y)\,, \mathbf{add}\,(x',y')]\right) \text{ avec}$$

$$\mathtt{addz} = \lambda\mathtt{x}\lambda\mathtt{y}\left(\mathtt{pair}\ \left(\mathtt{add}\ (\mathtt{fst\,x})\ (\mathtt{fst\,y})\right)\left(\mathtt{add}\ (\mathtt{snd\,x})\ (\mathtt{snd\,y})\right)\right)$$

En fait, pour réaliser cette preuve, on se place sous le contexte $\Gamma = \mathtt{x} : Z[x,x'], \mathtt{y} : Z[y,y']$. De ce contexte, on déduit d'une part $R[x,x']$ et $R[y,y']$ (formules sans contenu algorithmique) où $R[x,x'] = N \times N[x] \wedge N \times N[y] \wedge \mathbf{add}(\mathbf{fst}(x), \mathbf{snd}(y)) = \mathbf{add}(\mathbf{fst}(y), \mathbf{snd}(x))$, d'où l'on déduit $R[\mathbf{add}(x,y), \mathbf{add}(x',y')]$ en utilisant la commutativité et l'associativité de l'addition. D'autre part, on obtient $\mathtt{x} : N \times N[x]$ et $\mathtt{y} : N \times N[y]$ d'où l'on déduit :

$$\left(\mathtt{pair}\ \left(\mathtt{add}\ (\mathtt{fst\,x})\ (\mathtt{fst\,y})\right)\left(\mathtt{add}\ (\mathtt{snd\,x})\ (\mathtt{snd\,y})\right)\right) : N \times N[\mathbf{add}\,(x,y)\,, \mathbf{add}\,(x',y')].$$

On obtient alors le résultat voulu en utilisant l'introduction du connecteur restriction.

Conclusion

Notre propos était d'étendre le système AF_2, afin de permettre l'utilisation de types de données infinies. Dans quelle mesure avons-nous réussi ? Pour répondre à cette question, nous allons analyser nos résultats de deux points de vue : les résultats théoriques généraux, et ceux concernant la programmation et les types de données.

Les résultats généraux.

Nous avons tout d'abord prouver la viabilité de notre système, grâce au lemme de réduction qui assure la préservation du type pendant la réduction, et grâce au lemme de conservation qui prouve la correction de la notion sémantique d'interprétation. De plus, l'appendice B montre que cette extension est bien conservative par rapport à la logique intuitionniste du second ordre.

Ainsi nous restons avec un système construit au-dessus de la logique du second ordre que l'on peut étudier en utilisant les mêmes outils que pour AF_2.

Une question qui se pose pour tout système de type concerne l'étude de la normalisation des termes typés. Comme prévu, on n'a aucune propriété de normalisation forte ou faible, puisque l'un des buts du système est justement de pouvoir typer des termes non-normalisables. Notre étude a donc été centrée sur la résolubilité et la ω-résolubilité des termes typés (cf appendice A pour les définitions).

Dans le cas du système n'utilisant que le plus grand point fixe, on a prouvé que si l'on n'utilise pas de formules de la forme $\nu X \lambda \overline{x}\, F\langle \overline{t} \rangle$ où X est la variable la plus à droite dans F (par exemple $\nu X X$), alors tous les termes typés sont bien ω-résolubles.

Par contre, le plus petit point fixe pose un problème plus important, du fait d'une interaction avec la définition du faux par $\forall X\ X$ en logique du second ordre. On a néanmoins réussi à obtenir une condition sémantique qui assure la ω-résolubilité des termes typés, que l'on peut interpréter comme l'interdiction d'utiliser l'absurde sous toutes ses formes (on définit ainsi, en quelque sorte, la logique minimale du second ordre).

A quel point ces résultats sont-ils dépendants du choix de nos règles ? On aurait pu choisir des règles n'utilisant pas de combinateur de point fixe pour le connecteur μ (ce genre de règles permet de définir un système fortement normalisable avec types inductifs). Mais il semble difficile de trouver des règles analogues pour le plus grand point fixe. Il est pourtant souhaitable de traiter les deux connecteurs μ et ν de manière similaire.

On peut se demander s'il est possible de trouver un système où plus grand et plus petit point

fixe sont traités uniformément, et tel qu'une condition plus simple assure la ω-résolubilité des termes typés.

Types de données et programmation.

Toutefois, ces problèmes n'interviennent pas dans la justification de la méthode de programmation, qui repose entièrement sur la notion sémantique de type de données. L'extension de la définition de Krivine fonctionne parfaitement, et chaque élément d'un type de données infinies trouve bien une unique représentation pour la ω-équivalence.

Ce résultat sur les types de données permet de montrer que l'on extrait bien un programme correct à partir d'une preuve de totalité d'une fonction définie par des équations.

De plus, les deux exemples d'utilisation montrent que l'on peut réellement utiliser ces types de données infinies, tout en mettant en évidence de nouveaux concepts :

- Le premier concerne les types de données utilisant les deux points fixes. Le crible d'Eratosthène fait naturellement apparaître le type des listes infinies de 0 et 1 comprenant toujours une infinité de 1 (pour exprimer l'existence d'une infinité de nombres premiers). Ces types de données sont-ils de simples anecdotes ou peuvent-ils être d'une utilité plus générale ?

- Le deuxième concept important, révélé par l'utilisation des types de données infinies, est celui du constructeur muet. Nous avons montré que l'ajout d'un constructeur unaire, que l'on efface à la lecture du résultat, permet de définir une représentation non-standard des types de données, autorisant la programmation de toutes les fonctions récursives, séparant ainsi preuve de terminaison et extraction de programme.

  De plus, on pourrait voir et implémenter ce constructeur muet comme une unité de temps. Les programmes utilisant le constructeur muet seraient alors aussi efficaces que les autres.

  On peut aussi se demander quel classe d'algorithmes l'on peut programmer en utilisant cette méthode (Il est facile de voir que cette méthode permet de programmer des algorithmes tels que le "ou parallèle").

L'utilisation des types de données avec égalité introduite dans l'appendice 8, est une autre voix prometteuse. Il est possible de les utiliser pour obtenir des représentations des types quotients. Ainsi, la notion de type de données avec égalité, appliquée à des types infinies, permet au moins théoriquement de représenter dans notre système des objets mathématiques complexes, tels les nombres réels. Peut-on obtenir des représentations informatiquement raisonnables de ces notions ?

Annexe A

Termes ω-résolubles et ω-équivalence.

On introduit, dans cet appendice, les deux notions essentielles de λ-calcul pur utilisées dans cette thèse. On suppose que le lecteur est déjà familier avec les notions de α, β et η-équivalence.

