Bonnes démonstrations en déduction modulo

Guillaume Burel

Nancy-Université
Université Henri Poincaré

UFR STMIA **École doctorale IAEM Lorraine**

Département de formation doctorale en informatique

Bonnes démonstrations en déduction modulo

THÈSE

présentée et soutenue publiquement le 23 mars 2009

pour l'obtention du

doctorat de l'université Henri Poincaré

(spécialité informatique)

par

Guillaume Burel

Composition du jury

Président :	Delia Kesner	Professeur, université Denis Diderot, Paris, France
Rapporteurs :	Nachum Dershowitz	Professeur, université de Tel Aviv, Israël
	Gilles Dowek	Professeur, École polytechnique, Palaiseau, France
Examinateurs :	Patrick Blackburn	Directeur de recherche, Inria, Nancy, France
	Claude Kirchner	Directeur de recherche, Inria, Bordeaux, France (directeur de thèse)
	Daniel Leivant	Professeur, université de l'Indiana, Bloomington, ÉUA
	Dominique Mery	Professeur, université Henri Poincaré, Nancy, France
	Alexandre Miquel	Maître de conférences, université Denis Diderot, Paris, France

Laboratoire Lorrain de recherche en informatique et ses applications — UMR 7503

Loria

Mis en page avec LaTeX.

Remerciements

Je tiens à exprimer ma reconnaissance envers toutes les personnes grâce auxquelles ces trois années (et demi) de thèse se sont déroulées si plaisamment. Mes premiers remerciements sont évidemment pour mon directeur de thèse, Claude Kirchner, qui, il y a bientôt six ans de cela, et avec Olivier Bournez, me fit découvrir le monde de la recherche. Je tiens à souligner sa disponibilité, surtout en début de thèse où sa capacité à pointer des références intéressantes m'a été précieuse et où il m'a encouragé et aidé dans les différentes pistes que j'ai explorées. Si ses nouvelles responsabilités, et surtout son nouveau lieu de travail, ont bien entendu réduit la possibilité d'interactions scientifiques régulières, il avait déjà donné l'impulsion nécessaire qui m'a permis d'accomplir le travail décrit dans ce manuscrit, et il a su être disponible aux moments nécessaires.

Je souhaite aussi remercier tous les membres qui m'ont fait l'honneur d'accepter de faire partie de mon jury de soutenance. Répondre à leurs pertinentes questions me donnera du travail pour longtemps. Merci spécialement à Nachum Dershowitz et à Gilles Dowek d'avoir accepté de rapporter ce manuscrit. J'apprécie également leur intérêt constant pour mon travail.

Grand merci à tous les membres et anciens membres des équipes Prothéo puis Paréo. La bonne ambiance qui y règne favorise grandement les échanges et donc les avancées scientifiques. Merci en particulier à ceux que j'ai côtoyés alors qu'ils étaient encore thésards : Benjamin, Clara, Antoine, Florent, Fabrice, Germain, Colin, Anderson, Radu, Oana et Émilie. Merci à ceux qui auront bientôt à remplir une telle page : Clément, Paul, Cody, Claudia et Tony. Merci aussi à Jean-Christophe avec qui j'ai partagé le bureau vers la fin, à Pierre-Étienne qui dirige Paréo admirablement, à Hélène, Frédéric, Horatiu et Yves, et surtout à Chantal qui sait combattre efficacement les hydres administratives.

J'aimerais également remercier tous ceux qui travaillent ou ont travaillé autour de la déduction modulo et qui ont pu apporter un regard critique sur mes avancées : Claude et Gilles, que j'ai déjà cités, mais également Thérèse, Benjamin, Alexandre, Olivier, Richard, Denis, Lisa et Mathieu. Merci aussi aux personnes qui m'ont accompagné au cours de mes études : Rémi, Christophe, Gim, David, Samuel, Florent, Laurent et tous les autres.

Je voudrais témoigner ma gratitude envers ceux qui, volontairement ou non, m'ont fortement influencé : Herr Krameyer qui a inclinée durablement ma Weltanschauung, Kai qui m'a ouvert à de nouveaux horizons et M. Soyeur grâce à qui j'ai découvert ce que sont les mathématiques et l'informatique théorique. Je remercie également mes

parents et ma sœur qui m'ont toujours soutenu en me laissant libre de mes choix. Ces remerciements seraient néanmoins incomplets s'ils n'étaient en grande partie adressés à Céline, qui a pourtant tort de croire plus en moi qu'en elle-même.

Merci à ceux que j'aurais pu oublier, j'espère qu'ils ne s'en formaliseront pas.

Je remercie enfin tous ceux qui savent associer réécriture et beauté.

Table des matières

Extended abstract

Automation of mathematical reasoning has achieved a high level of maturity so that it is now possible to formalize big proofs like the one of the four-color theorem by Georges Gonthier (2005), or to prove the software correctness of the most modern airplanes (Patrick Cousot, 2007). If the automated search for a proof can be seen as a computation, the proofs themselves can include some calculations. For instance, the original proof by Kenneth Appel, Wolfgang Haken and John Koch (1977) of the four-color theorem uses a computer to check that a finite number of cases can be four-colored, the proof written by hand consisting in showing that these cases are enough to get the whole result. The completely formalized proof by Georges Gonthier (2005) with the help of Benjamin Werner in Coq[1] reflects also this fact, and makes extensive use of computations, through the computational-reflection method. It is therefore of higher importance to be able to represent computations faithfully in proof systems. In many modern proof assistants, computations are done either through axiomatization or with the help of a λ-calculus with inductive types. The former is not well adapted for proof search, because there is no clue what axiom is needed and at what moment in the proof. The latter may seem enough, because it is the base of functional programming language à la ML. However, this often constraints functions to be defined by structural induction, and one may want more liberality in these definitions. It is well accepted that rewriting provides a model for computations that is simple, flexible and executable, and that has been extensively studied. It is therefore no surprise that the integration of rewriting techniques in proof assistants is an active research field. Isabelle[2] offers for instance the possibility to simplify goals using rewrite rules.

The need to better understand interconnections between computation and deduction leads Gilles Dowek, Thérèse Hardin and Claude Kirchner (2003) to propose a formalism in which those are clearly identified. In the so called deduction modulo, deduction rules are applied modulo some congruence over formulæ that represents the computations. This congruence is better represented by means of a rewrite system which rewrites terms and also *formulæ*. Although this idea can theoretically be applied to any kind of deductive system, it is often used with inference systems for first-order logic, such as the natural deduction and the sequent calculus of Gerhard Gentzen (1934). This comes from the fact that higher-order logic can be encoded in first-order logic modulo, as shown by Gilles Dowek, Thérèse Hardin and Claude Kirchner (2001).

[1] http://coq.inria.fr/
[2] http://isabelle.in.tum.de/

The aim of this thesis is to study how the integration of computation in deductive systems, in particular through deduction modulo, may help at simplifying proofs. Numerous systems have been introduced since the early work of Gottlob Frege on the formalization of proofs. One of the principal objective of their conception was to simplify the building of proofs. Nevertheless, as far as we know, none of these takes explicitly into account the notion of simplicity of a proof, except the framework of the abstract canonical systems of Nachum Dershowitz and Claude Kirchner (2006); Maria Paola Bonacina and Nachum Dershowitz (2007*a*). Simplicity is but a crucial property of proofs, either from the point of view of a mathematician, which will seek the mathematical beauty of simple proofs, or from the point of view of automated theorem provers, which find simple proofs more easily. David Hilbert seemed to agree with this idea, because he apparently envisaged to include the study of the simplicity of proofs in his famous list of twenty-three problems, as found out by Rüdiger Thiele (2003). The notion of simplicity is of course highly subjective, and may vary whether one considers the position of a mathematician or the one of an automated theorem-proving tool. In this thesis, we focus mostly on the perspective of mechanized theorem provers. We have particularly highlighted three criteria.

We have firstly investigated the admissibility of cut. A deduction mechanism needs to be analytic, that is, inference rules should only need the information provided by the current goal. This is for instance the case in the sequent calculus without cut, because rules are determined by the formulæ occurring in the sequent. This implies that the proof search is more restricted, because it is not needed to guess extra formulæ as in the case of the cut rule. The tableau method is based on this fact and can be seen as a proof search in the sequent calculus without cut. The admissibility of cut, i.e. the fact that every sequent that can be proved with cuts can be proved without, is therefore crucial to show the completeness of these methods. It is but not always true in deduction modulo, as shown by Gilles Dowek *et al.* (2003). We even prove in this thesis that the admissibility of cut is undecidable in deduction modulo. To overcome this issue, we propose a completion procedure that transforms a rewrite system such that the sequent calculus modulo the final rewrite system admits cut. Indeed, Gilles Dowek (2003) has shown that when only terms are rewritten, and not formulæ, the admissibility of cut is equivalent to the confluence of the rewrite system. In that case, if the sequent calculus modulo does not admit cut, that means that the rewrite system is not confluent, and one may use the standard completion procedure of Donald Knuth and Peter Bendix (1970) to get an equivalent rewrite system which is confluent, and the sequent calculus modulo this system will admit cuts. In the case where formulæ can be rewritten, the equivalence is no longer true, so that we propose another completion procedure based on the abstract-canonical-system framework of Nachum Dershowitz and Claude Kirchner (2006); Maria Paola Bonacina and Nachum Dershowitz (2007*a*). This framework is developed upon an well founded ordering over proofs, representing their quality or their simplicity. This

ordering induces notions of completeness, redundancy and critical proofs, and abstract completion procedure are build on these. We have proved that deduction modulo can be seen as an instance of this framework where critical proofs are proofs with a cut. The instantiation of the abstract completion procedure eliminates the need for such proofs, so that the sequent calculus modulo the rewrite systems produced by this procedure does admit cuts. Besides, this study has led us to show how to transform a first-order theory into a rewrite system such that proving using deduction modulo this rewrite system is equivalent to proving in the theory. The completion procedure is described accurately enough to be implemented, what we did using the language TOM/OCaml. This implementation includes also a tableau method for deduction modulo based on the work of Richard Bonichon and Olivier Hermant (2006*b*). All this work is fully detailed in the paper (Guillaume Burel and Claude Kirchner, 2008).

We then study the length of the proofs as a simplicity criterion. This is of course an important criterion for practical purposes: short proofs will be found earlier for obvious reasons, and furthermore they are better to share and to maintain from a proof-engineering viewpoint. This study has also a theoretical interest, because it can be linked with complexity issues such that the famous question whether P=NP. The initial interrogation that led us to investigate this is a problem formulated by Kurt Gödel and proved later by Rohit Parikh (1973), Jan Krajíček (1989) and Samuel Buss (1994): proofs in $i+1^{\text{st}}$-order arithmetic are unboundedly shorter than proofs in i^{th}-order arithmetic. As mentioned above, it is possible to encode higher order logic in first-order systems modulo while preserving the length of proofs. Using deduction modulo therefore appears as a way to circumvent the speedup between different orders of arithmetic. Indeed, we show how to get such speedups in deduction modulo, and we propose two length-preserving encodings of higher-order arithmetic in deduction modulo: the first one encodes all higher mechanisms with the help of a very simple rewrite system (finite, terminating, confluent, left-linear); the second extends the work of Gilles Dowek and Benjamin Werner (2005) and presents higher-order arithmetic as a purely computational theory, that is, we define a rewrite system such that the formulæ proved in the deduction modulo this system corresponds exactly to the valid sentences of higher-order arithmetic. The results corresponding to this part are available in the paper (Guillaume Burel, 2008*a*).

The length-preserving encodings of higher-order logics in deduction modulo led us to study the logical order as a simplicity criterion. Indeed, first-order logics are considered to be more close to implementations, and they have been extensively studied so that they are also more easily understood. For instance, to implement the λ-calculus, one often use of a calculus of explicit substitutions, which is expressible in first-order logic. Another perspective was to show that deduction modulo, or related formalisms such as superdeduction (Benjamin Wack, 2006; Paul Brauner, Clément Houtmann and Claude Kirchner, 2007*b*), can be used as logical framework. In other words, it should

be possible to encode every inference system into deduction modulo as was done for the inference system for HOL by Gilles Dowek *et al.* (2001). To justify this claim, we have encoded every functional pure type systems, which are generic type systems for the λ-calculus used as the foundations of most of the theorem provers, into supernatural deduction, which is an extension of natural deduction with new rules derived from a rewrite system over formulæ and which can be related with natural deduction modulo. This is a step further the work of Denis Cousineau and Gilles Dowek (2007), who encode pure type system into $\lambda\Pi$ modulo, that is a very simple pure type system used in deduction modulo. We therefore argue how first-order logic can be used as a base language to check proofs developed in other systems. We also enlighten a link between proof normalization in the pure type system and in its encoding in supernatural deduction. These results can be found in (Guillaume Burel, 2008*b*).

All these results tends to convince that deduction modulo, or related formalisms such as superdeduction, are appropriate for mechanized proof search. For fully automated proof search, the completion procedure we developed and the work on the length of proofs provide a way to obtain complete and efficient proof search method. For interactive proof search, superdeduction can be used to simulate interactive provers, at least their pure-type-system part. The logical-framework aspect may also let superdeduction be used as a joint between heterogeneous part of a proof developed on several theorem-proving tools.

Chapitre 1

Introduction

> Quelle est la différence entre un psychotique et un névrotique ?
> Le psychotique est celui qui sait que 2+2 font 5 et qui s'en fout.
> Le névrotique est celui qui sait que 2+2 font 4 et ça le rend malade.
>
> Pierre Desproges

Plus encore que l'invention de l'outil, qui paraît relever avant tout de l'artisanat, c'est celle de l'agriculture qui semble faire entrer définitivement l'être humain dans l'histoire de l'automatisme : de chasseur-cueilleur se nourrissant au hasard des rencontres il développe un mécanisme lui permettant d'obtenir plus ou moins automatiquement des moyens de subsistance. Il est souvent considéré que cette découverte a constituée une étape importante dans l'émancipation de l'homme vis-à-vis de son milieu naturel. Toutefois, certains rétorquent qu'au contraire l'être humain s'est retrouvé dépendant de son champ et des nouvelles tâches qu'il doit y accomplir, et arguënt que le travail apparu conséquemment demande un temps bien supérieur à celui requis pour la recherche de nourriture des chasseurs-cueilleurs. Il est bien évident que l'objectif de cette thèse n'est pas de trancher entre ces deux points de vue, et de déterminer si oui ou non l'automatisme constitue un réel progrès pour l'humanité. Nous nous contenterons de constater que l'histoire humaine s'est accompagnée d'un nombre toujours croissant d'automatisations qui ont touchés des domaines de plus en plus vaste. En particulier, on retrouve dans la formalisation du raisonnement, telle qu'on peut la trouver dès la Grêce antique dans les travaux de l'école d'Aristote, une tentative de mécanisation des productions de l'esprit à l'aide de règles simples. Cette formalisation a fait d'importants progrès grâce aux travaux précurseurs de Gottlob Frege dans la deuxième moitié du XIX^{e} siècle. David Hilbert s'est alors demandé s'il est possible de modéliser l'ensemble des raisonnements mathématiques à l'aide d'un nombre fini de principes de bases clairement identifiables. Les années 1930 ont été témoins d'importantes avancées, avec l'introduction par Gerhard Gentzen de systèmes formels de déduction toujours utilisés aujourd'hui mais aussi avec le fameux théorème d'incomplétude de Kurt Gödel qui répond par la négative à l'interrogation de David Hilbert, puisqu'il dit que tout système logique suffisamment

puissant n'est pas capable de montrer ou de réfuter certains énoncés, notamment sa propre cohérence. Ces années ont également vue la naissance de l'ordinateur, qui a permis de mettre en œuvre de manière effective les techniques d'automatisation. La logique a bien entendu profité de cet apport, et on trouve aujourd'hui des logiciels qui permettent d'effectuer des démonstrations de façon totalement formelle.

De nombreux formalismes ont ainsi été proposés pour représenter les démonstrations mathématiques. Un certain nombre de ces formalismes sera présenté au fil de ce manuscrit. Les objectifs principaux avancés pour motiver l'introduction de ces formalismes sont d'une part la tentative de se rapprocher du raisonnement mathématique et de son expressivité et d'autre part de faciliter la recherche de démonstration mécanisée, que cette dernière soit entièrement automatisée ou qu'elle nécessite une interaction avec l'utilisateur, comme dans le cas des assistants à la démonstration. Ces formalismes répondent donc à une certaine volonté implicite de simplification de la construction des démonstrations. Néanmoins, à notre connaissance, aucun d'entre eux, hormis le cadre des systèmes canoniques abstraits de Nachum Dershowitz et Claude Kirchner (2006); Maria Paola Bonacina et Nachum Dershowitz (2007*a*), ne prend explicitement en compte la simplicité des démonstrations. Il s'agit pourtant d'un caractère important, que ce soit du point de vue du mathématicien qui recherchera la beauté associée à une démonstration simple, ou du point de vue de la machine, qui ne sera capable a priori de démontrer que les démonstrations les plus simples dans un temps donné. Des recherches épistémologiques de Rüdiger Thiele (2003) ont montré que David Hilbert s'est intéressé à cette thématique, et qu'il aurait même un temps envisagé de l'inclure dans sa célèbre liste de vingt-trois théorèmes qu'il a présenté au début du siècle dernier (1901). Il va de soit que la notion de simplicité est éminemment subjective, et qu'elle peut même différer suivant les objectifs recherchés. Ainsi, il semble probable que l'on n'utilisera pas les mêmes critères pour un mathématicien ou pour un logiciel de démonstration automatique. Dans cette thèse, on se placera avant tout du point de vue de la recherche mécanisée de démonstration.

Si la recherche automatique de démonstration s'apparente à du calcul, il est également possible de faire intervenir du calcul dans les démonstrations elles-mêmes. Par exemple la démonstration du théorème des quatre couleurs, telle qu'elle a été présentée originellement par Kenneth Appel, Wolfgang Haken et John Koch (1977), utilise l'ordinateur pour vérifier un certain nombre de cas dont il a été au préalable démontré qu'ils suffisent pour entraîner la véracité du théorème. Il est aujourd'hui possible de formaliser entièrement de telles démonstrations, et c'est ce qu'a fait Georges Gonthier (2005), avec l'aide de Benjamin Werner, dans l'assistant de démonstration Coq[1]. On retrouve également dans cette démonstration formelle l'importance du calcul. En effet, elle utilise de façon cruciale la technique dite de réflexion calculatoire, qui consiste à programmer une procédure de décision pour une partie du problème à traiter et d'utiliser implicitement cette

[1] http://coq.inria.fr/

procédure dans la partie calculatoire des démonstrations pour vérifier certains sous-buts (cf. Samuel Boutin, 1997). On voit donc l'importance de bien intégrer le calcul dans les systèmes de démonstrations. Dans le cas de Coq, on peut utiliser le λ-calcul avec types inductifs. On peut considérer que c'est suffisant, puisque les langages de programmation fonctionnelle du type ML ne contiennent pas plus. Toutefois on peut avoir envie d'exprimer le calcul à l'aide de mécanismes différents, pour par exemple sortir du schéma habituel des fonctions définies par induction structurelle. Il est reconnu que la réécriture est un formalisme qui permet à la fois d'avoir une description simple et exécutable du calcul, mais aussi d'offrir une souplesse au niveau de la définition des fonctions à calculer. De plus, ce formalisme a été abondamment étudié, tant d'un point de vue théorique que pratique. Il n'est donc pas étonnant que l'on cherche à ajouter des aspects de réécriture dans les assistants à la démonstration. Ainsi, dans Isabelle[2], il est possible de réécrire le but à démontrer pour le simplifier, chaque égalité démontrée pouvant servir de règle de réécriture.

Récemment, Gilles Dowek, Thérèse Hardin et Claude Kirchner (2003) ont introduit un nouveau formalisme pour étudier la combinaison entre déduction et calcul : la déduction modulo. Se fondant sur le principe d'Henri Poincaré, qui stipule qu'il faut bien distinguer dans les démonstrations les parties purement déductives des parties faisant intervenir le calcul, l'idée est alors d'appliquer les étapes du raisonnement modulo une congruence sur les formules logiques qui représente le calcul. Une des manières de représenter cette congruence est alors d'utiliser la réécriture. Cette technique peut s'appliquer sur la plupart des systèmes de déduction, mais on préférera en général l'appliquer à des systèmes pour la logique du premier ordre, c'est-à-dire la logique dans laquelle on ne raisonne que sur les objets eux-mêmes, et non pas sur les propriétés. Ceci a essentiellement pour raison qu'il a été démontré par Gilles Dowek, Thérèse Hardin et Claude Kirchner (2001) que la logique d'ordre supérieure, dans laquelle on peut prendre en compte les propriétés des objets, peut être encodée en logique du premier ordre en travaillant modulo un système de réécriture. Dans cette thèse, nous allons étudier quels sont les bénéfices que peut apporter l'utilisation de la déduction modulo pour la simplicité des démonstrations.

Guidés par l'objectif de mécanisation des démonstrations, nous avons retenus principalement trois critères qui témoignent de la simplicité des démonstrations. Le premier de ces critères est l'admissibilité des coupures. Dans une démonstration, il est possible de faire des détours : pour montrer une formule P, il est possible de démontrer d'abord une certaine formule Q, puis de démontrer que cette formule Q implique P. On appelle un tel détour une coupure. Pour un outil de démonstration automatique, ceci n'est pas satisfaisant car il faut alors « deviner » la formule Q. On dit alors qu'une telle démonstration n'est pas analytique, parce que les étapes de déduction utilisées ne dépendent pas uniquement des formules que l'on est en train de démontrer. Un système de déduction

[2] http://isabelle.in.tum.de/

admettra alors les coupures si toutes les formules qu'il peut démontrer peuvent l'être sans coupure. Quand Gerhard Gentzen (1934) introduit son calcul des séquences[3], le théorème principal (*Hauptsatz*) qu'il démontre est justement l'admissibilité des coupures. Si on se place dans le cadre de la déduction modulo, l'admissibilité des coupures n'est plus garantie, comme l'ont démontré Gilles Dowek et collaborateurs (2003). Nous montrons ici que cette propriété est même indécidable en général.

Nous avons alors cherché le moyen de contourner cette difficulté. Gilles Dowek (2003) a démontré que si le système de réécriture modulo lequel on se place ne réécrit que des termes (et non pas aussi des formules comme cela peut être le cas en général en déduction modulo), alors l'admissibilité des coupures est équivalente à la confluence du système de réécriture. Dans ce cas, pour avoir un système de déduction qui admet les coupures, il faut donc rendre le système de réécriture confluent. C'est justement ce que fait la complétion standard de Donald Knuth et Peter Bendix (1970). Dans le cas général où on peut également réécrire des formules, l'équivalence entre confluence et admissibilité des coupures n'est plus vraie. On est toutefois tenté de trouver une généralisation de la complétion standard qui permette de modifier un système de réécriture de façon à ce que la déduction modulo le système final admette les coupures. C'est ce que nous avons fait, en nous reposant sur le cadre théorique des systèmes canoniques abstraits de Nachum Dershowitz et Claude Kirchner (2006); Maria Paola Bonacina et Nachum Dershowitz (2007*a*). Ce cadre est fondé sur l'introduction d'un ordre bien fondé sur les démonstrations qui représente leur qualité ou leur simplicité. À partir de cet ordre sont dérivées des notions de complétude, de saturation et de redondances qui permettent de définir des procédures de complétion abstraites. Le but de ces procédures est de transformer un système pour pouvoir construire avec le système final toutes les démonstrations minimales, les meilleures donc. Nous montrons que l'on peut instancier le cadre des systèmes canoniques abstraits par une variante du calcul des séquences modulo, de façon à ce que les démonstrations minimales soient sans coupure. L'instance de complétion abstraite qui en résulte permet donc d'obtenir des systèmes de réécriture pour lesquels la déduction modulo admet les coupures. Au passage, l'instanciation fournit un algorithme qui transforme toute théorie de la logique classique du premier ordre en un système de réécriture tel qu'il est équivalent de démontrer des formules dans la théorie ou modulo le système de réécriture. Nous présentons également une implémentation de cette procédure de complétion.

Si l'admissibilité des coupures est une propriété intéressante du point de vue de la recherche automatique de démonstrations, elle a aussi ses limites. Par exemple, les démonstrations sans coupure peuvent être plus longues que les démonstrations avec. Richard

[3]Nous préférons traduire les *Sequentz* de Gerhard Gentzen (1934) par le terme « séquences », comme on peut le trouver chez Robert Feys et Jean Ladrière (1965) ou chez Jean Largeault (1971), plutôt que d'utiliser le néologisme « séquent » qui provient d'un détour par l'anglais. L'usage de ce dernier est néanmoins largement répandu dans la communauté.

Statman (1978) a ainsi démontré que dans le cas de la logique propositionnelle, certaines formules ont une démonstration de taille polynomiale en calcul des séquences avec coupures mais au mieux exponentielle sans coupure. Nous nous sommes donc intéressés à la longueur des démonstrations comme second critère de simplicité. Il est évident que d'un point de vue pratique, il est important de pouvoir construire des démonstrations courtes : ce sont celles qui seront trouvées le plus rapidement (puisqu'il faut compter au moins le temps de les construire), et de plus elles seront plus faciles à communiquer et probablement plus aisées à maintenir dans le cadre de l'ingénierie de la démonstration. La longueur des démonstrations a également une importance d'un point de vue théorique, puisqu'elle peut être reliée à des problèmes de complexité, comme la célèbre question P≠NP ?. On peut à ce propos par exemple lire la préface par Rohit Parikh de la traduction en anglais d'un article de Kurt Gödel (1986) sur ce sujet. C'est cet article qui énonce le théorème suivant sans donner de démonstration : il existe des formules valides en arithmétique d'un certain ordre i dont les démonstrations les plus courtes en arithmétique d'ordre i sont plus longues de façon non bornée que les démonstrations les plus courtes en arithmétique d'ordre $i+1$. En d'autres termes, non seulement l'arithmétique d'ordre $i+1$ permet de démontrer plus de formules que celle d'ordre i, mais elle offre également des démonstrations infiniment plus courtes de certaines formules démontrables à l'ordre inférieur. Ce théorème a été démontré par Rohit Parikh (1973) pour $i = 1$ avec une définition de l'addition et de la multiplication à l'aide de prédicat ternaires. Il a été étendu par Jan Krajíček (1989) à tous les ordres, et Samuel Buss (1994) l'a démontré pour les définitions plus naturelles de l'addition et de la multiplication à l'aide de symboles de fonction.

Comme nous l'avons indiqué ci-dessus, la logique d'ordre supérieur peut-être encodée au premier ordre en déduction modulo. De plus, cet encodage préserve la taille des démonstrations. Il semble donc que la déduction modulo permette de dépasser les réductions de tailles des démonstrations qui apparaissent en augmentant l'ordre en arithmétique. En effet, nous montrons qu'il est relativement simple de réduire la taille des démonstrations grâce à la déduction modulo, sans même se placer dans le cadre de l'arithmétique d'ordre supérieur. Nous avons toutefois aussi cherché à encoder l'arithmétique d'ordre supérieur en déduction modulo du premier ordre, en prenant soin de conserver la taille des démonstrations. Pour cela, nous avons définis deux systèmes de réécriture : le premier, très simple d'un point de vue calculatoire, encode tous les mécanismes d'ordre supérieur ; le second, qui étend le premier, permet de se passer de toute théorie axiomatique : l'arithmétique d'ordre supérieur est équivalente à la déduction en logique du premier ordre pure modulo ce système. On obtient donc des systèmes de déduction du premier ordre qui produisent des démonstrations qui sont de même longueur que celles des systèmes d'ordre supérieur qui leur correspondent.

L'encodage pour la déduction modulo que nous donnons de l'ordre supérieur en arithmétique est fondé sur un travail de Florent Kirchner (2006) qui utilise un système

de substitutions explicites. Celui de la logique d'ordre supérieur par Gilles Dowek et collaborateurs (2001) repose aussi sur l'utilisation d'un tel calcul. En règle général, il semble donc que la déduction modulo associée à un calcul de substitutions explicites permet d'encoder n'importe quel système d'inférence. La déduction modulo apparaît donc comme un cadre logique, c'est-à-dire, selon la définition de Frank Pfenning (1996), un méta-langage pour la spécification de système de déduction. Pour justifier cette affirmation, nous proposons une méthodologie qui indique comment trouver le système de réécriture qui permet d'encoder un système d'inférence. Pour des raisons de commodité, nous ne nous plaçons pas en déduction modulo, mais dans le formalisme de la surdéduction introduit par Benjamin Wack (2005) et étendu par Paul Brauner, Clément Houtmann et Claude Kirchner (2007*b*), qui lui est équivalent. Nous appliquons cette méthodologie à une large classe de systèmes de type pour le λ-calcul, à savoir les systèmes de type purs fonctionnels. Du fait de l'isomorphisme de Curry-Howard, ces systèmes de type servent souvent de base aux assistants à la démonstration. Ainsi, les premières versions de Coq reposaient sur le calcul des constructions qui est un système de type pur, même si on y a depuis rajouté des types (co-)inductifs. Un des intérêt de fournir un encodage des systèmes de type purs est donc d'obtenir un langage universel pour la vérification des démonstrations, qui sera donc la logique du premier ordre modulo. Cet objectif a déjà été atteint par Denis Cousineau et Gilles Dowek (2007), qui ont encodés les systèmes de type purs fonctionnels dans le système $\lambda\Pi$ modulo. $\lambda\Pi$, parfois appelé λP ou LF, est un système de type pur très simple. Toutefois, si $\lambda\Pi$ modulo peut être utilisé pour vérifier des démonstrations produites par des assistants à la démonstration différents, il n'est toujours pas possible de faire s'échanger des démonstrations entre assistants. L'encodage dans un système du premier ordre semble résoudre ceci, car il existe des outils, comme Fellowship[4] de Claudio Sacerdoti Coen et Florent Kirchner, qui permettent de partager des démonstrations en logique du premier ordre entre différents assistants à la démonstration (à savoir Coq et PVS[5] pour l'instant dans le cas de Fellowship). En plus de cet intérêt vis-à-vis de la coopération entre assistants à la démonstration, l'encodage des systèmes de type purs en surdéduction permet également d'envisager une meilleure compréhension de ceux-ci, puisque nous conjecturons un lien entre la normalisation des démonstrations dans un système de type pur et dans son encodage. Cela ouvre également des perspectives concernant la recherche automatique de démonstration, via les outils de démonstration automatique dérivés de la déduction modulo.

[4] http://www.lix.polytechnique.fr/Labo/Florent.Kirchner/fellowship/trunk/
[5] http://pvs.csl.sri.com/

Plan de la thèse

Dans un premier chapitre (chapitre 2), nous introduisons toutes les notions qui seront utiles pour ce manuscrit. En particulier, nous esquissons des définitions de base comme celle de la réécriture ou du λ-calcul, et nous tentons de formaliser ce qu'est une logique. Nous définissons ce qu'est un système de déduction et nous présentons les systèmes d'inférence usuels pour la logique du premier ordre. Nous en discutons l'équivalence ainsi que quelques propriétés, en particulier l'élimination des coupures du calcul des séquences.

Dans le chapitre 3 nous examinons les relations entre calcul et déduction. Nous commençons par parler de l'isomorphisme de Curry-Howard qui établit une équivalence formelle entre une démonstration et un programme (représenté sous la forme d'un λ-terme), la formule démontrée correspondant alors au type du programme. Pour intégrer le calcul au niveau des formules à démontrer, et pouvoir utiliser ainsi le calcul dans les démonstrations, une possibilité est alors de supprimer la distinction entre programme et type, ce qui engendre les λ-calculs avec types dépendants. Nous présentons en particulier la définition des systèmes de type purs, qui sont une classe de tels calculs. L'autre approche pour intégrer le calcul dans les démonstrations est celle de la déduction modulo qui consiste à appliquer les règles de déduction modulo ce calcul. Nous introduisons un certain nombre de variantes de système d'inférences pour la déduction modulo, et nous montrons leur équivalence. Nous discutons de l'admissibilité des coupures en déduction modulo en rappelant qu'elle n'est pas toujours vérifiée, puis en donnant des critères qui l'assure et en démontrant enfin son indécidabilité dans le cas général. Nous parlons des méthodes de démonstration automatique basées sur la déduction modulo, en particulier la méthode des tableaux TaMed de Richard Bonichon et Olivier Hermant (2006*b*). Nous introduisons ensuite un formalisme voisin de la déduction modulo, à savoir la surdéduction, et nous présentons en particulier les systèmes qu'il engendre pour la logique du premier ordre, à savoir la déduction surnaturelle de Benjamin Wack (2005) et le calcul des séquences extensibles de Paul Brauner et collaborateurs (2007*b*). Nous discutons des propriétés de ces systèmes, en particulier nous démontrons l'orthogonalité entre les règles de déduction purement logique et celles liées au calcul dans ces systèmes (théorème 3.30).

Le chapitre 4 débute par un exemple illustrant la méthode employée pour retrouver l'admissibilité des coupures. Les systèmes canoniques abstraits sont alors présentés en détail, et nous démontrons ensuite qu'une des variantes du calcul des séquences modulo en est une instance, ce que nous faisons en définissant un ordre bien fondé sur les démonstrations de cette variante. Ce travail nous conduit à élaborer un algorithme qui permet de transformer toute présentation finie d'une théorie du premier ordre en système de réécriture de telle façon qu'il est équivalent de faire des démonstrations dans la théorie en question ou de démontrer modulo le système de réécriture. Le fait de définir la déduction modulo comme une instance de système canonique abstrait nous fournit

une procédure de complétion qui permet de transformer un système de réécriture pour qu'au final la déduction modulo ce système admette les coupures. Nous détaillons le fonctionnement de cette procédure, qui est basée sur la notion de démonstration critique qui dans le cas de la déduction modulo correspond à une démonstration minimale possédant une coupure. Nous présentons ensuite une implémentation d'un prototype de cette procédure, qui inclut également un prototype de méthode des tableaux modulo.

Au cours du chapitre 5, nous cherchons à voir comment la déduction modulo permet des réductions dans la taille des démonstrations. Nous commençons par rappeler le cadre formel existant autour de notions de longueur de démonstration et leurs relations avec la complexité, et nous formulons quelques uns des enjeux dans ce domaine. Nous donnons ensuite deux exemples simples de réduction de la taille des démonstrations grâce à la déduction modulo, le deuxième prenant même en compte les étapes de calcul dans la taille des démonstrations. Nous nous intéressons ensuite à l'encodage de l'arithmétique d'ordre supérieur en déduction modulo en prenant soin de préserver la taille des démonstrations, et nous appliquons ces résultats à l'étude des réductions des longueurs des démonstrations quand on augmente d'ordre en arithmétique.

Le chapitre 6 est consacré à l'étude de la surdéduction comme cadre logique. Après avoir esquissé l'intérêt d'un cadre logique, nous présentons une méthodologie applicable pour la surdéduction. Nous appliquons ensuite cette méthodologie aux systèmes de type purs fonctionnels, après avoir rappelé la travail de Denis Cousineau et Gilles Dowek (2007). Nous discutons alors de ses conséquences pour la recherche de démonstration mécanisée, et nous tentons de comparer la normalisation des démonstrations dans les systèmes de type purs et dans leur encodage en surdéduction.

Dans le dernier chapitre, nous concluons en ouvrant une réflexion sur l'extension de nos méthodes à d'autres critères de simplicité des démonstrations, parmi lesquels leur normalisation ou la décidabilité de leur vérification. Nous nous penchons également sur l'utilisation de la déduction modulo et de la surdéduction comme cadre logique, en essayant de déterminer comment automatiser au maximum le passage d'une théorie ou d'un système d'inférence à son encodage en déduction modulo. Nous proposerons d'étudier un cadre théorique qui combine les bénéfices apportés par l'utilisation d'un ordre sur les démonstrations dans le cadre des systèmes canoniques abstraits et les liens entre longueurs de démonstrations et théorie de la complexité tels qu'initiés par Stephen Cook et Robert Reckhow (1979). Enfin nous discuterons de la mise en œuvre effective des résultats que nous avons obtenus pour implémenter des outils de démonstration automatique efficaces. En particulier, nous soulignerons la nécessité d'utiliser des stratégies pour cela.

Contributions

Nous avons essayé de rédiger ce manuscrit suivant un ordre naturel. C'est la raison pour laquelle nos contributions personnelles sont mêlées à des travaux existants. Afin de mieux faire ressortir celles-ci, nous en dressons la liste, même si elles n'ont pas toutes la même importance.

Chapitre 3

- Nous introduisons deux nouvelles variantes de calcul des séquences modulo : le calcul des séquences avec dépliage et le calcul des séquence avec dépliage polarisé, pour lesquels les règles de réécriture ne peuvent s'appliquer qu'à des formules atomiques et pour une étape de réécriture seulement, ce qui permet plus de contrôle sur la forme des démonstrations. Nous montrons l'équivalence entre ces deux versions (théorème 3.10) et celle envers les variantes originelles (théorème 3.8), y compris en ce qui concerne l'admissibilité des coupures.
- Nous démontrons qu'il est indécidable, étant donné un système de réécriture, de savoir si le calcul des séquences modulo ce système admet les coupures (théorème 3.14).
- Nous proposons de nouveaux termes de démonstration pour la déduction surnaturelle, dans le cas avec des conjonctions. Ces termes, basés sur un calcul distributif, permettent d'étendre le travail de Paul Brauner, Gilles Dowek et Benjamin Wack (2007*a*) sur l'équivalence de la normalisation des démonstrations en déduction naturelle modulo et en déduction surnaturelle (théorème 3.22).
- Nous proposons plusieurs approches pour obtenir un calcul des séquences extensible pour la logique intuitionniste, dont l'une est fondée sur l'équivalence avec la déduction surnaturelle (propositions 3.24 et 3.27).
- Nous montrons que pour une certaine classe de systèmes de réécriture, les systèmes de surdéduction intuitionnistes associés n'ont pas besoin des règles logiques pour démontrer les formules atomiques (théorème 3.30).

Chapitre 4

- Nous proposons un algorithme qui transforme une séquence en système de réécriture compatible. Ceci permet d'intégrer toute présentation finie d'une théorie en logique du premier ordre classique via un système de réécriture de telle façon qu'il soit équivalent d'être valide dans la théorie et d'être démontrable en déduction modulo le système (corollaire 4.10).
- Nous montrons que le calcul des séquences avec dépliage polarisé peut être vu comme une instance de système canonique abstrait (théorème 4.12). Pour cela, nous introduisons un ordre sur les démonstrations tel que les démonstrations critiques (les démonstrations minimales qui ne respecte pas la propriété recherchée) ont la forme d'une simple coupure suivie de dépliages (proposition 4.14).

– Nous en déduisons une procédure de complétion qui permet de retrouver l'admissibilité des coupures (théorème 4.17).
– Nous décrivons l'implémentation de cette procédure de complétion et de la méthode des tableaux modulo de Richard Bonichon et Olivier Hermant (2006*b*) dans le langage TOM/OCaml (section 4.4).

Les résultats de ce chapitre, ainsi que certains du précédent, ont été présentés avec Claude Kirchner en partie à la conférence *Logical Foundations of Computer Science* (2007) et en intégralité dans un article soumis au journal *Information and Computation* (2008).

Chapitre 5

– Nous démontrons sur un exemple simple que la déduction modulo permet des réductions arbitraires des longueurs de démonstrations (théorème 5.3).
– Nous encodons le calcul des structures de Kai Brünnler (2003) en déduction modulo, ce qui nous permet de profiter des résultats sur la longueur des démonstrations de Paola Bruscoli et Alessio Guglielmi (2009) pour avoir des réductions de longueur y compris en prenant en compte les étapes de calcul (proposition 5.5).
– Nous encodons tout ce qui concerne l'ordre supérieur en arithmétique d'ordre i dans un système de réécriture $\mathcal{HO}_i$ qui est fini, confluent, terminant (au moins pour $i = 2$) et linéaire à gauche. Les démonstrations en arithmétique d'ordre i sont de même longueur que dans une théorie du premier ordre *f*A modulo $\mathcal{HO}_i$ (théorème 5.12).
– Nous étendons cet encodage pour nous passer de la théorie *f*A, et ainsi obtenir une présentation purement calculatoire de l'arithmétique d'ordre i, tout en préservant la taille des démonstrations (théorème 5.14).
– Nous séparons la réduction de longueur des démonstrations obtenue en changeant d'ordre en arithmétique en deux : il existe une réduction de longueur dans *f*A modulo $\mathcal{HO}_i$ (et par conséquent en arithmétique d'ordre i) par rapport *f*A plus une théorie compatible avec $\mathcal{HO}_i$ (proposition 5.16) ; il existe une réduction de longueur dans *f*A plus une théorie compatible avec $\mathcal{HO}_i$ par rapport à l'arithmétique d'ordre $i-1$ (proposition 5.17).

Nous avons présentée une version préliminaire de ces travaux à la conférence *Computer Science and Logic* (2007). La version actuelle a été soumise au journal *ACM Transactions On Computational Logic* (2008*a*).

Chapitre 6

– Nous développons une méthodologie pour la spécification de systèmes d'inférence en surdéduction, en se basant sur : la définition de l'encodage des formules modulo un système de réécriture ; l'utilisation de λ-abstractions pour simuler les lieurs, avec

un calcul de substitutions explicites pour les manipuler ; l'utilisation du théorème 3.30 pour séparer l'encodage des règles du système d'inférence et les règles logiques du système de surdéduction (section 6.1.2).
- Nous encodons les systèmes de type purs fonctionnels en déduction surnaturelle de façon correcte et conservative (théorèmes 6.14 et 6.15).
- Ceci implique l'existence d'un calcul des séquences extensible correspondant (figure 6.4) qui est mieux adapté à la recherche de démonstration dans les systèmes de type purs. Nous proposons des pistes pour contrôler celle-ci de façon plus efficace.
- Nous montrons que la déduction surnaturelle associée à un système de type pur ne normalise pas fortement, même si c'est le cas pour ce dernier (proposition 6.20). Toutefois, nous montrons que si toutes les démonstrations en déduction surnaturelle dans un contexte bien formé normalisent fortement, alors c'est le cas dans le système de type pur (corollaire 6.19), et nous conjecturons la réciproque. Nous montrons déjà que la normalisation faible du système de type pur implique l'admissibilité des coupures pour la déduction surnaturelle associée, pour les formules dans un contexte bien formé (proposition 6.21).

L'encodage des systèmes de type purs en surdéduction a été présenté à la conférence Logic In Computer Science (2008*b*).

Chapitre 2

Notions préliminaires

> Il est toujours aisé d'être logique. Il est presque impossible d'être logique jusqu'au bout.
>
> Albert Camus, *Le mythe de Sisyphe*

Cette thèse n'a pas pour but de définir les fondements des mathématiques ou de la logique. Nous nous placerons donc dans un cadre mathématique usuel, celui de la théorie des ensembles. La définition de la sémantique des logiques utilisées par la suite, donnée à la façon d'Alfred Tarski, doit être vue avant tout comme un moyen de pouvoir parler des fondements logiques en tant qu'objets mathématiques[1]. Il ne s'agit pas pour autant d'adopter un point de vue essentialiste en déclarant que la notion de vérité existe *ex nihilo*, mais uniquement de présupposer qu'il a été possible d'en définir une de façon satisfaisante.

2.1 Notions mathématiques de base

Nous rappelons ici les définitions de quelques notions que nous utiliserons par la suite. On pourra se référer aux ouvrages de Franz Baader et Tobias Nipkow (1998) et Claude Kirchner et Hélène Kirchner (2001) pour plus de détails sur la réécriture, et de Henk Barendregt (1984) pour le λ-calcul.

Définition 2.1 (Relation)**.** *Étant donné un ensemble A, une* relation n*-aire sur A est un sous-ensemble de A^n.*

Une relation binaire $\rho \subseteq A \times A$ est dite :

- réflexive *si pour tout $x \in A$, on a $(x, x) \in \rho$;*
- antisymétrique *si pour tout $x, y \in A$, si $(x, y) \in \rho$ et $(y, x) \in \rho$ alors $x = y$;*
- transitive *si pour tout $x, y, z \in A$, si $(x, y) \in \rho$ et $(y, z) \in \rho$ alors $(x, z) \in \rho$;*
- bien fondée *(ou Noethérienne) si pour tout sous-ensemble non vide $X \subseteq A$ il existe un élément $x \in X$ minimal pour ρ, c'est-à-dire tel que pour aucun $y \in X$ on ait $(x, y) \in \rho$.*

[1] On pourrait faire une grossière analogie en informatique avec un compilateur *bootstrappé*.

Cette dernière propriété est équivalente à l'inexistence de chaîne infinie décroissante, c'est-à-dire de suite $(a_n)_{n\in\mathbb{N}}$ d'éléments de A tels que pour tout i on ait $(a_i, a_{i+1}) \in \rho$.

On notera $x\rho y$ pour $(x, y) \in \rho$.

Définition 2.2 (Ordre)**.** *Un* préordre *sur A est une relation binaire sur A réflexive et transitive.*

Un ordre *sur A est un préordre sur A antisymétrique.*

L'ordre strict $>$ associé à un ordre $\geq$ est la relation $\geq \setminus \{(x, x) : x \in A\}$.

Par extension, un ordre sera dit bien fondé si son ordre strict l'est.

Définition 2.3 (Multiensemble)**.** *Un* multiensemble *m sur A est une fonction de A dans $\mathbb{N}$. Si $m(a) \neq 0$ pour seulement un nombre fini d'éléments $a \in A$, alors on dit que m est fini, et on peut noter m comme un ensemble où l'élément a apparaît $m(a)$ fois.*

*Étant donné un ordre $\succ$ sur A, l'*extension multiensemble *$\succcurlyeq$ de cet ordre sur les multiensembles sur A est défini par : $m \succcurlyeq n$ si pour tout $a \in A$, si $n(a) > m(a)$ pour l'ordre usuel sur les entiers, alors il existe un $b \in A$ tel que $b \succ a$ et $m(b) > n(b)$.*

Remarque 2.1 *:* Un multiensemble sur A peut également être vu comme un élément du monoïde commutatif libre engendré par A.

Définition 2.4 (Arbre)**.** *Une* position *est un mot sur l'alphabet des entiers naturels non nuls $\mathbb{N} \setminus \{0\}$. On note $\mathfrak{e}$ la position correspondant au mot vide.*

Un support d'arbre est un ensemble non vide de positions fermé par préfixe.

Un arbre *à valeur dans A est la donnée d'un support d'arbre et d'une fonction d'étiquetage de ce support dans A.*

Le sous-arbre *de (P, f) à la position $\mathfrak{p}$ possède comme support $\{\mathfrak{q} : \mathfrak{pq} \in P\}$ et comme fonction d'étiquetage $\mathfrak{q} \mapsto f(\mathfrak{pq})$.*

Un fils *d'un arbre t est un sous-arbre s à une position $n \in \mathbb{N}$. On dit alors que t est le parent de s.*

Le remplacement *du sous-arbre de (P, f) à la position $\mathfrak{p}$ par l'arbre (Q, g) est l'arbre dont le support est l'union de l'ensemble des positions de P dont $\mathfrak{p}$ n'est pas préfixe et des positions $\mathfrak{pq}$ pour tout $\mathfrak{q} \in Q$, et dont la fonction d'étiquetage associe $f(\mathfrak{r})$ aux positions $\mathfrak{r}$ dont $\mathfrak{p}$ n'est pas préfixe et $g(\mathfrak{q})$ aux positions $\mathfrak{pq}$.*

Une feuille *d'un arbre est une position du support de l'arbre qui n'est le préfixe d'aucune autre position de ce support, ou par extension la valeur étiquetée par cette position.*

Définition 2.5 (Terme)**.** *Une* signature *est la donnée d'un ensemble de symboles de fonction F, une fonction $a : F \to \mathbb{N}$ qui associe à un symbole de fonction son arité, c'est-à-dire un entier naturel.*

Étant donnés une signature $\Sigma = (F, a)$ et un ensemble dénombrable de variables V disjoint de celui des symboles de fonctions, l'ensemble des termes *$\mathcal{T}(\Sigma, V)$ est le plus petit sous-ensemble des arbres à valeur dans $F \cup V$ tel que*

- *pour toute variable $x \in V$ on a $(\{\mathfrak{e}\}, \mathfrak{e} \mapsto x)$ appartient à $\mathcal{T}(\Sigma, V)$; on identifie alors x et $(\{\mathfrak{e}\}, \mathfrak{e} \mapsto x)$ pour plonger V dans $\mathcal{T}(\Sigma, V)$;*
- *si g est un symbole de fonction d'arité $n = a(g)$ dans Σ et $(P_1, f_1), \ldots, (P_n, f_n) \in \mathcal{T}(\Sigma, V)^n$ alors l'arbre de support $\{i\mathfrak{p} : 1 \leq i \leq n \wedge \mathfrak{p} \in P_i\}$ et de fonction d'étiquetage qui à $\mathfrak{e}$ associe g et à $i\mathfrak{p}$ associe $f_i(\mathfrak{p})$ appartient à $\mathcal{T}(\Sigma, V)$. En notant t_i l'arbre (P_i, f_i), on notera cet arbre $g(t_1, \ldots, t_n)$.*

La notion de sous-arbre s'étend en une notion de sous-terme. *On notera $t_{|\mathfrak{p}}$ le sous-terme de t à la position $\mathfrak{p}$, et $t[s]_\mathfrak{p}$ le remplacement du sous-terme de t à la position $\mathfrak{p}$ par s.*

Une constante *est un terme de support $\{\mathfrak{e}\}$ étiqueté par un symbole de fonction nullaire. On notera alors f (sans empattement) au lieu de $f()$.*

Les variables libres *d'un terme sont les variables qui apparaissent dans ce terme. On note $FV(t)$ l'ensemble des variables libres d'un terme t.*

Un contexte *est un terme dont une (et une seule) feuille est étiquetée par un symbole spécial $[]$. Appliquer un contexte à un terme s consiste à remplacer le sous-terme $[]$ du contexte par le terme s. On note $t[]$ un contexte et $t[s]$ l'application de ce contexte à s.*

Une substitution *σ est une fonction de l'ensemble des variables V dans l'ensemble des termes $\mathcal{T}(\Sigma, V)$ avec un nombre fini de variables $x \in V$ telles que $\sigma(x) \neq x$. Appliquer une substitution σ à un terme t consiste à remplacer les variables x dans t par $\sigma(x)$. On note le terme résultant σt. La substitution qui remplace x_1 par t_1, …, x_n par t_n est notée $\{t_1/x_1, \ldots, t_n/x_n\}$.*

Définition 2.6 (Ordre récursif sur les chemins, Nachum Dershowitz, 1982)**.** *Étant donnée une signature Σ, on considère un ordre $\succ$ sur les symboles de fonction que l'on appellera* précession.

À tout symbole de fonction on choisit d'associer un statut soit lexicographique soit multiensemble.

*On définit alors un ordre sur les termes $\mathcal{T}(\Sigma, V)$ de la façon suivante : pour tout $s = f(s_1, \ldots, s_n)$ et $t = g(t_1, \ldots, t_m)$ dans $\mathcal{T}(\Sigma, V)$, s est grand que t pour l'*ordre récursif sur les chemins *(en court, RPO, noté $s >_{RPO} t$) si*

- *$f = g$ et $\{s_1, \ldots, s_n\}$ est plus grand que $\{t_1, \ldots, t_n\}$ pour l'extension lexicographique ou multiensemble de l'ordre récursif sur les chemins, suivant le statut de f ; ou*
- *$f > g$ et pour tout j tel que $1 \leq j \leq m$, on a $f >_{RPO} t_j$; ou*
- *il existe un i tel que $1 \leq i \leq n$ et $s_i >_{RPO} t$.*

Proposition 2.1. *L'ordre récursif sur les chemins possède les propriétés suivantes :*

- *il est monotone, c'est-à-dire pour tout contexte $t[]$ et tous termes s et u, si $s >_{RPO} u$ alors $t[s] >_{RPO} t[u]$;*
- *il est stable, c'est à dire pour toute substitution σ et tous termes s et u, si $s >_{RPO} u$ alors $\sigma s >_{RPO} \sigma u$;*
- *pour tout terme t et toute position $\mathfrak{p}$, $t \geq_{RPO} t_{|\mathfrak{p}}$;*

ces trois propriétés font d'un ordre un ordre dit de simplification.

– *Si la précession est bien fondée, alors il est bien fondé* (Nachum DERSHOWITZ, 1982).

Dans la suite, on utilisera que des statuts multiensemble dans les ordres RPO que nous définirons.

Définition 2.7 (Système de réécriture)**.** *Une* règle de réécriture *est un couple de terme* (s, t) *tel que* $FV(t) \subseteq FV(s)$*. On la note* $s \to t$*.*

Un terme t_1 *se réécrit en un terme* t_2 *par la règle de réécriture* $s \to u$ *à la position* $\mathfrak{p}$ *si il existe une substitution* σ *telle que* $t_{1|\mathfrak{p}} = \sigma s$ *et* $t_2 = t_1[\sigma u]_\mathfrak{p}$*. On note alors* $t_1 \xrightarrow[s\to u]{\mathfrak{p},\sigma} t_2$*, ou parfois plus simplement* $t_1 \xrightarrow[s\to u]{} t_2$*.*

Un système de réécriture *est un ensemble de règle de réécriture. Un terme* t_1 *est réécrit en* t_2 *par un système de réécriture* $\mathcal{R}$ *si il l'est par une de ses règles. On note alors* $t_1 \xrightarrow[\mathcal{R}]{} t_2$*.* $\xrightarrow[\mathcal{R}]{*}$ *est la fermeture réflexive et transitive de cette relation,* $\xleftrightarrow[\mathcal{R}]{*}$ *sa fermeture réflexive, symétrique et transitive.*

Un terme t_1 *se surréduit en un terme* t_2 *par la règle de réécriture* $s \to u$ *à la position* $\mathfrak{p}$ *si il existe une substitution* σ *telle que* $\sigma t_{1|\mathfrak{p}} = \sigma s$ *et* $t_2 = \sigma(t_1[u]_\mathfrak{p})$*. La* surréduction *(en anglais* narrowing*) est à la réécriture ce qu'est le filtrage de motif à l'unification.*

Remarque 2.2 *:* Ces notions seront étendues à des règles de réécriture sur les formules dans la section 3.2.

Définition 2.8 (Confluence)**.** *Un système de réécriture* $\mathcal{R}$ *est* confluent *si pour tous termes* u*,* s *et* t *tels que* $s \xleftarrow[\mathcal{R}]{*} u \xrightarrow[\mathcal{R}]{*} t$ *il existe un terme* v *tel que* $s \xrightarrow[\mathcal{R}]{*} v \xleftarrow[\mathcal{R}]{*} t$*. Cette définition est équivalente à la propriété de* CHURCH-ROSSER *qui dit que pour tous termes* s *et* t*, si* $s \xleftrightarrow[\mathcal{R}]{*} t$ *alors il existe un terme* v *tel que* $s \xrightarrow[\mathcal{R}]{*} v \xleftarrow[\mathcal{R}]{*} t$*, ce que l'on notera* $s \xrightarrow[\mathcal{R}]{*} \xleftarrow[\mathcal{R}]{*} t$

Définition 2.9 (λ-calcul)**.** *Les termes du* λ-calcul *sont formés par la grammaire suivante :*

$$t \ ::= \ x \mid \lambda x,\ t \mid t\ t$$

Dans $\lambda x,\ t$*, la variable* x *est liée, ce qui veut dire qu'elle n'apparaît pas dans* $FV(\lambda x,\ t)$*, et que* $\{s/y\}(\lambda x,\ t) = \lambda z,\ \{s/y\}\{z/x\}t$ *pour* $z \notin FV(s, t, y)$*.* $\lambda x,\ t$ *est dit* α*-équivalent à* $\lambda y,\ \{y/x\}t$ *pour* $y \notin FV(t)$*. Les* λ*-termes sont alors définis modulo cette relation d'équivalence.*

Un λ*-terme* t_1 *se* β*-réduit en* t_2 *à la position* $\mathfrak{p}$ *si* $t_{1|\mathfrak{p}} = (\lambda x,\ s)\ u$ *pour certains* λ*-termes* s *et* u*, et* $t_2 = t_1[\{u/x\}s]_\mathfrak{p}$*. On note alors* $t_1 \xrightarrow[\beta]{} t_2$*.*

Un λ*-terme* t_1 *se* η*-réduit en* t_2 *à la position* $\mathfrak{p}$ *si* $t_{1|\mathfrak{p}} = \lambda x,\ (s\ x)$ *pour un certain* λ*-terme* s *et* $x \notin FV(s)$*, et* $t_2 = t_1[s]_\mathfrak{p}$*. On note alors* $t_1 \xrightarrow[\eta]{} t_2$*.*

$$\textsf{Variable}\ \frac{}{\Gamma \vdash x : A}\ x : A \in \Gamma$$

$$\textsf{Abstraction}\ \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A,\ t : A \Rightarrow B}$$

$$\textsf{Application}\ \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

FIG. 2.1 : Règles de typage du λ-calcul simplement typé

Ici aussi, on note $\xrightarrow[\beta]{*}$, $\xleftrightarrow[\beta]{*}$, $\xrightarrow[\eta]{*}$, $\xleftrightarrow[\eta]{*}$ *les fermetures réflexives et transitives, et réflexives, symétriques transitives de* $\xrightarrow[\beta]{}$ *et* $\xrightarrow[\eta]{}$.

Remarque 2.3 *:* La règle $(\lambda x,\ s)\ u \to \{u/x\}s$ n'est pas une règle de réécriture, du fait de la substitution $\{u/x\}$ qui se fait en une étape. Si on veut exprimer la β-réduction sous forme de système de réécriture, il faut utiliser un calcul de substitutions explicites (cf. section 6.1.2).

Définition 2.10. *On se dote d'un certain ensemble* $\mathcal{B}$ *de type dit de base, par exemple* ι, o, *etc. Les types simples sont les termes de la signature* $(\mathcal{B} \cup \{\Rightarrow\}, a)$ *où* $a(b) = 0$ *si* $b \in \mathcal{B}$ *et* $a(\Rightarrow) = 2$.

Pour avoir un λ*-terme à la* CHURCH, *il faut annoter par un type les variables liées par un* λ, *par exemple* $\lambda x : \iota,\ \lambda y : o \Rightarrow \iota,\ z\ x\ y$.

Un λ*-terme* t *sans variable libre est dit* simplement typé *si on peut construire une dérivation fermée de* $\vdash t : A$ *pour un certain type* A *dans le système d'inférence de la figure 2.1 (cf. la définition 2.24 des systèmes d'inférence ci-dessous). On dit alors que* t *est de type* A.