Définition A.1 ω-équivalence. La ω-équivalence est définie en introduisant la famille de relations d'équivalence $\{\sim_{\omega_n}\}_{n\in N}$ vérifiant :

- $\mathtt{u} \sim_{\omega_0} \mathtt{v}$ pour tous termes $\mathtt{u}$ et $\mathtt{v}$.
- $\mathtt{u} \sim_{\omega_{n+1}} \mathtt{v}$ si et seulement si $\mathtt{u} \sim_\beta \lambda\overline{\mathtt{x}}(\mathtt{x_i}\ \mathtt{u_1} \ldots \mathtt{u_p})$ est équivalent à il existe p termes $\mathtt{v_1} \ldots \mathtt{v_p}$ tel que $\mathtt{v} \sim_\beta \lambda\overline{\mathtt{x}}(\mathtt{x_i}\ \mathtt{v_1} \ldots \mathtt{v_p})$ et pour tout $i \in \{1, \ldots, p\}$, $\mathtt{u_i} \sim_{\omega_n} \mathtt{v_i}$

La ω-équivalence est alors l'intersection de toutes les relations $\sim_{\omega_n}$.

De même, on définit la $\omega\eta$-équivalence en remplaçant dans la définition ci-dessus $\sim_\beta$ par $\sim_{\beta\eta}$.

Voici quelques résultats sur la ω-équivalence (que l'on trouvera dans [1]) :

- Tous les termes non résolubles sont ω-équivalents et $\omega\eta$-équivalents.
- Restreinte aux termes normalisables, la ω-équivalence et la β-équivalence sont identiques, de même pour la $\omega\eta$-équivalence et la $\beta\eta$-équivalence.
- La ω-équivalence correspond exactement à l'égalité des arbres de Böhm. On en déduit que deux termes sont ω-équivalents si et seulement s'ils ne sont pas séparables ($\mathtt{u}$ et $\mathtt{v}$ sont séparables si et seulement si il existe une substitution τ et un terme $\mathtt{t}$ tel que $(\mathtt{t}\ \tau\mathtt{u}) \sim_\beta \lambda\mathtt{x}\lambda\mathtt{y}\ \mathtt{x}$ et $(\mathtt{t}\ \tau\mathtt{v}) \sim_\beta \lambda\mathtt{x}\lambda\mathtt{y}\ \mathtt{y}$).
- La ω-équivalence passe au contexte.

Définition A.2 Termes ω-résolubles. La notion de termes ω-résolubles est définie en introduisant la famille d'ensembles de termes $\{\Omega_n\}_{n\in N}$ vérifiant:

- $\Omega_0 = \Lambda$.
- $\mathtt{u} \in \Omega_{n+1}$ si et seulement s'il existe p termes $\mathtt{u_1} \ldots \mathtt{u_p}$ tel que $\mathtt{u} \sim_\beta \lambda\mathtt{x_1} \ldots \lambda\mathtt{x_n}(\mathtt{x_i}\ \mathtt{u_1} \ldots \mathtt{u_p})$ et pour tout $i \in \{1, \ldots, p\}$, $\mathtt{u_i} \in \Omega_n$

L'ensemble des termes ω-résolubles est alors défini comme l'intersection de tous les ensembles Ω_n.

On remarquera que les termes ω-résolubles sont les termes qui n'ont pas de "bottom" dans leur arbre de Böhm.

La propriété importante de la ω-résolubilité est la compatibilité avec la β-équivalence (qui se déduit de la compatibilité avec la ω-équivalence) : si `u` et `v` sont β-équivalents, `u` est ω-résoluble si et seulement si `v` l'est aussi.

Annexe B

Codage des points fixes.

Nous allons montrer dans cet appendice, que les points fixes n'ajoutent rien à la logique intuitionniste du second ordre. Plus précisément, on va prouver que le codage usuel et bien connu des plus petits et plus grands points fixes en logique du second ordre est équivalent aux connecteurs μ et ν. Ceci implique que d'un point de vue logique, l'extension que nous proposons est conservative. Son seul intérêt est donc l'optimisation du contenu algorithmique des programmes.

Un autre intérêt de cet appendice est la justification de la définition d'occurrence visiblement positive ou négative. En effet, on montrera la correction de ces règles en se ramenant à un cas plus simple (cf lemme B.5). La complexité de cette preuve montrant l'économie que l'on fait au niveau du terme extrait, en utilisant nos règles de points fixes.

On commence par introduire un certain nombre de notations pour la logique du second ordre :

Notation : On notera LI_2 la logique intuitionniste du second ordre (la logique sous-jacente à AF_2), et on notera $LI_2^{\mu\nu}$ notre extension avec point fixe (la logique sous-jacente à $AF_2^{\mu\nu}$). On notera $\Gamma \vdash_L F$ un séquent prouvé dans la logique L.

Notation : On définit la conjonction, la disjonction, l'équivalence, la quantification existentielle, l'inclusion, et l'égalité de la manière suivante :

$$\begin{array}{rcl} F_1 \wedge \ldots \wedge F_n & = & \forall X((F_1, \ldots, F_n \to X) \to X) \\ F \leftrightarrow G & = & (F \to G) \wedge (G \to F) \\ \exists X F & = & \forall Y(\forall X(F \to Y) \to Y) \\ \lambda \overline{x}\, F \subset \lambda \overline{x}\, G & = & \forall \overline{x}(F \to G) \\ \overline{u} = \overline{v} & = & \forall X(X\overline{u} \to X\overline{v}) \end{array}$$

Dans le cas des variable, la simplification $X = \lambda \overline{x}\, X\overline{x}$ permettra d'écrire $X \subset Y$ pour $\forall \overline{x}(X\overline{x} \to Y\overline{x})$.

Le lecteur vérifiera aisément que ces définitions donnent les connecteurs usuels.

Notation : De plus, on introduit les deux notations suivantes :

$$\begin{array}{rcl} \mu_0 X \lambda \overline{x}\, F\langle \overline{t} \rangle & = & \forall X((\lambda \overline{x}\, F \subset X) \to X\overline{t}) \\ \nu_0 X \lambda \overline{x}\, F\langle \overline{t} \rangle & = & \exists X((X \subset \lambda \overline{x}\, F) \wedge X\overline{t}) \end{array}$$

Définition B.1. Définissons par induction, la traduction $F \mapsto F^\circ$ suivante, qui associe à chaque formule de $LI_2^{\mu\nu}$ une formule de LI_2 (sans μ et ν) :

$$\begin{array}{rclcrcl} X(\overline{t})^\circ & = & X(\overline{t}) & & (F \to G)^\circ & = & F^\circ \to G^\circ \\ (\forall x F)^\circ & = & \forall x F^\circ & & (\forall X F)^\circ & = & \forall X F^\circ \\ (\mu X \lambda \overline{x}\, F\langle \overline{t}\rangle)^\circ & = & \mu_0 X \lambda \overline{x}\, F^\circ\langle \overline{t}\rangle & & (\nu X \lambda \overline{x}\, F\langle \overline{t}\rangle)^\circ & = & \nu_0 X \lambda \overline{x}\, F^\circ\langle \overline{t}\rangle \end{array}$$

Le fait que notre extension n'ajoute rien à la logique du second ordre s'exprime alors par la proposition suivante :

Proposition B.2. *Pour toute formule F et tout contexte Γ (du système avec point fixe) on a :*

$$\Gamma \vdash_{LI_2^{\mu\nu}} F \text{ si et seulement si } \Gamma^\circ \vdash_{LI_2} F^\circ$$

De cette proposition, on déduit le corollaire suivant :

Corollaire B.3. *Notre système est une extension conservative de la logique intuitionniste du second ordre. C'est à dire que pour tout contexte Γ et toute formule F de LI_2, on a :*

$$\Gamma \vdash_{LI_2^{\mu\nu}} F \text{ si et seulement si } \Gamma \vdash_{LI_2} F$$

Démonstration : Le corollaire découle de la proposition B.2 car $\Gamma^\circ = \Gamma$ et $F^\circ = F$ si Γ et F n'utilisent pas les points fixes. ■

Afin de prouver la proposition B.2, prouvons d'abord quelques lemmes :

Lemme B.4. *Si X n'a que des occurrences positives dans F, on peut prouver les séquents suivants dans LI_2 :*

1. $\vdash_{LI_2} \forall Y \forall Y'((Y \subset Y') \to (F[X \Leftarrow Y] \to F[X \Leftarrow Y']))$
2. $\vdash_{LI_2} \forall \overline{x}(\mu_0 X \lambda \overline{x}\, F\langle \overline{x}\rangle \leftrightarrow F[X \Leftarrow \mu_0 X \lambda \overline{x}\, F])$
3. $\vdash_{LI_2} \forall \overline{x}(\nu_0 X \lambda \overline{x}\, F\langle \overline{x}\rangle \leftrightarrow F[X \Leftarrow \nu_0 X \lambda \overline{x}\, F])$

Démonstration : Montrons que chacun de ces séquents est vrai :

1. La preuve est facile et très classique, il suffit de montrer par induction sur la structure de la formule F que si X n'a que des occurrences positives (resp. négatives) dans F, on a $\vdash_{LI_2} \forall Y \forall Y'((Y \subset Y') \to (F[X \Leftarrow Y] \to F[X \Leftarrow Y']))$ (resp. $Y' \subset Y$). Ceci ne pose aucun problème.

2. Montrons d'abord le sens droite-gauche de l'équivalence. Notons $\Phi = \lambda \overline{x} \mu_0 X \lambda \overline{x}\, F\langle \overline{x}\rangle$. Sous les hypothèses $F[X \Leftarrow \Phi]$ (i), et $\forall \overline{x}(F \to X\overline{x})$ (ii), il faut montrer $X\overline{x}$. Or, par définition de μ_0, de (ii), on déduit $\forall \overline{x}(\mu_0 X \lambda \overline{x}\, F\langle \overline{x}\rangle \to X\overline{x})$. Donc, X n'ayant que des occurrences positives dans F, on obtient $\forall \overline{x}(F[X \Leftarrow \Phi] \to F)$. Finalement, on obtient F en utilisant (i), puis le résultat cherché avec (ii).

   Pour le sens gauche-droite, supposons $\mu_0 X \lambda \overline{x}\, F\langle \overline{x}\rangle$. En appliquant le sens droite-gauche et car X n'a que des occurrences positives dans F, on obtient $\forall x F[X \Leftarrow \lambda \overline{x}\, F[X \Leftarrow \Phi]] \to F[X \Leftarrow \Phi]$. D'où $F[X \Leftarrow \Phi]$, en substituant $\lambda \overline{x}\, F[X \Leftarrow \Phi]$ à X dans $\mu_0 X \lambda \overline{x}\, F\langle \overline{x}\rangle$.

3. La preuve est similaire au cas précédent, on prouve d'abord le sens gauche-droite, puis on en déduit la réciproque.

■

Lemme B.5. *Si X est visiblement négative dans F, alors il existe une formule G où la variable X n'est pas libre, tel que :*

$$\vdash_{LI_2} \forall X \left(F \leftrightarrow (X \subset \lambda \overline{x}\, G)\right)$$

De même, Si X est visiblement positive dans F, alors il existe une formule G où la variable X n'est pas libre, tel que :

$$\vdash_{LI_2} \forall X \left(F \leftrightarrow (\lambda \overline{x}\, G \subset X)\right)$$

Démonstration : Montrons d'abord le cas où X est visiblement positive dans F. Par définition, on est dans l'un des cas suivants :

- $F = X(\overline{t})$ donc avec $G = (\overline{x} = \overline{t})$, on a bien $\vdash_{LI_2} F \leftrightarrow (\lambda \overline{x}\, G \subset X)$.
- $F = H \rightarrow F'$, X étant visiblement positive dans F' et sans occurrence dans H. Par hypothèse d'induction, on trouve une formule G' tel que $\vdash_{LI_2} F' \leftrightarrow (\lambda \overline{x}\, G' \subset X)$. Il suffit donc de prendre $G = G' \wedge H$ et on obtient le résultat voulu.
- $F = \forall \chi F'$, avec $\chi \neq X$ et X visiblement positive dans F'. Par hypothèse d'induction, on trouve une formule G' tel que $\vdash_{LI_2} F' \leftrightarrow (\lambda \overline{x}\, G' \subset X)$. Or, on a pour toutes formules A et B, χ non libre dans B implique $\vdash_{LI_2} \forall \chi (A \rightarrow B) \leftrightarrow (\exists \chi\, A \rightarrow B)$. Donc il suffit de prendre $G = \exists \chi G'$.

Montrons maintenant le cas où X est visiblement négative dans F :

- $F = H \rightarrow F'$, X étant visiblement négative dans F' et sans occurrence dans H. Par hypothèse d'induction, on trouve une formule G' tel que $\vdash_{LI_2} F' \leftrightarrow (X \subset \lambda \overline{x}\, G')$. Il suffit donc de prendre $G = H \rightarrow G'$ et on obtient le résultat voulu.
- $F = X\overline{t} \rightarrow F'$, X étant sans occurrence dans F'. Il suffit de prendre $G = (\overline{x} = \overline{t}) \rightarrow F'$ et l'on obtient $\vdash_{LI_2} F \leftrightarrow (X \subset \lambda \overline{x}\, G)$.
- $F = \forall \chi F'$, avec $\chi \neq X$ et X visiblement positive dans F'. Par hypothèse d'induction, on trouve une formule G' tel que $\vdash_{LI_2} F' \leftrightarrow (X \subset \lambda \overline{x}\, G')$. Or, on a pour toutes formules A et B, χ non libre dans A implique $\vdash_{LI_2} \forall \chi (A \rightarrow B) \leftrightarrow (A \rightarrow \forall \chi\, B)$. Donc il suffit de prendre $G = \forall \chi G'$.