2.2 Logique(s)

Il existe diverses façons de définir ce qu'est une logique. Cette question en soit n'est d'ailleurs toujours pas résolue, comme en témoigne l'existence d'un congrès cherchant à définir une logique de manière universelle[2]. José MESEGUER (1989) définit par exemple une logique de façon catégorique par un système de déduction et une sémantique. Toutefois cette approche n'est pas totalement satisfaisante, car on peut imaginer des logiques pour lesquels il n'existe pas de système de déduction précis, ou alors pour lesquelles la sémantique n'est pas définie autrement que par celle d'un système de déduction. On ne peut pas se contenter de définir une logique par un système de déduction, du fait de

[2]On pourra consulter le site suivant : http://www.uni-log.org/second1.html.

l'incomplétude de certaines logiques (Kurt GÖDEL, 1931). Nous utiliserons plutôt une définition proche de celle développée par Alfred TARSKI dans les années 1930, à partir de la notion de théorie, et qu'on peut retrouver chez Nachum DERSHOWITZ et Claude KIRCHNER (2006).

Définition 2.11 (Logique)**.** *Une* logique *est définie par un couple* $(\mathbb{A}, Th)$ *où* Th *est un opérateur de fermeture sur* $(\mathcal{P}(\mathbb{A}), \subseteq)$*, c'est-à-dire pour toutes sous-parties* A *et* B *de* $\mathbb{A}$

- $A \subseteq B$ *implique* $Th\, A \subseteq Th\, B$ *;*
- $A \subseteq Th\, A$ *;*
- $Th\, Th\, A = Th\, A$*.*

On appelle présentation *toute sous-partie de* $\mathbb{A}$*.*

On dit qu'une logique $(\mathbb{A}, Th)$ *est* cohérente *si* $Th\, \emptyset \neq \mathbb{A}$*.*

On appelle $Th\, \emptyset$ *l'ensemble des* tautologies*, qui sont donc les formules qui sont valides sans supposer d'hypothèses.*

On aurait pu préférer la définition de Lutz STRASSBURGER (2005*b*) qui caractérise une logique comme un préordre $(\mathbb{A}, \Longrightarrow)$, mais on peut s'y ramener :

Définition 2.12 (Relation de conséquence)**.** *Étant donnée une logique* $(\mathbb{A}, Th)$ *on définit la* relation de conséquence $\Longrightarrow$ *comme l'ensemble des paires* (a, b) *telles que* $b \in Th\, \{a\}$*.*

Proposition 2.2. $\Longrightarrow$ *est un préordre sur* $\mathbb{A}$*.*

Démonstration. Pour tout $a \in \mathbb{A}$, on a $\{a\} \subseteq Th\, \{a\}$ donc $\Longrightarrow$ est réflexive. Pour toutes formules $a, b, c \in \mathbb{A}$, si $b \in Th\{a\}$ alors $Th\, \{b\} \subseteq Th\, (Th\, \{a\})$, et si en plus $c \in Th\, \{b\}$ alors $c \in Th\, (Th\, \{a\}) = Th\, \{a\}$, donc $\Longrightarrow$ est transitive. □

Remarque 2.4 *:* À partir de la définition de Lutz STRASSBURGER (2005*b*), c'est-à-dire à partir d'un préordre $(\mathbb{A}, \Longrightarrow)$, on peut également retrouver la définition 2.11, en posant $Th\, A \overset{\text{déf}}{=} \{b \in \mathbb{A} : \exists a \in A,\ a \Longrightarrow b\}$ pour toute présentation $A \subseteq \mathbb{A}$. On vérifie qu'il s'agit bien d'un opérateur de fermeture.

Ceci est toutefois moins général, car dans ce cas $Th\, (A \cup B) = Th\, A \cup Th\, B$.

Remarque 2.5 *:* Si l'ensemble des formules est fermé par un symbole de conjonction $\wedge$, on peut étendre la relation $\Longrightarrow$ sur le produit de l'ensemble des parties finies de $\mathbb{A}$ avec $\mathbb{A} : \{a_1, \ldots, a_n\} \Longrightarrow b$ ssi $a_1 \wedge \ldots \wedge a_n \Longrightarrow b$. On suppose alors que $Th\, \{a_1, \ldots, a_n\} = Th\, \{a_1 \wedge \ldots \wedge a_n\}$.

Ces définitions très générales ont l'avantage de permettre de définir une logique à la fois à l'aide de modèles ou de systèmes de déduction.

Définition 2.13 (Sémantique)**.** *On définit une* sémantique *comme le triplet d'un ensemble de formules* $\mathbb{A}$*, d'un ensemble* M *dont les éléments sont appelés* modèles *et d'une relation* $\models$ *entre les modèles et les formules (*$\models \subseteq M \times \mathbb{A}$*).*

La logique associée à une sémantique $(\mathbb{A}, M, \models)$ *est définie par* $(\mathbb{A}, Th)$ *où* $Th\, A \stackrel{\text{déf}}{=} \{b \in \mathbb{A} : \forall m \in M,\ (\forall a \in A,\ m \models a) \Rightarrow m \models b\}$*, ce qui fournit bien un opérateur de fermeture.*

Une formule a est dite satisfaisable *si il existe un modèle* $m \in M$ *tel que* $m \models a$*. On dit alors que m est un modèle de a. On dit aussi que m est un* modèle d'une présentation *si c'est un modèle de toutes ses formules. De même, on parle alors de* modèle d'une théorie *pour un modèle d'une présentation d'une théorie (ce qui ne dépend pas de la présentation choisie).*

Une formule a est dite valide *si pour tout modèle* $m \in M$ *on a* $m \models a$*. Dans ce cas, on notera* $\models a$*. Cela revient à dire que a est une tautologie de la logique associée à la sémantique.*

On définit également ce qu'est un système de déduction, en s'inspirant de Nachum Dershowitz et Claude Kirchner (2006).

Définition 2.14 (Système de déduction)**.** *Un* système de déduction *est un quadruplet* $(\mathbb{A}, \mathbb{P}, [\cdot]^{Pm}, [\cdot]_{Cl})$ *tel que* $\mathbb{A}$ *est l'ensemble des formules,* $\mathbb{P}$ *est l'ensemble des* démonstrations*,* $[\cdot]^{Pm}$ *est une fonction de* $\mathbb{P}$ *dans* $\mathcal{P}(\mathbb{A})$ *et* $[\cdot]_{Cl}$ *une fonction de* $\mathbb{P}$ *dans* $\mathbb{A}$*, que l'on peut toutes deux étendre de façon usuelle de* $\mathcal{P}(\mathbb{P})$ *dans* $\mathcal{P}(\mathbb{A})$*, tels que* $Th \stackrel{\text{déf}}{=} [\cdot]_{Cl} \circ [\cdot]^{Pm^{-1}}$ *soit un opérateur de fermeture.*

On définit la logique associée à un système de déduction comme cet opérateur de fermeture.

On définit les démonstrations que l'on peut produire à partir d'une présentation grâce à la fonction $Pf \stackrel{\text{déf}}{=} [\cdot]^{Pm^{-1}}$*.*

On appelle justification *toute sous-partie de* $\mathbb{P}$*.*

Définition 2.15 (Correction, complétude)**.** *On considère un ensemble de formules* $\mathbb{A}$ *et deux logiques sur cet ensemble de formules ayant comme relations de conséquence* $\Longrightarrow_1$ *et* $\Longrightarrow_2$*.*

On dit que la première est correcte *pour la seconde si* $\Longrightarrow_1 \subseteq \Longrightarrow_2$*.*

On dit que la première est complète *pour la seconde si* $\Longrightarrow_1 \supseteq \Longrightarrow_2$*.*

Définition 2.16 (Logique engendrée par une théorie)**.** *Étant donnée une logique* $(\mathbb{A}, Th)$ *et une présentation* $A \subseteq \mathbb{A}$ *on définit la* logique engendrée *par A à partir de* $(\mathbb{A}, Th)$ *comme* $(\mathbb{A}, Th')$ *où* Th' *est définie par* $Th'B \stackrel{\text{déf}}{=} Th\,(A \cup B)$*.*

Définition 2.17 (Extension, conservatisme)**.** *On considère deux ensembles de formules* $\mathbb{A} \subseteq \mathbb{A}'$ *et deux logiques* $(\mathbb{A}, Th_1)$ *et* $(\mathbb{A}', Th_2)$*.*

On dit que la deuxième est une extension *de la première si* $(\mathbb{A}', Th'_1)$ *est correcte pour* $(\mathbb{A}', Th_2)$ *où* Th'_1 *est le plus petit opérateur de clôture étendant* Th_1 *sur* $\mathbb{A}'$*, défini par* $Th'_1 A = (Th_1(A \cap \mathbb{A})) \cup (A \setminus \mathbb{A})$*.*

On dit de plus que cette extension est conservative *si* $(\mathbb{A}, Th_1)$ *est complète pour* $(\mathbb{A}, Th_2)$ *(en considérant bien entendu la restriction et corestriction de* Th_2 *sur* $\mathbb{A}$*).*

2.2.1 Logique du premier ordre

On étend la définition de signature pour y inclure des symboles de prédicat.

Définition 2.18 (Signature)**.** *Une* signature *du premier ordre est un triplet* (F, P, a) *constitué d'un ensemble des symboles de fonction* F*, d'un ensemble de symboles de prédicat distincts des symboles de fonction* P*, et d'une fonction d'arité de* $F \cup P$ *dans* $\mathbb{N}$*.*

La définition des termes est inchangée.

Définition 2.19 (Formule)**.** *Étant donnés une signature* Σ *et un ensemble dénombrable de variables* V*, étant donné un symbole de prédicat* A *d'arité* n *dans* Σ *et* $(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, V)^n$*, on dit que* $A(t_1, \ldots, t_n)$ *est une* proposition atomique*.*

On définit alors l'ensemble des formules de la logique du premier ordre *par la grammaire suivante :*

$$P ::= A \mid \bot \mid \top \mid \neg P \mid P \wedge P \mid P \vee P \mid P \Rightarrow P \mid \forall x,\ P \mid \exists x,\ P$$

où x *appartient à* V *et* A *a valeur dans les propositions atomiques.*

On utilisera les notations $A \Leftrightarrow B$ *pour* $(A \Rightarrow B) \wedge (B \Rightarrow A)$ *et, pour tout ensemble fini de formules* $\Gamma = \{P_1, \ldots, P_n\}$*, on notera* $\bigwedge \Gamma$ *pour* $P_1 \wedge \cdots \wedge P_n$ *et* $\bigvee \Gamma$ *pour* $P_1 \vee \cdots \vee P_n$*.*

Le quantificateur universel $\forall$ et le quantificateur existentiel $\exists$ lient une variable. Il faut donc définir l'ensemble des variables libres apparaissant dans une formule :

Définition 2.20 (Variables libres)**.** *L'ensemble des* variables libres $FV(P)$ *apparaissant dans une formule est défini par :*

- $FV(A(t_1, \ldots, t_n)) = \bigcup_{1 \leq i \leq n} FV(t_i)$;
- $FV(\bot) \stackrel{\text{déf}}{=} FV(\top) \stackrel{\text{déf}}{=} \emptyset$;
- $FV(\neg P) \stackrel{\text{déf}}{=} FV(P)$;
- $FV(P \wedge Q) \stackrel{\text{déf}}{=} FV(P \vee Q) \stackrel{\text{déf}}{=} FV(P \Rightarrow Q) \stackrel{\text{déf}}{=} FV(P) \cup FV(Q)$;
- $FV(\forall x,\ P) \stackrel{\text{déf}}{=} FV(\exists x,\ P) \stackrel{\text{déf}}{=} FV(P) \setminus \{x\}$.

Il faut alors considérer les formules à α-conversion près : si $y \notin FV(P)$ alors $\forall x,\ P$ et $\forall y,\ \{y/x\}P$ sont les mêmes formules.

On peut définir un ordre bien fondé sur les formules :

Définition 2.21 (Ordre sous-formule)**.** *L'ordre sous-formule* $>$ *est le plus petit ordre sur les formules tel que*

- $\neg P > P$;
- $P \wedge Q > P$ *et* $P \wedge Q > Q$*, de même pour* $P \vee Q$ *et* $P \Rightarrow Q$;
- $\forall x,\ P > \{t/x\}P$ *pour tout terme* t*, de même pour* $\exists x,\ P$*.*

Cet ordre est bien fondé (le nombre de connecteurs dans une suite décroissante diminue).

La logique classique du premier ordre peut être définie grâce à sa sémantique.

Définition 2.22 (Modèle classique)**.** *Étant donnés une signature* Σ*, un* modèle pour la logique classique du premier ordre *est définit par un ensemble non vide* D *appelé support du modèle, une fonction* $\hat{f} : D^n \to D$ *pour chaque symbole de fonction* f *d'arité* n *et un ensemble* $\hat{P} \subseteq D^n$ *pour chaque symbole de prédicat* P *d'arité* n *dans* Σ*.*

Une valuation *est une fonction de* V *dans* D*.*

Étant donnée une valuation ρ*, on définit une interprétation sur les termes par induction sur leur structure :*

- $[\![x]\!]_\rho = \rho(x)$
- $[\![f(t_1, \ldots, t_n)]\!]_\rho = \hat{f}([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho)$

puis on définit la relation suivante entre le modèle et une formule par induction sur la formule :

- $m \models_\rho A(t_1, \ldots, t_n)$ *ssi* $([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho) \in \hat{A}$
- $m \not\models_\rho \bot$
- $m \models_\rho \top$
- $m \models_\rho \neg P$ *ssi* $m \not\models_\rho P$
- $m \models_\rho P \wedge Q$ *ssi* $m \models_\rho P$ *et* $m \models_\rho Q$
- $m \models_\rho P \vee Q$ *ssi* $m \models_\rho P$ *ou* $m \models_\rho Q$
- $m \models_\rho P \Rightarrow Q$ *ssi* $m \models_\rho P$ *implique* $m \models_\rho Q$
- $m \models_\rho \forall x,\ P$ *ssi pour tout* $d \in D$ *on a* $m \models_{\rho'} P$ *où* ρ' *est la valuation* ρ *sauf pour* x *à qui elle associe* d*.*
- $m \models_\rho \exists x,\ P$ *ssi il existe un élément* d *de* D *tel que* $m \models_{\rho'} P$ *où* ρ' *est la valuation* ρ *sauf pour* x *à qui elle associe* d*.*

Finalement, $m \models P$ *ssi* $m \models_\rho P$ *pour toute valuation* ρ*.*

De façon équivalente, on peut dire que l'on interprète les symboles de prédicats d'arité n par une fonction de D^n dans $\{0; 1\}$. On dit alors que 0 et 1 sont des *valeurs de vérité*. C'est cette définition qu'il est plus facile de généraliser pour d'autres logiques.

On parle de logique propositionnelle du premier ordre quand on se restreint aux formules sans quantificateurs. Dans ce cas, il n'est pas nécessaire de parler de termes, puisqu'on peut utiliser un prédicat nullaire $A_{t_1,\ldots,t_n}$ au lieu de la proposition atomique $A(t_1, \ldots, t_n)$. Pour distinguer, on parle alors parfois de logique des prédicats du premier ordre quand on autorise les quantificateurs.

Pour définir des systèmes d'inférence, nous aurons besoin de schéma de formules, avec des variables pouvant être substituées par des formules.

Définition 2.23 (Méta-formule)**.** *On considère un ensemble dénombrable* MV *de variables appelées à jouer un rôle particulier dans les règles d'inférence des systèmes de déduction, les* méta-variables*. On considère également des ensembles dénombrables de* variables propositionnelles PV_n *pour chaque arité* n*.*

Étant donnée une signature du premier ordre (F, P, a) *et un ensemble de variables* V*, une* méta-formule *sera une formule du premier ordre sur la signature* $(F, P \cup \bigcup_n PV_n, a')$ *avec* $a'(X) = n$ *si* $X \in PV_n$ *et* $a'(f) = a(f)$ *sinon, et les variables* $V \cup MV$*, autrement dit les termes pourront contenir des méta-variables et les variables propositionnelles pourront être utilisées comme des symboles de prédicat.*

L'intérêt des méta-formules réside dans leur instanciation : on obtient une instance d'une méta-formule *en lui appliquant une* substitution *qui associe une variable à une méta-variable, un terme quelconque à une variable non méta, et une formule dans laquelle on a distingué un nombre de variables égal à son arité à une variable propositionnelle. L'application d'une substitution* σ *à* $X(t_1, \ldots, t_n)$ *renvoie* $\sigma(X)$ *dans laquelle on a remplacé les* n *variables libres distinguées par l'application de* σ *à* $t_1, \ldots, t_n$*.*

Il est également possible de définir la logique intuitionniste du premier ordre à l'aide de sa sémantique, principalement par deux méthodes distinctes : les algèbres d'Arend Heyting et les modèles de Saul Kripke. Toutefois, il nous semble plus clair de la définir comme la logique associée à la déduction naturelle (voir la section suivante).

Logique multisortée

En logique du premier ordre monosortée, tous les objets ont le même statut, et sont interprétés dans le même domaine. Toutefois, on a parfois envie de distinguer différents types d'objets, pour clarifier les formules. La logique du premier ordre multisortée permet ceci, en généralisant la notion d'arité.

On se donne un ensemble de *sortes* S, et on choisit l'*arité* d'un symbole de fonction ou d'un prédicat parmi les listes finies d'éléments de S. La *coarité* d'un symbole de fonction sera un élément de S (son « type de retour »). Pour chaque sorte s de S on considère un ensemble dénombrable de variable V_s. Pour une sorte $s \in S$ l'ensemble des termes de sorte s noté $\mathcal{T}_s\left(\Sigma, (V_s)_{s\in S}\right)$ est le plus petit ensemble contenant V_s et tel que si f est un symbole d'arité $s_1, \ldots, s_n$ et de coarité s, si $t_i \in \mathcal{T}_{s_i}\left(\Sigma, (V_s)_{s\in S}\right)$ alors $f(t_1, \ldots, t_n) \in \mathcal{T}_s\left(\Sigma, (V_s)_{s\in S}\right)$. De même une proposition atomique $A(t_1, \ldots, t_n)$ est construite à partir d'un symbole de prédicat A d'arité $s_1, \ldots, s_n$ et de $(t_1, \ldots, t_n) \in \mathcal{T}_{s_1}\left(\Sigma, (V_s)_{s\in S}\right) \times \cdots \times \mathcal{T}_{s_n}\left(\Sigma, (V_s)_{s\in S}\right)$. Pour plus de clarté, on indiquera en exposant la sorte des variables, par exemple $x^s \in V_s$.

Il faut aussi étendre la notion de modèle : un modèle est donné par un support D_s pour chaque $s \in S$, des interprétations de chaque symbole de fonction d'arité $s_1, \ldots, s_n$ et de coarité s par une fonction de $D_{s_1} \times \cdots \times D_{s_n}$ dans D_s, et des interprétations de chaque symbole de prédicat A d'arité $s_1, \ldots, s_n$ par un sous-ensemble de $D_{s_1} \times \cdots \times D_{s_n}$. Une valuation est donnée par une fonction de V_s dans D_s pour chaque $s \in S$. Tout le reste fonctionne alors comme dans le cas monosorté.

Il est possible de traduire la logique multisortée en logique monosortée en ayant recours à des prédicats d'arité 1 pour chacune des sortes, par exemple S pour la sorte s, et des

axiomes pour traduire les arités de symboles de fonctions (par exemple pour $f : [\mathrm{s}_1; \mathrm{s}_2] \to s$ on a $\forall x,\ S_1(x) \Rightarrow \forall y,\ S_2(y) \Rightarrow S(f(x,y))$) ainsi que l'existence de termes pour toute sorte (pour toute sorte s on considère un axiome $\exists x,\ S(x)$). On traduit alors les formules par induction :

$$\begin{aligned} |A(t_1,\ldots,t_n)| &\stackrel{\text{déf}}{=} A(t_1,\ldots,t_n) \\ |P \wedge Q| &\stackrel{\text{déf}}{=} |P| \wedge |Q| \qquad \text{et de façon similaire pour } \vee,\ \Rightarrow,\ \neg \\ |\forall x^{\mathrm{s}},\ P| &\stackrel{\text{déf}}{=} \forall x,\ S(x) \Rightarrow |P| \\ |\exists x^{\mathrm{s}},\ P| &\stackrel{\text{déf}}{=} \exists x,\ S(x) \wedge |P| \end{aligned}$$

Les formules sont alors démontrables en logique multisortée ssi leur traduction l'est en logique monosortée avec les axiomes supplémentaires.

2.2.2 Logiques d'ordre supérieur

Dans la logique du premier ordre, on peut quantifier sur les individus, mais pas sur leur propriétés. On peut par exemple exprimer la phrase « tous les hommes sont mortels », mais pas « aucun homme n'a toutes les qualités ».

On peut distinguer trois possibilités fortement reliées pour introduire cette possibilité :

Quantification sur les formules

Il s'agit d'étendre la syntaxe de façon à pouvoir quantifier sur les formules également. On obtient ainsi la logique des prédicats du second ordre avec la grammaire suivante :

$$P ::= A \mid \bot \mid \top \mid \neg P \mid P \wedge P \mid P \vee P \mid P \Rightarrow P \mid \forall x,\ P \mid \exists x,\ P \mid \forall X,\ P \mid \exists X,\ P$$

où X est une variable propositionnelle.

Il faut également étendre la notion de valuation, qui à toute variable de terme associe un élément du support du modèle, et à toute variable propositionnelle associe une relation de même arité sur ce support.

On peut alors étendre la relation $\models_\rho$:

- $m \models_\rho X(t_1,\ldots,t_n)$ ssi $([\![t_1]\!]_\rho,\ldots,[\![t_n]\!]_\rho) \in \rho(X)$
- $m \models_\rho \forall X,\ P$ ssi pour toute relation R de même arité que X sur le support de m, on a $m \models_{\rho'} P$ où ρ' est la valuation ρ sauf pour X à qui elle associe R.
- $m \models_\rho \exists X,\ P$ ssi il existe une relation R de même arité que X sur le support de m telle que $m \models_{\rho'} P$ où ρ' est la valuation ρ sauf pour X à qui elle associe R.

Une façon de généraliser ceci est d'utiliser la théorie des types, en particulier les systèmes de type purs présentés dans la section 3.1.3.

Schéma de compréhension

Cette autre approche consiste à faire passer les propositions dans la syntaxe des termes à l'aide d'un schéma d'axiomes, de façon à pouvoir quantifier dessus.

Le schéma d'axiomes de compréhension est le suivant :

$$\exists x,\ \forall y,\ y \in x \Leftrightarrow X(y) \qquad (x \text{ non libre dans } X(y)) \tag{2.1}$$

où X est une variable du second ordre. Toutefois, si on le considère sans restriction, on arrive rapidement à des paradoxes. Ainsi, si on applique ce schéma en instanciant $X(x)$ par $\neg x \in x$ on obtient le paradoxe de Russell : à partir de l'instance du schéma de compréhension on obtient un x tel que $\forall y,\ y \in x \Leftrightarrow \neg y \in y$, et si on applique ceci à x on obtient $x \in x \Leftrightarrow \neg x \in x$ d'où la contradiction.

Il existe plusieurs façon de dépasser ce paradoxe. La première consiste à restreindre les formules que l'on peut subsitituer pour X à une signature sans le symbole de prédicat $\in$. Toutefois, en faisant ceci, on n'obtient pas une logique plus riche car la logique avec ce schéma de compréhension est une extension conservative de celle sur la signature sans $\in$.

Une autre approche, celle de la théorie des ensembles d'Ernst Zermelo, consiste à ne considérer qu'un schéma de compréhension restreint :

$$\forall a,\ \exists x,\ \forall y,\ y \in x \Leftrightarrow y \in a \wedge X(y) \qquad (x \text{ non libre dans } X(y)) \tag{2.2}$$

Bien qu'il ne soit plus possible d'obtenir le paradoxe de Russell, et qu'on puisse prouver la cohérence de la théorie de Zermelo (dans une théorie strictement plus puissante), nous verrons que ce schéma pose toutefois des problèmes vis-à-vis de la normalisation des démonstrations, comme l'a remarqué Marcel Crabbé (1974).

La dernière technique, que nous allons utiliser dans la présentation de l'arithmétique d'ordre supérieur donnée dans le chapitre 5, consiste à travailler dans une logique du premier ordre multisortée, avec une sorte i pour chaque ordre $i \geq 1$ et un symbole de prédicat $\in^{\mathrm{i}}$ d'arité $[\mathrm{i}; \mathrm{i}+1]$ pour tout $i \geq 1$. Le schéma de compréhension devient un schéma de schémas d'axiomes

$$\exists x^{\mathrm{i}+1},\ \forall y^{\mathrm{i}},\ y \in^{\mathrm{i}} x \Leftrightarrow X(y) \qquad (x \text{ non libre dans } X(y)) \tag{2.3}$$

pour tout $i \geq 1$.

Le paradoxe de Russell ne peut alors plus être obtenu car x ne peut être à la fois dans une sorte i et $\mathrm{i}+1$. D'un point de vue sémantique, cela revient à considérer un modèle d'ordre supérieur comme un modèle du premier ordre avec plusieurs sortes d'individus dans le support : les individus d'ordre 1 ; les ensembles d'individus, ou de façon équivalente les prédicats sur ceux-ci ; les ensembles d'ensembles d'individus ; etc., le schéma de compréhension permettant de relier ceux-ci.

λ-calculs

Un autre domaine dans lequel on entend parler d'ordre supérieur est celui des langages avec lieurs, comme le λ-calcul. Frank Pfenning et Conal Elliot (1988) ont ainsi inroduites les syntaxes abstraites d'ordre supérieur.

Cette dénomination a pour origine la *théorie simple des types* d'Alonzo Church (1940), qui utilise le λ-calcul pour obtenir des quantifications d'ordre supérieur. Dans cette théorie les types sont obtenus à l'aide de la grammaire suivante :

$$\tau ::= \iota \mid o \mid \tau \Rightarrow \tau$$

où ι et o qui sont les types de base respectivement pour les individus et les valeurs de vérité. On a alors pour chaque type τ des symboles de fonction $\exists_\tau$ et $\forall_\tau$ de type $(\tau \Rightarrow o) \Rightarrow o$ et la quantification universelle d'ordre 2 $\forall X,\ P$ est par exemple représentée par $\forall_o\ (\lambda X,\ P)$.

On peut alors définir l'ordre d'un λ-terme typé par τ inductivement sur τ :

– l'ordre d'un type de base est 0

– l'ordre d'un type $\tau \Rightarrow \sigma$ est le maximum de l'ordre de τ plus 1 et de l'ordre de σ.

Il est à noter que le système proposé par Alonzo Church (1940) est incomplet pour la sémantique standard, dans laquelle les termes de type $\alpha \Rightarrow \beta$ sont interprétés par des fonctions de l'ensemble des interprétations des termes de type α vers l'ensemble des interprétations des termes de type β.

Dans la même veine, on parle d'unification d'ordre supérieur : un problème d'unification d'ordre supérieur consiste en un ensemble d'égalités entre λ-termes qu'il faut résoudre en trouvant une substitution telle que les membres de chaque égalité deviennent β- (voire $\beta\eta$-)équivalents quand on l'applique.

2.3 Systèmes de déduction pour la logique du premier ordre

2.3.1 Systèmes d'inférence

Nous donnons ici une définition assez générale de ce qu'est un système d'inférence, pour pouvoir l'appliquer ensuite aux différents systèmes d'inférence connus.

Définition 2.24 (Système d'inférence)**.** *On suppose donnés un ensemble de* jugements *J et un ensemble de fonctions $\mathfrak{S}$ de J vers J appelées* substitutions*.*

Une règle d'inférence est donnée par un ensemble fini de jugements $\{j_1, \ldots, j_n\}$ appelés les prémisses *de la règle, un jugement j appelé sa* conséquence *et un sous-ensemble de $V \subseteq \mathfrak{S}$ qui représente les* substitutions valables*. On la note*

$$\frac{j_1 \qquad \cdots \qquad j_n}{j}\, V$$

Une instance d'une règle d'inférence *est donnée par cette règle et une de ces subsitutions valables. Par abus de langage on appellera l'application de la substitution à une prémisse (resp. à la conséquence) de la règle une prémisse (resp. la conséquence) de l'instance de la règle.*

Un système d'inférence *est composé d'un ensemble fini de règles d'inférence.*

Une dérivation *(dans un système d'inférence donné) est un arbre fini étiqueté par des instances de règles d'inférences telles que la conséquence d'un nœud corresponde à une prémisse de son parent et qu'à une prémisse du parent ne correspondent qu'au plus un fils.*

Les prémisses des nœuds d'une dérivation qui ne correspondent pas à la conséquence de l'un de leurs fils sont appelées les prémisses *de la dérivation. La conséquence de la racine d'une dérivation est sa* conséquence*.*

Une dérivation est dite fermée *si l'ensemble de ses prémisses est vide.*

Nous distinguons prémisses et hypothèses, conséquence et conclusion, ainsi que dérivation et démonstration car ces notions ne coïncident pas pour certains systèmes de déduction comme nous le verrons par la suite.

Définition 2.25 (Dérivabilité, admissibilité)**.** *Une règle d'inférence est dite* dérivable *dans un système d'inférence I si pour chaque instance de cette règle il existe (au moins) une dérivation dans I ayant les mêmes prémisses et la même conclusion que l'instance.*

Une règle d'inférence est dérivable en hauteur bornée *dans I si il existe une borne h et pour chaque instance de cette règle une dérivation de hauteur au plus h dans I ayant les prémisses et la même conclusion que l'instance.*

Une règle d'inférence r est admissible *dans un système d'inférence I si pour toute instance de cette règle, s'il existe une dérivation fermée de chacune des prémisses de l'instance dans I alors il existe une dérivation fermée de sa conclusion dans I.*

On voit que la dérivabilité correspond à un changement local, tandis que l'admissibilité est une propriété plus globale. La dérivabilité peut également être vue comme l'admissibilité pour laquelle on considère toutes les dérivations, fermée ou non.

2.3.2 Systèmes schématiques

Il s'agit de la forme la plus simple de systèmes de déduction. Les formules sont déduites les unes des autres à l'aide d'un nombre fini de règles d'inférence. Cette dénomination a été utilisée entre autres par Rohit Parikh (1973) et Samuel Buss (1994), on parle également de systèmes de Frege, ou encore de systèmes de Hilbert.

Définition 2.26 (Règle d'inférence)**.** *Pour les systèmes schématiques, les* jugements *sont les méta-formules sur une signature du premier ordre, les substitutions sont celles de la définition 2.23.*

Les substitutions valables sont données à l'aide de restrictions de la forme de la forme « x n'est pas libre dans Φ » ou « x peut être librement substitué par s dans Φ » où Φ est une méta-formule, x une variable de $V \cup MV$ et s un terme de $\mathcal{T}(\Sigma, V \cup MV)$. Une substitution ne sera donc valable que si la restriction est vérifiée une fois la substitution effectuée.

Un schéma d'axiome *est une règle d'inférence dont l'ensemble des prémisses est vide. Un* axiome *est un schéma d'axiome sans méta-variables ni variables propositionnelles. On peut donc identifier un axiome à une formule.*

Un *système schématique* donc est un ensemble fini de telles règles d'inférence. En suivant la définition 2.14, $\mathbb{A}$ sera l'ensemble des formules du premier ordre, $\mathbb{P}$ sera l'ensemble des dérivations (fermées ou non), $[p]^{Pm}$ l'ensemble des prémisses de la dérivation p et $[p]_{Cl}$ la conséquence de p.

En général, on parle de système de Frege uniquement quand la seule règle d'inférence qui ne soit pas un schéma d'axiome est le *modus ponens* :

$$\frac{X \Rightarrow Y \qquad X}{Y} \tag{2.4}$$

Gerhard Gentzen (1934) propose un système schématique correct et complet pour la logique classique du premier ordre. Il comporte les schémas d'axiomes suivants :

$$X \Rightarrow X \tag{2.5}$$
$$X \Rightarrow Y \Rightarrow X \tag{2.6}$$
$$(X \Rightarrow X \Rightarrow Y) \Rightarrow X \Rightarrow Y \tag{2.7}$$
$$(X \Rightarrow Y \Rightarrow Z) \Rightarrow Y \Rightarrow X \Rightarrow Z \tag{2.8}$$
$$(X \Rightarrow Y) \Rightarrow (Y \Rightarrow Z) \Rightarrow X \Rightarrow Z \tag{2.9}$$
$$(X \wedge Y) \Rightarrow X \tag{2.10}$$
$$(X \wedge Y) \Rightarrow Y \tag{2.11}$$
$$(X \Rightarrow Y) \Rightarrow (X \Rightarrow Z) \Rightarrow X \Rightarrow (Y \wedge Z) \tag{2.12}$$
$$X \Rightarrow (X \vee Y) \tag{2.13}$$
$$Y \Rightarrow (X \vee Y) \tag{2.14}$$
$$(X \Rightarrow Z) \Rightarrow (Y \Rightarrow Z) \Rightarrow (X \vee Y) \Rightarrow Z \tag{2.15}$$
$$(X \Rightarrow Y) \Rightarrow (X \Rightarrow Y \Rightarrow \bot) \Rightarrow X \Rightarrow \bot \tag{2.16}$$
$$(X \Rightarrow \bot) \Rightarrow X \Rightarrow Y \tag{2.17}$$
$$(\forall x.\ X(x)) \Rightarrow X(t) \qquad (x \text{ peut être librement substitué par } t \text{ dans } X(x)) \tag{2.18}$$
$$X(t) \Rightarrow \exists x.\ X(x) \qquad (x \text{ peut être librement substitué par } t \text{ dans } X(x)) \tag{2.19}$$
$$X \vee (X \Rightarrow \bot) \tag{2.20}$$

ainsi que le *modus ponens* (2.4) et les deux règles d'inférence :

$$\frac{X \Rightarrow Y(y)}{X \Rightarrow \forall x.\ Y(x)}\ (y \text{ n'est pas libre dans } X \Rightarrow \forall x.\ Y(x)) \tag{2.21}$$

$$\frac{Y(y) \Rightarrow X}{(\exists x.\ Y(x)) \Rightarrow X}\ (y \text{ n'est pas libre dans } (\exists x.\ Y(x)) \Rightarrow X) \tag{2.22}$$

Si on considère en plus un prédicat binaire d'égalité (infixe) $=$, on rajoute alors un axiome de réflexivité

$$\forall x,\ x = x \tag{2.23}$$

et le schéma d'axiome de Leibniz

$$\forall x,\ \forall y,\ x = y \Rightarrow X(x) \Rightarrow X(y) \tag{2.24}$$

2.3.3 Déduction naturelle

Pour tout système schématique raisonnable on peut démontrer le théorème de déduction : si on a une démonstration de b sous l'hypothèse a alors on peut démontrer $a \Rightarrow b$ sans hypothèse. Toutefois ceci n'est pas reflété au niveau de la syntaxe : il n'y a pas de moyen de passer de la première démonstration à la seconde de façon syntaxique.

La déduction naturelle a été introduite de façon indépendante par Stanisław Jaśkowski (1934) et par Gerhard Gentzen (1934). Elle consiste à avoir une nouvelle règle d'inférence, non exprimable dans un système schématique, de la forme

$$\overset{\vdash}{\Rightarrow}\ \frac{\begin{array}{c}[X]\\ \vdots\\ Y\end{array}}{X \Rightarrow Y}$$

où X est une hypothèse utilisée pour démontrer Y, mais n'est plus une hypothèse de la démonstration de $X \Rightarrow Y$. (On dit que l'hypothèse X est déchargée.)

On voit que cette règle d'inférence $\overset{\vdash}{\Rightarrow}$ permet d'introduire le connecteur $\Rightarrow$, tandis que le *modus ponens* (2.4) permet de l'éliminer. On pourra donc noter ce dernier $\overset{\Rightarrow}{\vdash}$, et on dira que $X \Rightarrow Y$ est sa *prémisse principale*. On peut généraliser ceci à tous les connecteurs et quantificateurs de la logique du premier ordre et obtient alors la déduction naturelle du premier ordre, dénotée NJ, formé par les règles d'inférence de la figure 2.2.

On remarque que les règles d'inférence qui déchargent des hypothèses ne rentrent pas dans le cadre de la définition 2.24. Néanmoins, comme remarqué par Robert Feys et Jean Ladrière (1965) dans leur traduction de Gerhard Gentzen (1934), il est également possible de présenter la déduction naturelle sous forme d'inférence entre séquences[3]. On

[3] Cf. note 3 page 8.

$$\vdash_{\Rightarrow} \frac{\begin{array}{c}[X]\\ \vdots\\ Y\end{array}}{X \Rightarrow Y} \qquad \vdash_{\Rightarrow} \frac{X \Rightarrow Y \qquad X}{Y}$$

$$\vdash_{\wedge} \frac{X \qquad Y}{X \wedge Y} \qquad \vdash_{\wedge 1} \frac{X \wedge Y}{X} \qquad \vdash_{\wedge 2} \frac{X \wedge Y}{Y}$$

$$\vdash_{\vee 1} \frac{X}{X \vee Y} \qquad \vdash_{\vee 2} \frac{Y}{X \vee Y} \qquad \vdash_{\vee} \frac{X \vee Y \qquad \begin{array}{c}[X]\\ \vdots\\ Z\end{array} \qquad \begin{array}{c}[Y]\\ \vdots\\ Z\end{array}}{Z}$$

$$\vdash_{\forall} \frac{X(y)}{\forall x,\ X(x)}$$ y non libre dans $X(x)$ ni dans les hypothèses de la démonstration

$$\vdash_{\forall} \frac{\forall x,\ X(x)}{X(t)}$$

$$\vdash_{\exists} \frac{X(t)}{\exists x, X(x)} \qquad \vdash_{\exists} \frac{\exists x, X(x) \qquad \begin{array}{c}[X(y)]\\ \vdots\\ Y\end{array}}{Y}$$ y non libre dans $X(x), Y$ ni dans les hypothèses de la démonstration hormis $X(y)$

$$\vdash_{\top} \frac{}{\top} \qquad \vdash_{\perp} \frac{\perp}{X}$$

$$\vdash_{\neg} \frac{\begin{array}{c}[X]\\ \vdots\\ \perp\end{array}}{\neg X} \qquad \vdash_{\neg} \frac{\neg X \qquad X}{Y}$$

FIG. 2.2 : Règles de la déduction naturelle

$$\overset{\frown}{\vdash}\ \frac{}{\Gamma, X \vdash X}$$

$$\underset{\Rightarrow}{\vdash}\ \frac{\Gamma, X \vdash Y}{\Gamma \vdash X \Rightarrow Y} \qquad \underset{\vdash}{\Rightarrow}\ \frac{\Gamma \vdash X \Rightarrow Y \qquad \Gamma \vdash X}{\Gamma \vdash Y}$$

$$\underset{\wedge}{\vdash}\ \frac{\Gamma \vdash X \qquad \Gamma \vdash Y}{\Gamma \vdash X \wedge Y} \qquad \underset{\vdash}{\wedge^1}\ \frac{\Gamma \vdash X \wedge Y}{\Gamma \vdash X} \qquad \underset{\vdash}{\wedge^2}\ \frac{\Gamma \vdash X \wedge Y}{\Gamma \vdash Y}$$

$$\underset{\vee 1}{\vdash}\ \frac{\Gamma \vdash X}{\Gamma \vdash X \vee Y} \qquad \underset{\vee 2}{\vdash}\ \frac{\Gamma \vdash Y}{\Gamma \vdash X \vee Y} \qquad \underset{\vdash}{\vee}\ \frac{\Gamma \vdash X \vee Y \qquad \Gamma, X \vdash Z \qquad \Gamma, Y \vdash Z}{\Gamma \vdash Z}$$

$$\underset{\forall}{\vdash}\ \frac{\Gamma \vdash X(y)}{\Gamma \vdash \forall x,\ X(x)}\ y \text{ non libre dans } \Gamma, X(x) \qquad \underset{\vdash}{\forall}\ \frac{\Gamma \vdash \forall x,\ X(x)}{\Gamma \vdash X(t)}$$

$$\underset{\exists}{\vdash}\ \frac{\Gamma \vdash X(t)}{\Gamma \vdash \exists x, X(x)} \qquad \underset{\vdash}{\exists}\ \frac{\Gamma \vdash \exists x, X(x) \qquad \Gamma, X(y) \vdash Y}{\Gamma \vdash Y}\ y \text{ non libre dans } \Gamma, X(x), Y$$

$$\underset{\top}{\vdash}\ \frac{}{\Gamma \vdash \top} \qquad \underset{\vdash}{\bot}\ \frac{\Gamma \vdash \bot}{\Gamma \vdash X}$$

$$\underset{\neg}{\vdash}\ \frac{\Gamma, X \vdash \bot}{\Gamma \vdash \neg X} \qquad \underset{\vdash}{\neg}\ \frac{\Gamma \vdash \neg X \qquad \Gamma \vdash X}{\Gamma \vdash Y}$$

FIG. 2.3 : Présentation des règles de la déduction naturelle avec des séquences

considère les hypothèses Γ utilisées à une position dans la démonstration en utilisant ce qu'on appellera la séquence $\Gamma \vdash q$ au lieu de la formule q.

Définition 2.27 (Séquence (intuitionniste))**.** *Une* séquence *est un couple formé par un multiensemble fini de formules et une formule. On notera la séquence formée par* Γ *et* q *par* $\Gamma \vdash q$*.* Γ *représente les* hypothèses *de la séquence,* q *sa* conclusion*.*

Les jugements seront donc des méta-séquences dans lesquelles pourront apparaître des variables de contexte Γ pouvant être substituées par des multiensemble de formules. Les règles de la déduction naturelle peuvent alors s'écrire selon la figure 2.3. La règle $\overset{\frown}{\vdash}$ permet de refermer les dérivations en utilisant les hypothèses disponibles. On définit alors la déduction naturelle comme un système de déduction avec pour $\mathbb{A}$ les formules du premier ordre, pour $\mathbb{P}$ les dérivations fermées, pour $[p]^{Pm}$ les hypothèses de la conséquence de p, pour $[p]_{Cl}$ la conclusion de sa conséquence.

Notons, par souci de complétude, qu'il existe également une notation basée sur des indentations successives proposée par Frederic FITCH (1952), mais que nous n'utiliserons pas.

La logique associé à la déduction naturelle est correcte pour la logique classique du premier ordre, mais pas complète, sauf si on autorise le tiers exclus $X \vee \neg X$ comme

schéma d'axiome supplémentaire. L'absence de la dérivabilité de ce principe, décrié par le courant intuitionniste de Luitzen BROUWER, en déduction naturelle, a conduit à appeler la logique associée *logique intuitionniste*[4].

2.3.4 Calculs des séquences

Les règles d'introduction de la déduction naturelle ont donné à Gerhard GENTZEN (1934) l'idée du calcul des séquences : pour démontrer une formule qui n'est pas une proposition atomique, on cherche une démonstration de ces sous-formules directes. On décompose de la même façon les hypothèses, ce qui fait qu'on ne travaille en fait pas sur une formule mais sur une séquence.

Les jugements considérés sont donc des méta-séquences. Dans le cas intuitionniste il s'agira de séquence telle que définies en 2.27. Dans le cas classique, on autorisera également d'avoir plusieurs formules à droite de la séquence.

Définition 2.28 (Séquence (classique))**.** *Une* séquence *est une couple formé par deux multiensembles finis de formules. On notera la séquence formée par deux multiensemble de formules* Γ *et* Δ *par* $\Gamma \vdash \Delta$*.* Γ *représente les* hypothèses *de la séquence,* Δ *ses* conclusions.

Le système d'inférence du *calcul des séquences* pour la logique classique du premier ordre, appelé LK, est donné dans la figure 2.4. On distingue trois groupes de règles d'inférence : les règles d'identité permettent de fermer une démonstration en identifiant une hypothèse et une conclusion X ($\frac{}{X \vdash X}$) ou de faire une coupure autour d'une formule X ($\frac{}{\vdash}$). Les règles structurelles permettent de gérer les séquences : Les règles de contractions ($\cdot, \cdot \vdash$ et $\vdash \cdot, \cdot$) servent à identifier des hypothèses ou des conclusions apparaissant en double, ce qui permet de considérer les multiensembles des hypothèses et des conclusions comme des ensembles. Les règles d'affaiblissement ($\cdot \vdash$ et $\vdash \cdot$) servent à ajouter des hypothèses ou des conclusions qui ne sont pas strictement utiles à la démonstration. Il est à noter que si on avait définies les séquences comme des couples de listes ordonnées et non pas comme des couples de multiensembles, il y aurait également eu besoin de deux règles permettant d'échanger l'ordre de deux formules dans un côté d'une séquence. Le troisième groupe, celui des règles logiques, consiste à décomposer une des formules de la séquence, appelée *formule active* de la règle, en une ou plusieurs de ses sous-formules directes. On peut également parler de formule active dans le cas des règles d'identité et structurelles.

Le calcul des séquences est complet pour la logique classique du premier ordre. On remarquera la parfaite dualité des opérateurs $\vee$ et $\wedge$ ainsi que des quantificateurs $\exists$ et $\forall$.

Remarque 2.6 *:* On peut également considérer des règles gauche pour $\top$ et droite pour $\bot$:

[4]Plus précisément, la logique proposée par Arend HEYTING s'est avérée coïncider avec celle de la déduction naturelle.

Règles d'identité

$$\frown\vdash \frac{}{X \vdash X} \qquad \vdash\smile \frac{\Gamma, X \vdash \Delta \qquad \Gamma \vdash X, \Delta}{\Gamma \vdash \Delta}$$

Règles structurelles

$$\cdot,\cdot\vdash \frac{\Gamma, X, X \vdash \Delta}{\Gamma, X \vdash \Delta} \qquad \vdash\cdot,\cdot \frac{\Gamma \vdash X, X, \Delta}{\Gamma \vdash X, \Delta}$$

$$\cdot\vdash \frac{\Gamma \vdash \Delta}{\Gamma, X \vdash \Delta} \qquad \vdash\cdot \frac{\Gamma \vdash \Delta}{\Gamma \vdash X, \Delta}$$

Règles logiques

$$\Rightarrow\vdash \frac{\Gamma \vdash X, \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, X \Rightarrow Y \vdash \Delta} \qquad \vdash\Rightarrow \frac{\Gamma, X \vdash Y, \Delta}{\Gamma \vdash X \Rightarrow Y, \Delta}$$

$$\wedge 1\vdash \frac{\Gamma, X \vdash \Delta}{\Gamma, X \wedge Y \vdash \Delta} \qquad \wedge 2\vdash \frac{\Gamma, Y \vdash \Delta}{\Gamma, X \wedge Y \vdash \Delta} \qquad \vdash\wedge \frac{\Gamma \vdash X, \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash X \wedge Y, \Delta}$$

$$\vee\vdash \frac{\Gamma, X \vdash \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, X \vee Y \vdash \Delta} \qquad \vdash\vee 1 \frac{\Gamma \vdash X, \Delta}{\Gamma \vdash X \vee Y, \Delta} \qquad \vdash\vee 2 \frac{\Gamma \vdash Y, \Delta}{\Gamma \vdash X \vee Y, \Delta}$$

$$\forall\vdash \frac{\Gamma, X(t) \vdash \Delta}{\Gamma, \forall x,\ X(x) \vdash \Delta} \qquad \vdash\forall \frac{\Gamma \vdash X(y), \Delta}{\Gamma \vdash \forall x,\ X(x), \Delta}\ \begin{array}{l} y \text{ non libre dans} \\ \Gamma, X(x), \Delta \end{array}$$

$$\exists\vdash \frac{\Gamma, X(y) \vdash \Delta}{\Gamma, \exists x, X(x) \vdash \Delta}\ \begin{array}{l} y \text{ non libre dans} \\ \Gamma, X(x), \Delta \end{array} \qquad \vdash\exists \frac{\Gamma \vdash X(t), \Delta}{\Gamma \vdash \exists x, X(x), \Delta}$$

$$\bot\vdash \frac{}{\bot \vdash} \qquad \vdash\top \frac{}{\vdash \top}$$

$$\neg\vdash \frac{\Gamma \vdash X, \Delta}{\Gamma, \neg X \vdash \Delta} \qquad \vdash\neg \frac{\Gamma, X \vdash \Delta}{\Gamma \vdash \neg X, \Delta}$$

FIG. 2.4 : Règles du calcul des séquences classique

$$\top\vdash \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta} \qquad\qquad \vdash\bot \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}$$

Elles sont redondantes, puisque ce sont des instances des règles d'affaiblissement $\cdot\vdash$ et $\vdash\cdot$. Néanmoins elles permettent pour toute formule non atomique dans une séquence d'avoir une règle logique pour la décomposer.

On définit alors le calcul des séquences classique comme un système de déduction avec pour $\mathbb{A}$ les formules du premier ordre, pour $\mathbb{P}$ les dérivations fermées, pour $[p]^{Pm}$ les hypothèses de la conséquence de p, pour $[p]_{Cl}$ la disjonction des conclusions de sa conséquence. On remarque qu'une séquence $\Gamma \vdash \Delta$ correspond à la formule close $\forall x_1, \ldots, x_n,\ (\bigwedge \Gamma) \Rightarrow (\bigvee \Delta)$, avec $\{x_1, \ldots, x_n\} = FV(\Gamma, \Delta)$. On notera $\mathcal{P}(\Gamma \vdash \Delta)$ cette formule.

Obtenir un calcul des séquences pour la logique intuitionniste à partir de LK est très simple : il suffit de ne considérer que des séquences avec une seule conclusion à droite. Les règle de contraction et d'affaiblissement à droite ($\vdash\cdot,\cdot$ et $\vdash\cdot$) sont par conséquence interdites, et la règle $\overset{\vdash}{\smile}$ devient :

$$\overset{\vdash}{\smile} \frac{\Gamma, X \vdash Y \qquad \Gamma \vdash X}{\Gamma \vdash Y}$$

On peut supposer que si on a une séquence sans formules à droite alors on a en fait la formule $\bot$ à droite.

Le *calcul des séquences* pour la logique intuitionniste du premier ordre est appelé LJ. Vu en tant que système de déduction, $[p]_{Cl}$ sera définie comme la conclusion de la conséquence de p.

Gerhard Gentzen (1934) a démontré que le système schématique composé des règles d'inférence (2.4) à (2.22), la déduction naturelle avec en plus le schéma du tiers exclu, et le calcul des séquences classique sont tous corrects et complets les uns envers les autres. Ceci est également vrai quand on retire le schéma du tiers exclu aux deux premiers systèmes et qu'on considère LJ pour le troisième.

Inversibilité des règles

Contrairement à la déduction naturelle où on ne considère qu'une seule formule à la fois (celle à droite de la séquence), en calcul des séquences on peut potentiellement appliquer une règle d'inférence à chacune des formules non atomiques composant la séquence. Il est donc important de savoir quelles stratégies d'application des règles d'inférence restent complètes. Stephen Kleene (1952*c*) s'est en particulier intéressé à l'ensemble des règles qu'il est possible d'appliquer sans jamais avoir besoin de revenir en arrière, règles qui sont alors dites inversibles :

Définition 2.29 (Règle inversible)**.** *Une règle logique du calcul des séquences est dite* inversible *si pour toute instance de cette règle, sa conséquence est démontrable ssi ses prémisses le sont.*

Dans la suite, tous les résultats seront les mêmes qu'on considère le calcul des séquences avec $\overset{\frown}{\vdash}$ ou sans.

Par exemple, $\vdash\Rightarrow$ est inversible : on peut montrer que $\Gamma \vdash P \Rightarrow Q, \Delta$ est démontrable ssi $\Gamma, P \vdash Q, \Delta$ l'est. Toute règle n'est cependant pas inversible : ainsi, pour le calcul des séquences classique, $\vdash\exists$ ne l'est pas, car la séquence $\exists x,\ P(x) \vdash \exists x,\ P(x)$ possède la démonstration

$$\exists\vdash \dfrac{\vdash\exists \dfrac{\overset{\frown}{\vdash} \dfrac{}{P(x) \vdash P(x)}}{P(x) \vdash \exists x,\ P(x)}}{\exists x,\ P(x) \vdash \exists x,\ P(x)}$$

alors que $\exists x,\ P(x) \vdash P(t)$ n'est pas démontrable quelque soit t pour des raisons de liberté de la variable introduite par $\exists\vdash$.

Stephen Kleene (1952*c*) a montré que dans le calcul des séquences classiques, les règles non inversibles sont $\vdash\vee i$ et $\vdash\exists$ ainsi que leur duaux $\wedge i\vdash$ et $\forall\vdash$.

Une façon de s'affranchir de la non-inversibilité d'une règle est d'appliquer une contraction juste avant. Ceci permet de pouvoir appliquer la règle à nouveau au cas où la première application est effectuée trop tôt. Par exemple, on peut modifier la règle $\vdash\exists$ en :

$$\vdash\exists \dfrac{\Gamma \vdash X(t), \exists x,\ X(x), \Delta}{\Gamma \vdash \exists x,\ X(x), \Delta}$$

qui est inversible. On peut faire de même avec $\forall\vdash$, $\wedge i\vdash$ et $\vdash\vee i$. Dans ces deux derniers cas, on peut même encore simplifier les règles obtenues : en effet, après avoir appliqué $\vdash\vee 1$ et $\vdash\vee 2$ sur une formule, on n'obtiendra aucune nouvelle formule en les appliquant à nouveau. On considère alors la règle suivante :

$$\vdash\vee \dfrac{\Gamma \vdash X, Y, \Delta}{\Gamma \vdash X \vee Y, \Delta}$$

qui est inversible.

Avec ces nouvelles règles, il n'est plus nécessaire d'avoir de contraction ($\cdot,\cdot\vdash$ et $\vdash\cdot,\cdot$). Les affaiblissements peuvent quant à eux être remontés jusqu'aux règles $\overset{\frown}{\vdash}$, $\bot\vdash$ et $\vdash\top$, qui deviennent alors

$$\overset{\frown}{\vdash} \dfrac{}{\Gamma, X \vdash X, \Delta} \qquad \bot\vdash \dfrac{}{\Gamma, \bot \vdash \Delta} \qquad \vdash\top \dfrac{}{\Gamma \vdash \top, \Delta}$$

On obtient ainsi le système G4 de Stephen Kleene (1967)[5,6] qui est représenté dans la figure 2.5, qui est donc également complet pour la logique du premier ordre. On peut

Règles d'identité

$$\vdash\!\frown \frac{}{\Gamma, X \vdash X, \Delta} \qquad \vdash\!\smile \frac{\Gamma, X \vdash \Delta \qquad \Gamma \vdash X, \Delta}{\Gamma \vdash \Delta}$$

Règles logiques

$$\Rightarrow\!\vdash \frac{\Gamma \vdash X, \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, X \Rightarrow Y \vdash \Delta} \qquad \vdash\!\Rightarrow \frac{\Gamma, X \vdash Y, \Delta}{\Gamma \vdash X \Rightarrow Y, \Delta}$$

$$\wedge\!\vdash \frac{\Gamma, X, Y \vdash \Delta}{\Gamma, X \wedge Y \vdash \Delta} \qquad \vdash\!\wedge \frac{\Gamma \vdash X, \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash X \wedge Y, \Delta}$$

$$\vee\!\vdash \frac{\Gamma, X \vdash \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, X \vee Y \vdash \Delta} \qquad \vdash\!\vee \frac{\Gamma \vdash X, Y, \Delta}{\Gamma \vdash X \vee Y, \Delta}$$

$$\forall\!\vdash \frac{\Gamma, X(t), \forall x,\ X(x) \vdash \Delta}{\Gamma, \forall x,\ X(x) \vdash \Delta} \qquad \vdash\!\forall \frac{\Gamma \vdash X(y), \Delta}{\Gamma \vdash \forall x,\ X(x), \Delta}\ \begin{array}{l} y \text{ non libre dans} \\ \Gamma, X(x), \Delta \end{array}$$

$$\exists\!\vdash \frac{\Gamma, X(y) \vdash \Delta}{\Gamma, \exists x,\ X(x) \vdash \Delta}\ \begin{array}{l} y \text{ non libre dans} \\ \Gamma, X(x), \Delta \end{array} \qquad \vdash\!\exists \frac{\Gamma \vdash X(t), \exists x,\ X(x), \Delta}{\Gamma \vdash \exists x,\ X(x), \Delta}$$

$$\bot\!\vdash \frac{}{\Gamma, \bot \vdash \Delta} \qquad \vdash\!\top \frac{}{\Gamma \vdash \top, \Delta}$$

$$\neg\!\vdash \frac{\Gamma \vdash X, \Delta}{\Gamma, \neg X \vdash \Delta} \qquad \vdash\!\neg \frac{\Gamma, X \vdash \Delta}{\Gamma \vdash \neg X, \Delta}$$

FIG. 2.5 : Système G4 de Stephen KLEENE

montrer que l'affaiblissement et la contraction y sont admissibles. Si toutes les règles sont alors inversibles, il est possible de les appliquer à tout moment. Néanmoins, il se peut que l'on applique une règle alors que ce n'était pas nécessaire : on dira que l'application d'une règle à la racine d'une démonstration est *inutile* si on peut démontrer sa conséquence par affaiblissement d'une de ses sous-démonstrations directe, et l'application d'une règle à une autre position sera inutile si elle l'est dans la sous-démonstration à cette position.

En s'appuyant sur la logique linéaire, Jean-Marc Andreoli (1992) a obtenue une analyse plus fine de l'inversibilité. L'idée est de distinguer les connecteurs par le comportement de leurs règles. On dit qu'un connecteur est négatif, ou asynchrone, si sa règle droite est inversible. On dit qu'il est positif, ou synchrone, si sa règle gauche l'est. Comme on l'a vu, en logique classique, un connecteur peut être à la fois positif et négatif, auquel cas on dira plutôt qu'il est neutre, ce qui n'est pas le cas en logique linéaire. En logique classique, les seuls connecteurs non neutres sont $\forall$ qui est négatif et $\exists$ qui est positif. Quand une formule est à gauche d'une séquence, sa polarité est inversée. La *focalisation* consiste, quand on décompose une formule positive d'une séquence, à décomposer immédiatement les sous-formules obtenues si celles-ci sont positives. On obtient alors une méthode de recherche de démonstration complète qui consiste à choisir une formule positive et décomposer tous ses connecteurs positifs, puis décomposer tous les connecteurs négatifs des séquences résultantes, et recommencer jusqu'à obtenir une démonstration.