■

Lemme B.6. *Si X, d'arité n n'a que des occurrences positives dans G et est visiblement négative (resp. positive) dans F, on a :*

$$\vdash_{LI_2} \forall X \left(F \rightarrow F[X \Leftarrow \lambda \overline{x}\, G]\right) \rightarrow F[X \Leftarrow \mu_0 X \lambda \overline{x}\, G] \text{ (resp. } \nu_0\text{)}$$

Démonstration : Commençons par le cas du plus grand point fixe. En utilisant le lemme B.5,on trouve H où X n'est pas libre et tel que $\vdash_{LI_2} \forall X (F \leftrightarrow \forall \overline{x} (H \rightarrow X\overline{x}))$. Il suffit de prouver :

$$\vdash_{LI_2} \forall X \left(\forall \overline{x}\, (H \rightarrow X\overline{x}) \rightarrow \forall \overline{x}\, (H \rightarrow G)\right) \rightarrow \forall \overline{x} \left(H \rightarrow \nu_0 X \lambda \overline{x}\, G \langle \overline{x} \rangle\right)$$

Supposons $\forall X (\forall \overline{x} (H \rightarrow X\overline{x}) \rightarrow \forall \overline{x} (H \rightarrow G))$. On en déduit $\forall \overline{x} (H \rightarrow G[X \Leftarrow \lambda \overline{x}\, H]))$ (en substituant $\lambda \overline{x}\, H$ à X). Donc, on obtient $H \rightarrow (\forall \overline{x} (H \rightarrow G[X \Leftarrow \lambda \overline{x}\, H])) \wedge H)$, D'où $H \rightarrow \nu_0 X \lambda \overline{x}\, G \langle \overline{x} \rangle$.

Le cas du plus petit point fixe se démontre de manière analogue. ■

Lemme B.7. *Les deux équivalences suivantes sont prouvables dans notre système :*

- $\vdash_{LI_2^{\mu\nu}} \mu X \lambda \overline{x}\, F\langle \overline{t}\rangle \leftrightarrow \mu_0 X \lambda \overline{x}\, F\langle \overline{t}\rangle$
- $\vdash_{LI_2^{\mu\nu}} \nu X \lambda \overline{x}\, F\langle \overline{t}\rangle \leftrightarrow \nu_0 X \lambda \overline{x}\, F\langle \overline{t}\rangle$

Démonstration : Ces deux équivalences se démontrent sans problème avec nos règles de points fixes :

- $\forall X(\forall \overline{x}(F \to X\overline{x}) \to X\overline{t})$ implique

  $$\forall \overline{x}\left(F[X \Leftarrow \mu X \lambda \overline{x}\, F] \to \mu X \lambda \overline{x}\, F\langle \overline{x}\rangle\right) \to \mu X \lambda \overline{x}\, F\langle \overline{t}\rangle$$

  Et donc implique $\mu X \lambda \overline{x}\, F\langle \overline{t}\rangle$ car $\forall \overline{x}(F[X \Leftarrow \mu X \lambda \overline{x}\, F] \to \mu X \lambda \overline{x}\, F\langle \overline{x}\rangle)$ découle de la factorisation du point fixe.

- $\vdash_{LI_2^{\mu\nu}} \mu X \lambda \overline{x}\, F\langle \overline{t}\rangle \to \forall X(\forall \overline{x}(F \to X\overline{x}) \to X\overline{t})$ se démontre en utilisant la règle du plus petit point fixe, à partir de

  $$\left(X\overline{t} \to \forall X\left(\forall \overline{x}\,(F \to X\overline{x}) \to X\overline{t}\right)\right) \to \left(F\left[\overline{x} \Leftarrow \overline{t}\right] \to \forall X\left(\forall \overline{x}\,(F \to X\overline{x}) \to X\overline{t}\right)\right)$$

  Pour démontrer cette formule, on doit montrer $X'\overline{t}$ sous les hypothèses suivantes :

  1. $X\overline{t} \to \forall X(\forall \overline{x}(F \to X\overline{x}) \to X\overline{t})$
  2. $F[\overline{x} \Leftarrow \overline{t}]$
  3. $\forall \overline{x}(F[X \Leftarrow X'] \to X'\overline{x})$

  De 1 et 3 on déduit $X\overline{t} \to X'\overline{t}$. Or, X n'ayant que des occurrences positives dans F on en déduit $F[\overline{x} \Leftarrow \overline{t}] \to F[X \Leftarrow X'][\overline{x} \Leftarrow \overline{t}]$. Puis, par 2 on obtient $F[X \Leftarrow X'][\overline{x} \Leftarrow \overline{t}]$, puis, $X'\overline{t}$ en utilisant 3.

- Le cas du plus grand point fixe se démontre de manière analogue.

■

On peut maintenant démontrer la proposition B.2:

Démonstration : Pour le sens gauche-droite ($\Gamma \vdash_{LI_2^{\mu\nu}} F$ implique $\Gamma \vdash_{LI_2} F^\circ$), on prouve le résultat par induction sur la preuve de $\Gamma \vdash_{LI_2^{\mu\nu}} F$. En fait, la seule chose à faire, est de montrer que les règles de points fixes sont correctes pour le codage des points fixes utilisé dans notre traduction. Or, la factorisation et le développement des points fixes correspondent aux cas 2 et 3 du lemme B.4, et les règles du plus petit et du plus grand point fixe correspondent au lemme B.6.

Pour le sens droite-gauche, Il faut remarquer que $\Gamma^\circ \vdash_{LI_2} F^\circ$ implique $\Gamma^\circ \vdash_{LI_2^{\mu\nu}} F^\circ$. Il suffit donc de prouver que pour toute formule F, $\vdash_{LI_2^{\mu\nu}} F \leftrightarrow F^\circ$. On montre cela par induction sur la structure de la formule F. Le seul cas non trivial étant le cas des points fixes, qui découle du lemme B.7. ■

Annexe C

Omission de contenu algorithmique.

Dans beaucoup de cas, les programmes extraits sont inefficaces car certaines parties de la preuve n'ont pas réellement de contenu algorithmique mais sont pourtant réduites. Pour tenter de pallier à ce défaut, nous allons ajouter de nouveaux connecteurs dont les règles omettent le contenue algorithmique d'une partie de la preuve.

C.1 Extension de la syntaxe.

Pour gérer l'omission de contenu algorithmique, on va reprendre l'idée du connecteur restriction de Michel Parigot [22]. En tout premier lieu, il est nécessaire d'utiliser deux types de séquents :

- des *séquents logiques*, c'est à dire sans aucune indication algorithmique, de la forme :

$$F_1, \ldots, F_n \vdash F.$$

- des *séquents algorithmiques*, de la forme :

$$\mathtt{x}_1 : F_1, \ldots, \mathtt{x}_p : F_p, F_{p+1}, \ldots, F_n \vdash \mathtt{t} : F.$$

  On remarquera la présence possible *d'hypothèses logiques* dans le contexte (les formules $F_{p+1}, \ldots F_n$).

On ajoute à la logique un connecteur binaire que l'on notera $F \restriction G$. Ce connecteur est logiquement équivalent à une conjonction, mais qui ne conserve que le contenu algorithmique de la formule F.