Dans le cadre de la logique intuitionniste, un nombre plus important de règles ne sont pas inversibles. Stephen Kleene (1952*c*) relève ainsi que pour LJ seules $\wedge\vdash$, $\vee\vdash$ et $\exists\vdash$ sont inversibles. Pour améliorer l'inversibilité des règles, on peut tenter de recourir aux mêmes techniques que pour obtenir G4, mais nous sommes limités par l'absence de contraction à droite dans LJ. Toutefois, une analyse des démonstrations de LJ montre que l'on peut s'autoriser à avoir plusieurs formules à droite et à faire des contractions à droite dans la plupart des cas, sauf quand on rencontre une règle $\vdash\Rightarrow$, $\vdash\neg$ ou $\vdash\forall$. En effet, au final, seules les formules utilisées dans $\frown$ sont importantes et si on a une démonstration avec plusieurs séquences à droite on peut la transformer pour ne conserver que la formule à droite de la séquence qui conduit à celles-ci, sauf dans le cas de $\vdash\Rightarrow$ puisque si la formule active est $A \Rightarrow B$ alors A peut être utilisé dans $\frown$ avec une autre formule à droite de la séquence, pareillement pour $\vdash\neg$, et dans le cas $\vdash\forall$ la variable introduite peut être utilisée pour instancier une autre formule à droite avec la règle $\vdash\exists$. On considérera donc un calcul des séquences avec des conclusions multiples similaire à LK, mais avec des règles $\vdash\Rightarrow$, $\vdash\neg$ et $\vdash\forall$

[5] Stephen Kleene (1952*b*) a introduit trois systèmes dérivés de LK : G1 correspondant à LK, G2 où la règle $\vdash$ est remplacée par une règle Mix pouvant couper plusieurs formules, et G3 où on applique une contraction avant toutes les règles logiques (y compris les inversibles). Il s'est par la suite inspiré du système LC de Stig Kanger (1957) pour G4.

[6] Le système G4 de Stephen Kleene (1967) n'autorise en fait $\frown$ que quand X est atomique, ce que nous ne ferons pas ici.

$$\vdash\Rightarrow \frac{\Gamma, X \vdash Y}{\Gamma \vdash X \Rightarrow Y, \Delta}$$

$$\vdash\forall \frac{\Gamma \vdash X(y)}{\Gamma \vdash \forall x,\ X(x), \Delta} \begin{array}{l} y \text{ non libre dans} \\ \Gamma, X(x) \end{array}$$

$$\vdash\neg \frac{\Gamma, X \vdash \bot}{\Gamma \vdash \neg X, \Delta}$$

FIG. 2.6 : Règles du système LB, différentes de celles de LK et G4

modifiées de façon à contraindre le choix de la formule à droite. Il semblerait qu'un calcul de ce type ait été introduit pour la première fois par Shôji MAEHARA (1954) sous le nom de L'J. On le retrouve ensuite sous plusieurs dénominations, par exemple LB chez Arild WAALER et Lincoln WALLEN (1999), ou LJm chez Grigori MINTS (2000), avec quelques petites différences suivant les auteurs. Dans la version que nous utiliserons, on essaiera d'avoir des règles inversibles, mais sans avoir recours à des contractions implicites, afin d'être sûr qu'appliquer une règle logique supprime un connecteur dans la séquence. Par conséquent, la règle de contraction ne sera pas admissible. Le calcul que nous appellerons LB sera donc constitué des règles $\cdot,\vdash$, $\vdash,\cdot$, $\forall\vdash$ et $\vdash\exists$ de LK, des règles $\Rightarrow\vdash$, $\wedge\vdash$, $\vdash\wedge$, $\vee\vdash$, $\vdash\vee$, $\exists\vdash$, $\vdash\forall$, $\bot\vdash$, $\vdash\top$ et $\vdash\neg$ de G4, et des règles de la figure 2.6. Il est correct et complet pour la logique intuitionniste du premier ordre, et en regardant les cas de permutabilité des règles donnés par Arild WAALER et Lincoln WALLEN (1999), on peut montrer que les seules règles non inversibles qui subsistent sont $\Rightarrow\vdash$, $\forall\vdash$, $\vdash\Rightarrow$, $\vdash\neg$, $\vdash\forall$ et $\vdash\exists$. Il est possible d'utiliser des contractions pour rendre $\Rightarrow\vdash$, $\forall\vdash$ et $\vdash\exists$ inversibles comme dans G4, mais ce n'est pas possible pour $\vdash\Rightarrow$, $\vdash\neg$ et $\vdash\forall$.

On peut tenter d'étendre la focalisation à LB. Néanmoins, on est restreint par l'absence de dualité. Ainsi, avec nos définitions, $\forall$ ne serait ni positif, ni négatif. On définit alors une polarité non pas sur les formules, mais sur les règles d'inférence.

Définition 2.30 (Polarité)**.** *Une règle d'inférence gauche (resp. droite) du calcul des séquences est dite :*

- négative *si elle est inversible et sa règle droite (resp. gauche) ne l'est pas ;*
- positive *si elle n'est pas inversible ;*
- neutre *si à la fois la règle gauche et droite sont inversibles.*

On voit que la polarité d'une formule pour le calcul des séquences classique peut alors être définie comme la polarité de sa règle droite. Dans LB, les règles négatives sont $\neg\vdash$ et $\vdash\exists$; les règles positives sont $\Rightarrow\vdash$, $\forall\vdash$, $\vdash\Rightarrow$, $\vdash\neg$, $\vdash\exists$ et $\vdash\forall$; toutes les autres sont par conséquent neutres. La focalisation consiste alors à choisir une formule et y appliquer ainsi qu'à ses sous-formules toutes les règles positives ou neutres possibles, puis à appliquer toutes les règles négatives ou neutres possibles, et de recommencer.

Élimination des coupures

Un des résultats les plus importants concernant le calcul des séquences est l'admissibilité de la règle $\vdash_\smile$. Gerhard Gentzen (1934) parle d'ailleurs d'*Hauptsatz*, théorème principal. En effet, ce théorème implique la propriété de la sous-formule, c'est-à-dire que pour démontrer une séquence on a besoin uniquement des sous-formules de cette séquence. L'admissibilité de la règle de coupure, associée à la complétude du calcul des séquences, implique également la cohérence de la logique du premier ordre : en effet, il est impossible de démontrer la séquence $\vdash \bot$, car aucune règle ne peut s'appliquer dessus à part $\vdash_\smile$. Elle implique également, en logique intuitionniste, la *propriété du témoin* : si $\exists x,\ P(x)$ est valide en logique du premier ordre intuitionniste, alors on peut exhiber un terme t tel que $P(t)$ soit valide. En effet, l'admissibilité de $\vdash_\smile$ implique que le calcul des séquences sans $\vdash_\smile$ intuitionniste est correct et complet pour cette logique, or seule $\vdash\exists$ est applicable sur $\vdash \exists x,\ P(x)$.

La propriété de la sous-formule peut également être utilisée pour avoir des méthodes de recherche de démonstrations avec un espace de recherche borné. La méthode des tableaux (voir par exemple Marcello D'Agostino, Dov Gabbay, Reiner Hähnle et Joachim Posegga, 1999) peut ainsi être vue comme une stratégie de recherche complète dans le calcul des séquences.

La démonstration de l'admissibilité de $\vdash_\smile$, telle qu'on peut par exemple la trouver chez Jean-Yves Girard, Yves Lafont et Paul Taylor (1989), est intéressante en elle-même, car elle fournit une procédure pour éliminer les coupures. Cette procédure, non déterministe et même non confluente, peut néanmoins être formalisée, comme l'a par exemple fait Frank Pfenning (2000) en Elf.

Nous allons utiliser cette procédure dans la suite de cette thèse, ce qui explique pourquoi nous en donnons le détail. On se place dans G4. On procède par induction sur la formule autour de laquelle on coupe et sur les sous-démonstrations directes. Plus précisément, on définit une notion de squelette de démonstration, et on définit un ordre dessus que la procédure d'élimination des coupures fait décroître.

Définition 2.31 (Squelette de démonstration)**.** *Le* squelette *d'une démonstration en calcul des séquences est l'arbre de même support que la démonstration et dont les positions sont étiquetées uniquement par les couples formés de la règle d'inférence et de la formules active à cette position dans la démonstration.*

On va utiliser l'ordre suivant entre les squelettes : On définit la précession suivante (infinie, mais Noethérienne) sur les couples règle d'inférence–formule active : $(\vdash_\smile, P) > (\vdash_\smile, Q)$ et $(\frown, P) > (\frown, Q)$ si $P > Q$ pour l'ordre sous-formule, et $(\vdash_\smile, P) > (\frown, Q) > (\mathsf{r}, R)$ pour toutes les règles d'inférence r autres que $\vdash_\smile$ et $\frown$. On considère alors l'ordre RPO basé sur cette précession. Ce n'est peut-être pas l'ordre le plus général pour démontrer l'admissibilité de $\vdash_\smile$, mais nous verrons qu'il est bien adapté pour faire des raisonnement par induction sur les preuves et aussi quand il s'agira de munir l'espace

des démonstration d'un ordre pour obtenir une procédure de complétion (cf. chapitre 4). Sur les démonstrations, on dit que $p < q$ si le squelette de p est plus petit que le squelette de q pour l'ordre RPO. Il est à noter que Marc Aiguier et Delphine Longuet (2007) ont généralisée cette utilisation d'un ordre RPO pour démontrer la normalisation d'un système de démonstration.

Il nous faut tout d'abord montrer trois lemmes qui montrent qu'on peut affaiblir une démonstration sans changer l'ordre, ou inverser des règles et faire des contractions en faisant décroître l'ordre.

Lemme 2.3 (Affaiblissement)**.** *Pour toute démonstration de $\Gamma \vdash \Delta$ on peut trouver des démonstrations de $\Gamma, P \vdash \Delta$ et de $\Gamma \vdash P, \Delta$ avec le même squelette.*

Démonstration. Il suffit de faire remonter P dans les séquences jusqu'à $\frown_{\vdash}$, $\bot\vdash$ et $\vdash\top$. On ne change ainsi pas le squelette de la démonstration. □

Corollaire 2.4. *Si $\pi > \pi'$ et si ϖ est obtenu de π par affaiblissement, alors $\varpi > \pi'$. De même si $\pi' > \pi$ et si ϖ est obtenu de π par affaiblissement, alors $\pi' > \varpi$.*

Lemme 2.5 (Inversibilité ordonnée)**.** *Si on a une démonstration de la conclusion d'une instance d'une règle logique* r*, alors on a des démonstrations de ses prémisses qui sont plus petites ou égales pour l'ordre RPO.*

Démonstration. On procède par induction sur la démonstration de la conclusion. Si dernière règle appliquée est $\frown_{\vdash}$ il y a deux cas :

- si la formule active dans $\frown_{\vdash}$ est celle de l'instance de r, alors la démonstration d'un prémisse consiste à potentiellement appliquer la règle opposée puis $\frown_{\vdash}$. On détaille uniquement les cas les plus intéressants qui sont $\forall\vdash$ et $\vdash\forall$:

$$\frown_{\vdash} \frac{}{\Gamma, \forall x,\ P(x) \vdash \forall x,\ P(x), \Delta}$$

donne

$$\frown_{\vdash} \frac{}{\Gamma, P(t), \forall x,\ P(x) \vdash \forall x,\ P(x), \Delta}$$

pour $\forall\vdash$ et

$$\forall\vdash \frac{\frown_{\vdash} \dfrac{}{\Gamma, P(y), \forall x,\ P(x) \vdash P(y), \Delta}}{\Gamma, \forall x,\ P(x) \vdash P(y), \Delta}$$

pour $\vdash\forall$;

- sinon, on peut aussi utiliser $\frown_{\vdash}$ pour les démonstrations des prémisses. Par exemple, pour $\forall\vdash$

$$\frown_{\vdash} \frac{}{\Gamma, P, \forall x,\ Q(x) \vdash P, \Delta}$$

donne

$$\frown_{\vdash} \frac{}{\Gamma, P, Q(t), \forall x,\ Q(x) \vdash P, \Delta}$$

.

Si la dernière règle appliquée est $\vdash\!\supseteq$ ou une règle logique r' qui n'est pas l'instance de la règle considérée, on obtient le résultat par hypothèse d'induction sur les sous-démonstrations, puis on applique à nouveau $\vdash\!\supseteq$ ou r'. Par exemple, si r est $\exists\vdash$ et r' est $\vdash\forall$ alors

$$\vdash\forall\ \frac{\overset{\pi}{\Gamma, \exists x,\ P(x) \vdash Q(y), \Delta}}{\Gamma, \exists x,\ P(x) \vdash \forall x,\ Q(x), \Delta}$$

devient

$$\vdash\forall\ \frac{\overset{HI(\pi)}{\Gamma, P(z) \vdash Q(y), \Delta}}{\Gamma, P(z) \vdash \forall x,\ Q(x), \Delta}$$

où $HI(\pi)$ est la démonstration obtenue en appliquant l'hypothèse d'induction sur π.

Sinon, la dernière règle appliquée est une instance de r avec la même formule active. Si r est $\forall\vdash$ ou $\vdash\exists$, alors la démonstration voulue est un simple affaiblissement de la démonstration existante. Par exemple, si on a une démonstration

$$\vdash\exists\ \frac{\overset{\pi}{\Gamma \vdash P(t), \exists x,\ P(x), \Delta}}{\Gamma \vdash \exists x,\ P(x), \Delta}$$

alors on a la démonstration suivante de la prémisse d'une instance de $\vdash\exists$

$$\vdash\exists\ \frac{\overset{\varpi}{\Gamma \vdash P(s), P(t), \exists x,\ P(x), \Delta}}{\Gamma \vdash P(s), \exists x,\ P(x), \Delta}$$

Sinon, à α-conversion près, les sous-démonstrations directes sont bien des démonstrations des prémisses qui sont plus petites. □

Lemme 2.6 (Contraction ordonnée)**.** *Si on a une démonstration de* $\Gamma, P, P \vdash \Delta$ *alors on a une démonstration plus petite ou égale de* $\Gamma, P \vdash \Delta$*.*

De façon duale, si on a une démonstration de $\Gamma \vdash P, P, \Delta$ *alors on a une démonstration plus petite ou égale de* $\Gamma \vdash P, \Delta$*.*

Démonstration. On procède par induction sur la démonstration de $\Gamma, P, P \vdash \Delta$ avec l'ordre RPO. Si la dernière règle est $\subseteq\!\vdash$ alors on utilise au plus une occurence de P et on peut également appliquer $\subseteq\!\vdash$ pour démontrer $\Gamma, P \vdash \Delta$.

Si la formule active de la dernière règle r n'est pas P, alors on applique l'hypothèse d'induction sur les sous-démonstrations directes puis on applique r aux démonstrations obtenues.

Si la dernière règle r décompose un des P alors on utilise le lemme 2.5 sur les sous-démonstrations directes, on applique l'hypothèse d'induction (potentiellement plusieurs

fois) sur les démonstrations obtenues (qui sont bien plus petites pour l'ordre RPO) et on applique ensuite r pour obtenir le résultat attendu. Nous détaillons uniquement les cas où $P = Q \vee R$: Si on suppose que l'on a

$$\vee\vdash \frac{\overset{\pi_1}{\Gamma, A, A \vee B \vdash \Delta} \qquad \overset{\pi_2}{\Gamma, B, A \vee B \vdash \Delta}}{\Gamma, A \vee B, A \vee B \vdash \Delta}$$

alors par le lemme 2.5 on obtient des démonstrations $\pi_1^1 \leq \pi_1$ de $\Gamma, A, A \vdash \Delta$, $\pi_1^2 \leq \pi_1$ de $\Gamma, A, B \vdash \Delta$, $\pi_2^1 \leq \pi_2$ de $\Gamma, B, A \vdash \Delta$ et $\pi_2^2 \leq \pi_2$ de $\Gamma, B, B \vdash \Delta$. On applique l'hypothèse d'induction sur π_1^1 et π_2^2 pour obtenir des démonstrations ϖ_1^1 de $\Gamma, A \vdash \Delta$ et ϖ_2^2 de $\Gamma, B \vdash \Delta$ qui permettent de construire la démonstration

$$\vee\vdash \frac{\overset{\varpi_1^1}{\Gamma, A \vdash \Delta} \qquad \overset{\varpi_2^2}{\Gamma, B \vdash \Delta}}{\Gamma, A \vee B \vdash \Delta}$$

qui est bien plus petite ou égale à la démonstration originelle. Si on suppose que l'on a

$$\vdash\vee \frac{\overset{\pi}{\Gamma \vdash A, B, A \vee B, \Delta}}{\Gamma \vdash A \vee B, A \vee B\Delta}$$

alors par le lemme 2.5 on obtient une démonstration $\pi' \leq \pi$ de $\Gamma \vdash A, B, A, B, \Delta$. On applique l'hypothèse d'induction sur π' pour obtenir une démonstration $\varpi' \leq \pi'$ de $\Gamma \vdash A, B, B, \Delta$, sur laquelle on peut de nouveau appliquer l'hypothèse d'induction pour obtenir une démonstration $\varpi'' \leq \varpi'$ de $\Gamma \vdash A, B, \Delta$ qui permet de construire la démonstration

$$\vdash\vee \frac{\overset{\varpi''}{\Gamma \vdash A, B, \Delta}}{\Gamma \vdash A \vee B, \Delta}$$

qui est bien plus petite ou égale à la démonstration originelle.

□

La procédure d'élimination des coupures consiste à se ramener à des cas de base où la formule X autour de laquelle la coupure est faite est éliminée tout de suite dans les deux sous-démonstrations. Ces cas de bases sont alors transformés en coupures autour de sous-formules de X, sauf pour $\exists$ et $\forall$, auquel cas on fait une coupure autour de la même formule, mais avec des sous-démonstrations plus petites. Dans tous les cas on vérifie que l'ordre RPO sur les démonstrations diminue. Les différents cas sont les suivants : (Les ϖ sont obtenus à partir des π par affaiblissement.)

$$\Rightarrow\vdash \frac{\dfrac{\overset{\pi_1}{\Gamma \vdash X, \Delta} \qquad \overset{\pi_2}{\Gamma, Y \vdash \Delta}}{\Gamma, X \Rightarrow Y \vdash \Delta}\ \vdash_{\smile} \qquad \vdash\Rightarrow \dfrac{\overset{\pi'}{\Gamma, X \vdash Y, \Delta}}{\Gamma \vdash X \Rightarrow Y, \Delta}}{\Gamma \vdash \Delta} \rightsquigarrow$$

$$\dfrac{\dfrac{\pi_2}{\Gamma, Y \vdash \Delta} \quad \overset{\pi'}{\underset{\smile}{\vdash}} \dfrac{\Gamma, X \vdash Y, \Delta \qquad \overset{\varpi_1}{\Gamma \vdash X, Y, \Delta}}{\Gamma \vdash Y, \Delta}}{\Gamma \vdash \Delta}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma, X, Y \vdash \Delta}}{\Gamma, X \wedge Y \vdash \Delta} \wedge\vdash \quad \vdash\wedge \dfrac{\overset{\pi'_1}{\Gamma \vdash X, \Delta} \qquad \overset{\pi'_2}{\Gamma \vdash Y, \Delta}}{\Gamma \vdash X \wedge Y, \Delta}}{\Gamma \vdash \Delta} \rightsquigarrow$$

$$\dfrac{\dfrac{\pi}{\Gamma, X, Y \vdash \Delta} \quad \overset{\varpi'_2}{\underset{\smile}{\vdash}} \dfrac{\Gamma, X \vdash Y, \Delta}{\Gamma, X \vdash \Delta} \qquad \overset{\pi'_1}{\Gamma \vdash X, \Delta}}{\Gamma \vdash \Delta}$$

$$\dfrac{\dfrac{\overset{\pi_1}{\Gamma, X \vdash \Delta} \qquad \overset{\pi_2}{\Gamma, Y \vdash \Delta}}{\Gamma, X \vee Y \vdash \Delta} \vee\vdash \quad \vdash\vee \dfrac{\overset{\pi'}{\Gamma \vdash X, Y, \Delta}}{\Gamma \vdash X \vee Y, \Delta}}{\Gamma \vdash \Delta} \rightsquigarrow$$

$$\dfrac{\dfrac{\pi_1}{\Gamma, X \vdash \Delta} \quad \overset{\varpi_2}{\underset{\smile}{\vdash}} \dfrac{\Gamma, Y \vdash X, \Delta \qquad \overset{\pi'}{\Gamma \vdash X, Y, \Delta}}{\Gamma \vdash X, \Delta}}{\Gamma \vdash \Delta}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma, \forall x,\ X(x), X(t) \vdash \Delta}}{\Gamma, \forall x,\ X(x) \vdash \Delta} \forall\vdash \quad \vdash\forall \dfrac{\overset{\pi'}{\Gamma \vdash X(y), \Delta}}{\Gamma \vdash \forall x,\ X(x), \Delta}\ y \text{ non libre dans } \Gamma, X(x), \Delta}{\Gamma \vdash \Delta} \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma, \forall x,\ X(x), X(t) \vdash \Delta}}{\Gamma, X(t) \vdash \Delta} \quad \vdash\forall \dfrac{\overset{\{z/y\}\varpi'}{\Gamma, X(t) \vdash X(z), \Delta}}{\Gamma, X(t) \vdash \forall x,\ X(x), \Delta}\ z \text{ non libre dans } \Gamma, X(x), X(t), \Delta \qquad \overset{\{t/y\}\pi'}{\Gamma \vdash X(t), \Delta}}{\Gamma \vdash \Delta}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma, X(y) \vdash \Delta}\ y \text{ non libre dans } \Gamma, X(x), \Delta}{\Gamma, \exists x, X(x) \vdash \Delta} \exists\vdash \quad \vdash\exists \dfrac{\overset{\pi'}{\Gamma \vdash \exists x, X(x), X(t), \Delta}}{\Gamma \vdash \exists x, X(x), \Delta}}{\Gamma \vdash \Delta} \rightsquigarrow$$

$$\vdash_{\smile} \dfrac{\{t/y\}\pi \quad \vdash_{\smile}^{\exists} \dfrac{\{z/y\}\varpi \quad \dfrac{\Gamma, X(z) \vdash X(t), \Delta}{\Gamma, \exists x, X(x) \vdash X(t), \Delta}\ z \text{ non libre dans } \Gamma, X(x), X(t), \Delta \quad \pi' \quad \Gamma \vdash \exists x, X(x), X(t), \Delta}{\Gamma \vdash X(t), \Delta}}{\Gamma \vdash \Delta}$$

$$\vdash_{\smile}^{\bot} \dfrac{\dfrac{}{\Gamma, \bot \vdash \Delta} \quad \pi' \quad \Gamma \vdash \bot, \Delta}{\Gamma \vdash \Delta} \rightsquigarrow \quad \varpi' \quad \Gamma \vdash \Delta$$

$$\vdash_{\smile} \dfrac{\pi \quad \Gamma, \top \vdash \Delta \quad \vdash^{\top} \dfrac{}{\Gamma \vdash \top, \Delta}}{\Gamma \vdash \Delta} \rightsquigarrow \quad \varpi \quad \Gamma \vdash \Delta$$

$$\vdash_{\smile}^{\neg} \dfrac{\dfrac{\pi \quad \Gamma \vdash X, \Delta}{\Gamma, \neg X \vdash \Delta} \quad \vdash^{\neg} \dfrac{\pi' \quad \Gamma, X \vdash \Delta}{\Gamma \vdash \neg X, \Delta}}{\Gamma \vdash \Delta} \rightsquigarrow \quad \vdash_{\smile} \dfrac{\pi' \quad \Gamma, X \vdash \Delta \quad \pi \quad \Gamma \vdash X, \Delta}{\Gamma \vdash \Delta}$$

On a aussi les cas où la formule introduite par $\vdash_{\smile}$ est utilisée par $\vdash_{\frown}$ juste au dessus, auquel cas on utilise le lemme 2.6 :

$$\vdash_{\smile}^{\frown} \dfrac{\dfrac{}{\Gamma, X \vdash X, \Delta} \quad \pi' \quad \Gamma \vdash X, X, \Delta}{\Gamma \vdash X, \Delta} \rightsquigarrow \quad \varpi' \quad \Gamma \vdash X, \Delta$$

$$\vdash_{\smile} \dfrac{\pi \quad \Gamma, X, X \vdash \Delta \quad \vdash_{\frown} \dfrac{}{\Gamma, X \vdash X, \Delta}}{\Gamma, X \vdash \Delta} \rightsquigarrow \quad \varpi \quad \Gamma, X \vdash \Delta$$

Il reste ensuite à faire permuter $\vdash_{\smile}$ avec les autres règles qui ne concernent pas la formule autour de laquelle on a coupé. Le schéma général suivant

$$\vdash_{\smile} \dfrac{\mathsf{r} \dfrac{\pi_i \quad \Gamma, \Gamma_i, Q \vdash \Delta_i, \Delta}{\Gamma, P, Q \vdash \Delta} \quad \pi' \quad \Gamma, P \vdash Q, \Delta}{\Gamma, P \vdash \Delta}$$

est transformé en

$$\mathsf{r} \dfrac{\vdash_{\smile} \dfrac{\pi_i \quad \Gamma, \Gamma_i, Q \vdash \Delta_i, \Delta \quad \varpi' \quad \Gamma, \Gamma_i \vdash Q, \Delta_i, \Delta}{\Gamma, \Gamma_i \vdash \Delta_i, \Delta}}{\Gamma, P \vdash \Delta}$$

où $\varpi' \leq \pi'$ est obtenu grâce au lemme 2.5, et pareillement pour le cas où P est à droite de la séquence. C'est ici qu'intervient l'indéterminisme de la procédure, puisqu'on peut faire permuter soit la règle de la sous-démonstration gauche, soit celle de la droite. Pour faire décroître l'ordre sur les démonstrations, il faut que r soit différent de $\vdash_{\smile}$, ce qui

revient à appliquer la procédure d'élimination des coupures d'abord sur les coupures les plus en haut dans la démonstration. Si on permet aux coupures de permuter les une envers les autres, alors la procédure d'élimination des coupures peut ne pas terminer.

Dans le cas du calcul des séquences pour la logique classique, on vient de voir qu'il existe une procédure d'élimination des coupures, ce qui implique l'admissibilité de la règle de coupure, ce qui implique la cohérence de la logique. Nous verrons dans le chapitre suivant que ces trois propriétés sont différentes, puisqu'on peut trouver des systèmes qui sont cohérents sans admettre $\vdash\!\!\!\supset$, ou des systèmes qui admettent $\vdash\!\!\!\supset$ mais pour lesquels la procédure d'élimination des coupures ne termine pas.

Une procédure similaire peut-être donnée pour LB. Dans ce cas, on peut démontrer un lemme similaire au lemme 2.3, et on aura besoin du lemme suivant, qui dit essentiellement qu'une démonstration de LB peut être transformée en démonstration de LJ quand on n'a affaire qu'aux connecteurs $\Rightarrow$, $\wedge$, $\forall$, $\bot$ et $\top$.

Lemme 2.7. *Si une séquence $\Gamma \vdash \Delta$ ($\Delta \neq \emptyset$) ne contenant que les connecteurs $\Rightarrow$, $\wedge$, $\forall$, $\bot$ et $\top$ possède une démonstration π dans* LB*, alors il existe $C \in \Delta$ tel que $\Gamma \vdash C$ possède une démonstration plus petite ou égale à π.*

Ceci n'est bien entendu pas vrai pour LK ou G4, puisque sinon LJ serait complet pour la logique classique.

Démonstration. On procède par induction sur π, avec l'ordre RPO.

- Si la dernière règle est $\subset\!\!\!\vdash$, on choisit pour C la formule sur laquelle $\subset\!\!\!\vdash$ agit.
- Dans le cas d'une règle logique à gauche autre que $\Rightarrow\!\vdash$ et $\neg\!\vdash$, on ne modifie pas la séquence à droite on peut donc appliquer la règle logique à gauche à la démonstration obtenue par hypothèse d'induction.
- Dans le cas d'une règle logique à droite autre que $\vdash\!\Rightarrow$ et $\vdash\!\neg$, on ne modifie pas la gauche de la séquence. On applique l'hypothèse d'induction sur les sous-démonstrations directes. Si les formules retenues à droite sont toutes les sous-formules de la formule active, alors on peut à nouveau appliquer la règle logique (avec un affaiblissement dans le cas de $\vdash\!\exists$). Sinon, il existe une sous-démonstration qui de Γ prouve une formule de Δ, qui est bien ce dont on a besoin.
- Si la dernière règle est $\Rightarrow\!\vdash$ (pareillement pour $\neg\!\vdash$ et $\vdash\!\!\!\supset$), on a

$$\Rightarrow\!\vdash \frac{\overset{\pi_1}{\Gamma' \vdash A, \Delta} \qquad \overset{\pi_2}{\Gamma', B \vdash \Delta}}{\Gamma', A \Rightarrow B \vdash \Delta}$$

  Par hypothèse d'induction il existe un C dans A, Δ avec une démonstration ϖ_1 de $\Gamma' \vdash C$ plus petite que π_1. Si $C \in \Delta$, alors par affaiblissement on a une démonstration de $\Gamma', A \Rightarrow B \vdash C$ comme attendu. Sinon, par hypothèse d'induction il existe un C' dans Δ avec une démonstration ϖ_2 de $\Gamma', B \vdash C'$ plus petite que π_2. On peut alors affaiblir ϖ_1 pour obtenir une démonstration ϖ_1' de $\Gamma' \vdash A, C'$ puis utiliser $\Rightarrow\!\vdash$ pour obtenir

$$\Rightarrow\vdash \frac{\begin{array}{cc}\varpi_1' & \varpi_2\\ \Gamma' \vdash A, C' & \Gamma', B \vdash C'\end{array}}{\Gamma', A \Rightarrow B \vdash C'}$$

comme attendu.

– Si la dernière règle est $\vdash\Rightarrow$ (pareillement pour $\vdash\neg$), on a

$$\vdash\Rightarrow \frac{\begin{array}{c}\pi_1\\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \Rightarrow B, \Delta'}$$

on donc également une démonstration

$$\vdash\Rightarrow \frac{\begin{array}{c}\pi_1\\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \Rightarrow B}$$

comme attendu.

□

Il est alors facile de démontrer l'admissibilité de la contraction à droite :

Lemme 2.8. *Si* $\Gamma \vdash P, P, \Delta$ *est démontrable dans* LB*, alors* $\Gamma \vdash P, \Delta$ *l'est aussi avec une démonstration plus petite ou égale.*

Démonstration. En utilisant le lemme précédent, on a une démonstration de $\Gamma \vdash C$ pour $C \in P, P, \Delta$ donc par affaiblissement une démonstration de $\Gamma \vdash P, \Delta$. □

La procédure d'élimination des coupures est alors la suivante : Les cas de base sont les mêmes, à affaiblissement près pour $\Rightarrow$ et $\forall$; Les cas avec $\frown$ sont les mêmes grâce au lemme 2.8 ; Les permutations ne fonctionnent plus exactement pareil, du fait de la non-inversibilité de certaines règles, mais on peut s'en sortir en utilisant le lemme 2.7.

2.3.5 Traductions entre calcul des séquences et déduction naturelle

Gerhard Gentzen (1934) a montré que les démonstrations de la déduction naturelle peuvent être transformées en démonstration du calcul des séquences LJ, et réciproquement. Ici, nous essaierons, quand ce sera possible, d'éviter que les traductions que nous donnons engendrent des détours supplémentaires, dans le sens où dans NJ un détour est l'application d'une règle d'introduction suivie immédiatement par une règle d'élimination ; dans LJ un détour est l'application d'une coupure $\smile$.

Les traductions sont basées sur l'idée que les règles droites sont les mêmes que les règles d'introduction, et les règles gauches sont des règles d'élimination vue de haut en bas. On donne les traductions entre NJ et LJ, mais grâce au lemme 2.7 on a des traductions similaires entre NJ et LB.

Proposition 2.9 (Traduction de NJ dans LJ)**.** *Une démonstration de* C *sous les hypothèses* Γ *en déduction naturelle peut être traduite en une démonstration de conséquence* $\Gamma \vdash C$ *en* LJ*.*

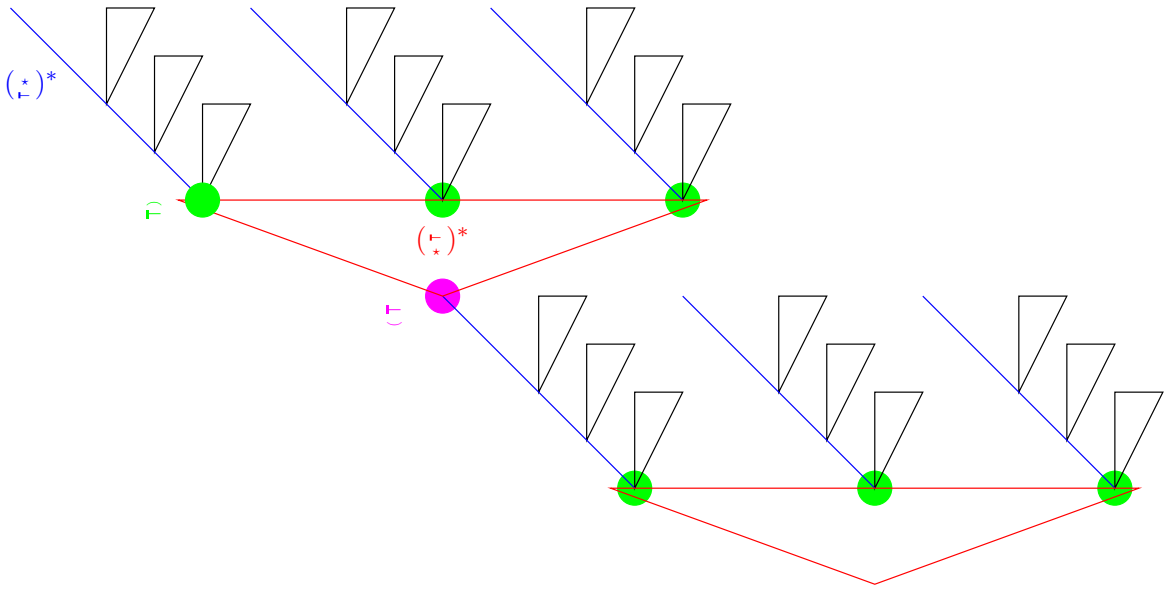


FIG. 2.7 : Schéma illustratif d'une démonstration en déduction naturelle, et indication de la traduction vers le calcul des séquences

Démonstration. On raisonne par induction sur la taille de la démonstration. Comme on peut le voir sur la figure 2.3 et la figure 2.4 dans laquelle on considère des séquences à une seule conclusion, les règles d'introduction $\vdash_\star$ et les règles droites $\vdash\star$ se correspondent parfaitement. On peut donc les traduire directement. Une démonstration en déduction naturelle à la forme suivante, du bas vers le haut : une série (potentiellement nulle) d'introduction, une suite d'élimination en suivant leurs prémisses principales, puis une série d'introduction, etc. (Cf. figure 2.7.) On traduit chaque bloc introductions–éliminations puis on les relie les uns aux autres à l'aide de coupures. Pour traduire un bloc, on commence par traduire les introductions par des règles droites en partant du bas, puis, quand il n'y a plus que des éliminations, on traduit l'élimination en haut à gauche par une règle gauche. Les cas de $\Rightarrow_\vdash$, $\wedge_\vdash$, $\forall_\vdash$ et $\neg_\vdash$ ne posent pas trop de problème :

– Si on a

$$\Rightarrow_\vdash \frac{\begin{array}{cc} & \pi \\ P \Rightarrow Q & P \end{array}}{\begin{array}{c} Q \\ \pi' \end{array}}$$

par hypothèse d'induction sur π on a une démonstration de $\Theta \vdash P$ et sur π' on a une démonstration de $\Delta, Q \vdash C$. En appliquant $\Rightarrow\vdash$ on a donc une démonstration de $\Theta, \Delta, P \Rightarrow Q \vdash C$ à laquelle on peut appliquer des contractions et des affaiblissements pour obtenir la démonstration attendue.

– Si on a

$$\overset{\wedge 1}{\vdash}\ \frac{P \wedge Q}{P}\ \pi'$$

par hypothèse d'induction sur π' on a une démonstration de $\Theta, P \vdash C$, donc par affaiblissement de $\Theta, P, Q \vdash C$. En appliquant $\wedge\vdash$ on obtient donc la démonstration attendue de $\Gamma', P \wedge Q \vdash C$ pour $\Gamma = \Gamma', P \wedge Q$.

– Si on a

$$\overset{\forall}{\vdash}\ \frac{\forall x,\ P}{\{t/x\}P}\ \pi'$$

par hypothèse d'induction sur π' on a une démonstration de $\Delta, \{t/x\}P \vdash C$. En appliquant $\forall\vdash$ on obtient donc la démonstration attendue de $\Gamma', \forall x,\ P \vdash C$ pour $\Gamma = \Gamma', \forall x,\ P$.

– Si on a

$$\overset{\neg}{\vdash}\ \frac{\neg P \qquad \overset{\pi}{P}}{Q}\ \pi'$$

par hypothèse d'induction sur π on a une démonstration de $\Theta \vdash P$. En appliquant $\neg\vdash$ et potentiellement quelques contractions et quelques affaiblissements, on obtient donc la démonstration attendue de $\Gamma', \neg P \vdash C$ pour $\Gamma = \Gamma', \neg P$.

Les autres cas sont plus problématiques car ils contiennent des coupures implicites. Dans ce cas, on n'aura plus la correspondance entre détours dans NJ et dans LJ.

– Si on a

$$\overset{\vee}{\vdash}\ \frac{P \vee Q \qquad \begin{matrix}[P]\\ \vdots\ \pi_1\\ D\end{matrix} \qquad \begin{matrix}[Q]\\ \vdots\ \pi_2\\ D\end{matrix}}{D}\ \pi'$$

par hypothèse d'induction sur π_1 et π_2 on a des démonstrations de $\Gamma_1, P \vdash D$ et $\Gamma_2, Q \vdash D$, donc en appliquant $\vee\vdash$ on a une démonstration de $\Gamma_1, \Gamma_2, P \vee Q \vdash D$. Par hypothèse d'induction sur π' on a des démonstrations de $\Delta, D \vdash C$, et donc en appliquant $\overset{\vdash}{\smile}$ et éventuellement des affaiblissements et des contractions on obtient la démonstration attendue de $\Gamma', P \vee Q \vdash C$ pour $\Gamma = \Gamma', P \vee Q$.

– Si on a

$$\overset{\exists}{\vdash}\ \frac{\exists x,\ P \qquad \begin{matrix}[\{t/x\}P]\\ \vdots\ \pi\\ Q\end{matrix}}{Q}\ \pi'$$

par hypothèse d'induction sur π on a une démonstration de $\Gamma_1, \{t/x\}P \vdash Q$, donc en affaiblissant et en appliquant $\exists\vdash$ on a une démonstration de $\Gamma_1, \exists x,\ P \vdash Q$. Par hypothèse d'induction sur π' on a des démonstrations de $\Delta, Q \vdash C$, et donc en appliquant $\overset{\vdash}{\smile}$ et éventuellement des affaiblissements et des contractions on obtient la démonstration attendue de $\Gamma', \exists x,\ P \vdash C$ pour $\Gamma = \Gamma', \exists x,\ P$.

– si on a

$$\overset{\bot}{\vdash}\ \frac{\bot}{P}\ \pi'$$

par hypothèse d'induction sur π' on a une démonstration de $\Theta, P \vdash C$. En appliquant $\bot\vdash$ on a une démonstration de $\bot \vdash P$, donc en appliquant $\overset{\vdash}{\smile}$ on en a une de $\Gamma', \bot \vdash C$ pour $\Gamma = \Gamma', \bot$.

Pour relier les blocs introductions–éliminations, on utilise des coupures : supposons que P est la formule apparaissant à la jointure entre deux blocs (de haut en bas, P est donc introduite puis éliminée), par hypothèse d'induction sur la démonstration au dessus on obtient une démonstration de $\Theta \vdash P$, et sur la démonstration en dessous une démonstration de $\Delta, P \vdash C$. En appliquant $\overset{\vdash}{\smile}$, et éventuellement quelques contractions et affaiblissement, on obtient une démonstration de $\Gamma \vdash C$. □

Cette traduction n'est pas la seule possible : on aurait pu choisir de traduire d'abord les éliminations en haut de chaque bloc avant les introductions, ou même de mélanger les deux. En règle générale, il existe plusieurs démonstrations de LJ correspondant à une démonstration de NJ. Ceci provient du fait que dans LJ les décompositions des formules à gauche et de la formule à droite peuvent être entremêlées les unes dans les autres, tandis que dans NJ elles se font en parallèle. L'utilisation de la focalisation pour limiter ces entremêlements, en particulier avec le calcul des séquences LJT de Hugo Herbelin (1995), permet d'obtenir une correspondance exacte.

Proposition 2.10 (Traduction de LJ dans NJ)**.** *Une démonstration de* $\Gamma \vdash C$ *dans* LJ *peut être traduite en une démonstration de* C *avec comme hypothèses un sous-ensemble de* Γ *en déduction naturelle.*

Démonstration. Ici aussi on raisonne par induction sur la taille de la démonstration. Les règles droites sont naturellement traduites en règles d'introduction. Pour les règles gauches, on utilise des règles d'élimination :

– Si on a

$$\Rightarrow\vdash\ \frac{\overset{\pi}{\Gamma' \vdash P} \qquad \overset{\pi'}{\Gamma', Q \vdash C}}{\Gamma', P \Rightarrow Q \vdash C}$$

par hypothèse d'induction on a des démonstrations ϖ de P sous les hypothèses Γ' et ϖ' de C sous les hypothèses Γ', Q. Il suffit de remplacer dans celle-ci les hypothèses Q par la dérivation

$$\overset{\Rightarrow}{\vdash} \frac{P \Rightarrow Q \qquad \overset{\varpi}{P}}{Q}$$

dans ϖ' pour obtenir une démonstration de C sous les hypothèses $\Gamma', P \Rightarrow Q$.

– Si on a

$$\wedge\vdash \frac{\overset{\pi}{\Gamma', P, Q \vdash C}}{\Gamma', P \wedge Q \vdash C}$$

par hypothèse d'induction on a une démonstration de C sous les hypothèses P, Q et Γ'. Il suffit d'y remplacer les hypothèses P par la dérivation

$$\overset{\wedge 1}{\vdash} \frac{P \wedge Q}{P}$$

et pareillement pour Q pour obtenir une démonstration de C sous les hypothèses $\Gamma', P \wedge Q$.

– Si on a

$$\wedge\vdash \frac{\overset{\pi}{\Gamma', \{t/x\}P \vdash C}}{\Gamma', \forall x,\ P \vdash C}$$

par hypothèse d'induction on a une démonstration de C sous les hypothèses $\{t/x\}P$ et Γ'. Il suffit d'y remplacer les hypothèses P par la dérivation

$$\overset{\forall}{\vdash} \frac{\forall x,\ P}{\{t/x\}P}$$

pour obtenir une démonstration de C sous les hypothèses $\Gamma', \forall x,\ P$.

– Si on a

$$\vee\vdash \frac{\overset{\pi}{\Gamma', P \vdash C} \qquad \overset{\pi'}{\Gamma', P \vdash C}}{\Gamma', P \vee Q \vdash C}$$

par hypothèse d'induction on a une démonstration ϖ de C sous les hypothèses P et Γ' et une démonstration ϖ' de C sous les hypothèses Q et Γ'. On a alors la démonstration

$$\overset{\vee}{\vdash} \frac{\begin{array}{ccc} & [P] & [Q] \\ & \vdots\, \varpi & \vdots\, \varpi' \\ P \vee Q & C & C \end{array}}{C}$$

de C sous les hypothèses $\Gamma', P \vee Q$.

– Si on a

$$\exists\vdash \frac{\overset{\pi}{\Gamma', \{y/x\}P \vdash C}}{\Gamma', \exists x,\ P \vdash C}\ y \text{ non libre dans } \Gamma', P, C$$

par hypothèse d'induction on a une démonstration ϖ de C sous les hypothèses $\{y/x\}P$. On a alors la démonstration

$$\vdash^{\exists} \frac{\exists x,\ P \qquad \begin{array}{c}[\{y/x\}P]\\ \vdots\,\varpi\\ C\end{array}}{C}$$

de C sous les hypothèses $\Gamma', \exists x,\ P$.

– Si on a

$$\bot\vdash \frac{}{\bot \vdash C}$$

on a une démonstration

$$\vdash^{\bot} \frac{\bot}{C}$$

de C sous l'hypothèse $\bot$.

– Si on a

$$\wedge\vdash \frac{\begin{array}{c}\pi\\ \Gamma' \vdash P\end{array}}{\Gamma', \neg P \vdash C}$$

par hypothèse d'induction on a une démonstration ϖ de P sous les hypothèses Γ'. On a alors la démonstration

$$\vdash^{\neg} \frac{\neg P \qquad \begin{array}{c}\varpi\\ P\end{array}}{C}$$

de C sous les hypothèses $\Gamma', \neg P$.

Pour traduire la règle de coupure, si on a

$$\vdash_{\subset} \frac{\begin{array}{c}\pi\\ \Gamma, P \vdash C\end{array} \qquad \begin{array}{c}\pi'\\ \Gamma \vdash P\end{array}}{\Gamma \vdash C}$$

alors par hypothèse d'induction on a des démonstrations ϖ de C sous les hypothèses Γ, P et ϖ' de C sous les hypothèses Γ. On a alors la démonstration

$$\begin{array}{l}\vdash\\ \vdash\end{array}^{\Rightarrow} \frac{\dfrac{\begin{array}{c}[P]\\ \vdots\,\varpi\\ C\end{array}}{P \Rightarrow C} \qquad \begin{array}{c}\varpi'\\ P\end{array}}{C}$$

de C sous les hypothèses Γ. □

On remarque qu'ici aussi les connecteurs $\vee, \bot, \exists$ joue un rôle différents. On voit également que ces deux traductions ne sont pas inverse l'une de l'autre.

Nous étendrons ces traductions à différents systèmes que nous allons introduire par la suite.

Si l'on veut ajouter du calcul dans les démonstrations, une des premières méthodes est d'utiliser une axiomatisation de ce calcul. Par exemple, dans la présentation usuelle de l'arithmétique de Peano (cf. section 5.3.1), l'addition est définie par les deux axiomes $\forall x,\ x + \mathsf{O} = x$ et $\forall x,\ \forall y,\ x + s(y) = s(x + y)$. Toutefois, cette méthode n'est pas satisfaisante pour la recherche de démonstration mécanisée, d'une part parce qu'au lieu d'utiliser directement l'ordinateur pour effectuer le calcul, on le fait simuler à l'aide d'application de règles logiques, et d'autre part parce qu'il est difficile de déterminer quel axiome il faut appliquer à un moment de la démonstration. Illustrons ces deux points par l'exemple suivant :

Exemple 2.1 *:* On considère la théorie présentée par l'axiome de réflexivité (2.23) et le schéma d'axiome de Leibniz (2.24), ainsi que les axiomes définissant l'addition donnés ci-dessus. On note $\underline{1}$ pour $s(\mathsf{O})$ et $\underline{2}$ pour $s(s(\mathsf{O}))$.

Pour démontrer $\underline{1} + \underline{1} = \underline{2}$, on a alors la démonstration suivante (On note Γ la présentation de la théorie utilisée, ou plutôt uniquement les instances des schémas d'axiomes que nous utilisons. Pour le schéma d'axiomes de Leibniz on ne considérera donc que l'instance $\forall x,\ \forall y,\ x = y \Rightarrow \underline{1} + \underline{1} = s(x) \Rightarrow \underline{1} + \underline{1} = s(y)$.) :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \underline{1}+\underline{1} = s(\underline{1}+\mathsf{O}) \vdash \underline{1}+\underline{1} = s(\underline{1}+\mathsf{O}), \underline{1}+\underline{1} = \underline{2}}\,\vdash}{\forall \vdash}}{\Gamma \vdash \underline{1}+\underline{1} = s(\underline{1}+\mathsf{O}), \underline{1}+\underline{1} = \underline{2}}\,\vdash \quad \dfrac{}{\Gamma, \underline{1}+\underline{1} = \underline{2} \vdash \underline{1}+\underline{1} = \underline{2}}\,\vdash}{\Gamma, \underline{1}+\underline{1} = s(\underline{1}+\mathsf{O}) \Rightarrow \underline{1}+\underline{1} = \underline{2} \vdash \underline{1}+\underline{1} = \underline{2}}\,\Rightarrow\vdash}{\Gamma, \underline{1}+\mathsf{O} = \underline{1} \vdash \underline{1}+\mathsf{O} = \underline{1}, \underline{1}+\underline{1} = \underline{2}}\,\vdash}{\forall \vdash}}{\Gamma \vdash \underline{1}+\mathsf{O} = \underline{1}, \underline{1}+\underline{1} = \underline{2}}\,\Rightarrow\vdash \quad \vdots}{\Gamma, \underline{1}+\mathsf{O} = \underline{1} \Rightarrow \underline{1}+\underline{1} = s(\underline{1}+\mathsf{O}) \Rightarrow \underline{1}+\underline{1} = \underline{2} \vdash \underline{1}+\underline{1} = \underline{2}}\,\forall \vdash}{\Gamma \vdash \underline{1}+\underline{1} = \underline{2}}$$

On voit qu'on a une démonstration bien compliquée pour un théorème si simple. En particulier, le choix de l'instance du schéma de Leibniz est difficile à automatiser, de même que les moments où il faut instancier les axiomes définissant l'addition. Dans le prochain chapitre, nous allons donc nous intéresser à des systèmes qui permettent de mieux intégrer le calcul dans les démonstrations.

Chapitre 3

Démonstration et calcul

> Civilization advances by extending the numbers of important operations which we can perform without thinking about them. Operations of thought are like cavalry charges in battle — they are strictly limited in number, they require fresh horses, and must only be made at decisive moments.
>
> Alfred Whitehead, *An Introduction to Mathematics*

Nous allons nous intéresser aux relations entre calculs et démonstrations. Parmi les logiciens aujourd'hui, ce lien est quelque peu biaisé par la correspondance de Curry-Howard, qui identifie démonstrations et programmes (λ-termes typés pour être plus précis). Néanmoins, ce qui nous intéresse est l'intégration du calcul dans les démonstrations. La premier façon d'y parvenir est d'axiomatiser le calcul. Nous avons vu à la fin du chapitre précédant que si cela est satisfaisant d'un point de vue théorique, cela entraîne en pratique des démonstrations dans lesquelles les étapes concernant le déroulement des calculs ne sont ni naturelles ni efficaces. Une autre manière est d'étendre l'isomorphisme de Curry-Howard en ne séparant plus le niveau des démonstrations (c'est-à-dire des programmes) de celui des formules (c'est-à-dire des types). On obtient alors les types dépendants et les systèmes de types purs. On peut aussi incorporer directement le calcul dans les démonstrations en appliquant les règles d'inférence des systèmes de déduction modulo une congruence représentant le calcul. On obtient alors la déduction modulo.

3.1 Identifier démonstration et calcul

3.1.1 Interprétation de Heyting-Brouwer-Kolmogorov

L'isomorphisme de Curry-Howard, qui permet d'identifier deux types d'objets a priori différents, à savoir les démonstrations et les programmes, peut avoir un côté magique pour le néophyte. Néanmoins, on peut trouver une explication assez satisfaisante

si on part de l'interprétation de HEYTING-BROUWER-KOLMOGOROV. Plus que d'une interprétation, il s'agit d'un postulat qui énonce quelles sont les propriétés que doit vérifier une démonstration pour pouvoir être qualifiée de constructive. Rappelons que Luitzen BROUWER et Arend HEYTING s'étaient proposés de poser les bases des mathématiques constructives par l'utilisation d'une logique dite alors intuitionniste. L'idée, développée également par Andreï KOLMOGOROV de façon indépendante, est que les démonstrations doivent pouvoir être interprétées comme des opérations sur des ensembles, à savoir :

- une démonstration de $A \wedge B$ doit être la paire d'une démonstration de A et d'une démonstration de B ;
- une démonstration de $A \vee B$ doit être une paire $(1, p)$ où p et une démonstration de A ou une paire $(2, q)$ où q et une démonstration de B ;
- une démonstration de $A \Rightarrow B$ doit être une fonction qui transforme les démonstrations de A en démonstrations de B ;
- une démonstration de $\forall x,\ A(x)$ doit être une fonction qui à tout objet o associe une démonstration de $A(o)$;
- une démonstration de $\exists x,\ A(x)$ doit être la paire d'un objet o et d'une démonstration de $A(o)$.

On peut d'ores et déjà remarquer que ces conditions ont un lien avec les règles d'introduction de la déduction naturelle. En particulier, on peut considérer $\overset{\vdash}{\Rightarrow}$

$$\overset{\vdash}{\Rightarrow} \frac{\begin{array}{c}[X] \\ \vdots \\ Y\end{array}}{X \Rightarrow Y}$$

comme une fonction qui prend une démonstration de X et qui donne une démonstration de Y en collant la démonstration de X à la place de l'hypothèse déchargée.

Si on veut avoir des démonstrations constructives, on va imposer que les fonctions considérées soient calculables. Or, un modèle des fonctions calculables assez naturel est le λ-calcul. Pousser plus avant la comparaison entre démonstrations et λ-termes va amener à l'isomorphisme de CURRY-HOWARD.

3.1.2 Isomorphisme de Curry-Howard

Sa forme la plus simple consiste à dire qu'il existe un isomorphisme entre l'ensemble des démonstrations en déduction naturelle et l'ensemble des λ-termes simplement typés, avec correspondance entre les formules démontrées et les types. Toutefois, cette isomorphisme a été découvert par Haskell CURRY d'abord entre un système schématique et un algèbre de combinateurs.

Nous allons tout d'abord nous restreindre à la logique minimale intuitionniste, où le seul connecteur est $\Rightarrow$. On peut reconnaître la similitude entre les règles d'inférence $\overset{\frown}{\vdash}$, $\overset{\vdash}{\Rightarrow}$ et $\overset{\Rightarrow}{\vdash}$ et les règles de typage Variable, Abstraction et Application. On peut donc associer

de façon bijective à chaque démonstration d'une formule P un λ-terme dont le type est P.

Un des aspects intéressants de cet isomorphisme est de voir comment la β-réduction se traduit au niveau des démonstrations. Rappelons qu'une étape de β-réduction se fait par

$$(\lambda x,\ t)\ u \underset{\beta}{\longrightarrow} \{u/x\}t\ .$$

Si on suppose que $(\lambda x,\ t)\ u$ est simplement typé dans un contexte Γ, cela veut dire qu'il existe une dérivation de typage de la forme

$$\dfrac{\dfrac{\vdots}{\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x,\ t : A \Rightarrow B}\ \mathsf{Abstraction} \qquad \vdots}{\Gamma \vdash u : A}\ \mathsf{Application}}{\Gamma \vdash (\lambda x,\ t)\ u : B}$$

qui correspond par l'isomorphisme à une démonstration

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \vdots t\\ B\end{array}}{A \Rightarrow B} \qquad \begin{array}{c}u\\ A\end{array}}{B}\ \begin{array}{l}\vdash\\ \Rightarrow\\ \vdash\end{array}$$

Le terme β-réduit possède alors le type B et la démonstration correspondant à son typage est alors

$$\begin{array}{c}u\\ A\\ \vdots t\\ B\end{array}$$

c'est-à-dire la démonstration de B dans laquelle on a remplacées toutes les hypothèses A par sa démonstration.

Si on en revient à l'interprétation de Heyting-Brouwer-Kolmogorov, $\lambda x,\ t$ est interprétée comme une fonction f qui transforme des démonstrations de A en démonstration de B. $\begin{array}{l}\vdash\\ \Rightarrow\\ \vdash\end{array}$ applique syntaxiquement cette fonction à une démonstration u de A. La β-réduction permet donc de réaliser cette application syntaxique pour retourner la démonstration résultant de l'application de la fonction f à u.

On remarque que la β-réduction définit un processus de normalisation des démonstrations en déduction naturelle qui élimine les détours. En effet, plutôt que de démontrer B en supposant A, puis de démontrer A avant de conclure par le *modus ponens*, la démonstration β-réduite montre directement B en intégrant la démonstration de A aux endroits où A était utilisée comme hypothèse. En ce sens, on voit qu'on se rapproche

de d'un processus d'élimination de la règle de coupure $\vdash_{\smile}$ du calcul des séquences. Ce n'est pas qu'une simple intuition, comme le met en avant par exemple Hugo HERBELIN (1995). On peut démontrer que ce processus de normalisation termine quelle que soit la démonstration de la déduction naturelle pour $\Rightarrow$, pour toute stratégie de réduction. Par l'isomorphisme de CURRY-HOWARD, cela revient à dire que tout λ-terme qui possède un type simple ne peut pas être infiniment β-réduit. On dit alors que la déduction naturelle (ou de façon équivalente le λ-calcul simplement typé) est *fortement normalisante*.

On peut étendre ceci à la logique intuitionniste du premier ordre, mais il faut pour cela étendre les λ-termes avec la grammaire suivante :

$$\begin{aligned} t ::= \quad & x \mid \lambda x,\ t \mid t\ t \mid (t,t) \mid \pi_1 t \mid \pi_2 t \mid \iota_1 t \mid \iota_2 t \mid \delta(x,\ t)(x,\ t)t \\ & \mid \Lambda a,\ t \mid t@b \mid \langle b, t\rangle \mid \Delta(x,\ a,\ t)t \mid 1 \mid \emptyset t \end{aligned}$$

avec a variable du premier ordre et b terme du premier ordre. Ceci permet construire des termes correspondant respectivement aux règles $\overset{\frown}{\vdash}$, $\underset{\Rightarrow}{\vdash}$, $\overset{\Rightarrow}{\vdash}$, $\underset{\wedge}{\vdash}$, $\overset{\wedge 1}{\vdash}$, $\overset{\wedge 2}{\vdash}$, $\underset{\vee 1}{\vdash}$, $\underset{\vee 2}{\vdash}$, $\overset{\vee}{\vdash}$, $\underset{\forall}{\vdash}$, $\overset{\forall}{\vdash}$, $\underset{\exists}{\vdash}$, $\overset{\exists}{\vdash}$, $\underset{\top}{\vdash}$, $\overset{\bot}{\vdash}$.

On peut alors considérer les réductions supplémentaires suivantes :

$$\begin{aligned} \pi_1(t_1, t_2) &\to t_1 \\ \pi_2(t_1, t_2) &\to t_2 \\ \delta(x,\ t_1)(y,\ t_2)(\iota_1 t) &\to \{t/x\}t_1 \\ \delta(x,\ t_1)(y,\ t_2)(\iota_2 t) &\to \{t/y\}t_2 \\ (\Lambda a,\ t)@s &\to \{s/a\}t \\ \Delta(x,\ a,\ t_1)\langle s, t_2\rangle &\to \{t_2/x\}\{s/a\}t_1 \end{aligned}$$

qui éliminent les détours constitués d'une introduction immédiatement suivie d'une élimination, et les réductions :

$$\begin{aligned} (\delta(x,\ t_1)(y,\ t_2)t)u &\to \delta(x,\ (t_1 u))(y,\ (t_2 u))t \\ \pi_i(\delta(x,\ t_1)(y,\ t_2)t) &\to \delta(x,\ (\pi_i t_1))(y,\ (\pi_i t_2))t \\ \delta(u,\ t_1)(v,\ t_2)(\delta(x,\ t_3)(y,\ t_4)t) &\to \delta(x,\ \delta(u,\ t_1)(v,\ t_2)t_3)(y,\ \delta(u,\ t_1)(v,\ t_2)t_4)t \\ (\delta(x,\ t_1)(y,\ t_2)t)@s &\to \delta(x,\ (t_1@s))(y,\ (t_2@s))t \\ \Delta(z,\ a,\ w)(\delta(x,\ t_1)(y,\ t_2)t) &\to \delta(x,\ \Delta(z,\ a,\ w)t_1)(y,\ \Delta(z,\ a,\ w)t_2)t \\ \emptyset(\delta(x,\ t_1)(y,\ t_2)t) &\to \delta(x,\ \emptyset t_1)(y,\ \emptyset t_2)t \\ (\Delta(x,\ a,\ t_1)t_2)u &\to \Delta(x,\ a,\ (t_1 u))t_2 \\ \pi_i(\Delta(x,\ a,\ t_1)t_2) &\to \Delta(x,\ a,\ (\pi_i t_1))t_2 \\ \delta(x,\ t_1)(y,\ t_2)(\Delta(x,\ a,\ t_3)t_4) &\to \Delta(x,\ a,\ (\delta(x,\ t_1)(y,\ t_2)t_3))t_4 \\ (\Delta(x,\ a,\ t_1)t_2)@s &\to \Delta(x,\ a,\ (t_1@s))t_2 \end{aligned}$$

$$\begin{aligned}
\Delta(z,\ a,\ w)(\Delta(x,\ a,\ t_1)t_2) &\to \Delta(x,\ a,\ \Delta(z,\ a,\ w)t_1)t_2 \\
\emptyset(\Delta(x,\ a,\ t_1)t_2) &\to \Delta(x,\ a,\ \emptyset t_1)t_2 \\
(\emptyset t)u &\to \emptyset t \\
\pi_i(\emptyset t) &\to \emptyset t \\
\delta(x,\ t_1)(y,\ t_2)(\emptyset t) &\to \emptyset t \\
(\emptyset t)@s &\to \emptyset t \\
\Delta(x,\ a,\ w)(\emptyset t) &\to \emptyset t \\
\emptyset(\emptyset t) &\to \emptyset t
\end{aligned}$$

qui permettent, via l'isomorphisme, de faire permuter les règles d'élimination $\overset{\vee}{\vdash}$, $\overset{\exists}{\vdash}$ et $\overset{\perp}{\vdash}$ avec les autres règles d'élimination. Ces dernières réductions sont appelées *coupures commutatives*.

Remarque 3.1 *:* Cet isomorphisme a été étendu à d'autres logiques et d'autres formes de calcul, pour lesquels on n'a pas forcément une bijection exacte. C'est la raison pour laquelle on trouve parfois dans la littérature l'appellation de « correspondance de Curry-Howard ». Notons également qu'on associe parfois à cet isomorphisme d'autres noms qui ont contribué à sa découverte, comme par exemple celui de Nicolas de Bruijn.

3.1.3 Extension : les systèmes de type purs

L'isomorphisme de Curry-Howard introduit donc un lien fort entre démonstration et calcul. Toutefois il n'est pas complètement satisfaisant, puisque nous aimerions être capable de raisonner sur le calcul, ce qui revient par l'isomorphisme à raisonner sur les démonstrations. C'est dans cette optique qu'ont été introduits les types dépendants. On considère alors une extension du λ-calcul dans laquelle on ne fait plus de distinction entre le niveau des types et le niveau de termes. On adopte donc la syntaxe suivante, dans le style de Church :

$$t ::= x \mid \lambda x : t,\ t \mid t\ t \mid \Pi x : t,\ t$$

dans laquelle $\Pi x : A,\ B$ représente un *produit dépendant* et peut être vu informellement comme l'espace des fonctions qui à un élément a de A associe un élément de $\{a/x\}B$. Le type $A \Rightarrow B$ devient alors une abréviation de $\Pi x : A,\ B$ où x n'est pas libre dans B.

Il faut étendre le système de type pour prendre en compte la nouvelle construction. La première idée est de définir une règle permettant la formation du produit dépendant. Pour cela, il nous faut définir les termes qui pourront intervenir dans ce produit, ceux que l'on voudrait faire correspondre à un type. On introduit pour cela un terme spécial $*$ qui sera informellement le type des types. On a alors la règle

$$\textsf{Produit}\ \frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A,\ B : *}$$

La règle Application est légèrement modifiée en

$$\textsf{Application}\ \frac{\Gamma \vdash T : \Pi x : A,\ B \qquad \Gamma \vdash U : A}{\Gamma \vdash (T\ U) : \{U/x\}B}$$

pour prendre en compte le fait que B dépend maintenant de x.

Pour la règle Abstraction on vérifie également que le produit dépendant est bien formé :

$$\textsf{Abstraction}\ \frac{\Gamma \vdash \Pi x : A,\ B : * \qquad \Gamma, x : A \vdash T : B}{\Gamma \vdash \lambda x : A,\ T : \Pi x : A,\ B}$$

Les contextes jouent ici un rôle primordial, car ils vont permettre entre autre de pouvoir déclarer les types de bases. Les types dans les contextes peuvent dépendre de variables définies auparavant dans le contexte. Ceci implique que les contextes ne sont pas commutatifs, et sont des listes ordonnées plutôt que des multiensembles. On doit alors éviter les cas problématiques où par exemple la variable associée à un type est libre dans ce type, d'où des règles de bonne formation du contexte :

$$\textsf{Vide}\ \frac{}{[]\ \text{bien formé}} \qquad \textsf{Déclaration}\ \frac{\Gamma\ \text{bien formé} \qquad \Gamma \vdash A : *}{\Gamma, x : A\ \text{bien formé}}\ x \text{ non libre dans } \Gamma, A$$

$$\textsf{Déclaration'}\ \frac{\Gamma\ \text{bien formé}}{\Gamma, A : *\ \text{bien formé}}\ A \text{ non libre dans } \Gamma$$

Il faut alors modifier la règle Variable pour qu'elle n'utilise que des contextes bien formés :

$$\textsf{Variable}\ \frac{\Gamma\ \text{bien formé}}{\Gamma \vdash x : A}\ x : A \in \Gamma$$

Si on se contente de ces règles, le système obtenu n'est pas plus puissant que celui des types simples. En effet, on peut montrer que les produits ne sont pas vraiment dépendant : si $\Gamma \vdash A : *$ est dérivable, alors aucune variable libre de A dont la type est typé par $*$ n'apparaît dans le contexte Γ. Par conséquent, si $\Gamma \vdash \Pi x : A,\ B : T$ ou $\Gamma \vdash T : \Pi x : A,\ B$ est dérivable alors x n'est pas une variable libre de B. On note alors ce système $\lambda_{\longrightarrow}$.

On aimerait étendre le système, en particulier pour essayer d'associer le produit dépendant avec la quantification universelle. Les formules seront liées par l'isomorphisme de Curry-Howard aux termes de type $*$. Un prédicat unaire sur un domaine $A : *$ sera donc assimilé à un terme de type $A \Rightarrow *$, c'est-à-dire $\Pi x : A,\ *$. Pour pouvoir construire un tel type avec la règle Produit, il faudrait pouvoir dériver $\Gamma \vdash * : *$. On est donc tenté d'ajouter un axiome de la forme

$$\textsf{Sorte}\ \frac{\Gamma\ \text{bien formé}}{\Gamma \vdash * : *}$$

Avec ceci, il est également possible d'avoir des quantifications du second ordre, avec un produit dépendant de la forme $\Pi A : *,\ B$, où B est de type $*$. Alors, la distinction entre niveau des termes et niveau des types n'existe plus vraiment. En particulier il est possible de dériver des termes de type $*$ contenant des β-radicaux. Moralement, deux types qui sont β-équivalents devraient typer le même ensemble de termes. Par conséquent on ajoute une nouvelle règle, qui va permettre d'intégrer le calcul dans les démonstrations :

$$\textsf{Conversion}\ \frac{\Gamma \vdash T : A \qquad \Gamma \vdash A : *}{\Gamma \vdash T : B}\ A \overset{*}{\underset{\beta}{\longleftrightarrow}} B$$

On obtient ainsi un système appelé $\lambda*$ proche de la première version de la théorie des types de Per Martin-Löf (1971). Malheureusement, Jean-Yves Girard (1972) a démontré que ce système est incohérent, dans le sens où tout terme de A de type $*$ est habitable, c'est-à-dire il existe un terme T tel que $\vdash T : A$ est dérivable.

Pour résoudre ce problème, on introduit un nouveau terme, $\Box$, qui sera le type de $*$. On utilise alors une règle

$$\textsf{Sorte}\ \frac{\Gamma \text{ bien formé}}{\Gamma \vdash * : \Box}$$

Une seule règle Déclaration est nécessaire :

$$\textsf{Déclaration}\ \frac{\Gamma \text{ bien formé} \qquad \Gamma \vdash B : s}{\Gamma, x : B \text{ bien formé}}\ s \in \{*; \Box\},\ x \text{ non libre dans } \Gamma, B$$

On peut alors généraliser la règle Produit. Pour pouvoir construire un prédicat de type $\Pi x : A,\ *$ on a besoin d'une règle de la forme

$$\textsf{Produit'}\ \frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : \Box}{\Gamma \vdash \Pi x : A,\ B : \Box}$$

On obtient alors le système appelé $\lambda\Pi$ ou λP ou encore LF pour *logical framework* (Robert Harper, Furio Honsell et Gordon Plotkin, 1993).

Pour retrouver la quantification du second ordre, on étend à nouveau la règle Produit :

$$\textsf{Produit''}\ \frac{\Gamma \vdash A : \Box \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A,\ B : *}$$

Sans la règle Produit', on a alors le λ-calcul avec types polymorphes, ou $\lambda 2$, proche du système F de Jean-Yves Girard (1972) pour la logique propositionnelle du second ordre.

Plus généralement, on voit que Produit, Produit' et Produit'' sont en fait des cas particuliers d'une règle

$$\textsf{Produit}\ \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A,\ B : s_3}$$

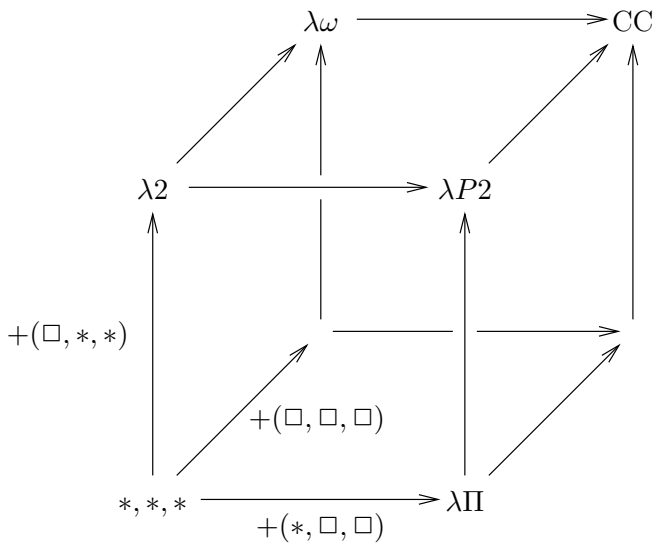


FIG. 3.1 : λ-cube de Henk BARENDREGT

avec s_1, s_2 et s_3 dans $\{*; \Box\}$. Il y a donc 8 règles envisageables, ce qui fait un total de 256 systèmes potentiels, suivant qu'on choisisse d'y inclure une de ces règles ou non. Le *λ-cube* de Henk BARENDREGT (1992) est le sous-ensemble de ces systèmes qui contiennent tous la règle Produit du système $\lambda\!\!\rightarrow$ équivalent au λ-calcul simplement typé (soit $s_1 = s_2 = s_3 = *$), et tels que $s_2 = s_3$ dans toutes les règles retenues. On a donc 8 systèmes, dont $\lambda\Pi$ et $\lambda 2$. On trouvera sur la figure 3.1 sa représentation en fonction des règles considérées. Le système $\lambda\omega$ permet des quantifications d'ordre supérieur et le coin du cube le plus éloigné du système des types simples est le *calcul des constructions* de Thierry COQUAND (1985) qui a été à la base de l'assistant à la démonstration Coq[1].

Les systèmes de type purs sont une façon de généraliser le λ-cube, le système $\lambda*$ et encore d'autres systèmes.

Définition 3.1 (Systèmes de types purs)**.** *Un* système de type pur *est donné par :*
- *un ensemble de* sortes $\boldsymbol{S}$ *;*
- *un ensemble de couples de sortes* $\boldsymbol{A} \subseteq \boldsymbol{S}^2$ *appelés* axiomes *;*
- *un ensemble de triplet de sortes* $\boldsymbol{R} \subseteq \boldsymbol{S}^3$ *appelés* règles *;*

qui définissent un système de type donné en figure 3.2.

Nous avons vu un certain nombre d'exemples de systèmes de type purs. La logique d'ordre supérieur basée sur la théorie simple des types d'Alonzo CHURCH (1940) (cf. section 2.2.2) peut également être représentée à l'aide du système de type pur λHOL

[1] http://coq.inria.fr/

$$\textsf{Vide}\ \frac{}{[]\ \text{bien formé}}$$

$$\textsf{Déclaration}\ \frac{\Gamma\ \text{bien formé} \qquad \Gamma \vdash A : s}{\Gamma, x : A\ \text{bien formé}}\ s \in \boldsymbol{S}\ \text{et}\ x\ \text{non libre dans}\ \Gamma, A$$

$$\textsf{Sorte}\ \frac{\Gamma\ \text{bien formé}}{\Gamma \vdash s_1 : s_2}\ (s_1, s_2) \in \boldsymbol{A}$$

$$\textsf{Variable}\ \frac{\Gamma\ \text{bien formé}}{\Gamma \vdash x : A}\ x : A \in \Gamma$$

$$\textsf{Produit}\ \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A,\ B : s_3}\ (s_1, s_2, s_3) \in \boldsymbol{R}$$

$$\textsf{Abstraction}\ \frac{\Gamma \vdash \Pi x : A,\ B : s \qquad \Gamma, x : A \vdash T : B}{\Gamma \vdash \lambda x : A,\ T : \Pi x : A,\ B}\ s \in \boldsymbol{S}$$

$$\textsf{Application}\ \frac{\Gamma \vdash T : \Pi x : A,\ B \qquad \Gamma \vdash U : A}{\Gamma \vdash T\ U : \{U/x\}B}$$

$$\textsf{Conversion}\ \frac{\Gamma \vdash T : A \qquad \Gamma \vdash A : s}{\Gamma \vdash T : B}\ s \in \boldsymbol{S}\ \text{et}\ A \overset{*}{\longleftrightarrow}_{\beta} B$$

FIG. 3.2 : Système de type pur $\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}$

dont les sortes sont $*$, $\Box$ et Δ ; les axiomes $(*, \Box)$ et $(\Box, \Delta)$; et les règles $(*, *, *)$, $(\Box, *, *)$ et $(\Box, \Box, \Box)$.

Propriétés des systèmes de type purs

Nous recensons ici quelques propriétés des systèmes de type purs, la plupart démontrées dans l'ouvrage de Henk Barendregt (1992) et la thèse de Herman Geuvers (1993).

Premièrement, la β-réduction préserve les types :

Lemme 3.1 (Réduction du sujet, Henk Barendregt, 1992, théorème 5.2.15)**.** *Si $\Gamma \vdash A : B$ est dérivable dans un système de type pur et $A \xrightarrow[\beta]{*} A'$ alors $\Gamma \vdash A' : B$ est aussi dérivable.*

Ceci est également vrai pour la η-réduction, même si elle ne fait pas partie de la règle Conversion :

Lemme 3.2 (Réduction du sujet pour η, Herman Geuvers, 1992, proposition 2.6)**.** *Si x n'est pas libre dans T et si $\Gamma \vdash \lambda x : A\ (T\ x) : B$ est dérivable, alors $\Gamma \vdash T : B$ l'est aussi.*

Ensuite, on peut démontrer que si on a dérivé qu'un terme est d'un certain type, alors ce type est soit une sorte, soit est typé par une sorte :

Lemme 3.3 (Correction des types, Henk Barendregt, 1992, corollaire 5.2.14)**.** *Si $\Gamma \vdash A : B$ est dérivable alors il existe une sorte $s \in \boldsymbol{S}$ telle que soit $B = s$ soit $\Gamma \vdash B : s$ est dérivable.*

On peut dériver de ces lemmes le résultat suivant :

Lemme 3.4. *Si $\Gamma \vdash A : B$ est dérivable et si $B \xrightarrow[\beta]{*} B'$, alors $\Gamma \vdash A : B'$ est dérivable.*

Démonstration. Pour pouvoir utiliser Conversion et ainsi obtenir le résultat attendu, il faut montrer que B est typé par une sorte. Le lemme précédent nous dit que soit c'est le cas, soit B est lui-même une sorte. Ce dernier cas est trivial, car B est alors irréductible et $B' = B$. □

Lemme 3.5 (Substitution, Henk Barendregt, 1992, lemme 5.2.11)**.** *Si $\Gamma, x : B \vdash \alpha$ et $\Gamma \vdash A : B$ sont dérivables, alors $\Gamma \vdash \{A/x\}\alpha$ l'est aussi.*

Définition 3.2. *Un système de type pur est dit* fonctionnel *si les deux conditions suivantes sont réunies :*

- *$\langle s_1, s\rangle \in \boldsymbol{A}$ et $\langle s_1, s'\rangle \in \boldsymbol{A}$ implique $s = s'$;*
- *$\langle s_1, s_2, s\rangle \in \boldsymbol{R}$ et $\langle s_1, s_2, s'\rangle \in \boldsymbol{R}$ implique $s = s'$.*

Cette définition est une condition nécessaire et suffisante pour démontrer qu'un λ-terme possède un type unique à β-réduction près :

Lemme 3.6 (Unicité, Henk Barendregt, 1992, Lemma 5.2.21)**.** *Si* $\Gamma \vdash A : B$ *et* $\Gamma \vdash A : B'$ *sont dérivable dans un système de type pur fonctionnel, alors* $B \overset{*}{\underset{\beta}{\longleftrightarrow}} B'$.

Systèmes de type purs et calculs

Comme nous l'avons dit plus haut, la règle Conversion permet d'introduire du calcul dans les démonstrations. Nous allons illustrer cette affirmation par un exemple.

Henri Poincaré (1902) s'est posé la question si on pouvait vraiment considérer un raisonnement formel conduisant à l'affirmation $2 + 2 = 4$ comme une véritable démonstration. Il tend à penser que non, car ce raisonnement n'est qu'une simple vérification, sa conclusion n'étant pas suffisamment générale vis-à-vis des hypothèses qui y ont conduit : « La démonstration véritable est féconde au contraire parce que la conclusion y est en un sens plus générale que les prémisses. »

Sans vouloir abonder totalement dans cette direction, car il est difficile de définir quand une conclusion est plus générale que ses prémisses, force est de constater qu'un tel raisonnement s'apparente plus à l'application simple d'un calcul qu'à une démonstration. De là provient le principe dit de Poincaré : dans une démonstration, on doit distinguer les étapes de déduction des étapes de calcul.

La règle Conversion permet de faire ceci, puisque le calcul, c'est-à-dire la β-réduction, est séparée des étapes de déduction. Penchons nous sur l'exemple de $2 + 2 = 4$. Il est connu depuis Alonzo Church que l'on peut encoder les entiers naturels par des λ-termes, qui sont par conséquent appelés *entiers de* Church : l'entier n est codé par le terme $\underline{n} \overset{\text{déf}}{=} \lambda x,\ \lambda f,\ \underbrace{f\ (f\ (\ldots (f}_{n \text{ fois}}\ (f\ x)) \ldots))$. On peut alors définir un terme *add* qui fait l'addition de deux entiers : $add \overset{\text{déf}}{=} \lambda m,\ \lambda n,\ \lambda x,\ \lambda f,\ m\ (n\ x\ f)\ f$.

Dans le calcul des constructions, un des types possible pour les entiers de Church est $* \Rightarrow (* \Rightarrow *) \Rightarrow *$, que l'on notera Nat. On vérifie alors que *add* peut bien être typé par $Nat \Rightarrow Nat \Rightarrow Nat$.

On peut alors définir le prédicat d'égalité entre entiers comme un terme *eq* de type $Nat \Rightarrow Nat \Rightarrow *$, avec un terme *refl* permettant de dire que ce prédicat est réflexif, de type $\Pi x : Nat,\ eq\ x\ x$. On travaille donc dans le contexte $\Gamma \overset{\text{déf}}{=} eq : Nat \Rightarrow Nat \Rightarrow *, refl : \Pi x : Nat,\ eq\ x\ x$, dont on peut vérifier qu'il est bien formé. On a alors la démonstration suivante que $2 + 2 = 4$:

$$\dfrac{\dfrac{\dfrac{\Gamma \text{ bien formé}}{\Gamma \vdash refl : \Pi x : Nat,\ eq\ x\ x}\ \text{Variable} \qquad \vdots \quad \Gamma \vdash \underline{4} : Nat}{\Gamma \vdash refl\ \underline{4} : eq\ \underline{4}\ \underline{4}}\ \text{Application} \qquad \vdots \quad \Gamma \vdash eq\ \underline{4}\ \underline{4} : *}{\Gamma \vdash refl\ \underline{4} : eq\ (\text{add}\ \underline{2}\ \underline{2})\ \underline{4}}\ \text{Conversion}$$

On voit que tout le calcul s'effectue lors de l'application de Conversion et que le reste de la démonstration est constitué uniquement de simples vérifications de type. En particulier, le terme de démonstration *refl* $\underline{4}$ ne contient absolument aucune indication concernant le calcul de add $\underline{2}$ $\underline{2}$.

Cette technique de démonstration utilisant le calcul pour simplifier des étapes de déduction peut être généralisée pour donner ce qu'on appelle la réflexivité calculatoire. Elle peut être employée dans de nombreux cas, et c'est l'un des grands avantages de l'assistant à la démonstration Coq. Elle est notamment intensivement utilisée dans la démonstration formelle du théorème des quatre couleurs par Georges Gonthier (2005).

3.2 Déduction modulo

Dans les systèmes de type purs, seule la β-réduction permet de faire des calculs. On peut considérer que c'est suffisant, car le λ-calcul a la même puissance que les machines de Turing. Il faut néanmoins se rappeler d'une part que les systèmes de type purs couramment utilisés sont fortement normalisants, ce qui veut dire que les termes typables ne sont pas suffisant pour encoder une machine de Turing, et d'autre part que l'encodage des fonctions usuelles en λ-calcul pur n'est pas forcément très naturel.

Une première approche pour avoir un calcul plus riche est d'introduire des types inductifs, comme le proposent Thierry Coquand et Christine Paulin-Mohring (1990). Si on les ajoute au calcul des constructions, on obtient le système appelé *calcul des constructions inductives* présenté par Christine Paulin-Mohring (1992) et étudié en détail par Benjamin Werner (1994). Toutefois, on est alors limité par le fait que les fonctions à calculer doivent être définies par induction structurelle. Par exemple, si on a un type inductif pour les entiers naturels

$$\begin{aligned} Nat : * \stackrel{\text{déf}}{=} \quad & O : Nat \\ & \mid s : Nat \Rightarrow Nat \end{aligned}$$

alors on peut définir l'addition par exemple par la fonction

$$add \stackrel{\text{déf}}{=} \lambda m,\ \lambda n,\ Nat\_réc\ Nat\ n\ (\lambda m',\ \lambda r,\ s\ r)\ m$$

où $Nat\_réc : \Pi p : *,\ p \Rightarrow (Nat \Rightarrow p \Rightarrow p) \Rightarrow Nat \Rightarrow p$ est le récurseur associé au type inductif *Nat*. On a alors $add\ O\ n \xrightarrow[\beta\iota]{*} n$ quel que soit n, mais par contre on n'a pas $add\ n\ O \xrightarrow[\beta\iota]{*} n$ si par exemple n est une variable. Dans le premier cas, on peut utiliser Conversion pour intégrer ce calcul aux démonstrations, dans le second, on est obligé de démontrer (par une simple induction sur n) un lemme $\Pi n : Nat,\ \Pi p : Nat \Rightarrow *,\ p\ (add\ n\ O) \Rightarrow p\ n$. Il n'y a pas de véritable raison pour laquelle on pourrait dans un cas se contenter du calcul, alors que dans le cas symétrique on aurait besoin

de déduction. Cela provient uniquement du choix de la variable sur laquelle on a défini inductivement *add*.

Pour résoudre ceci, on est tenté d'autoriser à définir les fonctions dont on a besoin par réécriture. Dans le cas de l'addition, on pourrait par exemple avoir le système

$$\begin{aligned} add\ O\ n &\to n \\ add\ m\ O &\to m \\ add\ (s\ m)\ (s\ n) &\to s\ (s\ (add\ m\ n)) \end{aligned}$$

qui possède une paire critique mais est confluent. Frédéric Blanqui, Jean-Pierre Jouannaud et Mitsuhiro Okada (1999) ont étudié l'ajout de la réécriture dans le calcul des constructions, ce qui a donné le *calcul des constructions algébriques*. Ils donnent des conditions pour que le système avec les nouvelles réductions reste cohérent.

La déduction modulo de Gilles Dowek, Thérèse Hardin et Claude Kirchner (2003) peut être vue comme une généralisation de cette approche à n'importe quel système d'inférence. On considère alors une congruence entre propositions, qui va représenter le calcul, et on applique les règles du système d'inférence modulo cette congruence. Comme il est dit plus haut, une des meilleures approche pour modéliser le calcul est la réécriture, et on va donc considérer que la congruence en question peut s'exprimer sous forme d'un système de réécriture entre propositions.

En général, on ne veut pas que le calcul interagisse trop avec le système d'inférence, et c'est la raison pour laquelle on impose que le système de réécriture réécrive des termes en termes, ou des formules *atomiques* en formules quelconques.

On se place à présent dans la logique du premier ordre.

Définition 3.3 (Système de réécriture propositionnel)**.** *Une* règle de réécriture propositionnelle *est donnée par un couple formé d'une proposition atomique A et d'une formule du premier ordre quelconque P, telles que $FV(P) \subseteq FV(A)$. On la note $A \to P$.*

Une formule P_1 se réécrit en une formule P_2 par la règle de réécriture $A \to P$ à la position $\mathfrak{p}$ si il existe une substitution σ telle que $P_{1|\mathfrak{p}} = \sigma A$ et $P_2 = P_1[\sigma P]_\mathfrak{p}$. On note alors $P_1 \xrightarrow[A\to P]{\mathfrak{p},\sigma} P_2$, ou parfois plus simplement $P_1 \underset{A\to P}{\longrightarrow} P_2$.

Un système de réécriture propositionnel *est un ensemble de règle de réécriture propositionnelle. Une formule P est réécrite en Q par un système de réécriture propositionnel $\mathcal{R}$ si elle l'est par une de ses règles. On note alors $P \underset{\mathcal{R}}{\longrightarrow} Q$. $\xrightarrow[\mathcal{R}]{*}$ est la fermeture réflexive et transitive de cette relation, $\xleftrightarrow[\mathcal{R}]{*}$ sa fermeture réflexive, symétrique et transitive.*

Le fait de pouvoir lier des variables dans la partie droite d'une règle de réécriture propositionnelle, par exemple dans $A(x) \to \exists y,\ B(x,y)$ fait qu'il ne s'agit pas exactement de systèmes de réécriture usuels. En particulier il faut prendre soin quand on applique les règles que la subsitution ne capture pas de variable à droite. Ainsi, $A(f(x,y))$ est réécrit par la règle ci-dessus en $\exists z,\ B(f(x,y),z)$.

En déduction modulo, on considère à la fois un système de réécriture usuel sur les termes, dont la relation de réécriture est étendue aux formules par congruence, et un système de réécriture propositionnel. On appellera donc système de réécriture l'union de deux tels systèmes.

Étant donné un tel système de réécriture $\mathcal{R}$, on peut alors définir comment appliquer un système d'inférence modulo $\mathcal{R}$: à chaque méta-formule Φ d'une règle d'inférence on associe une variable propositionnelle fraîche X_Φ de même arité, et on définit la nouvelle règle d'inférence comme la règle originelle dans laquelle on a remplacé les Φ par les X_Φ et dont les substitutions valables sont restreintes aux σ tels que pour toute méta-formule Φ apparaissant dans la règle originelle, si $(\Psi_i)_i$ sont les sous-méta-formules strictes de Φ apparaissant à un autre endroit dans la règle originelle, alors $\sigma(X_\Phi)$ est équivalent par la relation $\xleftrightarrow[\mathcal{R}]{*}$ à l'application de σ à la méta-formule Φ dans laquelle on a remplacés les Ψ_i par X_{Ψ_i}.

Plus concrètement, prenons l'exemple du calcul des séquences LK modulo un système de réécriture $\mathcal{R}$, tel qu'on peut le trouver chez Gilles Dowek et collaborateurs (2003). Les nouvelles règles d'inférence sont représentées en figure 3.3. On y a traduite la restriction sur les substitutions valables par des conditions sur les méta-formules.

De la même façon on peut définir la *déduction naturelle modulo* $\mathcal{R}$, dont on ne donne ici pour l'exemple que les règles pour $\vee$:

$$\vdash \vee \frac{\Gamma \vdash X}{\Gamma \vdash Z} \text{ il existe un } Y \text{ tel que } Z \xleftrightarrow[\mathcal{R}]{*} X \vee Y \text{ ou } Z \xleftrightarrow[\mathcal{R}]{*} Y \vee X \qquad \vee \vdash \frac{\Gamma \vdash W \quad \Gamma, X \vdash Z \quad \Gamma, Y \vdash Z}{\Gamma \vdash Z} \, W \xleftrightarrow[\mathcal{R}]{*} X \vee Y$$

Exemple 3.1 *:* Revenons sur l'exemple de la démonstration de $2 + 2 = 4$. On encode les entiers à des symboles de fonction O et s et on se place dans le système de réécriture $\mathcal{R}$ constitué de deux règles sur les termes et d'une règle propositionnelle

$$\begin{gathered} \mathsf{O} + n \to n \\ s(m) + n \to s(m + n) \\ x = x \to \top \end{gathered}$$

Alors on a

$$\begin{aligned} & s(s(\mathsf{O})) + s(s(\mathsf{O})) = s(s(s(\mathsf{O})))) \\ \xrightarrow[\mathcal{R}]{*} \; & s(s(s(s(\mathsf{O})))) = s(s(s(s(\mathsf{O})))) \\ \xrightarrow[\mathcal{R}]{} \; & \top \end{aligned}$$

et on a la démonstration

$$\vdash \top \frac{}{\vdash \underline{2} + \underline{2} = \underline{4}} \, \underline{2} + \underline{2} = \underline{4} \xleftrightarrow[\mathcal{R}]{*} \top$$

Règles d'identité

$$\vdash\frown \frac{}{X \vdash Y}\ X \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Y \qquad \smile\vdash \frac{\Gamma, X \vdash \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash \Delta}\ X \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Y$$

Règles structurelles

$$\cdot\cdot\vdash \frac{\Gamma, X, Y \vdash \Delta}{\Gamma, Z \vdash \Delta}\ X \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Z \qquad \vdash\cdot\cdot \frac{\Gamma \vdash X, Y, \Delta}{\Gamma \vdash Z, \Delta}\ X \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Z$$

$$\cdot\vdash \frac{\Gamma \vdash \Delta}{\Gamma, X \vdash \Delta} \qquad \vdash\cdot \frac{\Gamma \vdash \Delta}{\Gamma \vdash X, \Delta}$$

Règles logiques

$$\Rightarrow\vdash \frac{\Gamma \vdash X, \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, Z \vdash \Delta}\ Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} X \Rightarrow Y \qquad \vdash\Rightarrow \frac{\Gamma, X \vdash Y, \Delta}{\Gamma \vdash Z, \Delta}\ Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} X \Rightarrow Y$$

$$\wedge\vdash \frac{\Gamma, X \vdash \Delta}{\Gamma, Z \vdash \Delta}\ \begin{array}{l}\text{il existe un } Y \text{ tel que}\\ Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} X \wedge Y \text{ ou } Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Y \wedge X\end{array} \qquad \vdash\wedge \frac{\Gamma \vdash X, \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash Z, \Delta}\ Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} X \wedge Y$$

$$\vee\vdash \frac{\Gamma, X \vdash \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, Z \vdash \Delta}\ Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} X \vee Y \qquad \vdash\vee \frac{\Gamma \vdash X, \Delta}{\Gamma \vdash Z, \Delta}\ \begin{array}{l}\text{il existe un } Y \text{ tel que}\\ Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} X \vee Y \text{ ou } Z \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} Y \vee X\end{array}$$

$$\forall\vdash \frac{\Gamma, X(t) \vdash \Delta}{\Gamma, Y \vdash \Delta}\ Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \forall x,\ X(x) \qquad \vdash\forall \frac{\Gamma \vdash X(y), \Delta}{\Gamma \vdash Y, \Delta}\ \begin{array}{l}y \text{ non libre dans } \Gamma, X(x), \Delta\\ \text{et } Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \forall x,\ X(x)\end{array}$$

$$\exists\vdash \frac{\Gamma, X(y) \vdash \Delta}{\Gamma, Y \vdash \Delta}\ \begin{array}{l}y \text{ non libre dans } \Gamma, X(x), \Delta\\ \text{et } Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \exists x,\ X(x)\end{array} \qquad \vdash\exists \frac{\Gamma \vdash X(t), \Delta}{\Gamma \vdash Y, \Delta}\ Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \exists x,\ X(x)$$

$$\bot\vdash \frac{}{X \vdash}\ X \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \bot \qquad \vdash\top \frac{}{\vdash X}\ X \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \top$$

$$\neg\vdash \frac{\Gamma \vdash X, \Delta}{\Gamma, Y \vdash \Delta}\ Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \neg X \qquad \vdash\neg \frac{\Gamma, X \vdash \Delta}{\Gamma \vdash Y, \Delta}\ Y \overset{*}{\underset{\mathcal{R}}{\longleftrightarrow}} \neg X$$

FIG. 3.3 : Règles du calcul des séquences classique modulo $\mathcal{R}$

qui ne fait intervenir que du calcul.

Le fait de pouvoir réécrire des formules rend la déduction modulo très puissante. En particulier, il est même possible d'obtenir des système incohérents, par exemple en travaillant modulo le système de réécriture $A \to \neg A$. Toutefois, la logique associée au système d'inférence modulo un système de réécriture correspond à la logique engendrée par un certaine théorie que l'on peut associer à ce système, dont les présentations sont alors appelées compatibles :

Définition 3.4 (Présentation compatible)**.** *Étant donné un système de réécriture $\mathcal{R}$, une présentation A est dite* compatible *avec $\mathcal{R}$ si la logique associée au système d'inférence modulo $\mathcal{R}$ et la logique engendrée par $\mathrm{Th}\, A$ à partir du système originel coïncident. Plus concrètement, dans le cas du calcul des séquences, cela revient aux deux conditions suivantes :*

- *pour toutes formules P et Q, si $P \xleftrightarrow[\mathcal{R}]{*} Q$ alors $P \Leftrightarrow Q$ est démontrable sous les hypothèses A dans le système originel (sans modulo) ;*
- *pour toute formule P de A, on peut démontrer P sans hypothèse dans le système modulo $\mathcal{R}$.*

Si la logique associée au système d'inférence modulo $\mathcal{R}$ sans $\vdash\!\!\!\sim$ correspond à la logique engendrée par $\mathrm{Th}\, A$ à partir du système originel avec $\vdash\!\!\!\sim$, on parle alors de compatibilité forte*. Pour le calcul des séquences, cela revient à supposer que pour le deuxième point on a des démonstrations sans coupure.*

Avec cette définition, on voit que la compatibilité d'une présentation avec un système de réécriture dépend de la logique dans laquelle on se place. En particulier, nous montrerons un exemple de présentation compatible avec un certain système de réécriture en logique classique mais pas en logique intuitionniste (cf. remarque 4.1).

Gilles Dowek et collaborateurs (2003) montrent que l'on peut toujours trouver une telle présentation : il suffit de prendre la présentation (infinie) qui contient, pour toute formule P et Q telles que $P \xleftrightarrow[\mathcal{R}]{*} Q$, la formule $\forall x_1,\ \ldots,\ \forall x_n,\ P \Leftrightarrow Q$ où $x_1, \ldots, x_n$ sont les variables libres de P et Q. Dans le cas où on travaille dans une logique contenant l'égalité, ou en l'absence de règle sur les termes, on peut se contenter d'une présentation plus simple, qui contient un axiome $\forall x_1,\ \ldots,\ \forall x_n,\ s = t$ pour toute règle de réécriture sur les termes $s \to t$ avec $x_1, \ldots, x_n$ les variables libres de s ; et un axiome $\forall x_1,\ \ldots,\ \forall x_n,\ A \Leftrightarrow P$ pour toute règle de réécriture propositionnelle $A \to P$ avec $x_1, \ldots, x_n$ les variables libres de A.

La cohérence du système modulo $\mathcal{R}$ est donc équivalente à la cohérence de la logique engendrée par la théorie d'une présentation compatible à $\mathcal{R}$. Si le système originel est correct et complet pour la logique du premier ordre, alors la cohérence du système modulo $\mathcal{R}$ est équivalente à la satisfaisabilité d'une présentation compatible.

3.2.1 Variations sur les systèmes de déduction modulo

Il existe plusieurs manières plus ou moins équivalentes de considérer des systèmes de déduction modulo, suivant qu'on travaille modulo un système de réécriture de façon implicite comme nous venons de le voir, ou grâce à des règles d'inférence explicites ; ou bien qu'on oriente l'application des règles de réécriture dans les démonstrations ; ou encore qu'on ne puisse réécrire que des propositions atomiques ; ou enfin que l'on différencie les réécritures autorisées à gauche ou à droite d'une séquence.

On peut considérer une règle de conversion explicite, comme dans le cas des systèmes de type purs. Dans ce cas on garde les règles d'inférence du système originel, et on y a ajoute une ou plusieurs règles pour faire intervenir le système de réécriture. Dans le cas où on n'a pas de types dépendants, il n'y a pas besoin de vérifier que la proposition considérée est typée par une sorte, on obtient donc la règle, en déduction naturelle modulo $\mathcal{R}$

$$\mathsf{Conversion}\ \frac{\Gamma \vdash A}{\Gamma \vdash B}\ A \xleftrightarrow[\mathcal{R}]{*} B$$

Dans le cas du calcul des séquences modulo $\mathcal{R}$, on considère deux règles de conversion, une à gauche et une à droite.

$$\xleftrightarrow{*}\vdash\ \frac{\Gamma, A \vdash \Delta}{\Gamma, B \vdash \Delta}\ A \xleftrightarrow[\mathcal{R}]{*} B \qquad\qquad \vdash\xleftrightarrow{*}\ \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash B, \Delta}\ A \xleftrightarrow[\mathcal{R}]{*} B$$

Bien entendu, la version où on applique les règles d'inférence modulo le système de réécriture, et la version avec les règles de conversion explicites sont équivalentes.

Si on se place du point de vue de la démonstration automatique, on a également envie d'orienter l'applications des règles de réécriture dans les démonstrations, pour que la recherche de démonstrations soit effective. On obtient ainsi par exemple le *calcul des séquences asymétrique modulo* de Gilles Dowek (2003), dans lequel les conversions sont implicites. On obtient alors par exemple les règles d'identité et les règles pour la disjonction :

$$\vdash\frown\ \frac{}{X \vdash Y}\ X \xrightarrow[\mathcal{R}]{*} \xleftarrow[\mathcal{R}]{*} Y \qquad\qquad \vdash\smile\ \frac{\Gamma, X \vdash \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash \Delta}\ X \xleftarrow[\mathcal{R}]{*} \xrightarrow[\mathcal{R}]{*} Y$$

$$\vee\vdash\ \frac{\Gamma, X \vdash \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, Z \vdash \Delta}\ Z \xrightarrow[\mathcal{R}]{*} X \vee Y \qquad\qquad \vdash\vee\ \frac{\Gamma \vdash X, \Delta}{\Gamma \vdash Z, \Delta}\ \begin{array}{l}\text{il existe un } Y \text{ tel que}\\ Z \xrightarrow[\mathcal{R}]{*} X \vee Y \text{ ou } Z \xrightarrow[\mathcal{R}]{*} Y \vee X\end{array}$$

Quand on fait une recherche de démonstration par le bas, on a donc besoin de réécrire les formules que dans un sens, vers le haut de la démonstration. Sans la règle $\vdash\smile$, ce système n'est pas tout à fait équivalent au calcul des séquences modulo sans $\vdash\smile$. Par exemple, si on considère le système $\mathcal{R} \overset{\text{déf}}{=} \begin{cases} A \to B \\ A \to C \end{cases}$ alors $B \vdash C$ est démontrable en calcul des séquences modulo $\mathcal{R}$ sans coupure, mais on ne peut le démontrer en calcul des séquences

avec dépliage qu'avec $\vdash\!\!\!\subset$ (en coupant par A par exemple). Comme on le voit, à partir du moment où on s'intéresse aux coupures, on n'a l'équivalence que quand le système de réécriture est confluent.

On peut même se contenter pour les règles de conversions d'un seul pas de réécriture sur des propositions atomiques. Si on applique ceci à la déduction naturelle, on obtient la *déduction naturelle avec pliage–dépliage* (Dag Prawitz, 1965), avec deux nouvelles règles

$$\uparrow \frac{\Gamma \vdash \sigma P}{\Gamma \vdash \sigma A}\, A \to P \in \mathcal{R} \qquad\qquad \downarrow \frac{\Gamma \vdash \sigma A}{\Gamma \vdash \sigma P}\, A \to P \in \mathcal{R}$$

avec une nouvelle réduction de démonstration

$$\begin{array}{l} \uparrow \dfrac{\overset{\pi}{\Gamma \vdash \sigma P}}{\Gamma \vdash \sigma A}\, A \to P \in \mathcal{R} \\ \downarrow \dfrac{\Gamma \vdash \sigma A}{\Gamma \vdash \sigma P}\, A \to P \in \mathcal{R} \end{array} \quad \to \quad \overset{\pi}{\Gamma \vdash \sigma P}$$

Pour que cette règle de réduction fonctionne bien, on considère en général que le système de réécriture utilisé ne doit pas comprendre de paires critiques, de façon à ce qu'on ait bien le même P ci-dessus.

Pour étendre l'isomorphisme de Curry-Howard, les termes de démonstration correspondant sont alors étendus par des constructions $\uparrow_R t$ et $\downarrow_R t$ pour toute règle de réécriture propositionnelle R, et on ajoute la réduction

$$\downarrow_R \uparrow_R t \to t \ .$$

Gilles Dowek (2001) a démontré que la déduction naturelle avec pliage–dépliage et la déduction naturelle modulo sont équivalentes, y compris en ce qui concerne la terminaison de la réduction des démonstrations (en supposant qu'une proposition atomique donnée ne puisse être réécrite que d'une seule façon).

On peut appliquer la même idée au calcul des séquences. Si on voit les règles d'inférence $\uparrow$ et $\downarrow$ comme des règles d'introduction et d'élimination, alors on obtient des règles droite et gauche y correspondant, en ajoutant une contraction pour donner des règles inversibles :

$$\uparrow\vdash \frac{\Gamma, \sigma A, \sigma P \vdash \Delta}{\Gamma, \sigma A \vdash \Delta}\, A \to P \in \mathcal{R} \qquad\qquad \vdash\uparrow \frac{\Gamma \vdash \sigma P, \sigma A, \Delta}{\Gamma \vdash \sigma A.\Delta}\, A \to P \in \mathcal{R}$$

Nous appellerons par la suite le système G4 plus ces deux règles le *calcul des séquences avec dépliage*. On voit que l'application des règles de réécriture est orientée du bas vers le haut des démonstrations. Le calcul des séquences avec dépliage est en fait équivalent au calcul des séquences modulo asymétrique quand on tient compte des coupures.

Enfin, on a vu qu'une règle de réécriture propositionnelle $A \to P$ peut être, en première approximation, vue sémantiquement commme l'équivalence $A \Leftrightarrow P$. Ceci provient du fait

que l'on peut réécrire les formules à la fois à gauche ou à droite des séquences, ou plus précisément en position positive ou négative d'une formule. Pour avoir un contrôle plus fin, Gilles Dowek (2002) a introduit le calcul des séquences polarisé modulo. Pour cela, on donne tout d'abord une *polarité* + ou − aux règles de réécriture propositionnelles. On notera cette polarité sur la flèche de la règle, par exemple $A \rightarrow^{+} P$ ou $A \rightarrow^{-} P$. Un *système de réécriture polarisé* est un ensemble de règles de réécriture polarisées. On définit également un *polarité* sur les positions d'une formule : la racine est positive, et on ne change de polarité que sous $\neg$ et à gauche de $\Rightarrow$. On peut alors donner une polarité à la relation de réécriture :

Définition 3.5. *On dit que P se réécrit en Q positivement dans le système de réécriture polarisé $\mathcal{R}$, et on note $P \underset{\mathcal{R}\ +}{\longrightarrow} Q$ si P se réécrit en Q par une règle positive de $\mathcal{R}$ à une position positive de P, ou par une règle négative de $\mathcal{R}$ à une position négative de P.*

On dit que P se réécrit en Q négativement dans le système de réécriture polarisé $\mathcal{R}$, et on note $P \underset{\mathcal{R}\ -}{\longrightarrow} Q$ si P se réécrit en Q par une règle positive de $\mathcal{R}$ à une position négative de P, ou par une règle négative de $\mathcal{R}$ à une position positive de P.

On étend ces relations en leur fermeture transitive et réflexive-transitive que l'on notera respectivement $\underset{\mathcal{R}\ +}{\overset{+}{\longrightarrow}}$, $\underset{\mathcal{R}\ +}{\overset{*}{\longrightarrow}}$, $\underset{\mathcal{R}\ -}{\overset{+}{\longrightarrow}}$ et $\underset{\mathcal{R}\ -}{\overset{*}{\longrightarrow}}$

Remarque 3.2 *:* Ne pas confondre $\underset{\mathcal{R}}{\overset{+}{\longrightarrow}}$ et $\underset{\mathcal{R}\ +}{\longrightarrow}$.

Pour plus de simplicité, nous considérerons par la suite que les règles de réécriture sur les termes sont à la fois positives et négatives, ce qui fait qu'elles peuvent intervenir indifféremment à une position positive ou négative. On pourrait bien entendu considérer également des polarités sur ces règles. Le *calcul des séquences polarisé modulo* consiste alors à se placer dans le calcul des séquences asymétrique modulo et à ne réécrire les formules que positivement à droite des séquences, et que négativement à gauche. Les règles d'inférence sont représentées dans la figure 3.4. Il est à noter que pour retrouver les règles d'inférence du calcul des séquences asymétrique modulo $\mathcal{R}$, il suffit d'y supprimer les notions de polarité. Une règle $A \rightarrow^{+} P$ est équivalente à l'implication $P \Rightarrow A$. En effet il est possible de démontrer $\vdash P \Rightarrow A$ dans le calcul des séquences polarisé modulo $\{A \rightarrow^{+} P\}$. Dualement, $A \rightarrow^{-} P$ correspond à $A \Rightarrow P$.

Si on combine les polarités du calcul des séquences polarisé modulo avec la réécriture explicite et atomique du calcul des séquences avec dépliage, on obtient ce que nous appellerons le calcul des séquences avec dépliage polarisé. Les règles d'inférence sont celles de G4 plus celles de la figure 3.5. On notera que comme X est atomique dans les règles de dépliage, cela veut dire qu'on utilise une règle négative dans $\uparrow\vdash$ et positive dans $\vdash\uparrow$.

Pour plus de confort, nous introduisons des notations pour le calcul des séquences avec dépliage polarisé. On considère par la suite que le système de réécriture sur les termes est fixe. On le notera $\mathcal{R}_{\mathcal{T}(\Sigma,V)}$.

Règles d'identité

$$\frown\vdash \frac{}{X \vdash Y}\ X \xrightarrow[\mathcal{R}\ -]{*} \xleftarrow[\mathcal{R}\ +]{*} Y \qquad \vdash\smile \frac{\Gamma, X \vdash \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash \Delta}\ X \xleftarrow[\mathcal{R}\ -]{*} \xrightarrow[\mathcal{R}\ +]{*} Y$$

Règles structurelles

$$\cdot\vdash \frac{\Gamma, X, Y \vdash \Delta}{\Gamma, Z \vdash \Delta}\ X \xleftarrow[\mathcal{R}\ -]{*} Z \xrightarrow[\mathcal{R}\ -]{*} Y \qquad \vdash\cdot \frac{\Gamma \vdash X, Y, \Delta}{\Gamma \vdash Z, \Delta}\ X \xleftarrow[\mathcal{R}\ +]{*} Z \xrightarrow[\mathcal{R}\ +]{*} Y$$

$$\cdot\vdash \frac{\Gamma \vdash \Delta}{\Gamma, X \vdash \Delta} \qquad \vdash\cdot \frac{\Gamma \vdash \Delta}{\Gamma \vdash X, \Delta}$$

Règles logiques

$$\Rightarrow\vdash \frac{\Gamma \vdash X, \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, Z \vdash \Delta}\ Z \xrightarrow[\mathcal{R}\ -]{*} X \Rightarrow Y \qquad \vdash\Rightarrow \frac{\Gamma, X \vdash Y, \Delta}{\Gamma \vdash Z, \Delta}\ Z \xrightarrow[\mathcal{R}\ +]{*} X \Rightarrow Y$$

$$\wedge\vdash \frac{\Gamma, X \vdash \Delta}{\Gamma, Z \vdash \Delta}\ \text{il existe un } Y \text{ tel que } Z \xrightarrow[\mathcal{R}\ -]{*} X \wedge Y \text{ ou } Z \xrightarrow[\mathcal{R}\ -]{*} Y \wedge X \qquad \vdash\wedge \frac{\Gamma \vdash X, \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash Z, \Delta}\ Z \xrightarrow[\mathcal{R}\ +]{*} X \wedge Y$$

$$\vee\vdash \frac{\Gamma, X \vdash \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, Z \vdash \Delta}\ Z \xrightarrow[\mathcal{R}\ -]{*} X \vee Y \qquad \vdash\vee \frac{\Gamma \vdash X, \Delta}{\Gamma \vdash Z, \Delta}\ \text{il existe un } Y \text{ tel que } Z \xrightarrow[\mathcal{R}\ +]{*} X \vee Y \text{ ou } Z \xrightarrow[\mathcal{R}\ +]{*} Y \vee X$$

$$\forall\vdash \frac{\Gamma, X(t) \vdash \Delta}{\Gamma, Y \vdash \Delta}\ Y \xrightarrow[\mathcal{R}\ -]{*} \forall x,\ X(x) \qquad \vdash\forall \frac{\Gamma \vdash X(y), \Delta}{\Gamma \vdash Y, \Delta}\ \begin{array}{l} y \text{ non libre dans } \Gamma, X(x), \Delta \\ \text{et } Y \xrightarrow[\mathcal{R}\ +]{*} \forall x,\ X(x) \end{array}$$

$$\exists\vdash \frac{\Gamma, X(y) \vdash \Delta}{\Gamma, Y \vdash \Delta}\ \begin{array}{l} y \text{ non libre dans } \Gamma, X(x), \Delta \\ \text{et } Y \xrightarrow[\mathcal{R}\ -]{*} \exists x,\ X(x) \end{array} \qquad \vdash\exists \frac{\Gamma \vdash X(t), \Delta}{\Gamma \vdash \exists x, X(x), \Delta}\ Y \xrightarrow[\mathcal{R}\ +]{*} \exists x,\ X(x)$$

$$\bot\vdash \frac{}{X \vdash}\ X \xrightarrow[\mathcal{R}\ -]{*} \bot \qquad \vdash\top \frac{}{\vdash X}\ X \xrightarrow[\mathcal{R}\ +]{*} \top$$

$$\neg\vdash \frac{\Gamma \vdash X, \Delta}{\Gamma, Y \vdash \Delta}\ Y \xrightarrow[\mathcal{R}\ -]{*} \neg X \qquad \vdash\neg \frac{\Gamma, X \vdash \Delta}{\Gamma \vdash Y, \Delta}\ Y \xrightarrow[\mathcal{R}\ +]{*} \neg X$$

FIG. 3.4 : Règles du calcul des séquences classique polarisé modulo $\mathcal{R}$

Règles de dépliage

$$\uparrow\vdash \frac{\Gamma, X, Y \vdash \Delta}{\Gamma, X \vdash \Delta}\ X \xrightarrow[\mathcal{R}\ -]{} Y,\ X \text{ atomique} \qquad \vdash\uparrow \frac{\Gamma \vdash Y, X, \Delta}{\Gamma \vdash X, \Delta}\ X \xrightarrow[\mathcal{R}\ +]{} Y,\ X \text{ atomique}$$

FIG. 3.5 : Règles d'inférence du calcul des séquences avec dépliage polarisé supplémentaires par rapport à G4

Définition 3.6. *On dira qu'une démonstration de* $\Gamma \vdash \Delta$ *est dans* $\mathcal{R}$*, et on notera* $\Gamma \vdash_{\mathcal{R}} \Delta$*, si toutes les règles de réécriture utilisées dans les règles de dépliage sont dans* $\mathcal{R} \cup \mathcal{R}_{\mathcal{T}(\Sigma,V)}$*.*

On notera $\Gamma \vdash^*_{\mathcal{R}} \Delta$ *si la démonstration n'utilise pas* $\vdash_\smile$*.*

On notera $\Gamma \vdash \Delta$ *si la démonstration n'utilise pas* $\uparrow\vdash$ *ni* $\vdash\uparrow$*.*

Remarque 3.3 *:* $\Gamma \vdash \Delta$ et $\Gamma \vdash_\emptyset \Delta$ ne sont pas équivalents, puisqu'on peut utiliser les règles de réécriture sur les termes dans le second cas.

Nous montrons maintenant en détail que le calcul des séquences avec dépliage polarisé est équivalent au calcul des séquences polarisé modulo. On peut tout d'abord étendre les lemmes 2.3, 2.5 et 2.6 (affaiblissement, inversibilité et contraction). Pour cela, il suffit de prolonger le RPO défini sur les squelettes de démonstrations de G4 dans la section 2.3.4, en rajoutant à la précession $(\frown_\vdash, P) > (\uparrow\vdash, Q)$ et $(\frown_\vdash, P) > (\vdash\uparrow, Q)$ pour toutes formules P et Q. Les démonstrations des lemmes 2.3 et 2.5 ne changent pas. Pour le lemme 2.6, il y a un nouveau cas : supposons qu'on ait une démonstration de $\Gamma, A, A \vdash \Delta$ dont la dernière règle est $\uparrow\vdash$ sur un des A avec $A \underset{-}{\longrightarrow} P$. La sous-démonstration directe a pour conséquence $\Gamma, A, A, P \vdash \Delta$. Par hypothèse d'induction, on a une démonstration de $\Gamma, A, P \vdash \Delta$ sur laquelle on applique $\uparrow\vdash$ pour obtenir une démonstration de $\Gamma, A \vdash \Delta$.

Ensuite, il nous faut traduire les conversions implicites sur n'importe quelle formule en des conversions explicites sur des propositions atomiques.

Lemme 3.7. *Si on a* $\Gamma, Q \vdash_{\mathcal{R}} \Delta$ *et* $P \underset{\mathcal{R}\cup\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\overset{*}{\longrightarrow}}{}_{-} Q$*, alors* $\Gamma, P \vdash_{\mathcal{R}} \Delta$*.*

De façon duale, si on a $\Gamma \vdash_{\mathcal{R}} Q, \Delta$ *et* $P \underset{\mathcal{R}\cup\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\overset{*}{\longrightarrow}}{}_{+} Q$*, alors* $\Gamma \vdash_{\mathcal{R}} P, \Delta$*.*

Démonstration. On procède d'abord par induction mutuelle sur la longueur de la réécriture $P \underset{\mathcal{R}\cup\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\overset{*}{\longrightarrow}}{}_{\pm} Q$. Si elle est nulle le résultat est trivial, sinon il suffit de montrer le résultat pour un étape.

On procède alors par induction sur la profondeur de la position à laquelle est appliquée la réécriture $P \underset{\mathcal{R}}{\longrightarrow} Q$. Si cette profondeur est nulle, alors P est atomique. On peut donc simplement appliquer $\uparrow\vdash$. Sinon, cela veut dire que P n'est pas atomique. Supposons que P est de la forme $P_1 \Rightarrow P_2$ (les autres cas se traitant de la même façon). Alors $Q = Q_1 \Rightarrow Q_2$ et soit $P_1 \underset{\mathcal{R}\cup\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\longrightarrow}{}_{+} Q_1$ et $P_2 = Q_2$, soit $P_2 \underset{\mathcal{R}\cup\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\longrightarrow}{}_{-} Q_2$ et $P_1 = Q_1$. En utilisant le lemme 2.5 étendu, on a $\Gamma, Q_2 \vdash_{\mathcal{R}} \Delta$ et $\Gamma \vdash_{\mathcal{R}} Q_1, \Delta$. On utilise l'hypothèse d'induction sur l'une ou l'autre de ces démonstrations suivant les cas, et on obtient $\Gamma, P_2 \vdash_{\mathcal{R}} \Delta$ et $\Gamma \vdash_{\mathcal{R}} P_1, \Delta$. En appliquant $\Rightarrow\vdash$ on obtient $\Gamma, P \vdash_{\mathcal{R}} \Delta$. □

Nous pouvons maintenant démontrer l'équivalence :

Théorème 3.8.
Une séquence $\Gamma \vdash \Delta$ *possède une démonstration (resp. sans coupure) dans le calcul des*

séquences polarisé modulo $\mathcal{R} \cup \mathcal{R}_{\mathcal{T}(\Sigma,V)}$ ssi elle possède une démonstration (resp. sans coupure) dans $\mathcal{R}$ pour le calcul des séquences avec dépliage polarisé.

Démonstration. La direction « si » se démontre en faisant passer les règles de conversions explicite de façon implicite dans la ou les règles d'inférence qui, le cas échéant, décomposent la formule réécrite.