On verra qu'il est aussi utile d'ajouter une implication sans contenu algorithmique, que l'on notera $F \rightsquigarrow G$. Ce connecteur exprime que l'on n'a pas besoin du contenu algorithmique de la preuve de F pour déduire G.

Si $\Gamma = \mathtt{x}_1 : F_1, \ldots, \mathtt{x}_p : F_p, F_{p+1}, \ldots, F_n$ est un contexte algorithmique, on notera Γ° le contexte logique associé : $\Gamma^\circ = F_1, \ldots, F_p, F_{p+1}, \ldots, F_n$.

On a aussi besoin de considérer un terme $\star$ du λ-calcul ne se réduisant pas. On définit donc $\star$ comme une λ-variable que l'on ne déclarera jamais dans un contexte.

Les règles sont alors étendues de la manière suivante :

- Les règles sur les séquents logiques sont les règles usuelles de la logique classique du second ordre, sachant que le connecteur $\restriction$ est une conjonction et que les deux connecteurs $\rightarrow$ et $\rightsquigarrow$ sont des implications obéissant aux mêmes règles.
- Pour les séquents algorithmiques, les anciens connecteurs conservent les règles que l'on a introduites au chapitre 2, tandis que les nouveaux connecteurs utilisent les règles qui suivent.
- introduction de $\star$:
$$\frac{\Gamma^\circ \vdash \forall X\, X}{\Gamma \vdash \star : F}\star_i$$
- introduction de la restriction :
$$\frac{\Gamma \vdash \mathtt{t} : F \qquad \Gamma^\circ \vdash G}{\Gamma \vdash \mathtt{t} : F \restriction G}\restriction_i$$
- élimination de la restriction :
$$\frac{\Gamma \vdash \mathtt{t} : F \restriction G}{\Gamma \vdash \mathtt{t} : F}\restriction^l_e \qquad \frac{\Gamma \vdash \mathtt{t} : F \restriction G}{\Gamma^\circ \vdash G}\restriction^r_e$$
- introduction de $\rightsquigarrow$:
$$\frac{\Gamma, F \vdash \mathtt{t} : G}{\Gamma \vdash \mathtt{t} : F \rightsquigarrow G}\rightsquigarrow_i$$
- élimination de $\rightsquigarrow$:
$$\frac{\Gamma \vdash \mathtt{t} : F \rightsquigarrow G \qquad \Gamma^\circ \vdash F}{\Gamma \vdash \mathtt{t} : G}\rightsquigarrow_e$$

Proposition C.1. *Le lemme 2.1 (subject reduction) reste valide en présence des nouvelles règles.*

Démonstration : La preuve n'est pas très différente. On reprend l'induction sur la construction de la preuve :

- Si la dernière règle est l'une des règles originales à l'exception de la règle $\rightarrow_e$ (élimination de l'implication), la preuve reste inchangée.
- Pour les règles $\restriction_i$, $\restriction^l_e$, $\rightsquigarrow_i$ et $\rightsquigarrow_e$, l'hypothèse d'induction appliquée à la prémisse donne le résultat voulu.
- Le cas de la règle $\star_i$ ne se présente pas car $\star$ ne se réduit pas.
- Celui de la règle $\restriction^r_e$ non plus car sa conclusion est un séquent logique.
- Pour le cas de $\rightarrow_e$ (élimination de l'implication), il faut étudier comment on élimine les occurrences des règles $\restriction_i$, $\restriction^l_e$, $\rightsquigarrow_i$ et $\rightsquigarrow_e$ qui pourraient séparer l'introduction et l'élimination de l'implication. En fait, ces règles sont traitées comme les règles de quantification. En effet, elles permutent bien avec les règles Eq, μ_d, ν_d, μ_f et ν_f (équations, développement et factorisation des points fixes). De plus, comme pour les règles de quantification, on peut éliminer une introduction suivie d'une élimination sans problème.

■

C.2 Extension de la sémantique.

Pour étendre la sémantique, il suffit de considérer qu'à toute $\mathcal{C}$-interprétation σ est associé un modèle classique $\mathcal{M}_\sigma$ coïncidant avec σ sur les termes logiques. Ainsi, on donne la définition suivante :

Définition C.2. On appellera $\mathcal{C}^\star$-interprétation. un couple $(\sigma, \mathcal{M}_\sigma)$ où σ est une $\mathcal{C}$-interprétation et $\mathcal{M}_\sigma$ est un modèle classique coïncidant avec σ sur les termes logiques.

Pour toute $\mathcal{C}^\star$-interprétation $(\sigma, \mathcal{M}_\sigma)$, on définit simultanément $|F|^\sigma$ et $\mathcal{M}_\sigma \models F$ par induction sur la construction de la formule F :

- $\mathcal{M}_\sigma \models F$ est définie de manière usuelle, sachant que l'interprétation classique de $\restriction$ est celle d'une conjonction, tandis que l'interprétation classique de $\rightsquigarrow$ est celle d'une implication.
- $|F|^\sigma$ est définie par induction comme dans le chapitre 3, si le connecteur extérieur de F est l'un des anciens connecteurs.
- $|F \restriction G|^\sigma = |F|^\sigma$ si $\mathcal{M}_\sigma \models G$, sinon $|F \restriction G|^\sigma = \emptyset$
- $|F \rightsquigarrow G|^\sigma = |G|^\sigma$ si $\mathcal{M}_\sigma \models F$, sinon $|F \rightsquigarrow G|^\sigma = \Lambda$

On remarque que la satisfaction $\mathcal{M}_\sigma \models F$ est indépendante de l'interprétation $|F|^\sigma$, tandis que $|F|^\sigma$ subit l'influence de la satisfaction classique $\mathcal{M}_\sigma \models F$ au travers des nouveaux connecteurs.

On prouve alors que la notion sémantique d'interprétation, ainsi étendue, reste correcte :

Proposition C.3. *Le lemme de conservation reste valide sous la forme suivante : Pour toute $\mathcal{C}^\star$-interprétation $(\sigma, \mathcal{M}_\sigma)$, si l'on prouve $\mathtt{x}_1 : F_1, \ldots, \mathtt{x}_p : F_p, F_{p+1}, \ldots, F_n \vdash \mathtt{t} : F$, si $\mathcal{M}_\sigma \models F_i$ pour $1 \leq i \leq n$ et si $\mathtt{u_i} \in |F_i|^\sigma$ pour $1 \leq i \leq p$, alors on a $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F|^\sigma$.*

Démonstration : Là encore, la preuve change peu. On utilise en premier lieu la correction de la sémantique classique : si l'on prouve $F_1, \ldots, F_n \vdash F$ alors, pour tout modèle classique $\mathcal{M}$, $\mathcal{M} \models F_i$ pour $1 \leq i \leq n$ implique $\mathcal{M} \models F$ (les nouveaux connecteurs ne changent rien à ce résultat, puisque d'un point de vue logique ils se comportent comme une conjonction et une implication).