Réciproquement, on procède par induction sur le squelette de la démonstration en calcul des séquences polarisé modulo. Si une règle logique ou $\sqsubset$ utilise des conversions implicites, on applique cette règle sans conversion puis on utilise le lemme 3.7 pour obtenir la conséquence attendue. Par exemple, si on a une démonstration

$$\Rightarrow\vdash \frac{\overset{\pi}{\Gamma \vdash P, \Delta} \qquad \overset{\pi'}{\Gamma, Q \vdash \Delta}}{\Gamma, O \vdash \Delta}\, O \xrightarrow[\mathcal{R}\ -]{*} P \Rightarrow Q$$

on applique l'hypothèse d'induction sur π et π' pour avoir $\Gamma \vdash_{\mathcal{R}} P, \Delta$ et $\Gamma, Q \vdash_{\mathcal{R}} \Delta$, on applique $\Rightarrow\vdash$ pour obtenir $\Gamma, P \Rightarrow Q \vdash_{\mathcal{R}} \Delta$ puis grâce au lemme 3.7 on obtient $\Gamma, O \vdash_{\mathcal{R}} \Delta$. Pour $\wedge\vdash$, $\vdash\vee$, $\forall\vdash$ et $\vdash\exists$, on a également besoin du lemme 2.3 pour pouvoir appliquer la règle logique dans le calcul des séquences avec dépliage polarisé. Pour les règles de contraction (resp. $\vdash\!\supset$), on applique d'abord le lemme 3.7 pour se ramener à une séquence sur laquelle on peut utiliser le lemme 2.6 (resp. appliquer $\vdash\!\supset$). Par exemple, si on a une démonstration

$$\because\vdash \frac{\overset{\pi}{\Gamma, P, Q \vdash \Delta}}{\Gamma, O \vdash \Delta}\, P \xleftarrow[\mathcal{R}\ -]{*} O \xrightarrow[\mathcal{R}\ -]{*} Q$$

on applique l'hypothèse d'induction sur π pour obtenir $\Gamma, P, Q \vdash_{\mathcal{R}} \Delta$, puis on applique deux fois le lemme 3.7 ce qui donne $\Gamma, O, O \vdash_{\mathcal{R}} \Delta$, puis on utilise le lemme 2.6 ce qui donne $\Gamma, O \vdash \Delta$. Pour les règles d'affaiblissement, on utilise le lemme 2.3.

On a pris soin de ne pas rajouter de coupures dans les démonstrations des lemmes 2.3, 2.6 et 3.7. Si la démonstration de départ ne possède pas de coupures, la démonstration en calcul des séquences avec dépliage polarisé n'en aura pas non plus. □

Corollaire 3.9. *Le calcul des séquences asymétrique modulo est équivalent (avec la même signification) au calcul des séquences avec dépliage (non polarisé).*

Démonstration. Tout système de réécriture non polarisé $\mathcal{R}$ peut être vu comme un système polarisé $\mathcal{R}^{\pm}$ en considérant deux règles $A \rightarrow^{+} P$ et $A \rightarrow^{-} P$ pour chaque règle $A \rightarrow P$ du système. Le calcul des séquences asymétrique modulo $\mathcal{R}$ (resp. le calcul des séquences avec dépliage dans $\mathcal{R}$) correspond alors au calcul des séquences polarisé modulo $\mathcal{R}^{\pm}$ (resp. au calcul des séquences avec dépliage polarisé dans $\mathcal{R}^{\pm}$).

Cette transformation permet également de montrer que les lemmes 2.3, 2.5, 2.6 et 3.7 sont également valides pour le calcul des séquences avec dépliage (non polarisé). □

On vient de voir comment passer d'un système de réécriture non polarisé à un système équivalent polarisé. On s'intéresse maintenant à la réciproque. Nous avons vu qu'une règle positive $A \rightarrow^+ P$ correspond à un axiome $P \Rightarrow A$. Cet axiome est logiquement équivalent à $A \Leftrightarrow (A \vee P)$, qui correspond à la règle non polarisée $A \rightarrow A \vee P$. Nous proposons donc la traduction suivante d'un système polarisé $\mathcal{R}$ en un système non polarisé $\mathcal{R}^{\mp}$:

- une règle positive $A \rightarrow^+ P$ est traduite par une règle $A \rightarrow A \vee P$;
- une règle négative $A \rightarrow^- P$ est traduite par une règle $A \rightarrow A \wedge P$.

Théorème 3.10.
Le calcul des séquences avec dépliage polarisé dans $\mathcal{R}$ est équivalent au calcul des séquences avec dépliage dans $\mathcal{R}^{\mp}$.

Démonstration. Par induction sur la structure des démonstrations. Seules les règles de dépliage sont intéressantes.

Si on a une démonstration avec à la racine une règle de dépliage polarisé, par exemple

$$\uparrow\vdash \frac{\overset{\pi}{\Gamma, A, P \vdash \Delta}}{\Gamma, A \vdash \Delta}\, A \underset{B \rightarrow^- Q}{\longrightarrow} P$$

alors en appliquant l'hypothèse d'induction on a une démonstration de $\Gamma, A, P \vdash \Delta$ dans le calcul des séquences avec dépliage non polarisé. En appliquant $\wedge\vdash$ et en utilisant le lemme 2.3 on a une démonstration de $\Gamma, A, A \wedge P \vdash \Delta$. Comme $A \underset{B \rightarrow B \wedge Q}{\longrightarrow} A \wedge P$, on peut appliquer $\uparrow\vdash$ pour obtenir une démonstration de $\Gamma, A \vdash \Delta$.

Réciproquement, si on a une démonstration avec une règle de dépliage non polarisé, il y a deux cas : soit la règle de réécriture est appliqué du même côté que la règle polarisée qui l'a engendrée, soit elle est appliquée de l'autre côté. Dans le premier cas, on a par exemple

$$\uparrow\vdash \frac{\overset{\pi}{\Gamma, A, A \wedge P \vdash \Delta}}{\Gamma, A \vdash \Delta}\, A \underset{B \rightarrow B \wedge Q}{\longrightarrow} A \wedge P$$

Si on applique l'hypothèse d'induction sur π on obtient une démonstration de $\Gamma, A, A \wedge P \vdash \Delta$ dans le calcul des séquences avec dépliage polarisé. En appliquant les lemmes 2.5 et 2.6, on obtient une démonstration de $\Gamma, A, P \vdash \Delta$. Comme $A \underset{B \rightarrow^- Q}{\longrightarrow} P$ on peut appliquer $\uparrow\vdash$ et obtenir une démonstration de $\Gamma, A \vdash \Delta$. Dans le deuxième cas, on a par exemple

$$\uparrow\vdash \frac{\overset{\pi}{\Gamma, A, A \vee P \vdash \Delta}}{\Gamma, A \vdash \Delta}\, A \underset{B \rightarrow B \vee Q}{\longrightarrow} A \vee P$$

Par hypothèse d'induction sur π on a une démonstration de $\Gamma, A, A \vee P \vdash \Delta$. En utilisant les lemmes 2.5 et 2.6 on a une démonstration de $\Gamma, A \vdash \Delta$. □

Corollaire 3.11. *Le calcul des séquences polarisé modulo $\mathcal{R}$ est équivalent au calcul des séquences asymétrique modulo $\mathcal{R}^{\mp}$.*

Remarque 3.4 *:* Cette démonstration est également possible en logique intuitionniste, en se plaçant dans LB étendu avec des règles de dépliage. En effet les lemmes 2.3 et 2.6 (plus précisément 2.8) y sont valides, et les règles $\wedge\vdash$, $\vdash\wedge$, $\vee\vdash$ et $\vdash\vee$ sont inversibles, ce qui suffit.

Ceci répond plus ou moins à une question posée par Gilles Dowek (2002, fin de la section 4), qui demandait quels systèmes polarisés peuvent être représentés par des systèmes non polarisés. Néanmoins, il faut bien remarquer que le système non polarisé obtenu peut ne pas être confluent, ce qui veut dire qu'on ne sera pas équivalent avec la version originale du calcul des séquences modulo de Gilles Dowek et collaborateurs (2003), en tout cas en ce qui concerne l'admissibilité des coupures.

3.2.2 Sémantique

Nous aurons besoin, pour démontrer l'indécidabilité de l'admissibilité de $\vdash\!\!\!\smile$, de certaines notions sémantiques introduites par Olivier Hermant (2005).

Définition 3.7 (Modèle pour un système de réécriture)**.** *Étant donné un système de réécriture $\mathcal{R}$, on dit qu'un modèle pour la logique classique du premier ordre m est un modèle pour $\mathcal{R}$ si c'est un modèle d'une présentation compatible, ce qui revient à dire que pour toutes formules closes P et Q telles que $P \xleftrightarrow[\mathcal{R}]{*} Q$ l'interprétation de P et de Q est la même.*

Définition 3.8 (Modèle complet sans coupure)**.** *Un modèle m pour $\mathcal{R}$ d'une présentation Γ est dit* complet sans coupure *si pour toute formule P, $m \models P$ implique $\Gamma \vdash P$ est démontrable sans coupure dans le calcul des séquences modulo $\mathcal{R}$.*

Définition 3.9. *Une présentation Γ est dite* cohérente sans coupure *pour $\mathcal{R}$ si la logique engendrée par Γ à partir de la logique associée au calcul des séquences modulo $\mathcal{R}$ sans $\vdash\!\!\!\smile$ est cohérente.*

Ceci est équivalent à ce que $\Gamma \vdash$ ne soit pas démontrable sans coupure dans le calcul des séquences modulo $\mathcal{R}$.

On ajoute également une notion qui servira de critère à l'admissibilité des coupures.

Définition 3.10. *On dit qu'un système de réécriture est complet sans coupure si pour toute présentation Γ cohérente sans coupure pour ce système il existe un modèle pour $\mathcal{R}$ de Γ complet sans coupure.*

Pour démontrer qu'un système de réécriture est complet sans coupure, il faut en principe trouver un modèle pour toute présentation cohérente sans coupure. On peut en fait se restreindre à des présentations ayant des bonnes propriétés.

Définition 3.11. *Une présentation* Γ *cohérente sans coupure (pour* $\mathcal{R}$*) est complète sans coupure pour* $\mathcal{R}$ *si pour toute formule* P*, soit* $P \in \Gamma$ *soit* Γ, P *n'est pas cohérente sans coupures (c'est-à-dire* $\Gamma, P \vdash$ *est démontrable modulo* $\mathcal{R}$ *sans coupure).*

Définition 3.12. *Une présentation* Γ *cohérente sans coupure (pour* $\mathcal{R}$*) admet des témoins de* Henkin *pour* $\mathcal{R}$ *si pour toute formule* P *avec une variable libre* x*, il existe un terme* t *tel que si* $\Gamma, \exists x,\ P$ *ou* $\Gamma, \neg\forall x,\ P$ *est cohérente sans coupure alors* $\{t/x\}P \in \Gamma$*.*

Lemme 3.12. *Un système de réécriture est complet sans coupure si pour toute présentation* Γ *cohérente, complète et admettant des témoins de* Henkin *sans coupure pour ce système il existe un modèle pour* $\mathcal{R}$ *de* Γ *complet sans coupure.*

Démonstration. Olivier Hermant (2005) montre qu'on peut compléter toute présentation cohérente sans coupure Γ en une présentation Δ cohérente, complète et admettant des témoins de Henkin, tel que $m \models \Delta$ implique $m \models \Gamma$. Informellement, il s'agit de rajouter au fur et à mesure et arbitrairement des formules à Γ sans casser sa cohérence. □

3.2.3 Admissibilité des coupures en déduction modulo

Nous nous intéressons ici à l'admissibilité de la règle $\vdash_{\smile}$ dans le calcul des séquences LK modulo un système de réécriture. Gilles Dowek et collaborateurs (2003) ont montré qu'en général $\vdash_{\smile}$ n'est pas admissible. L'exemple qu'ils donnent est inspiré d'un manuscrit de Marcel Crabbé (1974) qui démontre la non-normalisation de la déduction naturelle si on y rajoute des règles d'inférence correspondant au schéma de compréhension restreinte (2.2) :

$$\vdash_{\in} \frac{\Gamma \vdash t \in A \qquad \Gamma \vdash X(t)}{\Gamma \vdash t \in \{x \in A :\ X(x)\}} \qquad\qquad \in\vdash \frac{\Gamma \vdash t \in \{x \in A :\ X(x)\}}{\Gamma \vdash X(t)}$$

On note $x \notin y$ pour $\neg(x \in y)$ et on pose $r_a \stackrel{\text{déf}}{=} \{x \in a :\ x \notin x\}$. On a alors une démonstration π de $r_a \in a \vdash r_a \notin r_a$

$$\vdash_{\neg} \frac{\vdash_{\in} \dfrac{\dfrac{}{r_a \in r_a, r_a \in a \vdash r_a \in r_a}}{r_a \in r_a, r_a \in a \vdash r_a \notin r_a} \qquad \vdash \dfrac{}{r_a \in r_a, r_a \in a \vdash r_a \in r_a}}{r_a \in r_a, r_a \in a \vdash \bot}}{r_a \in a \vdash r_a \notin r_a}$$

et donc une démonstration π' de $r_a \notin a$:

$$\vdash_{\neg} \frac{\pi \quad r_a \in a \vdash r_a \notin r_a \qquad \vdash_{\in} \dfrac{}{r_a \in a \vdash r_a \in a} \qquad \pi \quad r_a \in a \vdash r_a \notin r_a}{r_a \in a \vdash r_a \in r_a}}{r_a \in a \vdash \bot}}{\vdash r_a \notin a}$$

On remarque que dans cette démonstration il y a un détour au niveau du π à gauche ($\vdash_\neg$ suivi de $\neg\vdash$). Si on élimine cette coupure, on obtient la démonstration

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{r_a \in a \vdash r_a \in a}\quad \pi}{r_a \in a \vdash r_a \in r_a}\quad r_a \in a \vdash r_a \notin r_a}{r_a \in a \vdash r_a \notin r_a}\in\vdash \quad \dfrac{\dfrac{}{r_a \in a \vdash r_a \in a}\quad \pi}{r_a \in a \vdash r_a \in r_a}\vdash\in\quad r_a \in a \vdash r_a \notin r_a}{r_a \in a \vdash \bot}\neg\vdash}{\vdash r_a \notin a}\vdash\neg$$

Ici on a un détour avec $\vdash_\in$ suivi de $\in\vdash$. Pour éliminer ce détour on considère la réduction

$$\dfrac{\dfrac{\dfrac{\vdots\quad \pi}{\Gamma \vdash t \in A \quad \Gamma \vdash X(t)}\vdash\in}{\Gamma \vdash t \in \{x \in A \,:\, X(t)\}}\in\vdash}{\Gamma \vdash X(t)} \quad\rightarrow\quad \dfrac{\pi}{\Gamma \vdash X(t)}$$

Si on applique cette réduction à la démonstration ci-dessus, on retombe sur π'. Le système avec les règles d'inférence pour le schéma de compréhension et la réduction de démonstration associée ne normalise donc pas.

On peut adapter cet exemple en déduction modulo, en prenant comme système de réécriture $r_a \in r_a \rightarrow r_a \in a \wedge r_a \notin r_a$ ou plus simplement le système $\mathcal{C}rab : A \rightarrow B \wedge \neg A$, en assimilant $r_a \in r_a$ à A et $r_a \in a$ à B. On peut alors traduire π' en déduction naturelle modulo $\mathcal{C}rab$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B, A \vdash A}}{B, A \vdash \neg A}\vdash\wedge\neg \quad \dfrac{}{B, A \vdash A}\vdash\wedge}{B, A \vdash \bot}\vdash\neg}{B \vdash \neg A}\neg\vdash \quad \dfrac{\dfrac{}{B \vdash B}\quad \dfrac{\dfrac{\dfrac{}{B, A \vdash A}}{B, A \vdash \neg A}\vdash\wedge\neg \quad \dfrac{}{B, A \vdash A}\vdash\wedge}{B, A \vdash \bot}\vdash\neg}{B \vdash \neg A}\vdash\wedge}{B \vdash A}\neg\vdash}{B \vdash \bot}\vdash\neg}{\vdash \neg B}\vdash\neg$$

Le λ-terme associé à cette démonstration, $\lambda y,\ (\lambda x,\ (\pi_2 x)\ x)(y, \lambda x,\ (\pi_2 x)\ x)$, ne normalise pas, comme attendu.

Il est bien entendu possible de traduire cette démonstration en calcul des séquences (intuitionniste ou classique) modulo $\mathcal{C}rab$, et dans ce cas la procédure d'élimination des coupures ne termine pas. De plus, la règle $\vdash_\smile$ n'est alors plus admissible : on a vu qu'il existe une démonstration de $\neg B$, mais il n'en existe pas sans la règle $\vdash_\smile$. En effet, seule cette règle peut s'appliquer à la séquence $B \vdash$.

On voit qu'on a un système qui n'admet pas $\vdash_\smile$ et qui n'est pas normalisant. Toutefois ce système est cohérent : toute formule (en particulier B) n'est pas démontrable. En

TAB. 3.1 : Contre-exemples à des propriétés du calcul des séquences modulo. Pour une ligne donnée, l'ensemble des propriétés au-dessus sont vérifiées. La formule $y \approx z$ est une abréviation pour la formule $\forall x, y \in x \Rightarrow z \in x$.

Propriété	Contre-exemple minimal	Contre-exemple convergent
Cohérence	$A \to A \Rightarrow \bot$	$\mathsf{r} \in \mathsf{r} \to \forall y,\ y \approx \mathsf{r} \Rightarrow y \in \mathsf{r} \Rightarrow \bot$ (Gilles DOWEK et Benjamin WERNER, 2003)
Admissibilité de $\vdash\!\!\!\smile$	$A \to A \Rightarrow B$	$\mathsf{r} \in \mathsf{r} \to \forall y,\ y \approx \mathsf{r} \Rightarrow y \in \mathsf{r} \Rightarrow B$ (Olivier HERMANT, 2005)
Normalisation	$A \to A \Rightarrow A$	$\mathsf{r} \in \mathsf{r} \to \forall y,\ y \approx \mathsf{r} \Rightarrow y \in \mathsf{r} \Rightarrow A \Rightarrow A$ (Olivier HERMANT, 2005)

effet la présentation compatible $A \Leftrightarrow B \wedge \neg A$ est satisfaisable : elle possède par exemple le modèle de support $\{1\}$ avec $\hat{A} \stackrel{\text{déf}}{=} \emptyset$ et $\hat{B} \stackrel{\text{déf}}{=} \emptyset$.

On peut également avoir des systèmes qui admettent $\vdash\!\!\!\smile$, mais qui ne normalisent pas, dans le sens où les λ-termes associés aux démonstrations en déduction naturelle modulo ces systèmes ne sont pas normalisant. En particulier, le système $A \to A \Rightarrow A$ permet de donner pour tout λ-terme une démonstration de A correspondant à ce terme. On peut montrer par des méthodes sémantiques qu'il admet $\vdash\!\!\!\smile$ (Olivier HERMANT, 2005).

Il est alors naturel de chercher quels critères permettent de s'assurer qu'un système admet $\vdash\!\!\!\smile$, ou normalise. Par abus de langage, on dira qu'un système de réécriture admet $\vdash\!\!\!\smile$ si le calcul des séquences modulo $\mathcal{R}$ l'admet. Le premier critère qui vient à l'esprit est de se restreindre à des systèmes convergents, c'est-à-dire confluents et terminants. Néanmoins, il est possible de trouver de tels systèmes qui ne sont pas cohérents, qui n'admettent pas $\vdash\!\!\!\smile$, ou qui ne normalisent pas. L'idée est de rendre les systèmes vus précédemment terminants en bloquant la réduction à l'aide d'un quantificateur. Ainsi, au lieu de $A \to B \wedge \neg A$ on considère $A(\mathsf{r}) \to \forall y,\ y = \mathsf{r} \Rightarrow B \wedge \neg A(y)$ où r est une constante. Dans le cas où on n'a pas de prédicat d'égalité, on peut s'en sortir en utilisant un prédicat binaire : $A(\mathsf{r}, \mathsf{r}) \to \forall y,\ (\forall x,\ A(y, x) \Rightarrow A(\mathsf{r}, x)) \Rightarrow B \wedge \neg A(y, \mathsf{r})$. On trouvera dans le tableau 3.1 un résumé de tel systèmes.

Divers critères assurant l'admissibilité de $\vdash\!\!\!\smile$ ont été proposés. Dans le cas où seules la réécriture sur les termes est considérée, Gilles DOWEK (2003) a démontré que l'admissibilité des coupures dans le calcul des séquences modulo $\mathcal{R}$ asymétrique est équivalente à la confluence de $\mathcal{R}$.

Si on se replace dans le cas où on autorise des règles de réécriture propositionnelles, alors les contre-exemples que nous avons donnés ci-dessus montrent que la confluence n'est pas un critère suffisant. Dans la suite on ne considère que des systèmes confluents.

On dispose alors de plusieurs critères, que ce soit pour l'admissibilité des coupures ou la normalisation :

Compatibilité avec l'ordre sous-formule. Si on suppose qu'il existe un ordre bien fondé $\sqsupset$ incluant à la fois l'ordre sous-formule et la relation de réécriture $\underset{\mathcal{R}}{\longrightarrow}$, alors Olivier Hermant (2005) a montré l'admissibilité de $\vdash\!\!\!\smile$. Nous allons montrer dans la section 4.3.3 que ce critère assure en fait la terminaison de la procédure d'élimination des coupures. Gilles Dowek et Benjamin Werner (2003) ont montré que ce critère implique la normalisation forte des démonstrations en déduction naturelle pour la logique propositionnelle, et leur résultat peut s'étendre pour la logique des prédicats du premier ordre.

Positivité. Si on suppose que toutes les occurrences d'un atome dans les membres droits des règles sont positives, alors Olivier Hermant (2005) a montré l'admissibilité de $\vdash\!\!\!\smile$.

Combinaison de ces deux critères. Si on suppose que le système de réécriture peut se diviser en deux parties $\mathcal{R}_\sqsupset$ et R_+, telles que $R_\sqsupset$ satisfait le critère de compatibilité avec l'ordre sous-formule, R_+ le critère de positivité, et les instances des membres droits de R_+ par des termes non réductibles par $R_\sqsupset$ ne sont pas réductible par $R_\sqsupset$, alors Olivier Hermant (2005) a montré l'admissibilité de $\vdash\!\!\!\smile$.

Stratification. Dans un cadre proche, Peter Schroeder-Heister (1990) a défini un critère lié à des niveaux de formules. Si on donne à chaque prédicat A un niveau $nv(A)$ alors on peut définir le niveau d'une formule par induction structurelle :

- $nv(A(t_1, \ldots, t_n)) \stackrel{\text{déf}}{=} nv(A)$;
- $nv(\bot) \stackrel{\text{déf}}{=} nv(\top) \stackrel{\text{déf}}{=} 0$;
- $nv(\neg P) \stackrel{\text{déf}}{=} nv(P) + 1$;
- $nv(P \wedge Q) \stackrel{\text{déf}}{=} nv(P \vee Q) \stackrel{\text{déf}}{=} \max(nv(P), nv(Q))$;
- $nv(P \Rightarrow Q) \stackrel{\text{déf}}{=} \max(nv(P) + 1, nv(Q))$;
- $nv(\forall x,\ P) \stackrel{\text{déf}}{=} nv(\exists x,\ P) \stackrel{\text{déf}}{=} nv(P)$.

On dit alors que le système de réécriture est *stratifié* si les membres gauches des règles propositionnelles sont plus grands que les membres droits. Ce critère assure la terminaison de la procédure d'élimination des coupure dans le système de Peter Schroeder-Heister (1990). C'est aussi le cas en déduction modulo, puisque ce critère est en fait moins général que le critère de compatibilité avec l'ordre sous-formule : si on a un système stratifié, on peut définir l'ordre $P > Q$ si $nv(P) > nv(Q)$ ou ($nv(P) = nv(Q)$ et Q est une sous-formule de P), qui vérifie le critère.

Sémantique. Pour démontrer ses critères, Olivier Hermant (2005) utilise des notions sémantiques. Il donne deux démonstrations différentes, mais dont une partie est commune et est même utilisée dans la démonstration de l'admissibilité de $\leftharpoondown$ pour $\mathsf{r} \in \mathsf{r} \to \forall y,\ y \approx \mathsf{r} \Rightarrow y \in \mathsf{r} \Rightarrow A \Rightarrow A$. Nous allons donc factoriser cette partie, ce qui va donner un critère sémantique basé sur la notion de complétude sans coupure introduite à la section 3.2.2. Comme cette factorisation est nouvelle par rapport à Olivier Hermant (2005), nous détaillons la démonstration du critère.

Proposition 3.13. *Si un système de réécriture est complet sans coupure, alors il admet les coupures.*

Démonstration. Olivier Hermant (2005), par exemple, a démontré la correction du calcul des séquences modulo $\mathcal{R}$ vis-à-vis des modèles pour $\mathcal{R}$, c'est-à-dire que si $\Gamma \vdash \Delta$ est démontrable dans le calcul des séquences modulo $\mathcal{R}$ alors pour tout modèle m pour $\mathcal{R}$ de Γ, on a $m \models \bigvee \Delta$.

Soit une séquence $\Gamma \vdash \Delta$ démontrable (potentiellement avec des coupures) dans le calcul des séquences modulo $\mathcal{R}$. Par correction, pour tout modèle m pour $\mathcal{R}$ de Γ, on a $m \models \bigvee \Delta$. On distingue deux cas :

- Soit Γ n'est pas cohérente sans coupure pour $\mathcal{R}$, ce qui veut dire que $\Gamma \vdash$ est démontrable sans coupures dans le calcul des séquences modulo $\mathcal{R}$. Par affaiblissement, $\Gamma \vdash \Delta$ est également démontrable sans coupure.
- Soit Γ est cohérente sans coupure, et comme $\mathcal{R}$ est complet sans coupure, il existe un modèle m pour $\mathcal{R}$ de Γ tel que $m \models P$ implique $\Gamma \vdash P$ démontrable sans coupure. Comme $m \models \bigvee \Delta$, on peut démontrer $\Gamma \vdash \bigvee \Delta$ sans coupure, et par inversibilité de $\vdash\vee$ on peut démontrer $\Gamma \vdash \Delta$ sans coupure.

□

On peut alors montrer que les critères de positivité ou de compatibilité avec l'ordre sous-formule permettent de construire des modèles complets sans coupure pour toute présentation cohérente, complète et admettant des témoins de Henkin, ce qui permet de conclure avec le lemme 3.12 et la proposition précédente.

Pré-modèle. (Gilles Dowek et Benjamin Werner, 2003) Un *pré-modèle* est comme un modèle de la logique du premier ordre sauf que les formules sont interprétées par un candidat de réductibilité (Jean-Yves Girard, 1972) plutôt qu'une valeur de vérité. Un pré-modèle de $\mathcal{R}$ est tel que les formules équivalentes pour $\xleftrightarrow[\mathcal{R}]{*}$ ont la même interprétation. Si $\mathcal{R}$ possède un pré-modèle, alors la déduction naturelle modulo $\mathcal{R}$ normalise.

Supercohérence. (Gilles Dowek, 2006) Il s'agit d'une généralisation des pré-modèles : on interprète les formules dans ce qu'on appelle une *algèbre ordonnée de valeurs de vérité*. S'il existe un modèle de $\mathcal{R}$ pour toute algèbre ordonnée de valeurs de vérité, alors la déduction naturelle modulo $\mathcal{R}$ normalise.

Indécidabilité de l'admissibilité de la coupure

Dans tous le cas ces critères ne sont pas complets, dans le sens où il existe des systèmes de réécriture qui ne vérifient pas les critères mais vérifie la propriété qu'ils assurent. Notons toutefois que Denis Cousineau travaille à l'élaboration d'un critère sémantique correct et complet pour la normalisation forte. On aimerait aussi trouver un critère calculable qui assure de façon complète l'admissibilité de $\vdash\!\!\!\smile$. Malheureusement nous montrons que ce n'est pas possible, car il est indécidable de savoir si le calcul des séquences modulo un système de réécriture admet $\vdash\!\!\!\smile$.

Théorème 3.14.
Le problème qui prend en entrée un système de réécriture et qui dit si ce système admet les coupures est indécidable.

Démonstration. On le réduit au problème de la validité en logique du premier ordre, qui est indécidable dès qu'on a un prédicat binaire dans le langage.

Soit P une formule du premier ordre, et soit $\mathcal{R}_P$ le système $A(\mathsf{r},\mathsf{r}) \to \forall y,\ (\forall x,\ A(y,x) \Rightarrow A(\mathsf{r},x)) \Rightarrow A(y,\mathsf{r}) \Rightarrow P$ où r est une constante et A est un symbole de prédicat binaire n'apparaissant pas dans P. On montre que P est valide ssi $\mathcal{R}_P$ admet les coupures.

Si P est valide, alors montrons que $\mathcal{R}_P$ est complet sans coupures. Pour cela, utilisons le lemme 3.12. Soit Γ une présentation cohérente, complète et admettant des témoins de Henkin pour $\mathcal{R}_P$. On définit le modèle m suivant : Son support est l'ensemble des termes clos. Un symbole de prédicat A n-aire est interprété comme l'ensemble des n-uplets de termes clos $t_1,\dots,t_n$ tels que $\Gamma \vdash A(t_1,\dots,t_n)$ est démontrable sans coupure modulo $\mathcal{R}_P$. On montre que ceci définit bien un modèle de Γ par induction sur les formules, en utilisant le fait que Γ est complète et admet des témoins de Henkin. Montrons que c'est un modèle pour $\mathcal{R}$. P est supposé valide, par conséquent il est démontrable dans le calcul des séquences non modulo sans coupure, par complétude de LK pour la logique du premier ordre et par admissibilité de $\vdash\!\!\!\smile$ dans LK. Par conséquent $m \models P$. On a donc $m \models \forall y,\ (\forall x,\ A(y,x) \Rightarrow A(\mathsf{r},x)) \Rightarrow A(y,\mathsf{r}) \Rightarrow P$. Par ailleurs, $A(\mathsf{r},\mathsf{r})$ est démontrable sans coupure modulo $\mathcal{R}_P$ avec

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A(y,\mathsf{r}) \vdash A(y,\mathsf{r}), A(\mathsf{r},\mathsf{r}), P}\,\vdash\!\!\!\frown \quad \dfrac{}{A(\mathsf{r},\mathsf{r}), A(y,\mathsf{r}) \vdash A(\mathsf{r},\mathsf{r}), P}\,\vdash\!\!\!\frown}{A(y,\mathsf{r}) \Rightarrow A(\mathsf{r},\mathsf{r}), A(y,\mathsf{r}) \vdash A(\mathsf{r},\mathsf{r}), P}\,\Rightarrow\!\vdash}{\forall x,\ A(y,x) \Rightarrow A(\mathsf{r},x), A(y,\mathsf{r}) \vdash A(\mathsf{r},\mathsf{r}), P}\,\forall\!\vdash}{\vdash A(\mathsf{r},\mathsf{r}), (\forall x,\ A(y,x) \Rightarrow A(\mathsf{r},x)) \Rightarrow A(y,\mathsf{r}) \Rightarrow P}\,\vdash\!\Rightarrow}{\vdash A(\mathsf{r},\mathsf{r}), \forall y,\ (\forall x,\ A(y,x) \Rightarrow A(\mathsf{r},x)) \Rightarrow A(y,\mathsf{r}) \Rightarrow P}\,\vdash\!\forall}{\vdash A(\mathsf{r},\mathsf{r})}\,\vdash\!\!\cdot\!\cdot$$

Donc $m \models A(\mathsf{r},\mathsf{r})$ et m est un modèle pour $\mathcal{R}$. Montrons que m est complet sans coupure. Supposons que $m \models Q$. Si Q est atomique, alors $\Gamma \vdash Q$ est démontrable sans coupure modulo $\mathcal{R}_P$ par définition de m. Sinon, on procède par induction sur Q, et on a besoin

du fait que Γ est complète et admet des témoins de Henkin. Donc $\mathcal{R}_P$ est complet sans coupure, et le critère sémantique nous dit qu'il admet les coupures.

Réciproquement, supposons que $\mathcal{R}_P$ admet les coupures. Montrons d'abord que P est démontrable modulo $\mathcal{R}_P$. On a vu que $\vdash A(\mathsf{r},\mathsf{r})$ l'est, or $A(\mathsf{r},\mathsf{r}) \vdash P$ l'est aussi :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A(\mathsf{r},\mathsf{r}), A(\mathsf{r},x) \vdash P, A(\mathsf{r},x)}\,\vdash\frown}{A(\mathsf{r},\mathsf{r}) \vdash P, A(\mathsf{r},x) \Rightarrow A(\mathsf{r},x)}\,\vdash\Rightarrow}{A(\mathsf{r},\mathsf{r}) \vdash P, \forall x,\ A(\mathsf{r},x) \Rightarrow A(\mathsf{r},x)}\,\vdash\forall \qquad \Rightarrow\vdash\dfrac{\dfrac{}{A(\mathsf{r},\mathsf{r}) \vdash P, A(\mathsf{r},\mathsf{r})}\,\frown\vdash \qquad \dfrac{}{A(\mathsf{r},\mathsf{r}), P \vdash P}\,\frown\vdash}{A(\mathsf{r},\mathsf{r}), A(\mathsf{r},\mathsf{r}) \Rightarrow P \vdash P}}{A(\mathsf{r},\mathsf{r}), (\forall x,\ A(\mathsf{r},x) \Rightarrow A(\mathsf{r},x)) \Rightarrow A(\mathsf{r},\mathsf{r}) \Rightarrow P \vdash P}\,\Rightarrow\vdash}{A(\mathsf{r},\mathsf{r}), \forall y,\ (\forall x,\ A(\mathsf{r},x) \Rightarrow A(\mathsf{r},x)) \Rightarrow A(\mathsf{r},\mathsf{r}) \Rightarrow P \vdash P}\,\forall\vdash}{A(\mathsf{r},\mathsf{r}) \vdash P}\,\therefore\vdash$$

En appliquant $\therefore\vdash$, P est démontrable dans le calcul des séquences modulo $\mathcal{R}_P$. Par hypothèse, il l'est sans utiliser $\therefore\vdash$. Mais une démonstration de P sans $\therefore\vdash$ n'applique aucune règle d'inférence modulo $\mathcal{R}_P$: en effet à aucun moment A ne peut apparaître dans une telle démonstration. Par conséquent P est démontrable dans le calcul des séquences sans modulo, et comme ce dernier est correct pour la logique du premier ordre, P est valide pour la logique du premier ordre. □

Remarque 3.5 *:* La même démonstration fonctionne en posant $\mathcal{R}_P \overset{\text{déf}}{=} A \to A \Rightarrow P$. Avec le système que nous proposons, on montre que l'admissibilité des coupures est indécidable y compris quand on sait que le système de réécriture est convergent (ce qui est déjà une propriété indécidable).

3.2.4 Méthodes de recherches de démonstration basées sur la déduction modulo

La déduction modulo a, entre autres buts, été introduite pour permettre de développer des méthodes de recherche de démonstration intégrant efficacement des raisonnements calculatoires. Dès l'origine, Gilles Dowek et collaborateurs (2003) ont définie une extension de la méthode de résolution de Allan Robinson (1965) appelée ENAR (*extended narrowing and resolution*). Olivier Hermant (2005) a montré que cette méthode est correcte et complète pour le calcul des séquences modulo sans $\therefore\vdash$. Autrement dit, on peut dériver la clause vide d'une ensemble de clauses C dans la méthode ENAR pour le système de réécriture $\mathcal{R}$ ssi $C \vdash$ est démontrable dans le calcul des séquences modulo $\mathcal{R}$ sans $\therefore\vdash$. On a donc la complétude de la méthode ssi $\mathcal{R}$ admet les coupures.

Une extension de la méthode des tableaux (appelée TaMed) a également été proposée par Richard Bonichon (2004); Richard Bonichon et Olivier Hermant (2006*b*), y compris pour la logique intuitionniste (Richard Bonichon et Olivier Hermant, 2006*a*). Ici aussi la méthode est correcte et complète pour le calcul des séquences modulo sans $\therefore\vdash$.

Nous allons détailler la version proposée par Richard Bonichon et Olivier Hermant (2006*b*) car nous l'utiliserons dans une implémentation par la suite, .

Du fait de la dualité des connecteurs et quantificateurs en logique classique, il y en a essentiellement quatre types : $\alpha(P,Q)$ correspond à $P \vee Q$ à droite des séquences, $\neg P \wedge \neg Q$ à gauche ou $(\neg P) \Rightarrow Q$ à droite ; $\beta(P,Q)$ correspond à $P \wedge Q$ à droite des séquences, $\neg P \vee \neg Q$ à gauche ou $P \Rightarrow \neg Q$ à gauche ; $\gamma(x,P)$ correspond à $\exists x,\ P$ à droite ou $\forall x,\ \neg P$ à gauche ; enfin $\delta(x,P)$ correspond à $\forall x,\ P$ à droite ou $\exists x,\ \neg P$ à gauche. On utilise des méta-variables pour les termes introduits par γ car on ne sait pas encore par quoi les instancier. Chaque formule est annotée par un ensemble de méta-variables qui surapproximera l'ensemble des méta-variables libres de la formule, et un ensemble d'égalités entre termes appelées contraintes. Un *tableau* est alors un multiensemble de *branches*, qui sont des multiensembles de formules ainsi annotées. Les règles de TaMed transforme un tableau jusqu'à ce qu'on obtienne le multiensemble vide avec un ensemble de contraintes satisfaisables.

$$\alpha\ \frac{\Gamma_1, \alpha(P,Q)^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}{\Gamma_1, \alpha(P,Q)^l_c, P^l_c, Q^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}$$

$$\beta\ \frac{\Gamma_1, \beta(P,Q)^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}{\Gamma_1, \beta(P,Q)^l_c, P^l_c | \Gamma_1, \beta(P,Q)^l_c, Q^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}$$

$$\gamma\ \frac{\Gamma_1, \gamma(x,P)^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}{\Gamma_1, \gamma(x,P)^l_c, \{X/x\}P^{l\cup\{X\}}_c | \ldots | \Gamma_n \cdot \mathcal{C}}$$

$$\delta\ \frac{\Gamma_1, \delta(x,P)^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}{\Gamma_1, \delta(x,P)^l_c, \{\mathrm{sko}(l)/x\}P^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}$$

$$\mathsf{rw}\ \frac{\Gamma_1, P^l_c | \ldots | \Gamma_n \cdot \mathcal{C}}{\Gamma_1, P^l_c, (P[d]_{\mathfrak{p}})^l_k | \ldots | \Gamma_n \cdot \mathcal{C}}\ g \to d \in \mathcal{R} \text{ et } k = c \cup \{P_{|\mathfrak{p}} =^? g\}$$

$$\mathsf{closure}\ \frac{\Gamma_1, P^{l_1}_{c_1}, \neg P^{l_2}_{c_2} | \Gamma_2 | \ldots | \Gamma_n \cdot \mathcal{C}}{\Gamma_2 | \ldots | \Gamma_n \cdot \mathcal{C} \cup c_1 \cup c_2 \cup \{P^{l_1} =^? P^{l_2}\}}$$

On voit que α vue de haut en bas correspond à $\wedge\vdash$, $\vdash\vee$ et $\vdash\Rightarrow$ vues de bas en haut ; β à $\vee\vdash$, $\vdash\wedge$ et $\Rightarrow\vdash$; γ à $\forall\vdash$ et $\vdash\exists$ avec introduction d'une méta-variable fraîche X ; δ à $\exists\vdash$ et $\vdash\forall$ avec introduction d'un symbole de Skolem frais $\mathrm{sko}(l)$ basé sur les méta-variables dans l ; rw à une étape de conversion avec surréduction (les méta-variables sont unifiées et non pas seulement filtrées) ; closure correspond à $\frown$ avec unification globale au tableau des méta-variables.

Richard Bonichon et Olivier Hermant (2006*b*) proposent alors une stratégie d'application de ces règles et montrent par un argument sémantique qu'elle est complète par

rapport au calcul des séquences modulo $\mathcal{R}$ sans coupure. La stratégie est la suivante :

- Une règle α, β ou δ ne sera appliquée qu'au plus une fois sur une formule donnée (ceci traduit l'inversibilité des règles correspondantes) ;
- Une règle rw ne sera appliquée qu'au plus une fois par règle de réécriture pour une formule donnée ;
- Après chaque étape on cherche à fermer le tableau en entier par closure. Si ce n'est pas possible, mais que certaines branches peuvent être fermées sans unification, on les retire ;
- À une étape donnée, on choisit la branche contenant le moins de formule. Si plusieurs branches ont la même taille minimale, on choisit celle la plus à gauche. Dans cette branche, on essaie d'appliquer en priorité α, puis δ, β et alternativement γ ou rw. Une règle γ est appliquée à la fois à toutes les formules de type γ dans la branche. Si aucune règle n'est applicable, la formule en entrée n'est pas valide.

3.3 Surdéduction

Une autre façon d'enrichir l'isomorphisme de Curry-Howard est de considérer une extension du λ-calcul comme langage des termes de démonstration. Benjamin Wack (2005) s'est par exemple intéressé à l'extension de l'isomorphisme pour un sous-ensemble du ρ-calcul de Horatiu Cirstea et Claude Kirchner (2001). Le ρ-calcul, ou calcul de réécriture, est un cadre général permet d'incorporer le mécanisme de filtrage de motif (*pattern matching*), propre à la réécriture, dans le λ-calcul. On considère la syntaxe suivante :

$$t ::= x \mid \lambda P,\ t \mid t\ u$$
$$u ::= t \mid f(u_1, \ldots, u_n)$$

où f est un symbole de fonction d'arité n et P est un *motif*. Pour la surdéduction, on peut se contenter de motifs dits algébriques suivant la syntaxe

$$P ::= x \mid f(P_1, \ldots, P_n)$$

où où f est un symbole de fonction d'arité n. La β-réduction est étendue en ρ-réduction

$$(\lambda P,\ t_1)t_2 \to \sigma t_1 \qquad \text{si } \sigma = \mathrm{Match}(P, t_2)$$

Dans notre cas, Match est une fonction partielle qui représente le filtrage de motif syntaxique, et peut être calculée par un algorithme simple travaillant sur des conjonctions

de problèmes de filtrage $t \ll s$:

$$\begin{aligned}
t \ll t \wedge E &\rightsquigarrow E \\
f(t_1,\ldots,t_n) \ll f(s_1,\ldots,s_n) \wedge E &\rightsquigarrow t_1 \ll s_1 \wedge \cdots \wedge t_n \ll s_n \wedge E \\
f(t_1,\ldots,t_n) \ll g(s_1,\ldots,s_m) \wedge E &\rightsquigarrow \bot && f \neq g \\
x \ll t \wedge x \ll s \wedge E &\rightsquigarrow \bot && t \neq s \\
f(t_1,\ldots,t_n) \ll x \wedge E &\rightsquigarrow \bot && x \in V.
\end{aligned}$$

À la fin on a soit une conjonction de problèmes de filtrage de la forme $x_1 \ll t_1 \wedge \cdots \wedge x_n \ll t_n$ pour $x_i \in V$, que l'on transforme en la substitution $\{t_1/x_1,\ldots,t_n/x_n\}$ qui est l'unique solution du problème de filtrage initial, soit $\bot$, auquel cas le problème de filtrage initial n'a pas de solution, Match n'est pas définie et la ρ-réduction ne peut pas être appliquée.

Avec un tel calcul on voit qu'on peut par exemple simuler les paires du λ-calcul et les projecteurs π_1 et π_2. On suppose qu'on a un symbole de fonction *paire* d'arité 2, alors (t_1,t_2) est encodé par $paire(t_1,t_2)$ et les projecteurs sont $\pi_1 \stackrel{\text{déf}}{=} \lambda paire(x_1,x_2),\ x_1$ et $\pi_2 \stackrel{\text{déf}}{=} \lambda paire(x_1,x_2),\ x_2$. On a bien alors $\pi_i(t_1,t_2) = (\lambda paire(x_1,x_2),\ x_i)paire(t_1,t_2) \underset{\rho}{\longrightarrow} t_i$. On peut également simuler l'abstraction sur les variables du premier ordre Λ, en considérant un symbole de fonction *terme* et en définissant $\Lambda a,\ t$ par $\lambda terme(a),\ t$ et $t@s$ par $t\ terme(s)$. Ici aussi, on a $(\Lambda a,\ t)@s = (\lambda terme(a),\ t)terme(s) \underset{\rho}{\longrightarrow} \{s/a\}t$. Dans la suite, pour simplifier, on utilisera le même λ pour abstraire sur les λ-termes ou sur les termes du premier ordre.

Horatiu Cirstea, Clément Houtmann et Benjamin Wack (2006) ont étendu le ρ-calcul pour permettre de reconnaître différents motifs en distinguant des cas, ce qu'ils appellent alors le ρ-calcul distributif. On ajoute alors un symbole de fonction associatif et commutatif $\wr$.

De la même façon que la β-réduction est séparée en deux étapes dans les λ-calculs avec substitution explicite, une étape Beta qui lance la réduction en créant une substitution explicite puis des réductions qui propage la substitution dans les sous-termes, la ρ-réduction peut alors être décomposée en trois étapes : une étape permettant de distribuer les applications dans les structures

$$(t_1 \wr \cdots \wr t_n)\ u \rightarrow_\delta (t_1\ u) \wr \cdots \wr (t_n\ u)$$

la ρ-réduction proprement dite, identique à la précédente ; et une étape qui permet de nettoyer les structures des termes qui ne pourront jamais se réduire

$$\begin{aligned}
(\lambda f(x_1,\ldots,x_n),\ t)\ g(u_1,\ldots,u_m) &\rightarrow_{\mathsf{stk}} \mathsf{stk} && f \text{ différent de } g \text{ ou } n \neq m \\
t \wr \mathsf{stk} &\rightarrow_{\mathsf{stk}} t \\
\mathsf{stk}\ t &\rightarrow_{\mathsf{stk}} \mathsf{stk}
\end{aligned}$$

où stk est une constante spéciale indiquant qu'un radical potentiel ne peut pas être réduit (de l'anglais *stuck*). On notera $\longrightarrow_{dis}$ la relation qui consiste à appliquer autant de $\rightarrow_\delta$ que nécessaire, appliquer une étape de ρ-réduction, et appliquer $\rightarrow_{\mathsf{stk}}$ autant que possible.

On peut alors donner un autre encodage des paires. Si on a deux constantes g et d, la paire (t_1, t_2) est encodée par $(\lambda \mathsf{g},\ t_1)\wr(\lambda \mathsf{d},\ t_2)$, les projections $\pi_1 t$ et $\pi_2 t$ par $t\ \mathsf{g}$ et $t\ \mathsf{d}$. Ici aussi la *dis*-réduction permet de simuler les réductions sur les paires.

Pour toute démonstration de la déduction naturelle restreinte aux connecteurs $\Rightarrow$, $\wedge$ et $\forall$ il existe donc un ρ-terme correspondant. Réciproquement, on va chercher quel système de déduction on peut mettre en bijection avec les ρ-termes.

Si on considère un ρ-terme de la forme $\lambda f(x, y),\ t$, on voit qu'on abstrait sur deux variables x et y, qui devrait donc correspondre à deux hypothèses déchargées dans la démonstration correspondante. On devrait donc avoir une règle d'inférence de la forme

$$\vdash_f \frac{\Gamma, x : A, y : B \vdash t : C}{\Gamma \vdash \lambda f(x, y),\ t : D}$$

Pour qu'une telle règle soit correcte, il faut que $(A \Rightarrow B \Rightarrow C) \Rightarrow D$ soit valide.

Réciproquement, si on a un terme de la forme $t\ f(u, v)$ on a une règle de la forme

$$\vdash^f \frac{\Gamma \vdash t : D' \qquad \Gamma \vdash u : A' \qquad \Gamma \vdash v : B'}{\Gamma \vdash t\ f(u, v) : C'}$$

Pour que la ρ-réduction donne une démonstration correcte, il faut que $A = A'$, $B = B'$, $C = C'$ et $D = D'$. Pour que la dernière règle d'inférence soit correcte, il faut que $D \Rightarrow A \Rightarrow B \Rightarrow C$ soit valide, soit au final $D \Leftrightarrow (A \Rightarrow B \Rightarrow C)$. On peut donc se placer dans le même cadre que la déduction modulo et que la déduction naturelle avec pliage–dépliage, en considérant la règle de réécriture $D \rightarrow A \Rightarrow B \Rightarrow C$. Dans la déduction naturelle avec pliage–dépliage, les règles supplémentaires seraient

$$\uparrow \frac{\Gamma \vdash t : A \Rightarrow B \Rightarrow C}{\Gamma \vdash \uparrow_f t : D} \qquad \downarrow \frac{\Gamma \vdash t : D}{\Gamma \vdash \downarrow_f t : A \Rightarrow B \Rightarrow C}$$

Pour obtenir les règles d'inférence correspondant aux ρ-termes, on voit qu'il faut appliquer toutes les règles d'introduction possible au-dessus de $\uparrow$ et toutes les règles d'élimination au-dessous de $\downarrow$. C'est l'idée à l'origine de la déduction surnaturelle de Benjamin Wack (2005).

3.3.1 Déduction surnaturelle

L'idée est donc de calculer des nouvelles règles d'inférence correspondant aux règles de réécriture propositionnelle. À chaque règle de réécriture propositionnelle on va associer une règle d'introduction et une ou plusieurs règles d'élimination.

Pour calculer la règle d'introduction associée à $A \to P$, on applique toutes les règles d'introduction possibles à la séquence $\Gamma \vdash P$ où Γ est une variable de contexte et n'est donc pas modifiée par l'application des règles. Les prémisses de la nouvelle règle sont des celles de la dérivation, sa conclusion est $\Gamma \vdash A$, et les substitutions valables sont celles de l'intersection des substitutions valables des règles de la dérivation.

Pour calculer les règles d'élimination, on applique les règles d'élimination à $\Gamma \vdash P$ en l'utilisant comme prémisse principale, en descendant donc de haut en bas.

Pour des phénomènes liés à des coupures implicites, nous sommes obligés de nous restreindre aux connecteurs $\Rightarrow$, $\forall$ et $\wedge$. On peut également utiliser $\bot$ et $\top$, ce que nous ne ferons pas ici.

Exemple 3.2 *:* Si on prend l'exemple de la règle de réécriture propositionnelle $Max(x,a) \to x \in a \wedge \forall y,\ y \in a \Rightarrow y \leq x$ qui permet de définir le fait que x soit le maximum de a, on obtient, pour le calcul de la règle d'introduction, la dérivation

$$\vdash_{\wedge} \frac{\Gamma \vdash x \in a \qquad \vdash_{\forall} \dfrac{\vdash_{\Rightarrow} \dfrac{\Gamma, z \in a \vdash z \leq x}{\Gamma \vdash z \in a \Rightarrow z \leq x}}{\Gamma \vdash \forall y,\ y \in a \Rightarrow y \leq x}\ z \text{ non libre dans } \Gamma, y \in a \Rightarrow y \leq x}{\Gamma \vdash x \in a \wedge \forall y,\ y \in a \Rightarrow y \leq x}$$

ce qui permet de déduire la nouvelle règle d'introduction

$$\vdash_{Max^{\text{déf}}} \frac{\Gamma \vdash x \in a \qquad \Gamma, z \in a \vdash z \leq x}{\Gamma \vdash Max(x,a)}\ z \text{ non libre dans } \Gamma, a, x$$

Pour le calcul des règles d'élimination, on considère les deux dérivations

$$\vdash_{\wedge 1} \frac{\Gamma \vdash x \in a \wedge \forall y,\ y \in a \Rightarrow y \leq x}{\Gamma \vdash x \in a} \qquad \vdash_{\Rightarrow} \frac{\vdash_{\forall} \dfrac{\vdash_{\wedge 2} \dfrac{\Gamma \vdash x \in a \wedge \forall y,\ y \in a \Rightarrow y \leq x}{\Gamma \vdash \forall y,\ y \in a \Rightarrow y \leq x}}{\Gamma \vdash t \in a \Rightarrow t \leq x} \qquad \Gamma \vdash t \in a}{\Gamma \vdash t \leq x}$$

ce qui donne les deux nouvelles règles d'élimination associée à la définition du maximum

$$\vdash_{Max^{\text{déf}}_1} \frac{\Gamma \vdash Max(x,a)}{\Gamma \vdash x \in a} \qquad \vdash_{Max^{\text{déf}}_2} \frac{\Gamma \vdash Max(x,a) \qquad \Gamma \vdash t \in a}{\Gamma \vdash t \leq x}$$

Si on considère en plus la règle de réécriture $x = y \to x \leq y \wedge y \leq x$ correspondant à la réflexivité et à l'antisymétrie de $\leq$, on obtient la règle d'introduction

$$\vdash_{\text{réfl,antisym}} \frac{\Gamma \vdash x \leq y \qquad \Gamma \vdash y \leq x}{\Gamma \vdash y = x}$$

et les deux règles d'élimination

$$\vdash_{\text{réfl,antisym1}} \frac{\Gamma \vdash x = y}{\Gamma \vdash x \leq y} \qquad \vdash_{\text{réfl,antisym2}} \frac{\Gamma \vdash x = y}{\Gamma \vdash y \leq x}$$

On peut obtenir une démonstration très naturelle de l'unicité du maximum ($\Gamma \overset{\text{déf}}{=} Max(x,a), Max(y,a)$) :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash Max(y,a)}\,{}^{\frown}_{\vdash}Max^{\text{déf}_2} \quad \dfrac{\dfrac{}{\Gamma \vdash Max(x,a)}\,{}^{\frown}_{\vdash}Max^{\text{déf}_1}}{\Gamma \vdash x \in a}}{\Gamma \vdash x \leq y}\,{}_{\vdash}\text{réfl,antisym} \quad \dfrac{\dfrac{}{\Gamma \vdash Max(x,a)}\,{}^{\frown}_{\vdash}Max^{\text{déf}_2} \quad \dfrac{\dfrac{}{\Gamma \vdash Max(y,a)}\,{}^{\frown}_{\vdash}Max^{\text{déf}_1}}{\Gamma \vdash y \in a}}{\Gamma \vdash y \leq x}}{\Gamma \vdash x = y}}{\vdash Max(x,a) \Rightarrow Max(y,a) \Rightarrow x = y}\,{}^{\vdash}_{\Rightarrow}}{\vdash \forall x,\ \forall y,\ \forall a,\ Max(x,a) \Rightarrow Max(y,a) \Rightarrow x = y}\,{}^{\vdash}_{\forall}$$

Comme on le voit sur cet exemple, les nouvelles règles d'introduction et d'élimination obtenues paraissent très naturelles vis-à-vis de la définition du maximum et des propriétés d'antisymétrie et de réflexivité. C'est la raison pour laquelle Benjamin Wack (2006) a nommé ce moyen d'obtenir des systèmes de déduction adaptés à une théorie la déduction surnaturelle (en anglais, *supernatural deduction*). Les nouvelles règles sont appelées *surrègles*. Quand le système de réécriture considéré sera évident, on notera NJ^+ le système de déduction surnaturelle associé.

Termes de démonstration

Benjamin Wack (2006) a proposé des termes de démonstrations pour la déduction surnaturelle tels que les ρ-réductions correspondent à des réductions de démonstration naturelles. Toutefois, dans le cas où les parties droites des règles de réécriture comportent des conjonctions, les termes proposés ne sont pas complètement satisfaisant, car premièrement la construction des règles de typage pour les éliminations dépend de celle des règles d'introduction, deuxièmement les termes pour l'élimination peuvent contenir des variables qui n'ont aucune signification par rapport à la démonstration, et troisièmement une réduction de démonstration correspond à plusieurs ρ-réductions. Nous proposons ici de nouveaux termes de démonstration basés sur le ρ-calcul distributif, en étendant ceux sans la conjonction.

Pour calculer la règle de typage associée à une surrègle, on suit la formation de cette surrègle. Pour chaque règle de réécriture propositionnelle R on considère une famille de symboles de fonction f_R^w où w est un mot sur l'alphabet $\{1;2\}$. On notera simplement f_R dans le cas où w est le mot vide. Pour calculer la règle de typage correspondant à une surrègle d'introduction, on annote toutes les prémisses de la surrègle par un terme t_i, puis on annote les prémisses de la dérivation ayant permis de construire cette règle par $\lambda f_R(),\ t_i$. On annote la dérivation en entier en suivant les règles de la figure 3.6 de haut en bas. On annote finalement la conclusion de la surrègle comme la conclusion de la dérivation. Dans le cas où la formule à droite de la règle de réécriture propositionnelle ne comporte pas de conjonction, on voit que l'on n'a pas de symbole $\wr$, on retombe sur les termes proposés par Benjamin Wack (2006).

$$\vdash_{\Rightarrow} \frac{\Gamma, y : X \vdash (\lambda f_R^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ t_k) : Y}{\Gamma \vdash (\lambda f_R^{w_1}(y, \overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{w_k}(y, \overrightarrow{x_k}),\ t_k) : X \Rightarrow Y}$$

$$\vdash_{\wedge} \frac{\begin{array}{c}\Gamma \vdash (\lambda f_R^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \cdots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ t_k) : X \\ \vdots \qquad\qquad \Gamma \vdash (\lambda f_R^{w_{k+1}}(\overrightarrow{x_{k+1}}),\ t_{k+1}) \wr \cdots \wr (\lambda f_R^{w_l}(\overrightarrow{x_l}),\ t_l) : Y\end{array}}{\Gamma \vdash \begin{pmatrix} (\lambda f_R^{1w_1}(\overrightarrow{x_1}),\ t_1) \wr \cdots \wr (\lambda f_R^{1w_k}(\overrightarrow{x_k}),\ t_k) \wr \\ (\lambda f_R^{2w_{k+1}}(\overrightarrow{x_{k+1}}),\ t_{k+1}) \wr \cdots \wr (\lambda f_R^{2w_l}(\overrightarrow{x_l}),\ t_l) \end{pmatrix} : X \wedge Y}$$

$$\vdash_{\forall} \frac{\Gamma \vdash (\lambda f_R^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \cdots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ t_k) : X(x)}{\Gamma \vdash (\lambda f_R^{w_1}(x, \overrightarrow{x_1}),\ t_1) \wr \cdots \wr (\lambda f_R^{w_k}(x, \overrightarrow{x_k}),\ t_k) : \forall y,\ X(y)}$$

FIG. 3.6 : Calcul de la règle de typage associée à une surrègle d'introduction

$$\Rightarrow_{\vdash} \frac{\Gamma \vdash t_0\ f_R^w(u_1, \ldots, u_n) : X \Rightarrow Y \qquad \Gamma \vdash t_i : X}{\Gamma \vdash t_0\ f_R^w(u_1, \ldots, u_n, t_i) : Y}$$

$$\wedge^1_{\vdash} \frac{\Gamma \vdash t_0\ f_R^w(u_1, \ldots, u_n) : X \wedge Y}{\Gamma \vdash t_0\ f_R^{w1}(u_1, \ldots, u_n) : X}$$

$$\wedge^2_{\vdash} \frac{\Gamma \vdash t_0\ f_R^w(u_1, \ldots, u_n) : X \wedge Y}{\Gamma \vdash t_0\ f_R^{w2}(u_1, \ldots, u_n) : Y}$$

$$\forall_{\vdash} \frac{\Gamma \vdash t_0\ f_R^w(u_1, \ldots, u_n) : \forall x,\ X(x)}{\Gamma \vdash t_0\ f_R^w(u_1, \ldots, u_n, t) : X(t)}$$

FIG. 3.7 : Calcul de la règle de typage associée à une surrègle d'élimination

Remarque 3.6 *:* Si on utilise aussi $\top$ dans les formules du système de réécriture, on ajoute la règle de typage suivante :

$$\top_{\vdash} \frac{}{\Gamma \vdash \mathsf{f}_R : \top}$$

Pour les règles d'élimination, on annote toutes les prémisses d'une surrègle par un terme t_i, dont le terme t_0 pour la prémisse principale, puis on annote la prémisse principale de la dérivation ayant permis de construire cette règle par $t_0\ f_R()$. On propage ensuite les annotations dans la dérivation en suivant les règles de la figure 3.7 de haut en bas. On annote finalement la conclusion de la surrègle comme la conclusion de la dérivation.