Choisissons une $\mathcal{C}^\star$-interprétation $(\sigma, \mathcal{M}_\sigma)$ vérifiant les hypothèses de la proposition. On prouve alors le résultat par induction sur la construction de la dérivation en examinant la dernière règle appliquée. Dans le cas des anciennes règles, la preuve reste inchangée. Pour les nouvelles règles, on est dans l'un des cas suivants (on notera $\Gamma = \mathtt{x}_1 : F_1, \ldots, \mathtt{x}_p : F_p, F_{p+1}, \ldots, F_n$) :

- Si la dernière appliquée est l'introduction de $\star$:

$$\frac{\Gamma^\circ \vdash \forall X\, X}{\Gamma \vdash \star : F}\star_i$$

  Il ne peut y avoir de modèle classique $\mathcal{M}$ satisfaisant le contexte Γ° (sinon aurait $\mathcal{M} \models \forall X\, X$. La proposition est donc vrai car on ne peut pas satisfaire l'hypothèse.

- Si la dernière appliquée est l'introduction de la restriction :

$$\frac{\Gamma \vdash \mathtt{t} : F \qquad \Gamma^\circ \vdash G}{\Gamma \vdash \mathtt{t} : F \restriction G}\restriction_i$$

En appliquant l'hypothèse d'induction, on trouve $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F|^\sigma$ et $\mathcal{M}_\sigma \models G$. Par définition de l'interprétation du connecteur $\restriction$, on trouve bien $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F \restriction G|^\sigma$.

- élimination de la restriction :

$$\frac{\Gamma \vdash \mathtt{t} : F \restriction G}{\Gamma \vdash \mathtt{t} : F} \restriction_e^l \qquad \frac{\Gamma \vdash \mathtt{t} : F \restriction G}{\Gamma^\circ \vdash G} \restriction_e^r$$

Par hypothèse d'induction, on trouve $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F \restriction G|^\sigma$. Puis, par définition de l'interprétation du connecteur restriction, on en déduit $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F|^\sigma$ et $\mathcal{M}_\sigma \models G$.

- introduction de $\rightsquigarrow$:

$$\frac{\Gamma, F \vdash \mathtt{t} : G}{\Gamma \vdash \mathtt{t} : F \rightsquigarrow G} \rightsquigarrow_i$$

Si $\mathcal{M}_\sigma \models F$, l'hypothèse d'induction donne $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |G|^\sigma$. D'où le résultat par définition de l'interprétation. Sinon, on a $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F \rightsquigarrow G|^\sigma = \Lambda$.

- élimination de $\rightsquigarrow$:

$$\frac{\Gamma \vdash \mathtt{t} : F \rightsquigarrow G \qquad \Gamma^\circ \vdash F}{\Gamma \vdash \mathtt{t} : G} \rightsquigarrow_e$$

Par hypothèse d'induction, on trouve $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |F \rightsquigarrow G|^\sigma$ et $\mathcal{M}_\sigma \models F$. Donc, en appliquant la définition de l'interprétation on trouve $\mathtt{t}[\overline{\mathtt{x}} \Leftarrow \overline{\mathtt{u}}] \in |G|^\sigma$. ■

C.3 Définition et utilisation des sous-types.

On peut utiliser le connecteur restriction pour définir des sous-types. On exprime cela par la proposition suivante :

Proposition C.4. *Si $\mathcal{M}$ est un modèle intentionnel pour le type $D[x]$ et si F est une formule sans variable de prédicat libre (x ainsi que d'autres variables du premier ordre peuvent apparaître dans F), alors $\mathcal{M}$ est intentionnel pour le type $D[x] \restriction F$*

Démonstration : Choisissons un modèle intentionnel pour le type D et une $\mathcal{P}(\Lambda)^\star$-interprétation (interprétation standard pour la sémantique étendue) $(\sigma, \mathcal{M}_\sigma)$ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre. Par définition de l'interprétation, on a $\mathtt{t} \in |D[x] \restriction F|^{\sigma[x \Leftarrow \mathtt{u}]}$ si et seulement si $\mathtt{t} \in |D[x]|^{\sigma[x \Leftarrow \mathtt{u}]}$ et $\mathcal{M}_\sigma[x \Leftarrow \mathtt{u}] \models F$. Or, F n'ayant pas de variable de prédicat libre, et $\mathcal{M}_\sigma$ coïncidant avec $\mathcal{M}$ sur les termes du premier ordre, ceci est équivalent à $\mathcal{M}[x \Leftarrow \mathtt{u}] \models F$.

Donc, en appliquant le fait que $\mathcal{M}$ est intentionnel pour le type D, on trouve $\mathtt{t} \in |D[x] \restriction F|^{\sigma[x \Leftarrow \mathtt{u}]}$ si et seulement si $\mathtt{t} \sim_\omega \mathtt{u}$, $\mathcal{M}[x \Leftarrow \mathtt{u}] \models D[x]$ et $\mathcal{M}[x \Leftarrow \mathtt{u}] \models F$. Or, par définition de l'interprétation classique, ceci est équivalent à $\mathtt{t} \sim_\omega \mathtt{u}$ et $\mathcal{M}[x \Leftarrow \mathtt{u}] \models D[x] \restriction F$. ■

On va montrer un exemple d'utilisation simple qui met en évidence la nécessité du connecteur $\rightsquigarrow$. On considère le type des entiers $N[x]$ définit au chapitre 2, ainsi que la formule $x \neq \mathbf{0}$ définie par $\forall X(X\mathbf{0} \to Xx) \to \forall XX$. On considère aussi un symbole de fonction $\mathbf{p}$ tel que $\mathcal{E} \vdash \mathbf{p}(\mathbf{s}x) = x$. La fonction $\mathbf{p}$ est donc une fonction partielle pour le prédécesseur.

On va chercher un terme $\mathtt{t}$ tel que :

$$\vdash \mathtt{t} : \Big(N[x] \restriction x \neq 0\Big) \to N[\mathbf{p}x]$$

Pour cela, on suppose que l'on a $\Gamma = \mathtt{x} : N[x] \restriction x \neq 0$. En appliquant les règles d'élimination de la restriction, du développement du point fixe et de l'élimination du quantificateur universelle, on trouve :

$$\Gamma \vdash \mathtt{x} : \left(0 \neq 0 \rightsquigarrow N[\mathbf{p}0]\right), \forall y \left(N[y] \to \left(\mathbf{s}y \neq 0 \rightsquigarrow N[\mathbf{p}\,(\mathbf{s}y)]\right)\right) \to \left(x \neq 0 \rightsquigarrow N[\mathbf{p}x]\right)$$