Exemple 3.3 *:* Appliqué aux règles obtenues dans l'exemple 3.2, on obtient les règles de typage

$$\vdash_{Max^{\text{déf}}} \frac{\Gamma \vdash t : x \in a \qquad \Gamma, y : z \in a \vdash u : z \leq x}{\Gamma \vdash (\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u) : Max(x, a)}\ z \text{ non libre dans } \Gamma, a, x$$

$$\vdash_{Max^{\text{déf}_1}} \frac{\Gamma \vdash u : Max(x, a)}{\Gamma \vdash u\ Max^{\text{déf1}}() : x \in a} \qquad \vdash_{Max^{\text{déf}_2}} \frac{\Gamma \vdash u : Max(x, a) \qquad \Gamma \vdash v : t \in a}{\Gamma \vdash u\ Max^{\text{déf2}}(t, v) : t \leq x}$$

$$\vdash_{\text{réfl,antisym}} \frac{\Gamma \vdash t : x \leq y \qquad \Gamma \vdash u : y \leq x}{\Gamma \vdash (\lambda RA^1(),\ t) \wr (\lambda RA^2(),\ u) : y = x}$$

$$\vdash_{\text{réfl,antisym1}} \frac{\Gamma \vdash t : x = y}{\Gamma \vdash t\ RA^1() : x \leq y} \qquad \vdash_{\text{réfl,antisym2}} \frac{\Gamma \vdash t : x = y}{\Gamma \vdash t\ RA^2() : y \leq x}$$

La réduction de démonstration de la déduction naturelle, c'est-à-dire la β-réduction via l'isomorphisme de Curry-Howard, s'étend alors naturellement par la *dis*-réduction. Cette réduction préserve bien le typage :

Proposition 3.15 (Benjamin Wack, 2005)**.** *Si t est typé par P dans le système de déduction surnaturelle associé à un système de réécriture $\mathcal{R}$, et si $t \xrightarrow[dis]{*} s$, alors s est typable par P dans le même système.*

Démonstration. Benjamin Wack (2005) n'ayant pas exactement les mêmes termes de démonstrations que nous, la démonstration diffère un peu, mais reste essentiellement la même. On procède par induction sur les membres droits des règles de $\mathcal{R}$. □

Exemple 3.4 *:* Si on a une coupure avec les surrègles correspondants à la définition du maximum, elle est soit de la forme

$$\vdash_{Max^{\text{déf}}} \frac{\vdash_{Max^{\text{déf}_1}} \dfrac{\begin{array}{c}\vdots \\ \Gamma \vdash t : x \in a\end{array} \qquad \begin{array}{c}\vdots \\ \Gamma, y : z \in a \vdash u : z \leq x\end{array}}{\Gamma \vdash (\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u) : Max(x, a)}}{\Gamma \vdash ((\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u))\ Max^{\text{déf1}}() : x \in a}\ z \text{ non libre dans } \Gamma, a, x$$

auquel cas $((\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u))\ Max^{\text{déf1}}() \xrightarrow[dis]{} t$ qui correspond à la démonstration

$$\begin{array}{c}\vdots \\ \Gamma \vdash t : x \in a\end{array} \quad ;$$

soit de la forme

$$\vdash_{Max^{\text{déf}}} \frac{\vdots \quad \Gamma \vdash t : x \in a \qquad \Gamma, y : z \in a \vdash u : z \leq x \quad \vdots}{\Gamma \vdash (\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u) : Max(x, a)}\ z \notin FV(\Gamma)$$
$$\vdash_{Max^{\text{déf}}_2} \frac{\Gamma \vdash (\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u) : Max(x, a) \qquad \vdots \ \Gamma \vdash v : t \in a}{\Gamma \vdash ((\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u))\ Max^{\text{déf2}}(t, v) : t \leq x}$$

auquel cas $((\lambda Max^{\text{déf1}}(),\ t) \wr (\lambda Max^{\text{déf2}}(z, y),\ u))\ Max^{\text{déf2}}(t, v) \underset{dis}{\longrightarrow} \{v/y\}\{t/z\}u$ qui correspond à

$$\frac{\vdots}{\Gamma \vdash \{v/y\}\{t/z\}u : t \leq x}$$

Normalisation en déduction surnaturelle

Quand on prend des systèmes pour lesquels on n'a pas la normalisation forte en déduction modulo, le système de déduction surnaturelle correspondant ne normalise pas fortement non plus. Par exemple, le système qui ne normalise pas $A \to A \Rightarrow A$ possède les surrègles suivantes (annotées avec les ρ-termes)

$$\vdash_f \frac{\Gamma, x : A \vdash t : A}{\Gamma \vdash \lambda f(x),\ t : A} \qquad \vdash_f \frac{\Gamma \vdash u : A \qquad \Gamma \vdash v : A}{\Gamma \vdash u\ f(v) : A}$$

On peut alors d'abord construire la démonstration π :

$$\vdash_f \frac{\vdash_f \dfrac{\widehat{\vdash}\ \overline{x : A, y : A \vdash y : A} \qquad \widehat{\vdash}\ \overline{x : A, y : A \vdash y : A}}{x : A, y : A \vdash y\ f(y) : A}}{x : A \vdash \lambda f(y),\ y\ f(y) : A}$$

puis la démonstration

$$\vdash_f \frac{\vdash_f \dfrac{\overset{\pi}{x : A \vdash \lambda f(y),\ y\ f(y) : A} \qquad \overset{\pi}{x : A \vdash \lambda f(y),\ y\ f(y) : A}}{x : A \vdash (\lambda f(y),\ y\ f(y))\ f(\lambda f(y),\ y\ f(y)) : A}}{\vdash \lambda f(x), ((\lambda f(y),\ y\ f(y))\ f(\lambda f(y),\ y\ f(y))) : A}$$

On voit que $\lambda f(x), ((\lambda f(y),\ y\ f(y))\ f(\lambda f(y),\ y\ f(y)))$ se ρ-réduit en lui même.

Paul Brauner et collaborateurs (2007*a*) ont ainsi montré que dans le cas sans conjonction à droite des règles de réécriture, la normalisation forte pour un système de réécriture donné est équivalente en déduction naturelle modulo, en déduction naturelle avec pliage-dépliage et en déduction surnaturelle. On peut donc appliquer les critères de normalisation de la déduction modulo (pré-modèles, supercohérence, cf. section 3.2.3) pour démontrer la normalisation d'un système de déduction surnaturelle. Nous étendons ce résultat dans le cas avec des conjonctions, grâce aux nouveaux termes de démonstration.

Déduction naturelle avec pliage–dépliage vers déduction surnaturelle. On définit d'abord par induction sur la formule à droite d'une règle de réécriture $R : A \to P$ deux fonctions pour simuler le pliage et le dépliage, une démonstration t de A devenant une démonstration $[\![t]\!]_P^{f_R()}$ de P et une démonstration u de P devenant une démonstration $(\!|u|\!)_P^{f_R()}$ de A :

$$
\begin{array}{lcll}
[\![t]\!]_A^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & t\ f_R^w(x_1,\ldots,x_n) & A \text{ atomique} \\
[\![t]\!]_{P_1\Rightarrow P_2}^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & \lambda x_{P_1},\ [\![t]\!]_{P_2}^{f_R^w(x_{P_1},x_1,\ldots,x_n)} & \\
[\![t]\!]_{P_1\wedge P_2}^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & \left([\![t]\!]_{P_1}^{f_R^{w1}(x_1,\ldots,x_n)}, [\![t]\!]_{P_2}^{f_R^{w2}(x_1,\ldots,x_n)}\right) & \\
[\![t]\!]_{\forall y,\, P}^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & \lambda y,\ [\![t]\!]_P^{f_R^w(y,x_1,\ldots,x_n)} &
\end{array}
$$

$$
\begin{array}{lcll}
(\!|t|\!)_A^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & \lambda f_R^w(x_1,\ldots,x_n),\ t & A \text{ atomique} \\
(\!|t|\!)_{P_1\Rightarrow P_2}^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & (\!|t\ x_{P_1}|\!)_{P_2}^{f_R^w(x_{P_1},x_1,\ldots,x_n)} & \\
(\!|t|\!)_{P_1\wedge P_2}^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & (\!|\pi_1 t|\!)_{P_1}^{f_R^{w1}(x_1,\ldots,x_n)} \wr (\!|\pi_2 t|\!)_{P_2}^{f_R^{w2}(x_1,\ldots,x_n)} & \\
(\!|t|\!)_{\forall y,\, P}^{f_R^w(x_1,\ldots,x_n)} & \stackrel{\text{déf}}{=} & (\!|t\ y|\!)_P^{f_R^w(y,x_1,\ldots,x_n)} &
\end{array}
$$

On définit alors la traduction d'un terme de démonstration de la déduction naturelle avec pliage–dépliage en terme de démonstration de la déduction surnaturelle.

Définition 3.13 (Traductions de la déduction naturelle avec pliage–dépliage vers la déduction surnaturelle)**.** *On donne une traduction simple, par induction sur le terme de démonstration :*

- $[t] \stackrel{\text{déf}}{=} t$ *(t terme du premier ordre) ;*
- $[\lambda x,\ t] \stackrel{\text{déf}}{=} \lambda x,\ [t]$ *;*
- $[t\ u] \stackrel{\text{déf}}{=} [t]\ [u]$ *;*
- $[\pi_i t] \stackrel{\text{déf}}{=} \pi_i[t]$ *($i \in \{1,2\}$) ;*
- $[\uparrow_{A\to P} t] \stackrel{\text{déf}}{=} (\!|t|\!)_P^{f_{A\to P}()}$ *;*
- $[\downarrow_{A\to P} t] \stackrel{\text{déf}}{=} [\![t]\!]_P^{f_{A\to P}()}$.

On définit la traduction double d'un terme t correspondant à une démonstration de P comme la forme η-longue de sa traduction simple, à savoir :

- *si P est atomique* $[\![t]\!]_P^\eta \stackrel{\text{déf}}{=} t$ *;*
- *si* $P = P_1 \Rightarrow P_2$*, alors* $[\![t]\!]_P^\eta \stackrel{\text{déf}}{=} \lambda x, ([\![t\ x]\!]_{P_2}^\eta)$ *;*
- *si* $P = \forall x,\ Q$*, alors* $[\![t]\!]_P^\eta \stackrel{\text{déf}}{=} \lambda x, ([\![t\ x]\!]_Q^\eta)$ *;*
- *si* $P = P_1 \wedge P_2$*, alors* $[\![t]\!]_P^\eta \stackrel{\text{déf}}{=} ([\![\pi_1 t]\!]_{P_1}^\eta, [\![\pi_2 t]\!]_{P_2}^\eta)$.

Alors $[\![t]\!] \stackrel{\text{déf}}{=} [\![[t]]\!]_P^\eta$.

Lemme 3.16. – $[\![[\![t]\!]^\eta_P]\!]^\eta_P \xrightarrow[\beta]{*} [\![t]\!]^\eta_P$

– *Si t correspond à une démonstration en déduction naturelle avec pliage–dépliage de P, alors $[\![t]\!]^\eta_P$ correspond aussi à une telle démonstration.*

Démonstration. Par induction sur P. □

Proposition 3.17 (Correction)**.** *Si t est un terme correspondant à une démonstration de P en déduction naturelle avec pliage–dépliage, alors $[\![t]\!]$ est un terme pour une démonstration de P en déduction surnaturelle.*

Démonstration. On procède par induction sur t. Les seuls cas réellement intéressants sont ceux des pliages et dépliages. Si t a le type A alors $\downarrow_R t$ a le type P. Par hypothèse d'induction $[t]$ à le type A. Il s'agit donc de montrer qu'alors $\llbracket [t] \rrbracket^{f_{A\to P}()}_P$ a le type P . On démontre ceci par induction sur P. La cas du dépliage revient de même à montrer que $(\!|t|\!)^{f_{A\to P}()}_P$ a le type A si t est de type P. □

Lemme 3.18. $\left\llbracket (\!|t_1|\!)^{f^{w_1}_R(x^1_1,\ldots,x^1_{n_1})}_P \wr \cdots \wr (\!|t_m|\!)^{f^{w_m}_R(x^m_1,\ldots,x^m_{n_m})}_P \right\rrbracket^{f^{w_i}_R(x^i_1,\ldots,x^i_{n_i})}_P \xrightarrow[dis]{+} [\![t_i]\!]^\eta_P$ *si tous les w_j sont distincts.*

Démonstration. Par induction sur P. □

Proposition 3.19. *Pour tous termes t et t' correspondant à des démonstration en déduction naturelle avec pliage–dépliage, si $t \xrightarrow[\beta]{} t'$ alors $[\![t]\!] \xrightarrow[dis]{+} [\![t']\!]$.*

Démonstration. On procède par cas suivant le type de coupure. Le seul cas non trivial est celui de la coupure entre pliage et dépliage $\downarrow_R\uparrow_R t \xrightarrow[\beta]{} t$. $[\downarrow_R\uparrow_R t] = \left\llbracket (\!|[\![t]\!]|\!)^{f_R()}_P \right\rrbracket^{f_R()}_P$ se réduit d'après le lemme précédent en $[\![t]\!]^\eta_P$ en au moins une étape. Par conséquent $[\![\downarrow_R\uparrow_R t]\!]$ se réduit vers $[\![[\![t]\!]^\eta_P]\!]^\eta_P$ qui se réduit vers $[\![t]\!]^\eta_P$. □

Déduction surnaturelle vers déduction naturelle avec pliage–dépliage. Il s'agit essentiellement de remplacer les applications d'une surrègle d'introduction par l'application d'une règle de pliage suivie des introductions qui ont servi à construire la règle, et les applications d'une surrègle d'élimination par une règle de dépliage précédée des éliminations qui ont servi à construire la règle. Au niveau des termes, cela donne la traduction suivante.

Définition 3.14 (Traductions de la déduction surnaturelle vers la déduction naturelle avec pliage–dépliage)**.** *On définit d'abord les termes correspondant aux constructions des surrègles, pour les introductions*

– $(\!|\lambda f_R(),\ t|\!) \stackrel{\text{déf}}{=} t$;

– $\langle\!|(\lambda f_R^{w_1}(x, \overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{w_k}(x, \overrightarrow{x_k}),\ t_k)|\!\rangle \overset{\text{déf}}{=} \lambda x,\ \langle\!|(\lambda f_{A\to Q}^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_{A\to Q}^{w_k}(\overrightarrow{x_k}),\ t_k)|\!\rangle$ *si* R *est de la forme* $A \to P \Rightarrow Q$ *ou* $A \to \forall x,\ Q$ *;*

– $\langle\!|(\lambda f_R^{1w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{1w_k}(\overrightarrow{x_k}),\ t_k) \wr (\lambda f_R^{2w_{k+1}}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{2w_l}(\overrightarrow{x_l}),\ t_l)|\!\rangle \overset{\text{déf}}{=} \left(\langle\!|(\lambda f_{R_1}^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_{R_1}^{w_k}(\overrightarrow{x_k}),\ t_k)|\!\rangle, \langle\!|(\lambda f_{R_2}^{w_{k+1}}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_{R_2}^{w_l}(\overrightarrow{x_l}),\ t_l)|\!\rangle\right)$ *avec* $R = A \to P_1 \wedge P_2$, $R_1 = A \to P_1$ *et* $R_2 = A \to P_2$, *si il n'y a aucune variable commune à tous les* $\overrightarrow{x_i}$.

puis pour les éliminations

– $[\![t\ f_{A\to B}()]\!] \overset{\text{déf}}{=} t$ *si* B *est atomique ;*

– $[\![t\ f_{A\to P}^{w}(u, v_1, \ldots, v_n)]\!] \overset{\text{déf}}{=} \left[\!\!\left[(t\ u)\ f_{A\to Q}^{w}(v_1, \ldots, v_n)\right]\!\!\right]$ *si* P *est de la forme* $P' \Rightarrow Q$ *ou* $\forall x,\ Q$ *;*

– $[\![t\ f_{A\to P_1\wedge P_2}^{iw}(v_1, \ldots, v_n)]\!] \overset{\text{déf}}{=} [\![(\pi_i t)\ f_{A\to P_i}^{w}(v_1, \ldots, v_n)]\!]$ *pour* $i \in \{1; 2\}$ *;*

On peut donc définir ensuite une traduction des termes de démonstration de la déduction surnaturelle vers les termes de démonstration de la déduction naturelle avec pliage–dépliage

– $||t|| \overset{\text{déf}}{=} t$ *si* t *terme du premier ordre ;*

– $||\lambda x,\ t|| \overset{\text{déf}}{=} \lambda x,\ ||t||$ *;*

– $||t\ u|| \overset{\text{déf}}{=} ||t||\ ||u||$ *;*

– $||(\lambda f_R^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ t_k)|| \overset{\text{déf}}{=} \uparrow_R \langle\!|(\lambda f_R^{w_1}(\overrightarrow{x_1}),\ ||t_1||) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ ||t_k||)|\!\rangle$ *;*

– $||t\ f_R^{w}(u_1, \ldots, u_n)|| \overset{\text{déf}}{=} [\![(\downarrow_R ||t||)\ f_R^{w}(||u_1||, \ldots, ||u_n||)]\!]$.

Proposition 3.20 (Correction)**.** *Si* t *est un terme correspondant à une démonstration de* P *en déduction surnaturelle, alors* $||t||$ *correspond à une démonstration de* P *en déduction naturelle avec pliage–dépliage.*

Démonstration. Il s'agit de constater que les définitions données ci-dessus correspondent bien aux remplacements des surrègles par leur construction. □

Proposition 3.21. *Si* t *est un terme correspondant à une démonstration en déduction surnaturelle, et si* $t \underset{dis}{\longrightarrow} t'$, *alors* $||t|| \overset{+}{\underset{\beta}{\longrightarrow}} ||t'||$.

Démonstration. Seul le cas d'une coupure entre surrègles est intéressant. On a alors $((\lambda f_R^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ t_k))\ f_R^{w_i}(u_1, \ldots, u_n) \underset{dis}{\longrightarrow} \{u_1/x_{i,1}, \ldots, u_n/x_{i,n_i}\}t_i$.

$$
\begin{aligned}
&||((\lambda f_R^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ t_k))\ f_R^{w_i}(u_1, \ldots, u_n)|| \\
&= [\![(\downarrow_R ||(\lambda f_R^{w_1}(\overrightarrow{x_1}),\ t_1) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ t_k)||)\ f_R^{w_i}(||u_1||, \ldots, ||u_n||)]\!] \\
&= [\![(\downarrow_R \uparrow_R \langle\!|(\lambda f_R^{w_1}(\overrightarrow{x_1}),\ ||t_1||) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ ||t_k||)|\!\rangle)\ f_R^{w_i}(||u_1||, \ldots, ||u_n||)]\!] \\
&\underset{\beta}{\longrightarrow} [\![\langle\!|(\lambda f_R^{w_1}(\overrightarrow{x_1}),\ ||t_1||) \wr \ldots \wr (\lambda f_R^{w_k}(\overrightarrow{x_k}),\ ||t_k||)|\!\rangle\ f_R^{w_i}(||u_1||, \ldots, ||u_n||)]\!]
\end{aligned}
$$

On procède alors par induction sur la formule à droite de la règle de réécriture.

– $[\![\langle\!|\lambda f_{A\to B}(),\ t|\rangle\ \ f_{A\to B}()]\!] = t$
– si P est de la forme $P' \Rightarrow Q$ ou $\forall x,\ Q$, alors

$$\begin{aligned}
&[\![\langle\!|(\lambda f^{w_1}_{A\to P}(x,\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{A\to P}(x,\overrightarrow{x_k}),\ t_k)|\rangle\ \ f^{w_i}_{A\to P}(u,v_1,\ldots,v_n)]\!]\\
=&[\![\left(\lambda x,\ \langle\!|(\lambda f^{w_1}_{A\to Q}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{A\to Q}(\overrightarrow{x_k}),\ t_k)|\rangle\right)\ \ f^{w_i}_{A\to P}(u,v_1,\ldots,v_n)]\!]\\
=&[\![\left(\lambda x,\ \langle\!|(\lambda f^{w_1}_{A\to Q}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{A\to Q}(\overrightarrow{x_k}),\ t_k)|\rangle\right)\ \ u\ f^{w_i}_{A\to Q}(v_1,\ldots,v_n)]\!]\\
\xrightarrow[\beta]{}&[\![\{u/x\}\ \langle\!|(\lambda f^{w_1}_{A\to Q}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{A\to Q}(\overrightarrow{x_k}),\ t_k)|\rangle\ \ f^{w}_{A\to Q}(v_1,\ldots,v_n)]\!]\\
=&\{v_1/x_1,\ldots,v_n/x_n\}\{u/x\}t_i \text{ par hypothèse d'induction.}
\end{aligned}$$

– avec $R = A \to P_1 \wedge P_2$, $R_1 = A \to P_1$ et $R_2 = A \to P_2$, si $i \le k$,

$$\begin{aligned}
&[\![\left\langle\!\middle|\begin{matrix}(\lambda f^{1w_1}_{R}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{1w_k}_{R}(\overrightarrow{x_k}),\ t_k)\wr\\ (\lambda f^{2w_{k+1}}_{R}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{2w_l}_{R_1\wedge P_2}(\overrightarrow{x_l}),\ t_l)\end{matrix}\middle|\right\rangle\ \ f^{1w_i}_{A\to P_1\wedge P_2}(v_1,\ldots,v_n)]\!]\\
=&[\![\left\langle\!\middle|\begin{pmatrix}\langle\!|(\lambda f^{w_1}_{R_1}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{R_1}(\overrightarrow{x_k}),\ t_k)|\rangle,\\ \langle\!|(\lambda f^{w_{k+1}}_{R_2}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_l}_{R_2}(\overrightarrow{x_l}),\ t_l)|\rangle\end{pmatrix}\middle|\right\rangle\ \ f^{1w_i}_{A\to P_1\wedge P_2}(v_1,\ldots,v_n)]\!]\\
=&[\![\left\langle\!\middle|\left(\pi_1\begin{pmatrix}\langle\!|(\lambda f^{w_1}_{R_1}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{R_1}(\overrightarrow{x_k}),\ t_k)|\rangle,\\ \langle\!|(\lambda f^{w_{k+1}}_{R_2}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_l}_{R_2}(\overrightarrow{x_l}),\ t_l)|\rangle\end{pmatrix}\right)\middle|\right\rangle\ \ f^{w_i}_{A\to P_1}(v_1,\ldots,v_n)]\!]\\
\xrightarrow[\beta]{}&[\![\langle\!|(\lambda f^{w_1}_{R_1}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{R_1}(\overrightarrow{x_k}),\ t_k)|\rangle\ \ f^{w_i}_{A\to P_1}(v_1,\ldots,v_n)]\!]\\
=&\{v_1/x_1,\ldots,v_n/x_n\}t_i \text{ par hypothèse d'induction.}
\end{aligned}$$

le cas pour $k < i \le l$ étant symétrique.
Donc

$$\begin{aligned}
&||((\lambda f^{w_1}_{R}(\overrightarrow{x_1}),\ t_1)\wr\ldots\wr(\lambda f^{w_k}_{R}(\overrightarrow{x_k}),\ t_k))\ f^{w_i}_{R}(u_1,\ldots,u_n)||\\
\xrightarrow[\beta]{+}&\{||u_1||/x_{i,1},\ldots,||u_n||/x_{i,n_i}\}||t_i||\\
=&||\{u_1/x_{i,1},\ldots,u_n/x_{i,n_i}\}t_i||
\end{aligned}$$

. □

Théorème 3.22.
Étant donné un système de réécriture $\mathcal{R}$, la déduction naturelle modulo $\mathcal{R}$ est fortement normalisante ssi la déduction surnaturelle pour $\mathcal{R}$ l'est aussi.

Démonstration. C'est une conséquence directe des propositions 3.19 et 3.21, une dérivation infinie dans un des systèmes de déduction étant traduite par une dérivation infinie dans l'autre. □

3.3.2 Calcul des séquences extensible

Il est assez naturel de se demander s'il est possible d'obtenir des systèmes similaires à ceux obtenus grâce à la déduction surnaturelle, mais en partant du calcul des séquences. Paul Brauner et collaborateurs (2007*b*) ont ainsi introduit le *calcul des séquences extensible* pour la logique classique du premier ordre. L'idée est de calculer des nouvelles règles, cette fois des règles gauche et droite, en appliquant autant que possible les règles du calcul des séquences après un dépliage. Toutefois, la construction des nouvelles règles décompose peut-être trop les formules. Pour rester complet, il faut donc restreindre la décomposition des règles dans cette construction. Une version simple consisterait à n'utiliser que les règles inversibles. Une version plus évoluée consiste à n'utiliser que des règles de même polarité (les règles neutres pouvant être considérées à la fois comme positives et négatives). Pour le calcul des séquences classique, cela revient à ne pas appliquer $\exists\vdash$ ou $\vdash\forall$ après avoir appliqué $\forall\vdash$ ou $\vdash\exists$, et vice-versa.

Remarque 3.7 *:* On aurait pu décomposer toutes les sous-formules de la partie droite de la règle de réécriture avec des règles positives ou neutres, puis appliquer toutes les règles négatives ou neutres possibles. On serait ainsi resté complet, mais on perd la symétrie entre la construction d'une règle gauche et celle d'une règle droite, et l'élimination des coupures est alors moins claire, puisqu'il faut rajouter les étapes qui n'ont pas été effectuée pour une des surrègles. Par exemple, pour la règle de réécriture $R : A \to \forall x,\ \exists y,\ B(x,y)$, le calcul des séquences extensible donne les règles

$$R\vdash\ \frac{\Gamma, \exists y,\ B(t,y) \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \vdash R\ \frac{\Gamma \vdash \exists y,\ B(z,y), \Delta}{\Gamma \vdash A, \Delta}\ z \notin FV(\Gamma, \Delta)$$

avec une élimination des coupures quasi naturelle :

$$R\vdash\ \frac{\dfrac{\overset{\pi}{\Gamma, \exists y,\ B(t,y) \vdash \Delta}}{\Gamma, A \vdash \Delta}\ \subseteq \qquad \vdash R\ \dfrac{\overset{\pi'}{\Gamma \vdash \exists y,\ B(z,y), \Delta}}{\Gamma \vdash A, \Delta}\ z \notin FV(\Gamma, \Delta)}{\Gamma \vdash \Delta} \rightsquigarrow$$

$$\subseteq\ \frac{\overset{\pi}{\Gamma, \exists y,\ B(t,y) \vdash \Delta} \qquad \overset{\{t/z\}\pi'}{\Gamma \vdash \exists y,\ B(t,y), \Delta}}{\Gamma \vdash \Delta}$$

Si on décomposait les formules positives puis négative, on obtiendrait les règles

$$R\vdash\ \frac{\Gamma, B(t,y) \vdash \Delta}{\Gamma, A \vdash \Delta}\ y \notin FV(\Gamma, \Delta) \qquad \vdash R\ \frac{\Gamma \vdash \exists y,\ B(x,y), \Delta}{\Gamma \vdash A, \Delta}\ x \notin FV(\Gamma, \Delta)$$

avec une élimination des coupures moins évidente :

$$R\vdash\ \frac{\dfrac{\overset{\pi}{\Gamma, B(t,y) \vdash \Delta}}{\Gamma, A \vdash \Delta}\ y \notin FV(\Gamma, \Delta) \subseteq \qquad \vdash R\ \dfrac{\overset{\pi'}{\Gamma \vdash \exists y,\ B(x,y), \Delta}}{\Gamma \vdash A, \Delta}\ x \notin FV(\Gamma, \Delta)}{\Gamma \vdash \Delta} \rightsquigarrow$$

$$\exists\vdash,\ \vdash\!\smile \frac{\overset{\pi}{\Gamma, B(t,y) \vdash \Delta}}{\Gamma, \exists y,\ B(t,y) \vdash \Delta}\ y \notin FV(\Gamma,\Delta) \qquad \begin{matrix}\{t/x\}\pi' \\ \Gamma \vdash \exists y,\ B(t,y), \Delta\end{matrix}$$
$$\Gamma \vdash \Delta$$

Remarque 3.8 *:* Pour décomposer les formules au maximum, on considère également les règles $\top\vdash$ et $\vdash\bot$ dont il est question à la remarque 2.6 et qui sont neutres.

Exemple 3.5 *:* On considère toujours la définition du maximum, on obtient la règle gauche

$$Max^{\text{déf}}\vdash \frac{\Gamma, x \in a \vdash t \in a, \Delta \qquad \Gamma, x \in a, t \leq x \vdash \Delta}{\Gamma, Max(x,a) \vdash \Delta}$$

et la règle droite

$$\vdash Max^{\text{déf}} \frac{\Gamma \vdash x \in a, \Delta \qquad \Gamma, z \in a \vdash z \leq x, \Delta}{\Gamma \vdash Max(x,a), \Delta}\ z \text{ non libre dans } \Gamma, \Delta, a, x$$

Nous introduisons également une version du calcul des séquences extensible pour la logique intuitionniste du premier ordre, basée sur LB. Il y a essentiellement deux façon de l'obtenir : La première est de faire comme dans le cas classique et d'appliquer toutes les règles possibles de même polarité. Toutefois, comme on n'a plus de dualité, on n'aura dans ce cas plus de correspondance entre la surrègle gauche et la surrègle droite. Par exemple, la règle $R : A \to \neg(B \Rightarrow C)$ donnera les règles

$$R\vdash \frac{\Gamma, \vdash B \Rightarrow C, \Delta}{\Gamma, A \vdash \Delta} \qquad \vdash R \frac{\Gamma \vdash B, \Delta \qquad \Gamma, C \vdash \Delta}{\Gamma \vdash A, \Delta}$$

Autant, dans ce cas, utiliser la remarque 3.7 et utiliser d'abord toutes les règles positives ou neutres possibles, puis toutes les règles négatives ou neutres possibles.

La deuxième méthode est d'utiliser la traduction de la déduction naturelle vers le calcul des séquences pour définir une traduction des surrègles de la déduction naturelle en calcul des séquences extensible. On se restreint alors aux quantificateurs $\Rightarrow$, $\wedge$ et $\forall$. En utilisant la correspondance entre règle d'élimination et règle gauche, et entre règle d'introduction et règle droite, calculer la surrègle gauche (resp. droite) associée à une règle de réécriture propositionnelle $A \to P$ consiste à appliquer de bas en haut les règles de la figure 3.8 sur la séquence $\Gamma, P^* \vdash \Delta$ (resp. $\Gamma \vdash P^*, \Delta$) jusqu'à ce qu'il n'y ait plus d'étoile, et d'associer les prémisses obtenues avec la conclusion $\Gamma, A \vdash \Delta$ (resp. $\Gamma \vdash A, \Delta$). Quand le système de réécriture utilisé sera clair depuis le contexte, on utilisera la notation LB^+ pour désigner le système de déduction ainsi obtenu.

Exemple 3.6 *:* Pour l'exemple de la définition du maximum, les deux méthodes donnent les règles

$$Max^{\text{déf}}\vdash \frac{\Gamma, x \in a \vdash t \in a, \Delta \qquad \Gamma, x \in a, t \leq x \vdash \Delta}{\Gamma, Max(x,a) \vdash \Delta}$$

A atomique :

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A^* \vdash \Delta} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A^*, \Delta}$$

$$\frac{\Gamma, \top \vdash \Delta}{\Gamma, \top^* \vdash \Delta} \qquad \frac{}{\Gamma \vdash \top^*, \Delta}$$

$$\frac{\Gamma, Q^* \vdash \Delta \qquad \Gamma \vdash P, \Delta}{\Gamma, (P \Rightarrow Q)^* \vdash \Delta} \qquad \frac{\Gamma, P \vdash Q^*}{\Gamma \vdash (P \Rightarrow Q)^*, \Delta}$$

$$\frac{\Gamma, P^*, Q^* \vdash \Delta}{\Gamma, (P \wedge Q)^* \vdash \Delta} \qquad \frac{\Gamma \vdash P^*, \Delta \qquad \Gamma \vdash Q^*, \Delta}{\Gamma \vdash (P \wedge Q)^*, \Delta}$$

$$\frac{\Gamma, (\{t/x\}P)^* \vdash \Delta}{\Gamma, (\forall x.\ P)^* \vdash \Delta} \qquad \frac{\Gamma \vdash P^*}{\Gamma \vdash (\forall x.\ P)^*, \Delta}\ x \notin FV(\Gamma)$$

FIG. 3.8 : Règles de construction de surrègles gauche et droite pour LB^+

et

$$\vdash Max^{\text{déf}}\ \frac{\Gamma \vdash x \in a, \Delta \qquad \Gamma, z \in a \vdash z \leq x}{\Gamma \vdash Max(x, a), \Delta}\ z \text{ non libre dans } \Gamma, a, x$$

Remarque 3.9 *:* La version utilisant la polarité des règles est plus puissante, puisque par exemple $A \to (B \wedge C) \Rightarrow D$ produit une règle gauche plus décomposée. Néanmoins, on perd alors la correspondance avec la déduction surnaturelle, que nous allons utiliser par la suite.

La complétude (vis-à-vis de la logique engendrée par une présentation compatible avec le système de réécriture en question à partir de la logique intuitionniste du premier ordre) de la première méthode est une conséquence de celle de la focalisation que nous avons définie pour LB, que nous n'avons pas toutefois pas démontrée. La complétude de la deuxième méthode peut quant à elle être démontrée en utilisant la complétude de la déduction surnaturelle, en utilisant des traductions entre les deux systèmes. Ces traductions nous permettront également de démontrer certaines propriétés des systèmes de surdéduction. Il s'agit en fait d'étendre les traductions données dans la section 2.3.5.

De la déduction surnaturelle au calcul des séquences extensible

Il s'agit d'étendre la traduction entre déduction naturelle et calcul des séquences du chapitre précédent aux surrègles. Les règles de construction des surrègles du calcul des séquences extensible ont justement été choisies pour correspondre à celle de la déduction surnaturelle.

Dans la fin de ce chapitre, on écrira les démonstrations en déduction surnaturelle dans le style de Gerhard GENTZEN pour plus de facilité.

Lemme 3.23. *Soit R la règle de réécriture $A \to P$.*
Une surrègle d'élimination de R est de la forme

$$\overset{R}{\vdash} \frac{A \qquad H_1 \quad \cdots \quad H_n}{K}$$

La surrègle gauche de R est de la forme

$$R\vdash \frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_i}{\Gamma, A \vdash \Delta}$$

Si on suppose qu'il existe des démonstrations de $\Gamma \vdash H_j$ dans LB^+ pour tout j, alors on peut démontrer $\Gamma, \Gamma_i \vdash \Delta_i, K$ dans LB^+ pour tout i (et donc aussi $\Gamma, A \vdash K$).

Démonstration. La forme des surrègles s'obtient par simple induction sur leur construction.

Le reste se démontre par induction sur P :

– Si P est atomique, la surrègle d'élimination est

$$\overset{R}{\vdash} \frac{A}{P}$$

et la surrègle gauche

$$R\vdash \frac{\Gamma, P \vdash \Delta}{\Gamma, A \vdash \Delta}$$

On doit trouver une démonstration de $\Gamma, P \vdash P$:

$$\overset{\frown}{\vdash} \frac{}{\Gamma, P \vdash P}$$

– Si P est de la forme $P_1 \Rightarrow P_2$, soit R' la règle $D \to P_2$.
Les surrègles d'élimination de R' sont de la forme

$$\overset{R'p}{\vdash} \frac{D \qquad H_1 \quad \cdots \quad H_n}{K}$$

et dans ce cas les règles d'élimination de R sont de la forme

$$\overset{Rp}{\vdash} \frac{A \qquad P_1 \qquad H_1 \quad \cdots \quad H_n}{K}$$

La surrègle gauche de R' est de la forme

$$R'\vdash \frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_i}{\Gamma, D \vdash \Delta}$$

et dans ce cas la règle gauche de R est de la forme

$$R\vdash \frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_i \qquad \Gamma \vdash P_1, \Delta}{\Gamma, A \vdash \Delta}$$

Supposons qu'on ait des démonstrations π_j de $\Gamma \vdash H_j$ et une démonstration π_0 de $\Gamma \vdash P_1$.
Par hypothèse d'induction il existe une démonstration de $\Gamma, \Gamma_i \vdash \Delta_i, K$ pour tout i. Par affaiblissement π_0 is a proof of $\Gamma \vdash P_1, K$.

Remarque 3.10 *:* C'est ici qu'on voit l'intérêt d'utiliser LB plutôt que LJ car on a besoin d'avoir plusieurs formules à droite de la séquence (à savoir P_1 et K).

– Si P est de la forme $P_1 \wedge P_2$, soit R_1 la règle de réécriture $A_1 \to P_1$ et R_2 la règle $A_2 \to P_2$.
Pour $k \in \{1; 2\}$, une surrègle d'élimination de R_k est de la forme

$$\overset{R_{kp}}{\vdash} \frac{A_k \qquad H^1_{k,p} \quad \cdots \quad H^n_{k,p}}{K_{k,p}}$$

et dans ce cas les surrègles d'élimination de R sont de la forme

$$\overset{Rk,p}{\vdash} \frac{A \qquad H^1_{k,p} \quad \cdots \quad H^n_{k,p}}{K_{k,p}}$$

La surrègle gauche de R_k est de la forme

$$R_k\vdash \frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_{i \in I_k}}{\Gamma, A_k \vdash \Delta}$$

et dans ce cas la surrègle gauche de R est de la forme

$$R\vdash \frac{(\Gamma, \Gamma_i, \Gamma_j \vdash \Delta_i, \Delta_j, \Delta)_{i \in I_1, j \in I_2}}{\Gamma, A \vdash \Delta}$$

Si k est 1 ou 2, supposons qu'il existe une démonstration π_j de $\Gamma \vdash H^j_{k,p}$ pour tout j
Par hypothèse d'induction sur R_k, il existe des démonstrations de $\Gamma, \Gamma_i \vdash \Delta_i, K_{k,p}$ pour tout p et tout $i \in I_k$. Par affaiblissement, ce sont aussi des démonstrations de $\Gamma, \Gamma_i, \Gamma_j \vdash \Delta_i, \Delta_j, K_{k,p}$ pour tout $j \in I_{3-k}$.
– Si P est de la forme $\forall x.\, Q$, soit t le terme qui instancie x dans l'application de $R\vdash$.
Soit R' la règle de réécriture $D \to \{t/x\}Q$.
Les surrègles d'élimination de R' sont de la forme

$$\overset{R'p}{\vdash} \frac{D \qquad H_1 \quad \cdots \quad H_n}{K}$$

et dans ce cas les règles d'élimination de R sont de la forme

$$\overset{Rp}{\vdash} \frac{A \qquad H_1 \quad \cdots \quad H_n}{K}$$

La surrègle gauche de R' est de la forme

$$R'\vdash \frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_i}{\Gamma, D \vdash \Delta}$$

et dans ce cas la surrègle gauche de R est de la forme

$$R\vdash \frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_i}{\Gamma, A \vdash \Delta}$$

Une simple application de l'hypothèse d'induction donne le résultat attendu.

□

Remarquons également que les surrègles droite sont les mêmes que les surrègles d'introduction, si ce n'est qu'elles permettent plusieurs formules à droite de la séquence. On peut donc étendre la traduction de la proposition 2.9.

Proposition 3.24 (Traduction de NJ^+ dans LB^+)**.** *Une démonstration de C sous les hypothèses Γ en déduction surnaturelle peut être traduite en une démonstration de conséquence $\Gamma \vdash C$ en LB^+.*

Démonstration. On étend la démonstration de la proposition 2.9. Une surrègle d'introduction est traduite par une surrègle droite. Les surrègles d'élimination en haut à gauche sont traduite par des surrègles gauche à l'aide du lemme 3.23. On remarque que l'on conserve bien le rapport entre coupure en déduction surnaturelle et coupure en calcul des séquences extensible. □

Du calcul des séquences extensible à la déduction surnaturelle

On procède avec la même idée, qui est de traduire les surrègles gauche en surrègles d'élimination et les surrègles droites en surrègles d'introduction. On aura besoin du fait qu'on peut en fait se ramener à des séquences avec une seule formule à droite. On fera également attention au fait qu'on ne rajoute pas de règles logiques supplémentaire dans la traduction.

Lemme 3.25. *Supposons qu'il existe une démonstration dans LB^+ dont la dernière inférence est*

$$R\vdash \frac{(\Gamma_i \vdash \Delta_i)_i}{\Gamma, A \vdash \Delta}$$

avec R la règle $A \to P$. Supposons que pour tout i il existe un $C_i \in \Delta$ tel que C_i et une démonstration π_i de C_i sous les hypothèses Γ_i en déduction surnaturelle.

Alors il existe un $C \in \Delta$ et une démonstration de C sous les hypothèses Γ, A en déduction surnaturelle qui n'utilise que les π_i et les règles d'élimination de R.

Démonstration. Par induction sur P :

– Si P est atomique, alors la dernière inférence est en fait

$$R\vdash \frac{\Gamma, P \vdash \Delta}{\Gamma, A \vdash \Delta}$$

Par conséquent, si $C \in \Delta$ et π est une démonstration de C à partir de Γ et P, alors

$$\Gamma \quad \overset{R}{\vdash} \frac{A}{P}$$
$$\boxed{\pi}$$
$$C$$

est la démonstration attendue.

– Si P est de la forme $P_1 \Rightarrow P_2$, soit R' la règle de réécriture $D \to P_2$. Par construction de la surrègle gauche, il existe un i_0 tel que $(\Gamma_{i_0} \vdash \Delta_{i_0}) = (\Gamma \vdash P_1, \Delta)$ et la surrègle gauche de R' est

$$R' \vdash \frac{(\Gamma_i \vdash \Delta_i)_{i \neq i_0}}{\Gamma, D \vdash \Delta}$$

Supposons qu'il existe une démonstration π_{i_0} d'un $C_{i_0} \in P_1, \Delta$ à partir des hypothèses Γ. Si $C_{i_0} \in \Delta$, alors π_{i_0} est la démonstration attendue.
Sinon, $C_{i_0} = P_1$, supposons que pour tout $i \neq i_0$ nous avons des démonstrations π_i d'un certain $C_i \in \Delta_i$ à partir de Γ_i. Par hypothèse d'induction on peut construire une démonstration ϖ d'un certain C dans Δ à partir des hypothèses Γ, des π_i pour $i \neq i_0$ et des surrègles d'élimination de R'.
Les surrègles d'élimination de R' sont de la forme

$$\overset{R'p}{\vdash} \frac{D \qquad H_1 \quad \cdots \quad H_n}{K}$$

et dans ce cas les surrègles d'élimination de R sont de la forme

$$\overset{Rp}{\vdash} \frac{A \qquad P_1 \qquad H_1 \quad \cdots \quad H_n}{K}$$

En remplaçant $\overset{R'p}{\vdash}$ par $\overset{Rp}{\vdash}$ dans ϖ, on obtient une démonstration de C construite à partir des hypothèses Γ, P_1, des π_i et des surrègles d'élimination pour R. En remplaçant l'hypothèse P_1 qui est égale à C_{i_0} par la démonstration π_{i_0}, on obtient la démonstration attendue.

– Si P est de la forme $P_1 \wedge P_2$, soit R_1 la règle de réécriture $A_1 \to P_1$ et R_2 la règle $A_2 \to P_2$.
Pour $k \in \{1; 2\}$, la surrègle R_k est de la forme

$$R_k \vdash \frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_{i \in I_k}}{\Gamma, D \vdash \Delta}$$

et celle de R est alors

$$R \vdash \frac{(\Gamma, \Gamma_i, \Gamma_j \vdash \Delta_i, \Delta_j, \Delta)_{i \in I_1, j \in I_2}}{\Gamma, A \vdash \Delta}$$

Supposons que pour tous $i \in I_1, j \in I_2$ nous ayons des démonstrations en déduction surnaturelle $\pi_{i,j}$ d'un certain $C_i \in \Delta_i, \Delta_j, \Delta$ à partir des hypothèses $\Gamma, \Gamma_i, \Gamma_j$. Par hypothèse d'induction pour R_1, pour tout $j \in I_2$ il existe un $C_j \in \Delta_j, \Delta$ possédant une démonstration ϖ_j construite à partir des hypothèses Γ, Γ_j, A_1, des démonstrations $\pi_{i,j}$ et des surrègles d'élimination de R_1.
Une surrègle d'élimination de R_1 est de la forme

$$\overset{R_1 p}{\vdash} \frac{A_1 \qquad H_1 \quad \cdots \quad H_n}{K}$$

et une partie des surrègles d'élimination de R est alors

$$\overset{R1,p}{\vdash} \frac{A \qquad H_1 \quad \cdots \quad H_n}{K}$$

En remplaçant $\overset{R_1 p}{\vdash}$ par $\overset{R1,p}{\vdash}$ dans les ϖ_j, on obtient ainsi des démonstrations ϖ'_j de C_j construites à partir des hypothèses Γ, Γ_j, A, des démonstrations $\pi_{i,j}$ et des surrègles d'élimination de R.

Par hypothèse d'induction pour R_2 en utilisant ces ϖ'_j, il existe un $C \in \Delta$ possédant une démonstration ϖ construite à partir des hypothèses Γ, A, A_2, des démonstrations ϖ'_j et des surrègles d'élimination de R_2. Ici aussi, en remplaçant les surrègles d'élimination $\overset{R_2 p}{\vdash}$ par les surrègles $\overset{R2,p}{\vdash}$, on obtient une démonstration de C à partir des hypothèses Γ, A, des démonstrations ϖ'_j et des surrègles d'élimination de R, autrement dit une démonstration construite à partir des hypothèses Γ, A, des démonstrations $\pi_{i,j}$ et des surrègles d'élimination de R.

– Si P est de la forme $\forall x.\ Q$. Soit t le terme qui instancie x dans la démonstration de $\Gamma, A \vdash \Delta$. Soit R' la règle de réécriture $D \to \{t/x\}Q$. Par construction des surrègles, $\overset{R'}{\vdash}$ est de la forme

$$\overset{R'}{\vdash} \frac{(\Gamma_i \vdash \Delta_i)_i}{\Gamma, D \vdash \Delta}$$

Supposons que pour tout i nous ayons des démonstrations π_i d'un certain $C_i \in \Delta_i$ à partir des hypothèses Γ_i. Par hypothèse d'induction il existe un $C \in \Delta$ possédant une démonstration ϖ construite à partir des hypothèses Γ, D, des démonstrations π_i et des surrègles d'élimination de R'.

Les surrègles d'élimination de R' sont de la forme

$$\overset{R' p}{\vdash} \frac{D \qquad H_1 \quad \cdots \quad H_n}{K}$$

et dans ce cas les surrègles d'élimination de R sont de la forme

$$\overset{R p}{\vdash} \frac{A \qquad H_1 \quad \cdots \quad H_n}{K}$$

En remplaçant $\overset{R' p}{\vdash}$ par $\overset{R p}{\vdash}$ dans ϖ, on obtient donc une démonstration de C construite à partir des hypothèses Γ, A, des démonstrations π_i et des surrègles d'élimination pour R.

□

Pour les surrègles droites il faut s'assurer que l'on peut se ramener à une conclusion unique.

Lemme 3.26. *Supposons qu'il existe une démonstration en* LB^+ *dont la dernière inférence est*

$$\vdash R \frac{(\Gamma_i \vdash \Delta_i)_i}{\Gamma \vdash A, \Delta}$$

où R est la règle de réécriture $A \to P$. Supposons que pour tout i il existe une démonstration en déduction surnaturelle π_i d'un certain $C_i \in \Delta_i$ à partir des hypothèses Γ_i.

Alors on peut construire une démonstration en déduction surnaturelle d'un certain $C \in A, \Delta$ construite à partir des hypothèses Γ, des démonstrations π_i et si $C = A$ avec la surrègle d'introduction de R à la racine de la démonstration.

Démonstration. Les surrègles droites sont essentiellement les mêmes que les surrègles d'introduction. Il faut toutefois faire attention au fait qu'on a plusieurs conclusions. On procède par induction sur P.

– Si P est atomique, alors la dernière inférence de la démonstration est en fait

$$\vdash R \frac{\Gamma \vdash P, \Delta}{\Gamma \vdash A, \Delta}$$

Si on a une démonstration π de P à partir de Γ, alors

$$\overset{\Gamma}{\boxed{\pi}}\ \underset{R}{\vdash} \frac{P}{A}$$

est la démonstration attendue. Sinon on a une démonstration π d'une certain $C \in \Delta$ à partir des hypothèses Γ, qui convient également.

– Si P est de la forme $P_1 \Rightarrow P_2$, soit R' la règle de réécriture $D \to P_2$. Par construction des surrègles droites,

$$\vdash R' \frac{(\Gamma_i \vdash \Delta_i)_i}{\Gamma, P_1 \vdash D}$$

est une instance de $\vdash R'$. Remarquons qu'il n'y a que D dans la partie droite de la conséquence.

Supposons que pour tout i nous ayons une démonstration π_i d'un certain $C_i \in \Delta_i$ à partir des hypothèses Γ_i. Par hypothèse d'induction il existe une démonstration ϖ de D construite à partir des hypothèses Γ, P_1, des démonstrations π_i avec la surrègle d'introduction de R' en dernier.

La surrègle d'introduction de R' est de la forme

$$\underset{R'}{\vdash} \frac{[A_1^1, \cdots, A_1^{p_1}] \qquad [A_m^1, \cdots, A_m^{p_m}]}{B_1 \qquad\qquad B_m \qquad\qquad D}$$

et dans ce cas celle de R est de la forme

$$\underset{R}{\vdash} \frac{[P_1, A_1^1, \cdots, A_1^{p_1}] \qquad [P_1, A_m^1, \cdots, A_m^{p_m}]}{B_1 \qquad\qquad B_m \qquad\qquad A}$$

En remplaçant $\underset{R'}{\vdash}$ par $\underset{R}{\vdash}$ à la racine de ϖ on obtient une démonstration de A construite à partir des hypothèses Γ, des démonstrations π_i et de la surrègle d'introduction de R à la racine. (L'hypothèse P_1 a été déchargée par l'application de $\underset{R}{\vdash}$.)

Remarque 3.11 *:* C'est ici qu'intervient la différence entre LB^+ et LK^+. Si on autorise plusieurs formules à droite dans les prémisses de la règle droite pour $\Rightarrow$, on n'a plus nécessairement une démonstration de D à partir de Γ, P_1, et on ne peut donc plus conclure. Ceci explique pourquoi avec le système de réécriture $A \to (B \Rightarrow C) \wedge D$ on peut démontrer la séquence $D \vdash A, B$ dans LK^+ mais pas dans LB^+, pas plus que A ou B à partir de l'hypothèse D dans NJ^+.

– Si P est de la forme $P_1 \wedge P_2$, soit R_1 la règle de réécriture $A_1 \to P_1$ et R_2 la règle $A_2 \to P_2$. Par construction des règles droites, il existe I_1 et I_2 telle que pour $k \in \{1; 2\}$ $\vdash_{R_k}$ soit de la forme

$$\vdash_{R_k} \frac{(\Gamma_i \vdash \Delta_i)_{i \in I_k}}{\Gamma \vdash A_k, \Delta}$$

et $\vdash_R$ soit alors

$$\vdash_R \frac{(\Gamma_i \vdash \Delta_i)_{i \in I_1} \qquad (\Gamma_i \vdash \Delta_i)_{i \in I_2}}{\Gamma \vdash A, \Delta}$$

Supposons que pour tout $i \in I_1 \cup I_2$ nous ayons des démonstrations π_i d'un certain $C_i \in \Delta_i$ à partir des hypothèses Γ_i. Par hypothèse d'induction, il existe $C_k \in A_k, \Delta$ possédant une démonstration ϖ_k construite à partir des hypothèses Γ, des démonstrations π_i et si $C_k = A_k$ de de la surrègle d'introduction de R_k à la racine. Si C_1 ou C_2 est dans Δ, alors ϖ_k convient.
Sinon, la surrègle d'introduction de R_k est de la forme

$$\underset{R_k}{\vdash} \frac{[A^1_{k,1}, \cdots, A^{p_1}_{k,1}] \qquad [A^1_{k,m}, \cdots, A^{p_m}_{k,m}]}{B_{k,1} \qquad\qquad B_{k,m}}{A_k}$$

et dans ce cas celle de R est de la forme

$$\underset{R}{\vdash} \frac{\begin{pmatrix} [A^1_{k,j}, \cdots, A^{p_j}_{k,j}] \\ B_{k,j} \end{pmatrix}_{k,j}}{A}$$

Si on remplace $\underset{R_1}{\vdash}$ et $\underset{R_2}{\vdash}$ par $\underset{R}{\vdash}$ aux racines de ϖ_1 et ϖ_2, on obtient une démonstration de A construite à partir des hypothèses Γ, des démonstrations π_i et de la surrègle d'introduction de R à la racine.

– Si P est de la forme $\forall x.\, Q$, soit R' la règle de réécriture $D(x) \to Q$. En supposant que x n'est pas libre dans Γ, par construction de la règle droite

$$\vdash_{R'} \frac{(\Gamma_i \vdash \Delta_i)_i}{\Gamma \vdash D(x)}$$

est une instance de $\vdash_{R'}$.
Supposons que pour tout i nous ayons une démonstration π_i d'un certain $C_i \in \Delta_i$ à partir des hypothèses Γ_i. Par hypothèse d'induction il existe une démonstration ϖ de $D(x)$ construite à partir des hypothèses Γ, des démonstrations π_i avec la surrègle d'introduction de R' en dernier.

La surrègle d'introduction de R' est de la forme

$$\overset{\vdash}{}_{R'} \frac{[A_1^1, \cdots, A_1^{p_1}] \qquad [A_m^1, \cdots, A_m^{p_m}]}{\begin{array}{cc} B_1 & B_m \end{array} \qquad D(x)}$$

et dans ce cas celle de R est de la forme

$$\overset{\vdash}{}_{R} \frac{[A_1^1, \cdots, A_1^{p_1}] \qquad [A_m^1, \cdots, A_m^{p_m}]}{\begin{array}{cc} B_1 & B_m \end{array} \qquad A}$$

Comme on suppose que x n'est pas libre dans Γ, on peut remplacer $\overset{\vdash}{}_{R'}$ par $\overset{\vdash}{}_{R}$ dans Γ pour obtenir une démonstration de A construite à partir des hypothèses Γ, des démonstrations π_i et de la surrègle d'introduction de R à la racine.

□

On peut alors étendre la traduction de la proposition 2.10.

Proposition 3.27 (Traduction de LB^+ dans NJ^+)**.** *Une démonstration de* $\Gamma \vdash \Delta$ *dans* LB^+ *peut être traduite en une démonstration d'un certain* $C \in \Delta$ *avec comme hypothèses un sous-ensemble de* Γ *en déduction surnaturelle.*

De plus, si la démonstration en calcul des séquences n'utilise pas de règles logiques, c'est également le cas de la démonstration en déduction surnaturelle.

Démonstration. On étend la démonstration de la proposition 2.10. On utilise le lemme 3.25 pour traduire les surrègles gauche et le lemme 3.26 pour traduire les surrègles droite. On a besoin également d'une extension du lemme 2.7 pour se ramener à des séquences à une seule conclusion pour traduire les règles logiques.

La traduction n'ajoute pas de règles logiques comme on peut le remarquer dans les lemmes 3.25 et 3.26, sauf pour la traduction de $\overset{\vdash}{}_{\smile}$ qui rajoute une coupure $\frac{\overset{\vdash}{\Rightarrow}}{\overset{\Rightarrow}{\vdash}}$. On fait une étape de β-réduction sur la démonstration pour supprimer ces règles logiques. □

3.3.3 Propriétés des systèmes de surdéduction intuitionnistes

Nous allons nous intéresser au cas où les nouvelles règles d'inférence ne contiennent que des propositions atomiques. Pour cela, les règles de réécriture doivent être d'une certaine forme que nous appellerons atomisante.

Définition 3.15 (Règle de réécriture atomisante)**.** *Une règle de réécriture propositionnelle est dite* atomisante *si elle respecte la grammaire*

$$P \overset{\text{déf}}{=} A \mid A \Rightarrow P \mid P \wedge P \mid \forall x.\ P$$

où A *désigne une proposition atomique.*

Un système de réécriture est dit atomisant s'il est constitué de règles propositionnelles atomisantes et de règles sur les termes.

Proposition 3.28. *Les surrègles associées à une règle de réécriture propositionnelle en déduction surnaturelle ou en calcul des séquences extensible ne contiennent que des propositions atomiques et des variables de contexte.*

Démonstration. Par construction des surrègles. □

Proposition 3.29. *Si on se place dans un système de réécriture atomisant, une séquence composée uniquement de propositions atomiques est démontrable en* LB^+ *ssi elle l'est en utilisant uniquement des règles d'identité et des surrègles, avec* $\vdash\!\!\smile$ *appliquée uniquement sur des propositions atomiques.*

Démonstration. Comme indiqué plus haut, le procédure d'élimination des coupures définie dans la section 2.3.4 peut ne pas terminer, y compris pour des systèmes atomisants. On peut néanmoins l'utiliser pour réduire toutes les coupures en coupures autour de formules atomiques, ce qui montre que la règle $\vdash\!\!\smile$ générale est admissible dans le système avec la règle $\vdash\!\!\smile$ restreinte aux cas où X est instancié par une formule atomique.

Alors, à partir d'une séquence ne contenant que des propositions atomiques, on ne peut appliquer que des surrègles ou des coupures atomiques. Comme le système est atomisant, les prémisses obtenues ne contiennent donc elles aussi que des propositions atomiques. Par simple induction structurelle on voit que l'on ne pourra jamais utiliser de règle logique. □

Théorème 3.30.
Si on se place dans un système de réécriture atomisant, une séquence intuitionniste composée uniquement de propositions atomiques est démontrable en déduction surnaturelle ssi elle l'est en utilisant uniquement des surrègles et $\frown\!\!\vdash$.