Or, on prouve facilement $\vdash \star : 0 \neq 0 \rightsquigarrow N[\mathbf{p}0]$ et $\vdash \lambda \mathtt{p}\,\mathtt{p} : \forall y(N[y] \to (\mathbf{s}y \neq 0 \rightsquigarrow N[\mathbf{p}(\mathbf{s}y)]))$. Au total, on obtient :

$$\vdash \lambda \mathtt{x}\ (\mathtt{x}\ \star\ \lambda \mathtt{p}\,\mathtt{p}) : \left(N[x] \restriction x \neq 0\right) \to N[\mathbf{p}x].$$

Maintenant, si on essaye de prouver le même résultat sans utiliser le connecteur $\rightsquigarrow$, on ne peut plus dériver $x \neq 0 \rightsquigarrow N[\mathbf{p}x]$ par induction. Il faut utiliser une formule de la forme $(N[x] \restriction x \neq 0) \to N[\mathbf{p}x]$. On trouve alors

$$\vdash \lambda \mathtt{x}\ (\mathtt{x}\ \ \star\ \ \lambda \mathtt{p} \lambda \mathtt{c}\,\mathtt{p}\ \ \mathtt{x}) : \left(N[x] \restriction x \neq 0\right) \to N[\mathbf{p}x].$$

Or, le premier terme que l'on a obtenu est semblable au terme que l'on obtient pour une fonction prédécesseur totale (il suffit de remplacer $\star$ par `zero`). Par contre, le second terme est plus compliqué, on ne peut donc pas dire que le connecteur $\restriction$ suffise pour omettre le contenu non-algorithmique dans cette preuve.

C.4 L'induction généralisée.

En utilisant le connecteur restriction, Michel Parigot montre que pour toutes formules D et F tel que D définisse un type de données et $y \leq x$ définisse un ordre bien fondé sur le type D, on peut utiliser la règle suivante :

$$\frac{\Gamma \vdash \mathtt{t} : \forall x \left(\forall y \left(D[y] \restriction y \leq x \to F[y]\right) \to D[x] \to F[x]\right)}{\Gamma \vdash\ !\mathtt{t} : \forall x \left(D[x] \to F[x]\right)}$$

Cette règle permet d'utiliser le point fixe sans garder le contenu algorithmique de la preuve de décroissance lors des appels récursifs. Pour plus de détail sur cette règle, son utilisation et sa justification, se reporter à [22, 17].

Annexe D

Index des conventions et notations.

Afin de faciliter la lecture, nous allons détailler ici les conventions et les notations utilisées dans cette thèse.

- Notation d'intérêt général
  - $\emptyset$: l'ensemble vide.
  - $\mathcal{P}(E)$: l'ensemble des parties de l'ensemble E.
  - $E \to E'$: l'ensemble des fonctions de E dans E'.
  - $\bigcap E$: l'intersection des éléments de E (E étant un ensemble d'ensembles).
  - $\bigcup E$: la réunion des éléments de E (E étant un ensemble d'ensembles).
  - $\bigcap_{x \in E} f(x) : \bigcap \{f(x) \setminus x \in E\}$
  - $\bigcup_{x \in E} f(x) : \bigcap \{f(x) \setminus x \in E\}$
- Les termes logiques (ou termes du premier ordre).
  - Règle générale : tout en minuscule.
  - x, y, z, a, b, d, e, f : lettres minuscules utilisées pour les variables du premier ordre.
  - t, u, v, w : lettres minuscules utilisées pour désigner des termes logiques .
  - $\mathbf{0}, \mathbf{s}, \mathbf{car}, \ldots$: constantes de fonctions et d'individus (notées en gras).
  - $\mathbf{f}(t_1, \ldots, t_n)$: application d'une constante de fonction $\mathbf{f}$ à n termes logiques.
  - $\mathbf{s}t$ pour $\mathbf{s}(t)$: dans le cas des constantes unaires on omettra souvent les parenthèses.
  - $\overline{x}$ ou $\overline{t}$: pour désigner un vecteur de variables ou de termes du premier ordre. La taille du vecteur sera souvent implicite et imposée par le contexte (écrire $X\overline{t}$ implique que la taille du vecteur $\overline{t}$ est égale à l'arité de x). De plus, on supposera que l'on écrit $\forall \overline{x}$ ou $\lambda \overline{x}$ seulement si le vecteur $\overline{x}$ est composé de variables distinctes.

- Les formules.
  - Règle générale : tout en majuscule.
  - $K\ X\ Y\ Z$: lettres majuscules utilisées pour désigner des variables de prédicat.
  - $A\ B\ C\ F\ G\ H$: lettres majuscules utilisées pour désigner des formules.
  - $X(\overline{t})$: application d'une variable de prédicat à des termes logiques.
  - Les connecteurs utilisés pour construire les formules seront notés respectivement:
    - $A \to B$ pour l'implication.
    - $\forall x\, A$ et $\forall X\, A$ pour la quantification universelle.
    - $\mu X \lambda \overline{x}\, A\langle \overline{t} \rangle$ pour les types inductifs.
    - $\nu X \lambda \overline{x}\, A\langle \overline{t} \rangle$ pour les types coïnductifs.
  - On emploiera aussi les abréviations suivantes:
    - $X\,t$: application d'une variable de prédicat unaire à un terme logique.
    - $F_1, \ldots, F_n \to G$ pour $(F_1 \to (\ldots \to (F_n \to G)\ldots))$
    - $\forall x\, F \to G$ pour $((\forall x\, F) \to G)$
    - $\forall X\, F \to G$ pour $((\forall X\, F) \to G)$
  - $\mathrm{N}[t]$: utilisation d'une formule N à une variable libre appliquée au terme t.
  - $A[x \Leftarrow t]$: substitution d'une variable du premier ordre par un terme logique.
  - $A[X \Leftarrow \lambda x_1 \ldots x_n\, B]$: substitution d'une variable de prédicat d'arité n par une formule (où n variables sont distinguées).
  - $A\langle X \Leftarrow \lambda x_1 \ldots x_n\, B\rangle$: substitution physique (avec capture de variables) d'une variable de prédicat d'arité n par une formule (où n variables sont distinguées).
  - $A[X \Leftarrow Y]$ pour $A[X \Leftarrow \lambda \overline{x}\, Y(\overline{x})]$.
  - $A[X \Leftarrow \mu X \lambda \overline{x}\, B]$ (resp. ν) pour $A[X \Leftarrow \lambda \overline{x}\, \mu X \lambda \overline{x}\, B\langle \overline{x} \rangle]$ (resp. ν) (sorte de η-équivalence simplificatrice).
  - χ : pour désigner une variable d'ordre quelconque.
  - Φ : associée à χ pour désigner une expression substituable à χ (Φ sera un terme si χ est une variable du premier ordre et Φ sera de la forme $\lambda x_1 \ldots x_n\, A$ si χ est une variable de prédicat d'arité n).
- Les termes du λ-calcul.
  - Règle générale : tout en minuscule en utilisant une police de caractères style "machine à écrire'.
  - `a f r s x y z` : lettres minuscules utilisées pour désigner des variables du λ-calcul.