Démonstration. Si la séquence est démontrable en déduction surnaturelle, on peut traduire la démonstration en calcul des séquences extensible à l'aide de la proposition 3.24. La proposition précédente permet de se ramener à une démonstration sans règle logique, et la proposition 3.27 nous donne alors une démonstration en déduction surnaturelle utilisant uniquement des surrègles et $\frown\!\!\vdash$. □

Il est possible qu'on puisse démontrer le précédent théorème sans recourir à des traductions avec le calcul des séquences extensible. Ainsi, Gilles Dowek (1996) propose une démonstration d'un théorème similaire dans un système qui ressemble à un système de déduction surnaturelle, et dans lequel sont distinguées des règles d'inférence externes, correspondant aux règles logiques de la déduction naturelle, et des règles internes, similaires à des surrègles. Gilles Dowek montre que la réduction des coupures entre règles externes termine, ce qui est similaire à notre idée de se ramener à des coupures atomiques en calcul des séquences. La démonstration consiste à traduire les démonstration vers la déduction naturelle pure, en remplaçant toutes les propositions atomiques apparaissant

dans une règle de réécriture propositionnelle par une nouvelle proposition atomique E, puis en traduisant les règles internes par des dérivations avec cette proposition atomique E et en enrichissant les hypothèse avec E, par exemple une surrègle binaire

$$\mathsf{r}\,\frac{\Gamma, A_1, \ldots, A_n \vdash A \qquad \Gamma, B_1, \ldots, B_n \vdash B}{\Gamma \vdash C}$$

devient

$$\vdash_{\wedge}\,\frac{\Gamma, E, \ldots, E \vdash E \qquad \Gamma, E, \ldots E \vdash E}{\vdash_{\Rightarrow}\,\dfrac{\Gamma, E, \ldots, E \vdash E \wedge E}{\Rightarrow_{\vdash}\,\dfrac{\Gamma, E \vdash E \Rightarrow \ldots \Rightarrow (E \wedge E) \qquad \widehat{\vdash}\,\dfrac{}{\Gamma, E \vdash E}}{\wedge_{\vdash}\,\dfrac{\Gamma, E \vdash E \wedge E}{\Gamma, E \vdash E}}}}$$

Les β-réduction entre règles externes sont donc également présentes dans la traduction en déduction naturelle pure. Comme la β-réduction termine en déduction naturelle, on en déduit que la réduction des règles externes termine aussi. Il est à noter que la traduction n'est correcte que si on a des surrègles ne comportant que des proposition atomiques. Comme corollaire, on montre que si on n'a que des propositions atomiques, on n'a pas besoin de règles externes. Cette démonstration semble pouvoir s'adapter à la déduction surnaturelle quand le système de réécriture est atomisant, ce qui permettrait d'obtenir une démonstration du théorème 3.30 sans passer par le calcul des séquences extensible.

Le théorème 3.30 est intéressant car il permet de faire la distinction entre le système de déduction originel, avec des règles logiques, et celui correspondant à la logique engendrée par une théorie compatible avec le système de réécriture, dans lequel on n'a plus besoin de ces règles logiques. Cela suggère qu'un système de déduction associé à une théorie mathématique ne devrait pas faire appel à des notions purement logiques. Le théorème implique en effet une orthogonalité entre ce qui se passe au niveau logique, et ce qui se passe au niveau des nouvelles règles d'inférence. Ceci est également important si on veut considérer la surdéduction comme un cadre logique (cf. le chapitre 6) : le système de déduction que l'on veut simuler ne doit pas interférer avec les règles logiques.

Chapitre 4

Complétion abstraite pour l'obtention de l'admissibilité des coupures

> Calvin: Miss Wormwood, my Dad says when he was in school, they taught him to do math on a slide rule.
> He says he hasn't used a slide rule since, because he got a five-buck calculator that can do more functions than he could figure out if his life depended on it.
> Given the pace of technology, I propose we leave math to the machines and go play outside.
>
> Calvin (*disappointed*): My bills always die in subcommittee.
>
> Bill Watterson, *Calvin and Hobbes*, 25/11/1992

Nous allons maintenant proposer une méthode qui permet de transformer un système de réécriture de façon à ce que le calcul des séquences modulo le système de réécriture final admette $\vdash\!\!\!\smile$. Cette procédure est basée sur un cadre formel, celui des systèmes canoniques abstraits, qui cherche à généraliser les procédures de complétion telle la complétion standard de Donald Knuth et Peter Bendix (1970). Le principal ingrédient de ce cadre est l'utilisation d'un ordre bien fondé sur les démonstrations, qui mesure la qualité ou la simplicité d'une démonstration. Nous allons donc chercher à rendre les démonstrations sans coupure minimales dans ce formalisme.

4.1 Exemple introductif

Nous avons vu que les coupures ne sont pas toujours admissibles en déduction modulo. Toutefois, Gilles Dowek (2003) a démontré que cette admissibilité est équivalente à la

confluence du système de réécriture dans le cas où on ne considère que des règles de réécriture sur les termes. Si on a au départ un système non confluent, pour obtenir un système qui admet les coupures on peut alors appliquer la procédure de complétion standard de Donald Knuth et Peter Bendix (1970). Cette procédure a en effet pour but de transformer un système de réécriture pour le rendre confluent.

Gilles Dowek (2002) a donc proposé de définir une méthode de complétion qui permettent de retrouver l'admissibilité des coupures même dans le cas où on réécrit des formules. Il a définie une procédure dans le cas purement propositionnel qui s'appuie sur l'existence d'un modèle, et que nous décrivons maintenant.

Soit m un modèle de la théorie compatible avec le système de réécriture initial. Comme on est dans le cas propositionnel, on peut considérer qu'on a un support D avec un seul élément et que les propositions sont donc interprétées soit par $0 \stackrel{\text{déf}}{=} \emptyset$ soit par $1 \stackrel{\text{déf}}{=} D$. On considère alors que la présentation Γ de la théorie compatible est sous *forme conjonctive-disjonctive*, c'est-à-dire que c'est une conjonction de disjonctions de littéraux, un littéral étant une proposition atomique ou sa négation. On peut toujours se ramener à une présentation de cette forme. Les disjonctions sont alors appelées des *clauses*.

On prend alors une clause de Γ. Comme m est un modèle de Γ, il existe une proposition atomique A telle que A apparaissent dans Γ sans négation et $\hat{A} = 1$ ou $\neg A$ apparaissent dans Γ et $\hat{A} = 0$.

Dans le premier cas, on prend toutes les clauses où A apparaît sans négation $A \vee P_1, \ldots, A \vee P_n$ et on les remplace par la règle de réécriture polarisée

$$A \rightarrow^{+} \neg P_1 \vee \ldots \vee \neg P_n \ .$$

Dans le second cas, on prend toutes les clauses où $\neg A$ apparaît $\neg A \vee P_1, \ldots, \neg A \vee P_n$ et on les remplace par la règle de réécriture

$$A \rightarrow^{-} P_1 \wedge \ldots \wedge P_n \ .$$

On répète ceci jusqu'à ce qu'il n'y ait plus de clauses. Gilles Dowek (2002) montre que la déduction naturelle modulo le système de réécriture obtenu normalise fortement.

Exemple 4.1 *:* On applique cet algorithme à l'exemple de système de Marcel Crabbé $\mathcal{C}rab : A \rightarrow B \wedge \neg A$. Le seul modèle envisageable est $\hat{A} = 0$ et $\hat{B} = 0$.

Une présentation conjonctive-disjonctive compatible avec ce système est par exemple $(\neg A \vee B) \wedge \neg A \wedge (A \vee \neg B)$. Si on choisit la première clause $\neg A \vee B$, il faut choisir A comme proposition atomique. On considère donc l'ensemble de clauses $\neg A \vee B, \neg A$ ce qui donne la règle de réécriture $A \rightarrow^{-} B \wedge \bot$ puis on a l'ensemble singleton de clauses $\neg B \vee A$ qui donne la règle de réécriture $B \rightarrow^{-} A$.

La déduction naturelle polarisée modulo le système

$$A \rightarrow^{-} B \wedge \bot$$
$$B \rightarrow^{-} A$$

normalise donc fortement.

Les inconvénients d'une telle approche sont d'une part qu'elle nécessite un modèle, d'autre part qu'elle est difficilement extensible au cas avec des quantificateurs.

Pour cela, nous avons essayé de revenir à l'idée de généraliser une procédure de complétion du style Knuth-Bendix. Nous nous sommes appuyés sur le cadre formel des systèmes canoniques abstraits de Nachum Dershowitz et Claude Kirchner (2006); Maria Paola Bonacina et Nachum Dershowitz (2007*a*). Basés sur l'utilisation d'un ordre sur les démonstrations, ceux-ci permettent de définir des démonstrations critiques, qui peuvent être vues comme des contre-exemples minimaux à la propriété souhaitée. Dans le cas de la complétion standard, ces démonstrations critiques correspondent effectivement à des paires critiques. Dans notre cas, nous allons démontrer dans la proposition 4.14 que ces démonstrations critiques sont de la forme

$$\dfrac{\dfrac{\dfrac{\overset{\pi_1}{\Gamma, A, P \vdash \Delta}}{\Gamma, A \vdash \Delta}\uparrow\vdash}{\Gamma \vdash \Delta}\smile\vdash \qquad \dfrac{\overset{\pi_2}{\Gamma \vdash Q, A, \Delta}}{\Gamma \vdash A, \Delta}\vdash\uparrow}{}$$

avec π_1 et π_2 sans coupure. Pour pouvoir éliminer cette coupure, il suffirait de pouvoir démontrer $\Gamma \vdash \Delta$ sans coupures. Une idée est alors de rajouter des règles de réécriture qui permettront de fermer une démonstration de $\Gamma \vdash \Delta$ tout en restant correcte par rapport au système initial.

Exemple 4.2 *:* Pour le système $\mathcal{C}rab$, la seule démonstration critique intéressante est la suivante

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B, A, B \vdash A}\frown\vdash}{B, A, B, \neg A \vdash}\neg\vdash}{B, A, B \wedge \neg A \vdash}\wedge\vdash}{B, A \vdash}\uparrow\vdash}{\vdash A}\smile\vdash \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B, A}\frown\vdash \qquad \dfrac{\dfrac{}{B, A \vdash A}\frown\vdash}{B \vdash \neg A, A}\vdash\neg}{B \vdash B \wedge \neg A, A}\vdash\wedge}{B \vdash A}\vdash\uparrow}{}$$

Pour obtenir une démonstration de $B \vdash$ sans coupure, on peut par exemple rajouter la règle de réécriture $B \to^{-} \bot$. On peut alors montrer que le calcul des séquences modulo le système de réécriture

$$\begin{aligned} A &\to^{\pm} B \wedge \neg A \\ B &\to^{-} \bot \end{aligned}$$

admet les coupures.

Pour démontrer qu'une telle approche est correcte, on démontre que le calcul des séquences modulo, ou plus précisément le calcul des séquences avec dépliage polarisé,

peut être vu comme une instance de système canonique abstrait. L'admissibilité des coupures dans le calcul des séquence modulo le système final est alors un simple corollaire de la complétude de la méthode de complétion abstraite définie dans le cadre des systèmes canonique abstraits.

4.2 Systèmes canoniques abstraits

Les systèmes canoniques abstraits ont été introduits dans le but de définir un cadre général et formel qui englobe les différentes procédures ressemblant à la complétion standard dans lesquels on voit apparaître des notions de complétude, de saturation et de redondances. Parmi ces procédures, on retrouve bien entendu la complétion standard de Donald Knuth et Peter Bendix (1970) mais également la complétion close de Wayne Snyder (1989), la complétion modulo équations de Jean-Pierre Jouannaud et Hélène Kirchner (1986) et Gerald Peterson et Mark Stickel (1981), la complétion ordonnée initiée par Dallas Lankford (1975) et obtenue finalement par Jieh Hsiang et Michaël Rusinowitch (1991), la génération de bases de Gröbner par l'algorithme de Bruno Buchberger (1985), l'amélioration de bases de connaissances représentées sous forme de familles de Moore (Karell Bertet et Mirabelle Nebut, 2004), etc. Les systèmes canoniques abstraits sont fondés sur l'utilisation d'un ordre sur les démonstrations, ce qui reprend une idée développée par Leo Bachmair (1987, 1991) avec l'aide de Nachum Dershowitz (1989, 1994), Harald Ganzinger (1994), Jieh Hsiang (1986) et David Plaisted (1989).

Nachum Dershowitz (2003) a démontré que la complétion close rentrait dans ce cadre, nous avons fait de même avec Claude Kirchner pour la complétion standard (2006), et Maria Paola Bonacina et Nachum Dershowitz (2007*b*, 2008) se sont intéressés aux familles de Moore et aux théories de Horn.

4.2.1 Définitions et postulats

On se place dans un système de déduction $(\mathbb{A}, \mathbb{P}, [\cdot]^{Pm}, [\cdot]_{Cl})$ tel qu'introduit par la définition 2.14. Rappelons que cela implique que $Th : A \mapsto [[A]^{Pm^{-1}}]_{Cl}$ est un opérateur de fermeture.

On munit alors l'ensemble des démonstrations de deux ordres bien fondés $>$ et $\rhd$, le premier mesurant la qualité des démonstrations, le second représentant leur structure et étant par conséquent appelé *ordre de sous-démonstration*. Sans précision supplémentaire, quand on parlera de l'*ordre sur les démonstrations* on fera référence au premier. Par convention, on supposera qu'il compare uniquement des démonstrations ayant la même conclusion ($p > q$ implique $[p]_{Cl} = [q]_{Cl}$).

Exemple 4.3 *:* Pour donner des intuitions, nous illustrerons cette section avec la complétion standard, mais sans rentrer dans les détails qui pourront être trouvés dans notre

travail avec Claude Kirchner (2006).

Pour la complétion standard, les formules sont soit des règles de réécriture, soit des équations entre termes. Les démonstrations sont des démonstrations par réécriture : par exemple $a \underset{s\to t}{\longleftarrow} b \underset{u\to v}{\longrightarrow} c \underset{e=f}{\longleftrightarrow} d$ est une démonstration de $a = d$ dans $Pf(\{s \to t; u \to v; e = f\})$. L'ordre sur les démonstrations est choisi de telle sorte que la démonstration $a \underset{s\to t}{\longleftarrow} b \underset{u\to v}{\longrightarrow} c$ soit plus grande que $a \underset{a=c}{\longleftrightarrow} c$, elle-même plus grande qu'une démonstration de la forme $a \overset{*}{\longrightarrow} \overset{*}{\longleftarrow} c$.

Définition 4.1 (Démonstration triviale)**.** *Une démonstration p est dite* triviale *si elle n'a pas de sous-démonstration et démontre son unique hypothèse, c'est-à-dire $p \trianglerighteq q$ implique $p = q$ et $[p]^{Pm} = \{[p]_{Cl}\}$.*

Pour chaque formule $a \in \mathbb{A}$ on notera $\widehat{a}$ une démonstration triviale de conclusion a, qu'on suppose exister. Pour une présentation A, on notera $\widehat{A}$ l'ensemble des démonstration triviale des formules de A.

Exemple 4.4 *:* Pour la complétion standard, la démonstration triviale de $s \to t$ est $s \underset{s\to t}{\longrightarrow} t$.

Il est à noter que les démonstrations constituées de zéro étapes de réécriture, par exemple pour démontrer $s = s$, ne sont pas des démonstrations triviales, car elles n'ont pas d'hypothèses.

On suppose alors que la structure dont on dispose vérifie trois postulats.

Postulat A (Trivialité)**.**
Pour toutes démonstration p et formule a, si $a \in [p]^{Pm}$ alors $p \trianglerighteq \widehat{a}$.

Postulat B (Monotonie des hypothèses des sous-démonstrations)**.**
Pour toutes démonstrations p et q, si $p \trianglerighteq q$ alors $[p]^{Pm} \supseteq [q]^{Pm}$.

Postulat C (Remplacement)**.**
Pour toutes démonstrations p, q et r : si $p \rhd q > r$ alors il existe une démonstration $v \in Pf([p]^{Pm} \cup [r]^{Pm})$ telle que $p > v \rhd r$.

Le premier postulat énonce qu'une démonstration utilise vraiment ses hypothèses, le deuxième que les sous-démonstrations n'utilisent pas plus d'hypothèses que la démonstration en question, et le dernier que l'on peut remplacer une sous-démonstration par une démonstration plus petite et obtenir une démonstration plus petit que l'originelle. Nous avons démontré (2005) que l'on peut en fait se passer du postulat B, pour pouvoir l'appliquer par exemple à des systèmes intégrant le théorème de déduction et déchargeant par conséquent des hypothèses, comme la déduction naturelle. Toutefois, nous n'utiliserons pas ce fait ici.

Définition 4.2. *On note μP l'ensemble des démonstrations minimales de la justification P :*

$$\mu P \stackrel{\text{déf}}{=} \{p \in P \,:\, \neg\exists q \in P,\ p > q\} \ .$$

Une démonstration est en forme normale *pour une présentation A si elle est minimale quelle que soit la présentation de $Th\,A$. On dénote l'ensemble des démonstrations en forme normale pour A par $Nf(A)$ qui peut donc être défini par*

$$Nf(A) \stackrel{\text{déf}}{=} \mu Pf(Th\,A) \ .$$

Exemple 4.5 *:* Pour la complétion standard, les démonstration en forme normale sont alors les démonstrations en vallée, c'est-à-dire de la forme $s \xrightarrow{*} \xleftarrow{*} t$.

Étant donnée une théorie, le but est d'en trouver une présentation telle qu'on puisse se contenter des démonstrations minimales pour obtenir la théorie en entier. C'est ce qu'on va appeler la complétude

Définition 4.3 (Complétude)**.** *Une présentation A est dite* complète *si on peut démontrer toute la théorie en se contentant des démonstrations en forme normale, c'est-à-dire si*

$$Th\,A = [Pf(A) \cap Nf(A)]_{Cl} \ .$$

Exemple 4.6 *:* Pour la complétion standard, un système est complet si toutes les égalités démontrables dans ce système le sont en utilisant une démonstration en vallée, ce qui est équivalent à dire que le système est confluent.

4.2.2 Mécanismes de complétion

On peut maintenant définir des mécanismes de déduction qui permettent de transformer les présentations dans le but de les rendre complètes.

Définition 4.4. *Un* mécanisme de déduction *est une fonction des présentations dans les présentations. On notera $A \rightsquigarrow B$ si B est l'image de A par le mécanisme de déduction $\rightsquigarrow$.*

Une suite $(A_n)_{n\in\mathbb{N}}$ de présentations telles que $A_0 \rightsquigarrow A_1 \rightsquigarrow \cdots$ est appelées une dérivation*.*

Le résultat *A_∞ d'une dérivation $(A_n)_{n\in\mathbb{N}}$ est l'ensemble des formules persistantes défini par*

$$A_\infty \stackrel{\text{déf}}{=} \bigcup_{j>0} \bigcap_{i>j} A_i \ .$$

On peut alors définir un mécanisme de complétion, celui qui permettra d'obtenir des présentations complètes.

Définition 4.5 (Mécanisme de complétion)**.** *Un mécanisme de déduction $\rightsquigarrow$ est dit de complétion si*

- *si $A \rightsquigarrow B$, alors $Th\, A = Th\, B$;*
- *si $A \rightsquigarrow B$, alors pour toute démonstration $p \in Pf(A)$ il existe une démonstration $q \in Pf(B)$ qui est au moins aussi bien, c'est-à-dire $p \geq q$;*
- *pour toute dérivation $(A_n)_{n\in\mathbb{N}}$, la limite A_∞ est complète.*

Les mécanismes de complétion sont bien ceux que l'on attend, car leurs limites peuvent démontrer toute la théorie initiale en n'utilisant que des démonstrations en forme normales

Théorème 4.1 (Maria Paola Bonacina et Nachum Dershowitz (2007*a*, Lemma 5.13))**.**
Un mécanisme est de complétion ssi pour toute dérivation $(A_n)_{n\in\mathbb{N}}$ on a

$$Th\, A_0 = [Pf(A_\infty) \cap Nf(A_0)]_{Cl} \ .$$

Nachum Dershowitz et Claude Kirchner (2003) définissent un mécanisme de complétion particulier, basé sur la notion de démonstration critique.

Définition 4.6 (Démonstration critique)**.** *Une démonstration est dite* critique *pour une présentation A si elle est minimale pour la présentation A sans être en forme normale, et toutes ses sous-démonstrations sont en forme normale. On notera $Crit(A)$ l'ensemble des démonstrations critiques de A, qui est donc défini par*

$$Crit(A) \stackrel{\text{déf}}{=} \{p \in \mu Pf(A) \setminus Nf(A) \ : \ \forall q \in Pf(A),\ p \rhd q \Rightarrow q \in Nf(A)\} \ .$$

Exemple 4.7 *:* Pour la complétion standard, les démonstrations critiques sont les paires critiques non joignables, c'est-à-dire des démonstrations de la forme $u \underset{s[v]\to u}{\longleftarrow} s[v] \underset{v\to w}{\longrightarrow} s[w]$ telles qu'il n'existe pas de démonstration en vallée $u \xrightarrow{*} \xleftarrow{*} s[w]$. La complétion standard consiste alors à ajouter l'équation $u = s[w]$ dans la présentation, de façon à ce qu'il soit possible de construire la démonstration plus petite $u \underset{u=s[w]}{\longleftrightarrow} s[w]$.

Pour compléter une présentation, il suffit donc de lui ajouter la possibilité de construire des démonstrations plus petites que les démonstrations critiques. Une présentation complétante permettra ceci.

Définition 4.7 (Présentation complétante)**.** *Une présentation sera dite* complétante *pour A si elle est incluse dans la théorie de A et si elle contient les hypothèses de démonstrations strictement plus petites que les démonstrations critiques de A. En général on notera $C(A)$ une telle présentation. Cela revient à dire qu'il existe un ensemble de*

démonstrations $P(A)$ telles que pour toute démonstration critique de A il existe une démonstration plus petite dans $P(A)$, et

$$\bigcup_{p \in P(A)} \{[p]^{Pm}\} \subseteq C(A) \subseteq Th\, A \ .$$

Le mécanisme de déduction est alors défini par $A \rightsquigarrow A \cup C(A)$. On peut démontrer que c'est bien un mécanisme de complétion.

Proposition 4.2 (Nachum Dershowitz et Claude Kirchner (2003, Lemme 10))**.** *Ce mécanisme est un mécanisme de complétion.*

4.3 Le calcul des séquences avec dépliage polarisé comme instance de système canonique abstrait

Nous allons maintenant montrer que le calcul des séquences avec dépliage polarisé peut être vu comme un système canonique abstrait dont les démonstrations en forme normale seront les démonstrations sans coupures. Ceci va nous permettre de définir une procédure de complétion qui transformera un système de réécriture pour que le calcul des séquences avec dépliage polarisé admette les coupures.

Nous avons déjà défini le calcul des séquences avec dépliage polarisé dans la section 3.2.1, avec les règles d'inférence de la figure 3.5. On peut considérer que les règles de dépliage sont indexées par une règle de réécriture $s \to^{\pm} t$ propositionnelle ou sur les termes.

Au niveau des termes, Gilles Dowek (2003) a démontré que la confluence équivalait à l'admissibilité des coupures. On supposera donc que le système de réécriture sur les termes $\mathcal{R}_{\mathcal{T}(\Sigma,V)}$ est confluent et terminant. On rappelle que les notations $\Gamma \vdash_{\mathcal{R}} \Delta$, $\Gamma \vdash^*_{\mathcal{R}} \Delta$ et $\Gamma \vdash \Delta$ ont été introduite par la définition 3.6.

Rappelons que nous dirons qu'un système de réécriture admet les coupures si le calcul des séquences modulo ce système les admet, ce qui revient à dire que le calcul des séquences avec dépliage polarisé basé sur ce système les admet (cf. section 3.2.1).

Définition 4.8. *Un système de réécriture $\mathcal{R}$ admet $\overset{\smile}{\vdash}$ si $\Gamma \vdash_{\mathcal{R}} \Delta$ implique $\Gamma \vdash^*_{\mathcal{R}} \Delta$ pour toute séquence $\Gamma \vdash \Delta$.*

4.3.1 Système de réécriture et séquence

Ce sont essentiellement des systèmes de réécriture que nous voulons compléter ici, puisque nous voulons obtenir au final un système de réécriture qui admette $\overset{\smile}{\vdash}$. Dans le système canonique abstrait représentant le calcul des séquences avec dépliage, les formules devraient donc être des systèmes de réécriture. Néanmoins, les conclusions des démonstrations sont des séquences, et il nous faut donc établir un lien entre ces séquences

et les systèmes de réécriture. Nous établissons tout d'abord les propriétés nécessaires que doit avoir un tel lien pour que l'on ait effectivement une instance de système canonique abstrait. Puis nous donnons un algorithme qui permet de transformer les séquences en système de réécriture. Au passage, cet algorithme permet également de transformer toute présentation en logique classique du premier ordre d'une théorie en système de réécriture compatible que l'on pourra utiliser en déduction modulo ou en surdéduction.

Propriétés

On suppose qu'il existe une fonction Rew de l'ensemble des séquences vers l'ensemble des systèmes de réécriture. Cette fonction doit vérifier les hypothèses suivantes :

1. toute règle de réécriture polarisée appartient à l'image d'au moins une séquence par Rew ;
2. pour toute séquence $\Gamma \vdash \Delta$, la présentation $\mathcal{P}(\Gamma \vdash \Delta)$ est fortement compatible avec le système de réécriture $Rew(\Gamma \vdash \Delta)$;
3. pour tout système de réécriture $\mathcal{R}$, pour toute séquence $\Gamma \vdash \Delta$, si pour toute règle de réécriture $R \in Rew(\Gamma \vdash \Delta)$ il existe une séquence $\Gamma' \vdash \Delta'$ telle que $R \in Rew(\Gamma' \vdash \Delta')$ et $\Gamma' \vdash^*_{\mathcal{R}} \Delta'$, alors $\Gamma \vdash^*_{\mathcal{R}} \Delta$.

Rappelons que $\mathcal{P}(\Gamma \vdash \Delta) \stackrel{\text{déf}}{=} \forall x_1, \ldots, x_n,\ (\bigwedge \Gamma) \Rightarrow (\bigvee \Delta)$ où $\{x_1, \ldots, x_n\} = FV(\Gamma, \Delta)$.

La deuxième propriété permet de s'assurer que le système de réécriture obtenu correspond bien à la séquence initiale.

Un des intérêts de la compatibilité est le lemme suivant.

Lemme 4.3. *Si Θ est fortement compatible avec $\mathcal{R}_1$, alors pour toute séquence $\Gamma \vdash \Delta$ on a $\Gamma \vdash_{\mathcal{R}_1 \cup \mathcal{R}_2} \Delta$ ssi $\Gamma, \Theta \vdash_{\mathcal{R}_2} \Delta$.*

Démonstration. Supposons que $A \underset{B \to^- Q}{\longrightarrow} P$ pour $(B \to^- Q) \in \mathcal{R}_1$. Par hypothèse de compatibilité on a $\Theta \vdash A \Rightarrow P$. Par inversibilité de la règle $\vdash\Rightarrow$ il existe une démonstration π de $\Theta, A \vdash P$. On peut donc remplace les instance de $\uparrow\vdash$

$$\uparrow\vdash \frac{\Gamma, A, P \vdash \Delta}{\Gamma, A \vdash \Delta}$$

par la dérivation

$$\smile\vdash \frac{\Gamma, A, \Theta, P \vdash \Delta \qquad \Gamma, \Theta, A \overset{\pi}{\vdash} P, \Delta}{\Gamma, \Theta, A \vdash \Delta}$$

On procède dualement pour les instances de $\vdash\uparrow$.

Si on applique ceci à toutes les instances de règles de dépliage pour des règles de $\mathcal{R}_1$, on transforme une démonstration de $\Gamma \vdash \Delta$ dans $\mathcal{R}_1 \cup \mathcal{R}_2$ en démonstration de $\Gamma, \Theta \vdash \Delta$ dans $\mathcal{R}_2$.

Réciproquement, par hypothèse de compatibilité on a $\vdash_{\mathcal{R}_1} P$ donc pour toute formule P de T, donc on peut appliquer $\overset{\cup}{\vdash}$ avec ces démonstrations pour transformer une démonstration de $\Gamma, T \vdash \Delta$ dans $\mathcal{R}_2$ en $\Gamma \vdash \Delta$ en $\mathcal{R}_1 \cup \mathcal{R}_2$. □

Avoir la compatibilité forte permet également d'avoir des démonstrations sans coupure de toute règle de réécriture.

Lemme 4.4. *Pour toute séquence* $\Gamma \vdash \Delta$*, on a* $\Gamma \vdash^*_{Rew(\Gamma \vdash \Delta)} \Delta$.

Démonstration. Il suffit d'utiliser l'inversibilité des règles $\vdash\forall$, $\vdash\Rightarrow$, $\vdash\vee$ et $\wedge\vdash$ sur la démonstration sans coupure de $\mathcal{P}(\Gamma \vdash \Delta)$ dans $Rew(\Gamma \vdash \Delta)$. □

Algorithme de transformation des séquences en système de réécriture

Si on veut effectivement utiliser la fonction *Rew* dans la procédure de complétion, il faut la décrire comme un algorithme. Nous proposons donc l'algorithme qui suit, qui n'est peut être pas le meilleur, mais qui a les propriétés attendues. L'idée est d'appliquer les règles de G4 jusqu'à obtenir une proposition atomique quelque part, et de transformer la séquence avec cette proposition atomique en règle de réécriture.

On propose ici une stratégie qui permet de faire ceci, mais n'importe quelle autre fonctionnerait également. Pour être plus précis, on décrit l'algorithme non déterministe de la façon suivante. Pour être sûr de terminer, on n'appliquera les règles $\forall\vdash$ et $\vdash\exists$ qu'une seule fois par formule. On agit sur des multiensembles de séquences, et on produit des règles de réécriture propositionnelles. On suppose qu'on peut souligner les formules pour indiquer qu'elles ont déja été décomposées. On a alors trois étapes :

1. On choisit une séquence. Dans cette séquence, on supprime les négations en tête des formules en les faisant passer de l'autre côté de $\vdash$. Par exemple $A, \neg B \vdash \neg C, \neg\neg D$ devient $A, C \vdash B, D$. Si il n'y plus de formule non soulignée à gauche, on passe à l'étape 2, si il n'y en a plus à droite, on passe à l'étape 3, sinon on choisit l'une ou l'autre.
2. On choisit une formule non soulignée à droite et on passe toutes les autres à gauche en les niant. Par exemple $P_1, \ldots, P_n \vdash Q_1, \ldots, Q_m$ devient $P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_{m-1} \vdash Q_m$. Puis on décompose Q jusqu'à tomber sur une négation, auquel cas on retourne à l'étape 1 :

$$\begin{array}{rcl}
P_1, \ldots, P_n \vdash Q_1 \wedge Q_2 & \text{devient} & P_1, \ldots, P_n \vdash Q_1 \ ; \ P_1, \ldots, P_n \vdash Q_2 \\
P_1, \ldots, P_n \vdash Q_1 \vee Q_2 & " & P_1, \ldots, P_n, \neg Q_1 \vdash Q_2 \\
P_1, \ldots, P_n \vdash Q_1 \Rightarrow Q_2 & " & P_1, \ldots, P_n, Q_1 \vdash Q_2 \\
P_1, \ldots, P_n \vdash \forall x,\ Q & " & P_1, \ldots, P_n \vdash \{y/x\}Q \\
P_1, \ldots, P_n \vdash \exists x,\ Q & " & P_1, \ldots, P_n, \neg\underline{\exists x,\ Q} \vdash \{t/x\}Q \\
P_1, \ldots, P_n \vdash A & " & A \rightarrow^+ \underline{\exists x_1, \ldots, x_p.\ (P_1 \wedge \cdots \wedge P_n)}
\end{array}$$

où $y \notin FV(P_1, \ldots, P_n, Q)$, t est un terme quelconque, A est atomique et $\{x_1, \ldots, x_p\} = FV(P_1, \ldots, P_n) \setminus FV(A)$. Pour simplifier, on prendra $t = x$, mais l'algorithme pourrait fonctionner pour d'autres choix, qui pourraient d'ailleurs se révéler plus pertinents en fonction de la séquence qu'on est en train de transformer. On voit que dans le cas de la conjonction on produit deux séquences, l'algorithme s'applique alors ensuite aux deux. Dans le cas de la proposition atomique, on supprime la séquence de l'entrée, et on ajoute la règle de réécriture propositionnelle au système que l'on renverra en sortie.

3. On procède dualement à l'étape 2, en faisant passer toutes les formules sauf une à droite et en décomposant cette dernière jusqu'à obtenir une négation.

On recommence l'étape 1 jusqu'à ce qu'il n'y ait plus de séquence. On termine bien, puisque à chaque étape 2 ou 3 on fait décroître le nombre de connecteurs distincts de $\neg$ dans les formules non soulignées.

Nous devons maintenant démontrer que cet algorithme vérifie les propriétés attendues. Pour cela nous démontrons d'abord trois lemmes pour obtenir la compatibilité.

Lemme 4.5. *Si une séquence $\Gamma \vdash \Delta$ est transformée en une étape en l'ensemble de séquences $\{\Gamma' \vdash \Delta'\} \cup S'$ alors $\mathcal{P}(\Gamma \vdash \Delta) \vdash \mathcal{P}(\Gamma' \vdash \Delta')$.*

Démonstration. On procède par analyse de cas. Par exemple, à l'étape 2, $P_1, \ldots, P_n \vdash Q_1 \wedge Q_2$ est transformé en $P_1, \ldots, P_n \vdash Q_1; P_1, \ldots, P_n \vdash Q_2$.

Supposons que $FV(P_1, \ldots, P_n, Q_1) = \{x_1, \ldots, x_m\}$ et $FV(P_1, \ldots, P_n, Q_1 \wedge Q_2) = x_1, \ldots, x_k$; les variables libres uniquement dans Q_2 sont donc $x_{m+1}, \ldots, x_k$. On peut alors construire la démonstration suivante (seules les formules intéressantes sont indiquées dans les séquences)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Q_1, Q_2 \vdash Q_1}\,\frown\vdash}{Q_1 \wedge Q_2 \vdash Q_1}\,\wedge\vdash \qquad \dfrac{}{\bigwedge_i P_i \vdash \bigwedge_i P_i}\,\frown\vdash}{\bigwedge_i P_i \Rightarrow (Q_1 \wedge Q_2), \bigwedge_i P_i \vdash Q_1}\,\Rightarrow\vdash}{\bigwedge_i P_i \Rightarrow (Q_1 \wedge Q_2) \vdash \bigwedge_i P_i \Rightarrow Q_1}\,\vdash\Rightarrow}{\forall x_1, \ldots, x_k,\ (\bigwedge_i P_i \Rightarrow (Q_1 \wedge Q_2)) \vdash (\bigwedge_i P_i \Rightarrow Q_1)}\,\forall\vdash}{\forall x_1, \ldots, x_k,\ (\bigwedge_i P_i \Rightarrow (Q_1 \wedge Q_2)) \vdash \forall x_1, \ldots, x_m,\ (\bigwedge_i P_i \Rightarrow Q_1)}\,\vdash\forall$$

□

Lemme 4.6. *Pour toutes propositions $A, P_1, \ldots, P_n$ avec $\{x_1, \ldots, x_p\} = FV(P_1, \ldots, P_n) \setminus FV(A)$, on a $\mathcal{P}(P_1, \ldots, P_n \vdash A) \vdash (\exists x_1, \ldots, x_p,\ \bigwedge_i P_i) \Rightarrow A$ et $\mathcal{P}(A \vdash P_1, \ldots, P_n) \vdash A \Rightarrow \forall x_1, \ldots, x_p,\ \bigvee_i P_i$.*

Démonstration. Supposons que $FV(A, P_1, \ldots, P_n) = \{x_1, \ldots, x_k\}$, les variables libres de A étant donc $x_{p+1}, \ldots, x_k$. On peut construire la démonstration

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\widehat{\vdash} \quad \dfrac{}{\bigwedge_i P_i \vdash \bigwedge_i P_i}\,\widehat{\vdash}}{(\bigwedge_i P_i \Rightarrow A), \bigwedge_i P_i \vdash A}\Rightarrow\vdash}{\forall x_1, \ldots, x_k,\ (\bigwedge_i P_i \Rightarrow A), \bigwedge_i P_i \vdash A}\forall\vdash}{\forall x_1, \ldots, x_k,\ (\bigwedge_i P_i \Rightarrow A), \exists x_1, \ldots, x_p,\ \bigwedge_i P_i \vdash A}\exists\vdash}{\forall x_1, \ldots, x_k,\ (\bigwedge_i P_i \Rightarrow A) \vdash (\exists x_1, \ldots, x_p,\ \bigwedge_i P_i) \Rightarrow A}\vdash\Rightarrow$$

On vérifie que les applications de $\exists\vdash$ sont correctes, car pour $i \leq p$ la variable x_i n'est pas libre dans A.

La démonstration de l'autre séquence est duale. □

Lemme 4.7. *Pour toute proposition atomique A et pour toutes propositions $P_1, \ldots, P_n$ avec $\{x_1, \ldots, x_p\} = FV(P_1, \ldots, P_n) \setminus FV(A)$ alors*
$\vdash^*_{A \to^+ \exists x_1, \ldots, x_p.\ (P_1 \wedge \cdots \wedge P_n)} \mathcal{P}(P_1, \ldots, P_n \vdash A)$ *et* $\vdash^*_{A \to^- \forall x_1, \ldots, x_p.\ (P_1 \vee \cdots \vee P_n)} \mathcal{P}(A \vdash P_1, \ldots, P_n)$.

Démonstration. Supposons que $FV(A, P_1, \ldots, P_n) = \{x_1, \ldots, x_k\}$, les variables libres de A étant donc $x_{p+1}, \ldots, x_k$. On peut construire les démonstrations suivantes :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P_1 \vdash P_1}\,\widehat{\vdash} \quad \cdots \quad \dfrac{}{P_n \vdash P_n}\,\widehat{\vdash}}{P_1, \ldots, P_n \vdash A, P_1 \wedge \cdots \wedge P_n}\vdash\wedge}{\bigwedge_i P_i \vdash A, P_1 \wedge \cdots \wedge P_n}\wedge\vdash}{\bigwedge_i P_i \vdash A, \exists x_1, \ldots, x_p,\ (P_1 \wedge \cdots \wedge P_n)}\vdash\exists}{\bigwedge_i P_i \vdash A}\vdash\uparrow}{\vdash \bigwedge_i P_i \Rightarrow A}\vdash\Rightarrow}{\vdash \forall x_1, \ldots, x_k,\ (\bigwedge_i P_i \Rightarrow A)}\vdash\forall$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P_1 \vdash P_1}\,\widehat{\vdash} \quad \cdots \quad \dfrac{}{P_n \vdash P_n}\,\widehat{\vdash}}{A, P_1 \vee \cdots \vee P_n \vdash P_1, \ldots, P_n}\vee\vdash}{A, P_1 \vee \cdots \vee P_n \vdash \bigvee_i P_i}\vdash\vee}{A, \forall x_1, \ldots, x_p,\ (P_1 \vee \cdots \vee P_n) \vdash \bigvee_i P_i}\forall\vdash}{A \vdash \bigvee_i P_i}\uparrow\vdash}{\vdash A \Rightarrow \bigvee_i P_i}\vdash\Rightarrow}{\vdash \forall x_1, \ldots, x_k,\ (A \Rightarrow \bigvee_i P_i)}\vdash\forall$$

Dans $\vdash\exists$ et $\forall\vdash$ on instancie x_i par x_i. □

La troisième propriété est quand à elle une conséquence du lemme suivant

Lemme 4.8. *Pour tout système de réécriture propositionnel $\mathcal{R}$, si l'ensemble de séquences S est transformé en une étape en l'ensemble de séquence S', alors toutes les séquences de S ont une démonstration sans coupure dans $\mathcal{R}$ ssi toutes les séquences de S' ont une démonstration sans coupure dans $\mathcal{R}$.*

Démonstration. On procède par analyse de cas de la transformation. La direction « si » résulte du fait que les transformations sont des applications des règles d'inférence, tandis que la direction « seulement si » provient du fait que toutes les règles logiques sont inversibles (cf. lemme 2.5 étendu en section 3.2.1). C'est également le cas dans le système d'inférence sans $\leftharpoondown$.

Par exemple $P_1, \ldots, P_n \vdash \exists x.\ Q$ est transformé en $P_1, \ldots, P_n, \neg\exists x.\ Q \vdash Q$. Si $P_1, \ldots, P_n, \neg\exists x.\ Q \vdash_{\mathcal{R}} Q$, comme $\vdash\neg$ est inversible $P_1, \ldots, P_n \vdash_{\mathcal{R}} \exists x.\ Q, Q$ donc en appliquant $\vdash\exists$ on a $P_1, \ldots, P_n \vdash_{\mathcal{R}} \exists x,\ Q$. Réciproquement, si $P_1, \ldots, P_n \vdash_{\mathcal{R}} \exists x,\ Q$ alors comme $\vdash\exists$ est inversible on a $P_1, \ldots, P_n \vdash_{\mathcal{R}} \exists x.\ Q, Q$ donc on peut appliquer $\vdash\neg$ pour avoir $P_1, \ldots, P_n, \neg\exists x.\ Q \vdash_{\mathcal{R}} Q$.

□

On peut donc montrer que l'algorithme proposé a bien les propriétés attendues.

Proposition 4.9. *La fonction Rew qui a une séquence s différente de $\vdash$ associe le résultat de l'algorithme sur l'entrée $\{s\}$ possède les propriétés 1 et 2.*

Démonstration. On procède par récurrence sur le nombre d'étapes de l'algorithme.

Si $FV(P) \subseteq FV(A)$ on a $Rew(A \vdash P) = \{A \to^- P\}$ et $Rew(P \vdash A) = \{A \to^+ P\}$, ce qui montre que la première propriété est vérifiée.

Pour montrer la compatibilité forte, démontrons tout d'abord que pour toute règle de réécriture $A \to^- P$ dans $Rew(\Gamma \vdash \Delta)$, on a $\mathcal{P}(\Gamma \vdash \Delta) \vdash A \Rightarrow P$. Le lemme 4.6 donne le cas de base, quand une séquence est directement transformée en règle de réécriture. Le lemme 4.5 permet de démontrer le cas de récurrence : si $\Gamma \vdash \Delta$ est transformé en $\{\Gamma' \vdash \Delta'\} \cup \cdots$ et que $\mathcal{P}(\Gamma' \vdash \Delta') \vdash A \Rightarrow P$, comme par le lemme 4.5 on a $\mathcal{P}(\Gamma \vdash \Delta) \vdash \mathcal{P}(\Gamma' \vdash \Delta')$ on peut appliquer $\leftharpoondown$ pour obtenir $\mathcal{P}(\Gamma \vdash \Delta) \vdash A \Rightarrow P$. On fait de même pour les règles positives.

Démontrons maintenant que pour toute séquence $\Gamma \vdash \Delta$ on a $\vdash^*_{Rew(\Gamma\vdash\Delta)} \mathcal{P}(\Gamma \vdash \Delta)$. Le cas de base est une conséquence du lemme 4.7, quand une séquence est directement transformée en règle de réécriture. Pour démontrer le cas de récurrence, on utilise l'inversibilité des règles du calcul des séquences avec dépliage. Par exemple, si $\Gamma \vdash Q_1 \wedge Q_2$ est transformé en $\Gamma \vdash Q_1; \Gamma \vdash Q_2$, on suppose par hypothèse de récurrence que $\vdash^*_{Rew(\Gamma\vdash Q_i)} \mathcal{P}(\Gamma \vdash Q_i)$ pour $i \in \{1; 2\}$. Par inversibilité des règles $\vdash\forall$, $\vdash\Rightarrow$ et $\wedge\vdash$, on a $\Gamma \vdash^*_{Rew(\Gamma\vdash Q_i)} Q_i$. En appliquant $\vdash\wedge$ on a donc $\Gamma \vdash^*_{Rew(\Gamma\vdash Q_1)\cup Rew(\Gamma\vdash Q_2)} Q_1 \wedge Q_2$ puis en appliquant (potentiellement plusieurs fois) $\wedge\vdash$, $\vdash\Rightarrow$ et $\vdash\forall$, on obtient $\vdash^*_{Rew(\Gamma\vdash Q_1)\cup Rew(\Gamma\vdash Q_2)} \mathcal{P}(\Gamma \vdash Q_1 \wedge Q_2)$. Comme d'après l'algorithme $Rew(\Gamma \vdash Q_1) \cup Rew(\Gamma \vdash Q_2) = Rew(\Gamma \vdash Q_1 \wedge Q_2)$ on a bien la démonstration attendue.

Pour démontrer la troisième propriété, on utilise une induction sur le lemme 4.8, qui nous dit alors que $\Gamma \vdash^*_{\mathcal{R}} \Delta$ ssi pour chaque règle de $Rew(\Gamma \vdash \Delta)$ la séquence qui a donnée la règle est démontrable sans coupure dans $\mathcal{R}$. Soit un système de réécriture $\mathcal{R}$ et une séquence $\Gamma \vdash \Delta$. Supposons que pour chaque règle $R \in Rew(\Gamma \vdash \Delta)$ il existe une séquence $\Gamma' \vdash \Delta'$ telle que $\Gamma' \vdash^*_{\mathcal{R}} \Delta'$ et $R \in Rew(\Gamma' \vdash \Delta')$. Par conséquent, la séquence qui donne la règle R est démontrable sans coupure dans $\mathcal{R}$, pour toutes les règles $R \in Rew(\Gamma \vdash \Delta)$. On a donc $\Gamma \vdash^*_{\mathcal{R}} \Delta$ □

Cet algorithme permet donc d'intégrer toute présentation finie d'une théorie du premier ordre dans un système de réécriture modulo lequel on se place alors.

Corollaire 4.10. *Pour toute présentation du premier ordre Θ il existe un système de réécriture $\mathcal{R}$ tel que pour toute séquence $\Gamma \vdash \Delta$ et tout système de réécriture $\mathcal{R}'$ on ait $\Theta, \Gamma \vdash_{\mathcal{R}'} \Delta$ ssi $\Gamma \vdash_{\mathcal{R}' \cup \mathcal{R}} \Delta$*

Démonstration. Il suffit de prendre $\mathcal{R} \stackrel{\text{déf}}{=} \bigcup_{P \in \Theta} Rew(\vdash P)$. □

Remarque 4.1 *:* En logique classique, $\mathcal{P}(\Gamma \vdash \Delta)$ est compatible avec $Rew(\Gamma \vdash \Delta)$. Ceci n'est par contre pas vrai en logique intuitionniste : par exemple $Rew(\vdash \exists x,\ B(x)) = \{B(y) \rightarrow^+ \neg\exists x,\ B(x)\}$. Alors $\vdash \exists x,\ B(x)$ possède la démonstration suivante en calcul des séquences classique avec dépliage polarisé dans $Rew(\vdash \exists x,\ B(x))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overset{\frown}{\vdash}}{\exists x,\ B(x) \vdash \exists x,\ B(x), B(x)}}{\vdash \exists x,\ B(x), B(x), \neg\exists x,\ B(x)}\vdash\uparrow}{\vdash \exists x,\ B(x), B(x)}\vdash\exists}{\vdash \exists x,\ B(x)}\vdash\neg$$

mais ne possède pas de démonstration dans une version intuitionniste, par exemple en déduction naturelle polarisée modulo (Gilles Dowek, 2002). En règle générale, il n'est pas possible de transformer une présentation $\exists x,\ P(x)$ en un système de réécriture admettant les coupures et compatible en logique intuitionniste, car la théorie ne possède pas la propriété du témoin[1] tandis que le calcul des séquences intuitionniste modulo le système de réécriture l'aurait.

Remarque 4.2 *:* Le système de réécriture fourni par cet algorithme peut ne pas admettre $\overset{\smile}{\vdash}$, bien qu'on puisse démontrer $\mathcal{P}(\Gamma \vdash \Delta)$ sans coupure. Par exemple sur $\vdash A \Leftrightarrow (B \wedge \neg A)$ l'algorithme produit le système polarisé correspondant au système $\mathcal{C}rab$ de la section 3.2.3, qui n'admet pas les coupures.

[1] Rappelons que la propriété du témoin dit que d'une démonstration de $\exists x,\ Q(x)$ on sait exhiber un terme t tel que $Q(t)$ est démontrable.

4.3.2 Démonstrations, formules, ordres

Nous allons maintenant spécifier l'instance de système canonique abstrait correspondant au calcul des séquences avec dépliage polarisé. On fait l'hypothèse qu'on n'a que des systèmes de réécriture cohérents, pour ne pas avoir à traiter le cas de la séquence vide $\vdash$.

Les formules de cette instance seront, comme nous l'avons écrit plus haut, des règles de réécriture. Il faut donc bien distinguer les formules de la logique du premier ordre des formules du système canonique. Les démonstrations seront par contre bien celles du calcul des séquences avec dépliage polarisé. Les hypothèses d'une démonstration seront les règles de réécriture propositionnelles utilisées dans ses règles de dépliage. On aimerait que sa conclusion soit le système de réécriture obtenu en appliquant *Rew* à sa conséquence. Malheureusement il nous faut une seule formule en conclusion, donc une seule règle de réécriture. L'idée est alors de considérer plusieurs exemplaires de chaque démonstration, une pour chaque règle de réécriture de l'image par *Rew* de sa conséquence, et d'associer cette règle à la conclusion de la démonstration.

L'ordre sur les démonstrations sera similaire à celui donné dans la section 2.3.4 pour démontrer la terminaison de la procédure d'élimination des coupures dans le cas sans dépliage et étendu dans la section 3.2.1. Ce n'est probablement pas l'ordre le plus général pour avoir une instance de système canonique abstrait permettant de définir une procédure de complétion redonnant l'admissibilité des coupures. Néanmoins, cet ordre à l'avantage de bien fonctionner par rapport aux coupures, comme nous le verrons avec la forme des démonstrations critiques, et c'est également un ordre de simplification, ce qui sera utile pour démontrer le postulat C. De plus il est suffisamment fin pour avoir une caractérisation assez précise des démonstrations critiques. On veut également l'étendre pour prendre en compte la réécriture sur les termes. Ainsi, on dira que $(\vdash\!\!\smile, P) > (\vdash\!\!\smile, Q)$ si P est plus grand que Q pour l'ordre sous-formule, ou bien si $P \xrightarrow[\mathcal{R}_{\mathcal{T}(\Sigma,V)}]{*} Q$, et on étend par transitivité. Comme $\mathcal{R}_{\mathcal{T}(\Sigma,V)}$ réécrit uniquement des termes, il n'interfère pas avec l'ordre sous-formule et la précession reste bien fondée. L'ordre sur les démonstrations est donc le RPO basé sur la précession $(\vdash\!\!\smile, P) > (\vdash\!\!\smile, Q)$ et $(\vdash\!\!\frown, P) > (\vdash\!\!\frown, Q)$ si $P > Q$ pour l'ordre sous-formule ou si $P \xrightarrow[\mathcal{R}_{\mathcal{T}(\Sigma,V)}]{*} Q$, et $(\vdash\!\!\smile, P) > (\vdash\!\!\frown, Q) > (\mathsf{r}, R)$ pour toutes les règles d'inférence r autres que $\vdash\!\!\smile$ et $\vdash\!\!\frown$. Il faut ensuite restreindre cet ordre aux démonstrations qui ont la même *conséquence*, pour que le postulat C soit vérifié. Sinon, en considérant une règle étant à la fois dans $Rew(\Gamma \vdash \Delta)$ et dans $Rew(\Gamma' \vdash \Delta')$, on ne pourrait pas remplacer une démonstration de conséquence $\Gamma \vdash \Delta$ par une démonstration de conséquence $\Gamma' \vdash \Delta'$.

L'ordre de sous-démonstration le plus approprié semble celui des démonstrations du calcul des séquences. Nous voulons également prendre en compte le fait que quand on affaiblit une démonstration, on obtient en quelque sorte une sur-démonstration. On considère donc la fermeture transitive de l'union de ces deux ordres :

Définition 4.9. *On définit sur les démonstrations du calcul des séquences avec dépliage polarisé les relations suivantes :*
- $\pi \rhd_s \pi'$ *si* π' *est une sous-démonstration de* π *structurellement ;*
- $\pi \rhd_a \pi'$ *si* π *a été obtenu de* π' *par affaiblissement, autrement dit, s'il existe une formule que l'on peut élaguer dans toutes les séquences de* π *pour obtenir la démonstration valide* π' *;*
- $\rhd_d$ *est la fermeture transitive de* $\rhd_s \cup \rhd_a$.

Pour montrer que $\rhd_d$ est un ordre strict bien fondé, on a besoin du fait que $\rhd_s$ et $\rhd_a$ commutent partiellement.

Lemme 4.11. *Si* $\pi_1 \rhd_a \pi_2 \rhd_s \pi_3$ *alors il existe* π_2' *telle que* $\pi_1 \rhd_s \pi_2' \rhd_a \pi_3$.

Démonstration. π_2 est obtenu de π_1 en élaguant une formule A dans toutes ses séquences, et π_3 est une sous-démonstration de π_2. Soit π_2' la démonstration π_3 affaiblie en rajoutant A du même côté qu'on l'a supprimée dans π_2. π_2' est une sous-démonstration de π_1. □

Il reste néanmoins un problème, concernant les démonstrations triviales que l'on suppose exister. Pour qu'une démonstration utilise une formule abstraite, c'est à dire une règle de réécriture, il faut au moins une règle de dépliage dans la démonstration, ce qui implique que la démonstration n'est pas minimale pour l'ordre de sous-démonstration. Par conséquent, on ne peut trouver de démonstration du calcul des séquences avec dépliage qui conviendrait comme démonstration triviale. On ajoute donc artificiellement des démonstrations triviales $\widehat{R}$ pour toute règle de réécriture R. Au final, ça n'est pas trop problématique car le lemme 4.4 nous permettra de construire des démonstrations sans coupure correspondant aux démonstrations triviales, même si celles-ci comporteront des sous-démonstrations. Pour cela, on a néanmoins besoin de la première propriété de Rew. Une autre façon de voir ceci est de considérer deux versions des démonstrations correspondant aux démonstrations triviales : une avec ordre de sous-démonstration, considérées comme des démonstrations normales, et une sans, considérée comme démonstration triviale. Il faut alors étendre l'ordre sur les démonstrations à ces démonstrations triviales, ce qui peut être fait simplement en disant que les démonstrations triviales ne sont pas comparables aux autres. Pour respecter le postulat A, on définit alors l'ordre de sous-démonstration en disant que $p \trianglerighteq q$ si
- p n'est pas triviale et q est une sous-démonstration de p pour $\rhd_d$;
- ou $q = \widehat{a}$ avec $a \in [p]^{Pm}$.

Pour résumer, on définit l'instance suivante de système canonique abstrait :

Définition 4.10. *On peut voir le calcul des séquences avec dépliage comme un système canonique abstrait avec les définitions suivantes :*
- *Les formules sont les règles de réécriture propositionnelles polarisées :*

$$\mathbb{A} \stackrel{\text{déf}}{=} \{A \to^{\pm} P \,:\, FV(P) \subseteq FV(A)\} \ .$$

- *Les démonstrations sont les couples formés par une démonstration du calcul des séquences avec dépliage et une règle de réécriture de l'image par Rew de la conséquence de la démonstration, ainsi que des démonstrations triviales pour chaque règle de réécriture :*

$$\mathbb{P} \overset{\text{déf}}{=} \left\{ \left(\frac{\vdots}{\Gamma \vdash \Delta}, A \to^{\pm} P \right) \ : \ (A \to^{\pm} P) \in Rew(\Gamma \vdash \Delta) \right\} \cup \{\widehat{a} \ : \ a \in \mathbb{A}\} \ .$$

- *Les hypothèses d'une démonstration non triviale sont les règles de réécriture propositionnelles utilisées dans ses règles de dépliage. L'hypothèse unique d'une démonstration triviale est la formule qui lui est associée.*
- *La conclusion d'une démonstration non triviale est la règle de réécriture en deuxième composante. La conclusion d'une démonstration triviale est la formule qui lui est associée.*
- *L'ordre sur les démonstrations est l'ordre strict qui ne compare aucune démonstration triviale et qui dit que $p > q$ si les deuxièmes composantes de p et q sont égales, si les conséquences des premières composantes sont égales, et si le squelette de la première composante de p est strictement plus grand que celui de q pour l'ordre RPO basé sur la précession $(\vdash_{\smile}, P) > (\vdash_{\smile}, Q)$ et $(\vdash_{\frown}, P) > (\vdash_{\frown}, Q)$ si $P > Q$ pour l'ordre sous-formule ou si $P \xrightarrow[\mathcal{R}_{\mathcal{T}(\Sigma,V)}]{*} Q$, et $(\vdash_{\smile}, P) > (\vdash_{\frown}, Q) > (\mathsf{r}, R)$ pour toutes les règles d'inférence r autres que $\vdash_{\smile}$ et $\vdash_{\frown}$.*
- *L'ordre de sous-démonstration est l'ordre défini comme $p \trianglerighteq q$ si*
  - *p et q ne sont pas triviales et la première composante de q est une sous-démonstration de la première composante de p pour $\trianglerighteq_d$;*
  - *ou $q = \widehat{a}$ avec $a \in [p]^{Pm}$.*

Il reste maintenant à démontrer que ceci définit bien une instance de système canonique abstrait, c'est-à-dire qu'on a bien défini un système de déduction, que les ordres sont bien fondés et que les postulats A, B et C sont vérifiés.

Théorème 4.12.
Le système de la définition 4.10 est un système canonique abstrait.

Démonstration. Montrons tout d'abord qu'on a bien un système de déduction, ce qui revient à démontrer que $Th : A \mapsto [Pf(A)]_{Cl}$ est un opérateur de fermeture.

Th est monotone (ce qui est en fait toujours le cas du fait des définitions de Pf et $[\cdot]_{Cl}$) : si $A \subseteq B$ alors par définition de Pf sur les présentations $Pf(A) \subseteq Pf(B)$ et donc par définition de $[\cdot]_{Cl}$ sur les justifications $[Pf(A)]_{Cl} \subseteq [Pf(B)]_{Cl}$.

Pour toute présentation A, on a $A \subseteq Th\, A$: si $a \in A$, $\widehat{a} \in Pf(A)$ donc $a = [\widehat{a}]_{Cl} \in [Pf(A)]_{Cl}$.

Pour toute présentation A, on a $Th\,(Th\, A) \subseteq Th\, A$: Soit $a \in Th\,(Th\, A)$, alors $a = [p]_{Cl}$ pour $p \in Pf(Th\, A)$. Si p est triviale, alors $\{a\} = \{[p]_{Cl}\} = [p]^{Pm} \subseteq Th\, A$ donc $a \in Th\, A$.