- $\mathtt{t\ u\ v\ w}$: lettres minuscules utilisées pour désigner des termes du λ-calcul.
- $\overline{\mathbf{f}}$: pour désigner un terme associé à un symbole de fonction (c'est à dire un terme extrait d'une preuve de totalité de la fonction $\mathbf{f}$).
- $\lambda\mathtt{x\ t}$: abstraction.
- $(\mathtt{t\ u_1\ \ldots\ u_n})$: application.
- $\mathtt{t}[\mathtt{x} \Leftarrow \mathtt{u}]$: substitution.
- Λ : l'ensemble des termes du λ-calcul, quotienté par $\sim_\beta$.
- $\mathcal{V}_\Lambda$: l'ensemble des variables du λ-calcul.

- Les séquents.
  - $\mathcal{E}$: ensemble d'équations entre termes logiques.
  - $\mathcal{E} \vdash t = u$: si l'équation $t = u$ appartient au système $\mathcal{E}$.
  - $\mathtt{x_1} : F_1, \ldots, \mathtt{x_n} : F_n \vdash \mathtt{t} : F$: un séquent.

- La sémantique.
  - $\mathcal{C}$: domaine de variation pour les variables de prédicat d'arité 0 ($\mathcal{C} \subset \mathcal{P}(\Lambda)$).
  - $\mathcal{C}_n$: domaine de variation pour les variables de prédicat d'arité n ($\mathcal{C}_n = \Lambda^n \to \mathcal{C}$).
  - σ : une interprétation.
  - $\sigma[\chi \Leftarrow \Phi]$: substitution de la valeur d'une variable χ dans l'interprétation σ.
  - $|t|^\sigma$: interprétation d'un terme (à valeur dans Λ).
  - $|F|^\sigma$: interprétation d'une formule (à valeur dans $\mathcal{C}$).
  - $|\lambda\chi_1 \ldots \chi_n\, F|^\sigma$: la fonction qui à $\Phi_1, \ldots, \Phi_n$ associe $|F|^{\sigma[\chi_i \Leftarrow \Phi_i]}$, où pour chaque i on a ou bien χ_i est une variable du premier ordre et $\Phi_i \in \Lambda$, ou bien χ_i est une variable de prédicat et $\Phi_i \in \mathcal{C}_n$.
  - $\Phi \leq \Phi'$: l'ordre canonique sur $\Lambda^n \to \mathcal{P}(\Lambda)$, défini par $\Phi \leq \Phi'$ si et seulement si $\Phi(\overline{\mathtt{t}}) \subseteq \Phi'(\overline{\mathtt{t}})$ pour tout $\overline{\mathtt{t}} \in \Lambda^n$.
  - $\mathcal{M} \models F$: $|F|^{\mathcal{M}} = \Lambda$, $\mathcal{M}$ étant une interprétation classique.
  - $Sat^{\overline{x}}_{\mathcal{M}}(F)$: $\{\overline{\mathtt{t}} \in \Lambda^n \setminus \mathcal{M}[\overline{x} \Leftarrow \overline{\mathtt{t}}] \models F\}$

Bibliographie.

[1] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[2] C. Böhm. Alcune proprieta delle forme $\beta\eta$-normali nel λk-calculus. Pubblicazioni 696, Instituto per le Applicazioni del Calcolo, Roma, 1968.

[3] A. Church. *The calculi of lambda-conversion*. Princeton University Press, 1941.

[4] T. Coquand. Infinite objects in type theory. In *Informal proceedings of the workshop on Types for Proofs and Programs*, 1993.

[5] T. Coquand and G. Huet. The calculus of construction. In *Information and Computation*, pages 241–262, 1988.

[6] N. de Bruijn. The mathematical language automath, its usage and some of its extensions. In *Symp. on automatic demonstration*, pages 29–61. Springer Verlag, 1970. Lecture Notes in Mathematics Vol. 125.

[7] J.-Y. Girard. The system F of variable types: fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[8] W. Howard. The formulae-as-types notion of construction. *To H.B. Curry: Essays on combinatory logic, λ-calculus and formalism*, pages 479–490, 1980.

[9] Jean-Louis Krivine. Un algorithme non typable dans le systme f. In *Comptes Rendus de l'Acadmie des Sciences de Paris*, volume 304 SÉrie I, pages 123–126, 1987.

[10] Jean-Louis Krivine. Opérateurs de mise en mémoire et traduction de Godël. Technical Report 3, Equipe de Logique de Paris 7, December 1989.

[11] Jean-Louis Krivine. *Lambda-Calcul : Types et Modèles*. Etudes et Recherches en Informatique. Masson, 1990. Available now in english version.

[12] Jean-Louis Krivine and Michel Parigot. Programming with proofs. *Inf. Process. Cybern.*, EIK 26(3):149–167, 1990.

[13] F. Leclerc and C. Paulin-Mohring. Programming with Streams in Coq. A case study : The Seve of Eratosthenes. In K. Petersson B. Nordström and G. Plotkin, editors, *Proceedings of the 1992 Workshop on Types for Proofs ad Programs*, pages 245–262, 1992.

[14] Daniel Leivant. Reasoning about functional programs and complexity classes associated with type disciplines. In *24th Annual Symposium on Foundations of Computer Science*,

volume 44, pages 460–469, 1983.

[15] Daniel Leivant. Typing and computational properties of lambda expressions. *Theoretical Computer Science*, 44:51–68, 1986.

[16] Daniel Leivant. Contracting proofs to programs. *Logic and Computer Science*, pages 279–327, 1990.

[17] P. Manoury and M. Simonot. *Des preuves de totalits de fonctions comme synthèse de programmes*. PhD thesis, Equipe de logique de Paris VII, CNRS URA 753, 1993.

[18] P. Martin-Löf. Lecture notes on the domain interpretation of type theory. In Programming Methodology Group, editor, *Workshop on the Semantics of Programming Languages*, Göteborg, Sweden, 1983. Chalmers University of Technology.

[19] Paul Francis Mendler. *Inductive definition in type theory*. PhD thesis, Cornell University, 1988.

[20] M. Parigot P. Manoury and M. Simonot. `ProPre` : a programming language with proofs. In *LPAR 92*, 1992.

[21] Michel Parigot. Programming with proofs: a second order type theory. *Lecture Notes in Computer Science*, 300, 1988. Communication at ESOP 88.

[22] Michel Parigot. Recursive programming with proofs. *Theoritical Computer Science*, 94:335–356, 1992.

[23] C. Paulin-Mohring. Inductive definitions in the calculus of constructions. Technical report, INRIA, 1989. Technical Report Number 110.

[24] Paul Rozière. Constant time reduction in λ-calculus. In Springer-Verlag, editor, *Mathematical Foundation of Computer Science*, Gdansk, 1993. Lecture Note in Computer Science.

[25] M. Tatsuta. Realizability interpretation of coinductive definitions and program synthesis using streams. In *Proceedings of the fifth generation computer systems*, pages 666–673. ICOT, 1992.