Sinon, la première composante de p correpond à une démonstration π du calcul des séquences avec dépliage de conséquence $\Gamma \vdash \Delta$. Les règles de réécriture utilisées dans π sont donc dans $Th\,A$. Soit b une telle règle. Elle est la conclusion d'une démonstration $q \in Pf(A)$. Si q est triviale, alors $\{b\} = [q]^{Pm}$ donc $b \in A$. Sinon, la première composante de q est une démonstration π' de conséquence $\Gamma' \vdash \Delta'$ avec $b \in Rew(\Gamma' \vdash \Delta')$. Si $b \notin A$, en utilisant la propriété de compatibilité de Rew et le lemme 4.3 on peut transformer π en une démonstration ϖ de $\Gamma, \mathcal{P}(\Gamma' \vdash \Delta') \vdash \Delta$ dans $[p]^{Pm} \setminus Rew(\Gamma' \vdash \Delta')$. On peut appliquer $\wedge\vdash$, $\vdash\vee$, $\vdash\Rightarrow$ et $\vdash\forall$ sur π' pour obtenir une démonstration de $\vdash \mathcal{P}(\Gamma' \vdash \Delta')$ dans A, à laquelle on peut appliquer $\overset{\vdash}{\smile}$ avec ϖ pour obtenir une démonstration de $\Gamma \vdash \Delta$ dans $[p]^{Pm} \setminus Rew(\Gamma' \vdash \Delta') \cup A$. En particulier on n'utilise plus b comme hypothèse. En répétant ce procédé à toutes les hypothèses de p dans $Th\,A \setminus A$, on obtient au final une démonstration dans A.

Montrons maintenant que nous avons bien défini des ordres bien fondés. Il est assez simple de montrer que $>$ est un ordre strict. On peut montrer qu'il est bien fondé en disant que si on avait une suite infinie strictement décroissante pour $>$, alors on aurait également une suite infinie de squelettes strictement décroissante pour l'ordre RPO. La précession définie étant bien fondée, l'ordre RPO est bien fondé, d'où une contradiction. Pour montrer que $\trianglerighteq$ est un ordre, il suffit de remarquer que si $\widehat{a} \trianglerighteq q$ alors $q = \widehat{a}$. Pour montrer qu'il est bien fondé, on utilise le lemme 4.11 : si on avait une suite infinie strictement décroissante pour $\triangleright$, on pourrait en faire une suite infinie strictement décroissante pour $\triangleright_d$, ce qui impliquerait l'existence d'une suite infinie décroissante pour $\triangleright_a$ ou $\triangleright_s$.

Le postulat A est vérifié par définition de $\trianglerighteq$.

Montrons que le postulat B est vérifié : soient deux démonstrations p et q telles que $p \trianglerighteq q$. Si q n'est pas triviale alors p non plus et la première composante de q est un élagage d'une sous-démonstration de celle de p dans le calcul des séquences avec dépliage. Par conséquent, les règles de réécriture utilisées dans q sont incluses dans celles utilisées dans p. Si q est la démonstration triviale $\widehat{a}$, alors $[q]^{Pm} = \{a\} \subseteq [p]^{Pm}$ par définition de $\trianglerighteq$.

Montrons que le postulat C est vérifié : soient trois démonstrations p, q et r telles que $p \triangleright q > r$. Comme q et r sont comparables, aucune des deux n'est triviale. Par conséquent p est non triviale et est de la forme (π_p, a) et la première composante de q est un élagage d'une sous-démonstration de π_p dans le calcul des séquences avec dépliage par définition de $\trianglerighteq$. De plus, les premières composantes π_q de q et π_r de r possèdent la même conséquence par définition de $>$. On peut donc remplacer π_q par π_r dans π_p (après les avoir affaiblis) et obtenir une démonstration π_v valide dans le calcul des séquences avec dépliage. Les règles de réécriture utilisées dans cette démonstration sont incluses dans l'union des règles utilisées dans π_p et dans π_r. On pose donc $v \overset{\text{déf}}{=} (\pi_v, a)$. Comme π_v a la même conséquence que π_p, on a bien $v \in \mathbb{P}$. De plus, on a $[v]^{Pm} \subseteq [p]^{Pm} \cup [r]^{Pm}$. Comme π_r est un élagage d'une sous-démonstration de π_v différent de π_v on a $v \triangleright r$. Enfin, comme $q > r$, le squelette de π_q est strictement plus grand que celui de π_r pour l'ordre RPO. Comme l'ordre RPO est un ordre de simplification, il est monotone et le

squelette de π_p est strictement plus grand que celui de π_v, donc $p > v$. □

4.3.3 Procédure de complétion

Comme nous venons de montrer que le calcul des séquences avec dépliage peut être vu comme un système canonique, on obtient directement grâce à la proposition 4.2 un mécanisme de complétion. Il nous reste donc à voir quel est exactement ce mécanisme, et quelles sont les propriétés de ses limites.

Démonstrations critiques

Nous allons d'abord rechercher la forme des démonstrations en forme normale et des démonstrations critiques.

Proposition 4.13 (Caractérisation des démonstrations en forme normale)**.** *Une démonstration est en forme normale ssi c'est une démonstration triviale ou c'est une démonstration sans coupure qui n'utilise pas de règle d'inférence inutile, avec uniquement des* $\frown$ *autour de propositions atomiques.*

Démonstration. Une démonstration triviale est minimale, car on ne peut pas la comparer par définition de $>$. Par conséquent si $a \in A$ alors $\widehat{a} \in \mu Pf(Th\, A) = Nf(A)$.

Montrons par contraposé que si une démonstration non triviale n'est pas sans coupure ou utilise des règles inutile, ou ne décompose pas assez avant d'appliquer $\frown$, alors elle n'est pas en forme normale. Soit une démonstration $p \in Pf(A)$ non triviale dont la première composante π_p possède une coupure à une position $\mathfrak{p}$. La conséquence de la sous-démonstration $\pi_{p|\mathfrak{p}}$ est une séquence $\Gamma \vdash \Delta$. D'après le lemme 4.4 il existe une démonstration π' sans coupure de $\Gamma \vdash \Delta$ dans $Rew(\Gamma \vdash \Delta)$. Comme $\pi_{p|\mathfrak{p}}$ est dans A, on a $Rew(\Gamma \vdash \Delta) \subseteq Th\, A$ et par conséquent $\pi' \in Pf(Th\, A)$. Comme dans la précession $(\leftharpoondown, P) > (\mathsf{r}, Q)$ pour toute règle d'inférence r autre que $\leftharpoondown$, le squelette d'une démonstration sans coupure est forcément plus petit que celui d'une démonstration avec une coupure à la racine. Par conséquent le squelette de $\pi_{p|\mathfrak{p}}$ n'est pas minimal, et comme le RPO est un ordre de simplification celui de π_p non plus. Autrement dit, $p \notin \mu Pf(Th\, A)$.

Si on a une démonstration non triviale avec une règle inutile, alors la démonstration dans laquelle on a supprimée cette règle est plus petite car le RPO est un ordre de simplification.

Si une démonstration utilise $\frown$ sur une formule non atomique, alors il est toujours possible de se ramener à une démonstration en utilisant $\frown$ sur des propositions atomiques. La squelette de cette démonstration sera plus petit que celui de la démonstration de départ, du fait de la définition de la précession.

Réciproquement, si on a une démonstration p non triviale qui n'est pas minimale, cela veut dire qu'il existe une démonstration plus petite q dans la même présentation. Comme q est comparable à p, elle n'est pas triviale et on note π_p et π_q les premières

composantes de p et q. On procède par récurrence sur la structure de π_p. Si on regarde la définition d'un RPO, le fait que $\pi_p > \pi_q$ peut provenir de quatre causes :

- π_p et π_q ont la même première règle, et une des sous-démonstration de π_p est plus grande que celle de π_q correspondante. C'est le cas de récurrence ;
- la première règle de π_p est $\vdash_{\smile}$;
- la première règle de π_p est $\frown_{\vdash}$; dans ce cas, comme les feuilles de π_q sont toutes $\frown_{\vdash}$, cela veut dire que la formule active dans la règle $\frown_{\vdash}$ de π_p est plus grande pour l'ordre sous-formule que toutes celles des feuilles de π_q, donc qu'elle n'est pas atomique ;
- π_q est une sous-démonstration de π_p ; comme p et q sont comparables, elles ont la même conclusion, donc π_p utilise des règles inutiles à partir de π_q.

□

Proposition 4.14 (Caractérisation des démonstrations critiques)**.** *Les démonstrations critiques en calcul des séquences avec dépliage polarisé ne sont pas triviales et ont leur première composante de la forme*

$$\uparrow\vdash \dfrac{\vdash_{\smile} \dfrac{\dfrac{\overset{\pi}{\Gamma, A, P \vdash \Delta}}{\Gamma, A \vdash \Delta}\, A \underset{s_1 \to^- t_1}{\longrightarrow} P \qquad \uparrow\vdash \dfrac{\dfrac{\overset{\pi'}{\Gamma \vdash Q, A, \Delta}}{\Gamma \vdash A, \Delta}\, A \underset{s_2 \to^+ t_2}{\longrightarrow} Q}{}}{\Gamma \vdash \Delta}}{}$$

où

- π *et* π' *sont sans coupure ;*
- π *et* π' *n'appliquent pas inutilement des règles d'inférence ;*
- π *et* π' *n'appliquent* $\frown_{\vdash}$ *que sur des propositions atomiques ;*
- Γ *contient uniquement des formules quantifiées universellement et des propositions atomiques différentes de* A *;*
- Δ *contient uniquement des formules quantifiées existentiellement et des propositions atomiques différentes de* A *;*
- *toute les formules de* Γ, Δ *sont des formules principales soit pour* $\frown_{\vdash}$ *(mais pas pour* $\uparrow\vdash$ *ni pour* $\vdash\uparrow$*),* $\forall\vdash$ *ou* $\vdash\exists$ *dans* π *ou* π'*.*
- *au moins une de* $s_1 \to^- t_1$ *et* $s_2 \to^+ t_2$ *n'est pas une règle de réécriture sur les termes.*

Démonstration. Pour démontrer ceci, il suffit d'étendre la procédure d'élimination des coupures.

Tout d'abord, un démonstration critique p est minimale dans $Pf(A)$ mais pas dans $Pf(Th\, A)$. La proposition précédente nous montre que cela veut dire que p n'est pas triviale et que sa première composante π_p possède une coupure, utilise une règle d'inférence inutile, ou utilise $\frown_{\vdash}$ avec une formule non atomique. Dans les deux derniers cas, cela voudrait dire que p ne serait également pas minimale dans $Pf(A)$, puisqu'on peut

construire des démonstrations plus petites avec les mêmes règles de réécriture. Donc π_p possède une coupure.

Les sous-démonstrations de p sont en forme normale. Cela veut dire, en utilisant la proposition précédente, que les sous-démonstrations de π_p sont sans coupure. Par conséquent, l'unique coupure de π_p se trouve à sa racine, et π_p est de la forme

$$\dfrac{\dfrac{\mathsf{r}\ \dfrac{\pi_1 \quad \cdots \quad \pi_n}{\Gamma, P \vdash \Delta} \qquad \mathsf{r}'\ \dfrac{\pi'_1 \quad \cdots \quad \pi'_m}{\Gamma \vdash P, \Delta}}{\vdash\!\frown} }{\Gamma \vdash \Delta}$$

Si les règles r et r' ne sont pas des règles de dépliage, on peut appliquer une étape de la procédure d'élimination des coupures de la section 2.3.4. On obtient ainsi une démonstration dans le même système de réécriture dont le squelette est plus petit pour l'ordre RPO. Par conséquent p n'est pas minimal dans $Pf(A)$.

Si une des règles r ou r' est une règle de dépliage dont la formule active n'est pas P, alors on peut faire permuter cette règle avec la coupure comme on fait dans le même cas pour les règles logique. Ici aussi, on obtient ainsi une démonstration dans le même système de réécriture dont le squelette est plus petit pour l'ordre RPO, et par conséquent p n'est pas minimal dans $Pf(A)$.

On suppose donc que la formule active de r et r' est P. Si une des règles r ou r' est une règle de dépliage, alors P est atomique. Par conséquent l'autre règle d'inférence ne peut être que $\vdash\!\frown$ ou une règle de dépliage. Dans le cas de $\vdash\!\frown$ on peut appliquer une étape d'élimination des coupures et obtenir une démonstration plus petite. Il ne reste donc plus que le cas d'une démonstration de la forme donnée par l'énoncé. Si les deux règles de réécriture $s_i \to t_i$ utilisées dans les règles de dépliage sont des règles sur les termes, alors P et Q sont atomiques et comme $\mathcal{R}_{\mathcal{T}(\Sigma,V)}$ est confluent il existe R tel que $P \underset{\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\overset{*}{\longrightarrow}} R \underset{\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\overset{*}{\longleftarrow}} Q$. On peut donc construire la démonstration

$$\dfrac{\dfrac{\pi}{\Gamma, P \vdash \Delta} \qquad \dfrac{\dfrac{\dfrac{\vdash\!\frown}{\Gamma, R \vdash R, \Delta}}{\Gamma, Q \vdash P, \Delta}\ \uparrow\!\vdash,\vdash\!\uparrow \qquad \pi'}{\Gamma \vdash P, \Delta}\ \vdash}{\Gamma \vdash \Delta}\ \vdash\!\frown$$

dont le squelette est plus petit pour l'ordre RPO.

Dans la forme restante, π et π' sont les premières composantes de démonstrations en forme normale. Elles ne possèdent donc pas de coupures, et n'appliquent pas de règles inutilement d'après la proposition 4.13. Toutes les formules de Γ, Δ sont utilisées, car sinon on peut obtenir une démonstration plus petite pour $\rhd$ (pour $\rhd_a$ plus précisément) qui n'est pas en forme normale (elle possède une coupure). Si on avait une formule ni atomique, ni quantifiée universellement dans Γ, alors elle serait décomposée à l'aide d'une règle logique gauche que l'on pourrait faire permuter jusqu'en dessous de la coupure.

On obtiendrait ainsi une démonstration plus petite ce qui contredirait la minimalité des démonstrations critiques. Le même argument permet de montrer que les règles de dépliage ne peuvent s'appliquer aux formules de Γ, Δ. □

Au passage, cela permet de montrer qu'un des critère utilisé par Olivier Hermant (2005) pour démontrer l'admissibilité de $\vdash\!\!\!\smile$ permet en fait de démontrer la terminaison de la procédure d'élimination des coupures.

Proposition 4.15. *Si on suppose qu'il existe un ordre bien fondé $\sqsupset$ incluant à la fois l'ordre sous-formule et la relation de réécriture $\underset{\mathcal{R}\cup\mathcal{R}_{\mathcal{T}(\Sigma,V)}}{\longrightarrow}$, alors on peut définir, dans le calcul des séquences avec dépliage, une procédure d'élimination des coupures qui termine.*

Démonstration. La procédure en question est celle décrite dans la démonstration de la proposition précédente, si ce n'est que l'on élimine également les coupures dans le cas où on a des règles de réécriture propositionnelle, comme si on avait des règles sur les termes. Pour démontrer la terminaison de cette procédure, on modifie la précession de l'ordre RPO : on aura $(\vdash\!\!\!\smile, P) > (\vdash\!\!\!\smile, Q)$ si $P \sqsupset Q$. On vérifie alors que la procédure fait diminuer les squelettes pour cette ordre. □

Dans le mécanisme de complétion, on a besoin de trouver des démonstrations plus petites que toutes les démonstrations critiques. Le problème est que l'ensemble des démonstrations critiques n'est pas décidable.

Proposition 4.16. *Le problème qui prend en entrée un système de réécriture et une séquence, et qui dit si cette séquence est la conséquence de la première composante d'une démonstration critique pour ce système, est indécidable.*

Démonstration. On réduit ce problème à celui de la validité dans la logique du premier ordre.

Soient P une formule du premier ordre et A et B deux propositions atomiques n'apparaissant pas dans P. On considère le système de réécriture

$$\begin{array}{l} A \to^{-} B \\ A \to^{+} P \end{array} \quad .$$

Montrons que P est valide ssi $\vdash B$ est la conséquence de la première composante d'une démonstration critique.

En effet, d'après la proposition 4.14, la première composante d'une démonstration critique dont la conséquence est $\vdash B$ est obligatoirement de la forme

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash B}\,\vdash\!\!\!\frown}{A \vdash B}\,\uparrow\vdash \qquad \dfrac{\pi'}{\vdash A, P, B}\,\uparrow\vdash}{\vdash B}\,\vdash\!\!\!\smile$$

Dans la démonstration π', on peut voir qu'il est inutile de déplier A à nouveau. Comme A et B n'apparaissent pas dans P, π' est en fait une démonstration de $\vdash P$, qui n'utilise donc pas de dépliage. Si P est valide, on va donc pouvoir trouver une telle démonstration π' pour avoir une démonstration critique. Réciproquement, si $\vdash B$ n'est pas la conséquence de la première composante d'une démonstration critique alors il n'existe pas de démonstration de $\vdash P$ dans le calcul des séquences (sans dépliage), donc P n'est pas valide à cause de la complétude du calcul des séquences pour la logique du premier ordre. □

Le mécanisme de complétion

Pour définir le mécanisme de complétion de la fin de la section 4.2.2, il nous faut pour chaque présentation A définir une présentation $C(A)$ qui soit incluse dans $Th\,A$ et qui contienne au moins les hypothèses permettant de construire des démonstrations plus petites que les démonstrations critiques. Calculer une présentation $C(A)$ minimale en terme de taille est impossible du fait de la proposition 4.16. On peut alors utiliser un sur-ensemble en considérant toutes les démonstrations de la forme de la proposition 4.14.

Pour calculer ces démonstrations potentiellement critiques, on procédera donc comme suit : on appliquera une méthode des tableaux pour le calcul des séquences avec dépliage aux séquences $A, P \vdash$ et $\vdash A, Q$. Il est possible que la procédure ne termine pas, on fixera donc une limite à au nombre de décomposition de type γ et rw. On terminera donc sur des branches constituées uniquement de propositions atomiques et de formules universellement quantifiées à gauche et existentiellement à droite. On fermera ensuite toutes les branches restées ouvertes en rajoutant des propositions atomiques différentes de A dans les contextes, ce qui nous donnera les Γ et Δ de la conséquence de la démonstration potentiellement critique. Il est possible que ces propositions atomiques utilisent des variables propres introduites par des $\exists\vdash$ et $\vdash\forall$. Dans ce cas ces variables ne peuvent apparaître dans Γ, Δ et il faut les garder par des quantifications. Par exemple, si y est une varible propre de ce type, on ajoutera $\exists y,\ A(y)$ dans Δ plutôt que $A(y)$. Si on a arrêté la méthode des tableaux trop tôt, on aura éventuellement laissée ouverte une branche qu'il était possible de fermer. Cela veut dire que la démonstration finale ne sera pas critique, mais le système de réécriture qu'elle rajoutera dans $C(A)$ permettra de construire des démonstrations plus petites que la démonstration critique qu'on aurait pu obtenir en continuant la méthode des tableaux. Néanmoins, de façon à avoir $C(A)$ la plus minimale possible, il faut que éviter ce cas de figure au maximum.

Pour une démonstration dans A de cette forme donnée, le lemme 4.4 permet d'obtenir une démonstration dans $Th\,A$ sans coupure, qui est donc plus petite. Notre ensemble $C(A)$ sera donc l'union des $Rew(\Gamma \vdash \Delta)$ pour toutes les conséquence $\Gamma \vdash \Delta$ des démonstrations potentiellement critiques calculées comme indiqué ci-dessus. Le mécanisme de complétion consistera donc à calculer le plus petit point fixe de la fonction qui, étant donné un système de réécriture $\mathcal{R}_0$, associe à un système de réécriture $\mathcal{R}$ le système

$\mathcal{R}_0 \cup \mathcal{R} \cup C(\mathcal{R})$.

Théorème 4.17 (Admissibilité des coupures pour la limite)**.**
Pour tout système de réécriture $\mathcal{R}_0$, pour toute séquence $\Gamma \vdash \Delta$, on a $\Gamma \vdash_{\mathcal{R}_0} \Delta$ ssi $\Gamma \vdash^*_{\mathcal{R}_\infty} \Delta$.

Démonstration. Par la proposition 4.2 on sait que l'on a définit un mécanisme de complétion. Par conséquent, le théorème 4.1 nous dit que

$$Th\,\mathcal{R}_0 \subseteq [Pf(\mathcal{R}_\infty) \cap Nf(\mathcal{R}_0)]_{Cl} \ . \tag{4.1}$$

La direction « si » provient du fait que les règles que l'on rajoute à chaque étape sont bien dans $Th\,\mathcal{R}_0$. En utilisant la propriété de compatibilité de Rew on peut donc revenir à une démonstration dans $\mathcal{R}_0$.

Pour la direction « seulement si », en supposant que $\Gamma \vdash_{\mathcal{R}_0} \Delta$, alors $Rew(\Gamma \vdash \Delta) \subseteq Th\,\mathcal{R}_0$. En utilisant (4.1) on obtient pour toute règle de $Rew(\Gamma \vdash \Delta)$ une démonstration en forme normale dans $\mathcal{R}_\infty$. Si cette démonstration est triviale, c'est la démonstration triviale d'une formule qui par définition de $\mathbb{A}$ appartient à $Rew(\Gamma' \vdash \Delta')$ pour une certaine séquence $\Gamma' \vdash \Delta'$. On utilise le lemme 4.4 pour obtenir $\Gamma' \vdash^*_{\mathcal{R}_\infty} \Delta'$. Si la démonstration en forme normale n'est pas triviale, on sait grâce à la proposition 4.13 que la première composante correspond à une démonstration sans coupure. On utilise finalement la troisième propriété de Rew pour obtenir une démonstration sans coupure de conséquence $\Gamma \vdash \Delta$ dans $\mathcal{R}_\infty$. □

4.3.4 Exemples d'applications

On a déjà vu dans l'exemple 4.2 l'exemple de la complétion du système $\mathcal{C}rab$.

On peut regarder ce que donne le mécanisme de complétion sur des exemples avec des quantificateurs. Par exemple, on peut prendre l'exemple du tableau 3.1 du système avec quantificateurs qui n'admet pas $\not\subset$ introduit par Olivier Hermant (2005) :

$$\mathsf{r} \in \mathsf{r} \to \forall y,\ (\forall x,\ y \in x \Rightarrow \mathsf{r} \in x) \Rightarrow y \in \mathsf{r} \Rightarrow B \ .$$

Pour construire une démonstration critique, il faut donc chercher une démonstration π de $\Gamma, \mathsf{r} \in \mathsf{r}, (\forall x,\ y \in x \Rightarrow \mathsf{r} \in x) \Rightarrow y \in \mathsf{r} \Rightarrow B \vdash \Delta$ pour certains Γ et Δ. On applique la méthode des tableaux pour décomposer au maximum la séquence $\mathsf{r} \in \mathsf{r}, (\forall x,\ y \in x \Rightarrow \mathsf{r} \in x) \Rightarrow y \in \mathsf{r} \Rightarrow B \vdash$, en se restreignant à une seule application de $\forall\vdash$ et $\vdash\exists$ par formule. On obtient la dérivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\Theta, B \vdash \qquad \Theta \vdash Y \in \mathsf{r}}{\Theta, Y \in \mathsf{r} \Rightarrow B \vdash}\Rightarrow\vdash \qquad \dfrac{\dfrac{\Theta, Y \in x(Y) \vdash \mathsf{r} \in x(Y)}{\Theta \vdash Y \in x(Y) \Rightarrow \mathsf{r} \in x(Y)}\vdash\Rightarrow}{\Theta \vdash \forall x,\ Y \in x \Rightarrow \mathsf{r} \in x}\vdash\forall}{\Theta, (\forall x,\ Y \in x \Rightarrow \mathsf{r} \in x) \Rightarrow Y \in \mathsf{r} \Rightarrow B \vdash}\Rightarrow\vdash}{\Theta \vdash}\forall\vdash$$

en posant $\Theta = \mathsf{r} \in \mathsf{r}, (\forall x,\ y \in x \Rightarrow \mathsf{r} \in x) \Rightarrow y \in \mathsf{r} \Rightarrow B$ et avec Y est une méta-variable. On fait de même pour la séquence $\vdash \Theta$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{r} \in X, y \in \mathsf{r} \vdash \mathsf{r} \in \mathsf{r}, B \qquad y \in \mathsf{r} \vdash y \in X, \mathsf{r} \in \mathsf{r}, B}{\Rightarrow\vdash} \quad y \in X \Rightarrow \mathsf{r} \in X, y \in \mathsf{r} \vdash \mathsf{r} \in \mathsf{r}, B}{\forall\vdash} \quad \forall x,\ y \in x \Rightarrow \mathsf{r} \in x, y \in \mathsf{r} \vdash \mathsf{r} \in \mathsf{r}, B}{\vdash\Rightarrow} \quad \vdash \mathsf{r} \in \mathsf{r}, (\forall x,\ y \in x \Rightarrow \mathsf{r} \in x) \Rightarrow y \in \mathsf{r} \Rightarrow B}{\vdash\forall} \quad \vdash \Theta$$

avec X méta-variable.

Pour fermer le tableau entier sans rajouter la formule $\mathsf{r} \in \mathsf{r}$, il faut rajouter B à droite pour fermer la première branche et instancier Y et X par r. On obtient alors comme démonstration critique la démonstration donnée par Olivier Hermant (2005) pour montrer l'inadmissibilité de $\overset{\smile}{\vdash}$.

Nous devons rajouter $Rew(\vdash B) = \{B \rightarrow^{+} \top\}$ au système initial, et le système obtenu n'a plus de démonstrations critiques.

On peut être intéressé par un exemple dans lequel il resterait des quantificateurs dans la conséquence des démonstrations critiques. Pour cela, il suffit de remplacer B par exemple par $\exists x,\ \forall y,\ B \wedge C(x,y)$ dans $\mathcal{C}rab$. On applique la méthode des tableaux à $A, (\exists x,\ \forall y.\ B \wedge C(x,y)) \wedge \neg A \vdash$

$$\dfrac{\dfrac{\dfrac{\overset{\frown}{\vdash}\ \overline{A, (\exists x,\ \forall y,\ B \wedge C(x,y)) \vdash A}}{\neg\vdash} \quad A, (\exists x,\ \forall y,\ B \wedge C(x,y)), \neg A \vdash}{\wedge\vdash} \quad A, (\exists x,\ \forall y,\ B \wedge C(x,y)) \wedge \neg A \vdash}$$

puis on l'applique sur $\vdash A, (\exists x,\ \forall y,\ B \wedge C(x,y)) \wedge \neg A$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash A, \exists x,\ \forall y,\ B \wedge C(x,y), B \qquad \vdash A, \exists x,\ \forall y,\ B \wedge C(x,y), C(X, y(X))}{\vdash\wedge} \quad \vdash A, \exists x,\ \forall y,\ B \wedge C(x,y), B \wedge C(X, y(X))}{\vdash\forall} \quad \vdash A, \exists x,\ \forall y,\ B \wedge C(x,y), \forall y,\ B \wedge C(X,y)}{\vdash\exists} \quad \vdash A, \exists x,\ \forall y,\ B \wedge C(x,y)}{\vdash\wedge} \quad \dfrac{\overset{\frown}{\vdash}\ \overline{A \vdash A}}{\vdash\neg}\ \vdash A, \neg A}{\vdash A, (\exists x,\ \forall y,\ B \wedge C(x,y)) \wedge \neg A}$$

Pour fermer le tableau sans rajouter A, il faut rajouter B et une instance de $C(X, y(X))$ à gauche. Quand on calcule le système associé à $B, C(x, y(x)) \vdash$ on obtient par exemple $B \rightarrow^{-} \forall x,\ \neg C(x, y(x))$. On remarque que cette règle est plus générale que toutes les règles $B \rightarrow^{-} \neg C(t, y(t))$ obtenue avec des instances closes de $C(X, y(X))$, dans le sens où les règles de dépliage pour ces dernières peuvent sont dérivables dans le système avec la première règle. Il suffit donc de rajouter cette règle et on n'a plus de démonstration critique. En particulier, il est possible de démontrer l'équivalent de $B \vdash$ dans le système $\mathcal{C}rab$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall y,\ B \wedge C(x,y), B, C(x,y(x)), B, \forall x,\ \neg C(x,y(x)), C(x,y) \vdash C(x,y(x))}\,\widehat{\vdash}}{\forall y,\ B \wedge C(x,y), B \wedge C(x,y(x)), B, \forall x,\ \neg C(x,y(x)), C(x,y) \vdash C(x,y(x))}\,\wedge\vdash}{\forall y,\ B \wedge C(x,y), B, \forall x,\ \neg C(x,y(x)), C(x,y) \vdash C(x,y(x))}\,\forall\vdash}{\forall y,\ B \wedge C(x,y), B, \forall x,\ \neg C(x,y(x)), C(x,y) \vdash C(x,y(x))}\,\neg\vdash}{\forall y,\ B \wedge C(x,y), B, \forall x,\ \neg C(x,y(x)), \neg C(x,y(x)), C(x,y) \vdash}\,\forall\vdash}{\forall y,\ B \wedge C(x,y), B, \forall x,\ \neg C(x,y(x)), C(x,y) \vdash}\,\uparrow\vdash}{\forall y,\ B \wedge C(x,y), B, C(x,y) \vdash}\,\wedge\vdash}{\forall y,\ B \wedge C(x,y), B \wedge C(x,y) \vdash}\,\forall\vdash}{\forall y,\ B \wedge C(x,y) \vdash}\,\exists\vdash}{\exists x,\ \forall y,\ B \wedge C(x,y) \vdash}$$

Le construction de la conséquence des démonstrations critiques n'est pas exactement celle de la section précédente, du fait qu'on utilise des métavariables dans la méthode des tableaux. Si on avait utilisée la procédure de la section précédente, on aurait obtenu la conséquence $B, \forall y,\ C(x,y) \vdash$ ce qui aurait donné par exemple la règle $B \rightarrow^{-} \forall x,\ \neg\forall y,\ C(x,y)$, dont la version ci-dessus est simplement une forme skolémizée.

4.4 Implémentation

Pour nous convaincre que nous avons décrit ce mécanisme de complétion de manière effective, nous en avons réalisé un prototype.

4.4.1 TOM/OCaml

Pour pouvoir développer une implémentation du mécanisme de complétion, nous avons besoin essentiellement de coder une méthode des tableaux pour le calcul des séquences avec dépliage. Un tableau étant une structure de données associative et commutative, il nous a semblé que le filtrage syntaxique de OCaml n'était pas suffisant. Nous aurions probablement pu utiliser Maude, qui est un langage permettant de faire des spécification sous forme de réécriture et qui permet de faire du filtrage de motif modulo associativité, commutativité et unité. D'autre part, Moca, développé par Frédéric Blanqui, Thérèse Hardin et Pierre Weis (2007), permet de travailler en OCaml sur des représentants canoniques de structures modulo certaines équations. Néanmoins, cela ne permet pas, à notre connaissance, de faire de la reconnaissance de motif modulo ces équations sur une structure quelconque. Nous avons donc préféré utiliser TOM, également développé dans notre équipe, d'autant qu'il possède également une sortie pour OCaml. Basé sur la notion d'îlot formel d'Emilie Balland, Claude Kirchner et Pierre-Etienne Moreau (2006), TOM permet d'enrichir un langage de programmation hôte (parmi Java, C, OCaml, Python, etc.) avec du filtrage associatif. Une prochaine version devrait même proposer de faire du filtrage associatif-commutatif. Pour cela, il existe une directive `%match` qui

filtre un terme TOM et qui applique des instructions du langage hôte en fonction du filtrage. Le mieux est d'illustrer ceci sur un exemple, en prenant pour langage hôte OCaml :

```
let cherche_avant a b l =
  %match (Liste l) {
     (_*, a, _*, b, _*) ->
       { print_endline "Une occurrence trouvée." }
  }
```

Le motif `(_*, a, _*, b, _*)` désigne une liste associative possédant un élément correspondant à `a` et un élément correspond à `b`. On voit que l'on peut introduire des variables du langage hôte dans les motifs TOM. Lors de la compilation TOM, le filtrage est réécrit en instruction du langage hôte (on parle de dissolution). Pour cela, il faut indiquer comment les termes TOM correspondent aux expressions du langage hôte, ce qui s'appelle l'encrage. Pour que les listes d'entier OCaml soient vues comme des listes TOM, on rajoute le code

```
%typeterm int {
  implement   { int }
  is_sort(t)  { true }
  equals(i,j) { (i = j) }
}

%typeterm Liste {
  implement     { int list }
  is_sort(l)    { true }
  equals(l1,l2) { (l1 = l2) }
}

%oplist Liste concListe(int*) {
  is_fsym(t)       { (* statically by type checking *) true }
  make_empty()     { [] }
  make_insert(c,s) { (c :: s) }
  get_head(s)      { List.hd s }
  get_tail(s)      { List.tl s }
  is_empty(s)      { (s = []) }
}
```

L'opérateur `%typeterm` lie un type de OCaml a un type TOM, tandis que `%oplist` permet de définir un symbole de fonction associatif.

Le code ci-dessus est alors transformé à la compilation TOM en code OCaml qu'il est inutile de recopier ici car il serait illisible.

Pour écrire des expressions TOM au milieu de code du langage hôte, on utilise l'accent grave ‘. Par exemple ‘`(1,2,4,3)` sera réécrite à la compilation TOM en une liste OCaml égale à `[1;2;4;3]`.

Un des autres intérêts de TOM est la présence d'antimotifs introduits par Claude Kirchner, Radu Kopetz et Pierre-Etienne Moreau (2007), ce qui permet de filtrer sur des contraintes négatives, ou plus exactement des contraintes de complément ensembliste. Ainsi le motif `(_*, x, _*, !x, _*)` filtrera les listes qui contiennent au moins deux éléments distincts.

L'implémentation que nous avons réalisée est disponible dans la distribution de TOM par `svn`[2] dans le répertoire `application/completion`.

4.4.2 Implémentation de TaMed

La méthode des tableaux que nous avons implémentée est essentiellement celle de Richard Bonichon et Olivier Hermant (2006*b*) (cf. section 3.2.4), si ce n'est que nous n'appliquons rw que sur les propositions atomiques, ce qui revient à se placer en calcul des séquences avec dépliage plutôt que dans le calcul des séquences modulo avec règle de conversion explicite.

Les termes et les formules de la logique du premier ordre sont représentées par les structures de données OCaml

```
type term =
    Var of string
  | Fun of string * term list

type prop =
    Atomic of string * term list
  | True
  | False
  | Not of prop
  | Or of prop * prop
  | And of prop * prop
  | All of string * prop * int
  | Ex of string * prop * int
```

Pour `All` et `Ex`, l'entier associé correspondant au nombre de fois auquel on a déjà appliqué respectivement $\forall\vdash$ ou $\vdash\exists$. Ces structures de données sont ancrées en TOM de façon triviale.

On a également besoin d'équations pour les contraintes. Ces équations sont soit entre termes, soit entre formules, d'où la structure

[2] Cf. http://gforge.inria.fr/scm/?group_id=78 .

```
type eq =
    Term of term * term
  | Prop of prop * prop
```

Ce type servira également pour les règles de réécriture. On annote une formule par une liste de méta-variable (représentées par des chaînes), une liste de contraintes, et une liste de règles de réécriture déjà appliquée sur la formule. On a donc un type

```
type a_prop = prop * (string list * eq list * eq list)
```

Une branche est une liste de formules annotées, elle-même annotée par un booléen indiquant l'alternance de priorité entre γ et rw. Un tableau est une liste de branches.

Pour résoudre les contraintes, on a besoin d'un algorithme d'unification. Nous avons utilisée pour cela une des stratégies proposées par Claude Kirchner et Hélène Kirchner (2001, section 3.2.4) pour résoudre des ensembles d'équations à unifier.

On écrit alors une fonction `closure` qui vérifie si le tableau en entier peut-être fermé, ce qui est assez facile grâce au filtrage associative de TOM. En effet, il s'agit de collecter toutes les propositions atomiques potentielles qui pourraient s'unifier, et on n'utilise pour cela les motifs `(_*,a(not(atomic(s,a1)),c(_,c1,_)),_*,a(atomic(s,a2),c(_,c2,_)),_*)` et `(_*,a(atomic(s,a1),c(_,c1,_)),_*,a(not(atomic(s,a2)),c(_,c2,_)),_*)`. Pour chacun des choix potentiels pour chaque branche, on essaie d'unifier le tableau en entier en prenant soin de tenir compte des contraintes sur les formules (filtrées par `c1` et `c2` ci-dessus).

On peut alors écrire la méthode des tableaux. On utilise une table de hachage dans laquelle on stocke à chaque étape le tableau, pour vérifier qu'on ne boucle pas, car dans un tel cas on peut arrêter la méthode, puisqu'on ne fermera jamais le tableau. On a un certain nombre d'étapes qui ferment une branche quand on n'a pas besoin d'unification :

```
(b1*,b((_*,a(True(),_),_*),_),b2*) -> { next (`(b1*,b2*)) }

(b1*,b((_*,a(not(False()),_),_*),_),b2*) -> { next (`(b1*,b2*)) }

(b1*,b((_*,a(p,_),_*,a(not(p),_),_*),_),b2*) -> { next (`(b1*,b2*)) }

(b1*,b((_*,a(not(p),_),_*,a(p,_),_*),_),b2*) -> { next (`(b1*,b2*)) }
```

D'autres étapes suppriment les branches et les formules inutiles ou redondantes :

```
(b1*,b,b2*,b,b3*) -> { next (`(b1*,b,b2*,b3*)) }

(b1*,b((gamma*,a(False(),_),delta*),s),b2*) ->
  { next (`(b1*,b((gamma*,delta*),s),b2*)) }
```

```
(b1*,b((gamma*,a(not(True()),_),delta*),s),b2*) ->
  { next (‘(b1*,b((gamma*,delta*),s),b2*)) }

(b1*,b((gamma*,ap,eta*,ap,delta*),s),b2*) ->
  { next (‘(b1*,b((gamma*,ap,eta*,delta*),s),b2*)) }
```

On essaie ensuite de fermer le tableau à l'aide de la fonction `closure` définie précédemment. On peut alors appliquer les règles logiques (on n'indiquera que l'application sur les formules non niées, le reste étant dual). L'ordre d'application traduira la priorité définie par Richard Bonichon et Olivier Hermant (2006*b*). La fait de mettre la branche modifiée à la droite du tableau permet de s'assurer de ne pas rester toujours dans la même branche, ce qui revient à choisir en premier les branches qui auraient un nombre minimal de formules si on gardaient les formules de départ.

```
(b1*,b((gamma*,a(not(not(p)),l),delta*),s),b2*) ->
  { (* suppression double négation *)
    next (‘(b1*,b((gamma*,a(p,l),delta*),s),b2*)) }

(b1*,b((gamma*,a(or(p,q),l),delta*),s),b2*) ->
  { (* alpha *)
    next (‘(b1*,b2*,b((gamma*,a(p,l),a(q,l),delta*),s))) }

(b1*,b(conc(gamma*,a(all(x,p,_),cs@c(l,_,_)),delta*),s),b2*) ->
  { (* delta *)
    let e = Term(Var (‘x),
                 Fun(fresh_var "sk_",
                     List.rev_map (fun x -> Var x) (‘l))) in
    let q = subst [e] (‘p) (* on substitue x par un terme de Skolem *)
    next (‘(b1*,b2*,b((gamma*,a(q,cs),delta*),s))) }

(b1*,b((gamma*,a(and(p,q),l),delta*),s),b2*) ->
  { (* beta *)
    next (‘(b1*,b2*,b((gamma*,a(p,l),delta*),s),
                    b(conc(gamma*,a(q,l),delta*),s))) }
```

On applique en priorité γ devant `rw` qui si la branche est annotée par `true` et si on ne l'a pas appliqué plus de `lim` fois, `lim` étant un paramètre fourni à la méthode des tableaux.

```
(b1*,b(bg@(_*,a(ex(_,_,i),_),_*),true()),b2*) && i < lim ->
  { (* gamma *)
    let nbg = do_gamma (‘bg) lim in
    next (‘(b1*,b2*,b(nbg,false())))
  }
```

`do_gamma` est une fonction qui applique Γ à toutes les formules de type γ de la branche (écrite en OCaml) :

```
let do_gamma bg lim =
  let rec aux accu = function
    | (Ex(x,p,i),(l,c,rr))::ps when i < lim ->
         let y = fresh_var "v_" in
         let e = Term(Var x, Var y) in
         let q = subst [e] p
           (* on substitue x par une méta-variable fraiche *)
         and r = y::l, c, rr
           (* on rajoute cette variable dans les contraintes *)
         in aux ((Ex(x,p,i+1),(l,c,rr))::(q,r)::accu) ps
    | (Not(All(x,p,i)),(l,c,rr))::ps when i < lim ->
         (* cas dual *)
          ...
    | ap::ps -> aux (ap::accu) ps
    | [] -> List.rev accu
  in aux [] (bg)
```

Dans la méthode des tableaux, il faut maintenant le code pour rw, qui utilise le paramètre `rules` qui contient la liste des règles de réécriture disponibles. `narrow p r` surréduit une proposition `p` à l'aide d'une règle de réécriture `r`, qui retourne cette proposition et une ensemble de contraintes ou qui renvoie une exception `Not_match` si ce n'est pas possible.

```
(b1*,b((gamma*,a(p@atomic(_,_),(l,c,rr)),delta*),_),b2*) ->
  { (* rw *)
    List.iter
     (fun r ->
        if not (List.mem r rr)
        then try
          let q, nc = narrow (‘p) r in
          let nrr = r :: rr in
          next (‘(b1*,b2*,b((gamma*,delta*,
                             a(q,(l,nc,())),a(p,(l,c,nrr))),True())))
          with Not_match -> () )
     rules
  }
```

Il faut ensuite remettre la décomposition γ au cas où on ait une branche annotée par `False()` et dans laquelle on puisse appliquer γ mais pas rw.

Pour définir la méthode des tableaux finale, on procède par approfondissement itératif (en anglais *iterative deepening*) : on applique la méthode pour `lim = 0` puis on augmente progressivement la limite jusqu'à obtenir une démonstration.

4.4.3 Implémentation de la complétion

On va chercher maintenant à développer le mécanisme de complétion. Pour cela, il nous faut trouver toutes les conséquences des démonstrations potentiellement critiques, de la forme de la proposition 4.14. On voit que dans ce cas (P, Q) forme une paire critique. On va donc calculer tout d'abord toutes les paires critiques du système de réécriture initial, ce qui revient à prendre toutes les paires de règles de réécritures et à tenter de réécrire le membre gauche de l'une par l'autre. On peut se restreindre à une règle propositionnelle pour l'une des deux d'après la proposition 4.14.

Étant donnée une paire critique $P \underset{r_1}{\longleftarrow} A \underset{r_2}{\longrightarrow} Q$, on va chercher à démontrer les tableaux $\neg A, \neg P | A, Q$ et $\neg A, \neg Q | A, P$ et à compléter les branches restées ouvertes si ce n'est pas le cas. On applique donc la méthode des tableaux sur le tableau

`(b((a(not(`A`), ((),(),(`r_1`))), a(not(`P`), ((),(),()))), True()),`
`b((a(`A`, ((),(),(`r_2`))), a(`Q`, ((),(),()))), True()))`

et le tableau dual, pour une limite `lim` donné. Comme dit ci-dessus, plus la limite est haute, plus les démonstrations potentiellement critiques obtenues seront effectivement critique, mais cela nécessitera plus de temps. Si on arrive à démontrer le tableau, cela veut dire que le système de réécriture initial n'était pas cohérent. Sinon, on obtient un tableau dans lequel toutes les formules sont atomiques sauf les formules quantifiées appliquées plus souvent que `lim`. On va chercher à fermer les branches de ce tableau. Pour cela, on cherche d'abord à fermer un maximum de branches sans avoir à rajouter des formules. Si on a un tableau à n branches, on en considère d'abord tous les sous-tableaux à $n-1$ branches. Si on peut fermer un tel tableau, on rajoute son complémentaire (c'est-à-dire dans ce cas le tableau constitué de l'unique branche que l'on avait mis de côté) dans la liste des tableaux dans lesquels il faudra fermer les branches en rajoutant des formules. Sinon, on recommence avec ses sous-tableaux à $n-2$ branches et ainsi de suite. On obtient donc une liste de tableaux auxquels il faut rajouter des formules aux branches pour les fermer. Pour chacun de ces tableaux, on choisit une proposition atomique différente de A dans chaque branche et on met sa négation dans une nouvelle branche. Cette branche correspond à la conséquence de la démonstration potentiellement critique. Si il n'est pas possible de choisir une proposition atomique différente de A, cela veut dire qu'il n'y a pas de démonstration critique pour cette paire critique. On considère tous les choix potentiels de proposition atomiques différentes de A, ce qui donne les conséquences de plusieurs démonstrations potentiellement critiques. On rajoute alors au système de réécriture une règle pour chacune de ces conséquences, qu'il est facile d'obtenir puisqu'on n'a que des

propositions atomiques dans la branche. On réitère ce processus jusqu'à atteindre un point fixe pour le système de réécriture.

Chapitre 5

Taille des démonstrations en déduction modulo

> Ce qui manque aux orateurs en profondeur, ils vous le donnent en longueur.
>
> MONTESQUIEU, *Mes pensées*

Si les démonstrations sans coupure sont faciles à produire de manière automatique, grâce à l'analycité du calcul des séquences sans coupure, elles peuvent être plus longues que celles avec coupures, car ces dernières permettent de faire des raccourcis dans les démonstrations. Nous allons donc maintenant chercher à savoir comment la déduction modulo permet d'obtenir des démonstrations plus courtes.

5.1 Taille de démonstration

La réduction de la taille des démonstrations est bien entendu intéressante d'un point de vue pratique. En effet, une démonstration courte sera plus facile à trouver à la main, mais elle sera aussi plus rapide à obtenir par un outil de démonstration automatique. Elle sera également probablement plus facile à communiquer et à maintenir, si on cherche à faire de l'ingénierie de la démonstration.

La longueur des démonstrations est également importante d'un point de vue théorique. En effet, le problème P=NP ? en théorie de la complexité peut être vu comme un problème de réduction de taille de démonstration, comme le souligne Rohit PARIKH dans le paragraphe préfaçant un article de Kurt GÖDEL (1936) sur la taille des démonstrations dans la collection traduite en anglais de ses travaux (1986). Pour comprendre cela, il faut étudier les systèmes de déduction pour la logique propositionnelle du premier ordre et, comme nous allons le détailler par la suite, il faut trouver ou infirmer l'existence d'un système de déduction possédant des démonstrations de taille polynomiale pour toutes les tautologies. Afin que ces systèmes aient un sens du point de vue de la complexité, on doit néanmoins faire une hypothèse supplémentaire par rapport à la définition 2.14. En effet, on pourrait sinon imaginer un système de déduction avec pour chaque tautologie

P de la logique propositionnelle une démonstration p_P de taille un. Dans leur travaux fondateurs, Stephen Cook et Robert Reckhow (1974, 1979) imposent donc que la fonction $[\cdot]_{Cl}$ soit calculable polynomialement. Plus exactement, ils définissent un système de déduction comme une fonction (totale) surjective calculable en temps polynomial qui va d'un langage quelconque vers les tautologies, ce qui revient avec nos définitions à prendre comme langage de départ $Pf(\emptyset)$.

Dans la pratique, cette condition revient à dire que la vérification des démonstrations peut se faire en temps polynomial. En effet, si le système de déduction qu'on utilise est un système d'inférence, alors le calcul de $[\cdot]_{Cl}$ ne peut se contenter de renvoyer uniquement la conséquence des dérivations, il faut aussi vérifier que ces dérivations sont bien correctes. En général, dans ce cas, on définit le langage de départ comme celui de toutes les dérivations, y compris celles qui ne sont pas correctes, et la fonction surjective qui va vers les tautologies prend en entrée une dérivation, la vérifie, et renvoie sa conclusion si elle est correcte ou une tautologie quelconque sinon. Si on ne se place pas dans ce cadre et qu'on ne cherche pas à démontrer des résultats de théorie de la complexité, on pourrait bien entendu imaginer des systèmes de déduction dont la vérification des démonstrations, si elle reste décidable, n'est plus exponentielle. Par exemple, en déduction modulo, on peut utiliser un système de réécriture dont la congruence n'est pas calculable en temps polynomial, et si on considère qu'une démonstration ne comporte que les étapes de déduction et que c'est au cours de la vérification que les étapes de réécriture sont validées, alors on ne rentre pas dans le cadre de Stephen Cook et Robert Reckhow (1979). On peut alors se demander si on a affaire à des démonstrations, ou à des esquisses de démonstrations retraçant uniquement les grandes étapes. Il est à noter que pour vraiment être dans le cadre Stephen Cook et Robert Reckhow (1979), il faut également faire attention à ce qu'on appelle la taille de la démonstration. Telle qu'ils l'utilisent, il s'agit de la taille des mots du langage utilisé en entrée de la fonction qui leur sert de système de déduction. Cela implique que si on considère un système d'inférence, la taille d'une démonstration est la somme de la taille de toutes les formules apparaissant dans la dérivation. Toutefois, dans la suite de ce chapitre, on considérera souvent pour des questions de simplicité que la taille des démonstrations est le nombre d'étapes, c'est-à-dire le nombre d'applications de règles d'inférence, ce qui est une mesure différente. Le théorème 7 de Samuel Buss (1994) (qui reprend les travaux de Rohit Parikh (1973) et Jan Krajíček (1989)) établit toutefois un lien entre les deux, puisqu'il énonce que dans les systèmes schématiques, la profondeur des formules présentes dans une démonstration peut être bornée par la profondeur de la conclusion plus un facteur linéaire du nombre d'étapes dans la démonstration.

Précisons les liens entre longueur des démonstration et théorie de la complexité. Stephen Cook et Robert Reckhow (1979) ont démontré que la fermeture de NP par complément est équivalente à l'existence d'un système de déduction pour la logique propositionnelle dans lequel il est possible de trouver une démonstration de taille polynomiale de chaque tautologie. Autrement dit, si on veut démontrer que NP=coNP, il

faut trouver un système de déduction correct et complet pour la logique propositionnelle du premier ordre et un polynôme f tels que pour toute démonstration $p \in Pf(\emptyset)$, on ait $|p| \leq f(|[p]_{Cl}|)$ (en définissant correctement la taille $|\cdot|$ des démonstrations et des formules). On dit alors qu'un tel système de déduction est *borné polynomialement*. Stephen Cook et Robert Reckhow (1979) conjecture l'inexistence d'un tel système, ce qui aurait pour conséquence la différence entre P et NP. Il existe certains systèmes dont on peut démontrer qu'ils ne sont pas bornés polynomialement, mais il existe avant tout des résultats de *simulation polynomiale* :

Définition 5.1. *Un système de déduction* $(\mathbb{A}, \mathbb{P}_1, [\cdot]_1^{Pm}, [\cdot]_{Cl_1})$ *simule polynomialement un système de déduction* $(\mathbb{A}, \mathbb{P}_2, [\cdot]_2^{Pm}, [\cdot]_{Cl_2})$ *si il existe un fonction de traduction* g *de* $\mathbb{P}_2$ *vers* $\mathbb{P}_1$ *calculable polynomialement telle que pour tout* $p \in \mathbb{P}_2$ *on a* $[p]_{Cl_2} = [g(p)]_{Cl_1}$.

Ceci permet de définir des classes de système de déduction se simulant polynomialement les uns les autres, avec comme propriété que si un de ces systèmes est (resp. n'est pas) borné polynomialement, alors toute la classe l'est (resp., et de façon plus intéressante, ne l'est pas), d'où l'utilité de l'étude de ces classes. Stephen Cook et Robert Reckhow (1979) montrent ainsi que tous les systèmes schématiques corrects et complets pour la logique propositionnelle du premier ordre, ainsi que la déduction naturelle et le calcul des séquences restreints à cette logique se simulent tous polynomialement les uns les autres. Au contraire, le calcul des séquences sans coupure ne simule pas polynomialement ces systèmes, car il existe une famille de formules exhibée par Richard Statman (1978) qui ont une démonstration de taille au moins exponentielle sans coupure, mais qui ont des démonstrations de taille polynomiale avec coupures. On parle alors de *réduction de taille* exponentielle entre le calcul des séquences sans coupure et avec.

Définition 5.2 (Réduction de taille)**.** *Pour une fonction* h *sur les entiers, un système de déduction* $(\mathbb{A}, \mathbb{P}_1, [\cdot]_1^{Pm}, [\cdot]_{Cl_1})$ *réduit par* h *la taille des démonstrations d'un système de déduction* $(\mathbb{A}, \mathbb{P}_2, [\cdot]_2^{Pm}, [\cdot]_{Cl_2})$ *s'il existe une famille infinie* $(a_i)_{i\in\mathbb{N}}$ *de tautologies du second système (donc de formules de* $[Pf_2(\emptyset)]_{Cl_2}$*) telles que pour tout* i *il existe une démonstration de* a_i *dans le premier système (donc une démonstration* $p \in Pf_1(\emptyset) \cap [a_i]^{-1}_{Cl\,1}$*) dont la taille est plus petite que* h *appliquée à la taille de toute démonstration* q *dans le second système (*$q \in Pf_2(\emptyset) \cap [a_i]^{-1}_{Cl\,2}$*) :* $|p| \leq h(|q|)$.

On dira qu'un système de déduction réduit la taille des démonstrations d'un autre système arbitrairement *s'il la réduit par toute fonction sur les entiers.*

La proposition suivante précise une des méthodes souvent utilisée pour montrer qu'un système de déduction réduit arbitrairement la taille des démonstrations d'un autre :

Proposition 5.1. *Un système de déduction* $(\mathbb{A}, \mathbb{P}_1, [\cdot]_1^{Pm}, [\cdot]_{Cl_1})$ *réduit la taille des démonstrations d'un système de déduction* $(\mathbb{A}, \mathbb{P}_2, [\cdot]_2^{Pm}, [\cdot]_{Cl_2})$ *arbitrairement s'il existe une famille de formules* $(a_i)_{i\in\mathbb{N}}$ *telle que*

$$\mathsf{ai}{\downarrow}\,\frac{\top}{X \vee \neg X} \qquad \mathsf{aw}{\downarrow}\,\frac{\bot}{X} \qquad \mathsf{ac}{\downarrow}\,\frac{X \vee X}{X} \qquad X \text{ atomique pour } \mathsf{ai}{\downarrow},\ \mathsf{aw}{\downarrow} \text{ et } \mathsf{ac}{\downarrow}$$

$$\mathsf{s}\,\frac{X \wedge (Y \vee Z)}{(X \wedge Y) \vee Z} \qquad\qquad \mathsf{m}\,\frac{(X \wedge Y) \vee (Z \wedge W)}{(X \vee Z) \wedge (Y \vee W)}$$

FIG. 5.1 : Système d'inférence profonde KS pour la logique classique propositionnelle

- *il existe une borne $k \in \mathbb{N}$ telle que pour tout i il existe une démonstration $p \in Pf_1(\emptyset)$ telle que $[p]_{Cl_1} = a_i$ et $|p| \leq k$;*
- *pour tout i il existe une démonstration $p \in Pf_2(\emptyset)$ telle que $[p]_{Cl_2} = a_i$;*
- *il n'existe pas de borne k telle que pour tout i il existe une démonstration $p \in Pf_2(\emptyset)$ telle que $[p]_{Cl_2} = a_i$ et $|p| \leq k$.*

Dans ce chapitre, nous allons essentiellement étudier dans quelle mesure la déduction modulo permet de réduire la taille des démonstrations. Pour cela, on s'appuiera sur deux approches.

Premièrement, Alessio GUGLIELMI (2007) a introduit le formalisme de l'inférence profonde, qui consiste à utiliser des systèmes de déduction qui peuvent s'appliquer en profondeur sur les formules, de la même façon qu'on réécrit des termes en profondeur. Kai BRÜNNLER (2003) généralise donc le calcul des séquences pour pouvoir appliquer les règles d'inférence à tout niveau. Le système KS ainsi obtenu est représenté dans la figure 5.1. Ces règles peuvent s'appliquer à n'importe quelle profondeur dans une formule, et il faut les appliquer modulo l'associativité et la commutativité de $\vee$ et $\wedge$, et le fait que $\top$ est neutre pour $\wedge$ et idempotent pour $\vee$, et que dualement $\bot$ est neutre pour $\vee$ et idempotent pour $\wedge$. Le système KS est analytique, dans le sens où quand on fait une recherche de démonstration de bas en haut, on n'a pas besoin de deviner des formules supplémentaires, celles présentes dans le but courant suffisant à définir les instances des règles d'inférence que l'on peut y appliquer. En particulier, il n'existe pas de règle équivalente à $\vdash$, qui est présente dans d'autres systèmes d'inférence profonde et qui est alors

$$\mathsf{ai}{\uparrow}\,\frac{X \wedge \neg X}{\bot}\ X \text{ atomique}$$

Paola BRUSCOLI et Alessio GUGLIELMI (2009) ont montré que KS réduit exponentiellement la taille des démonstrations du calcul des séquences sans $\vdash$. Pour obtenir ceci, ils démontrent que les formules de STATMAN qui ne possèdent pas de démonstration de taille polynomiale en calcul des séquences sans coupures possèdent des démonstrations de taille polynomiale dans KS.

Il semble donc que le fait d'appliquer des règles d'inférence en profondeur dans les formules permet de réduire la taille des démonstrations. Dans une certaine mesure, la

déduction modulo permet également d'appliquer des règles en profondeur, puisqu'on les applique modulo une congruence sur les formules. Nous allons donc voir comment propager les résultats de Paola Bruscoli et Alessio Guglielmi (2009) à la déduction modulo.

Le deuxième point qui permet de penser que la déduction modulo permet de réduire la taille des démonstrations est le fait qu'elle permet de simuler des systèmes de déduction pour la logique d'ordre supérieur dans des systèmes de premier ordre modulo sans changer la taille des démonstrations, comme l'on montré Gilles Dowek et collaborateurs (2001). Or, un résultat énoncé par Kurt Gödel (1936) et démontré plus tard par Rohit Parikh (1973) dit que l'arithmétique d'ordre deux réduit arbitrairement la taille des démonstrations de l'arithmétique d'ordre un. Jan Krajíček (1989) a généralisé ce résultat à tous les ordres, et Samuel Buss (1994) l'a démontré pour le vrai langage de l'arithmétique, alors que les précédents résultats le faisait avec des prédicats ternaire pour définir l'addition et la multiplication. Le théorème prouvé par Samuel Buss est le suivant :

Théorème 5.2 (Samuel Buss, 1994, théorème 3)**.**
Pour tout système schématique A_i *pour l'arithmétique d'ordre* i *et* A_{i+1} *pour l'arithmétique d'ordre* $i+1$*, il existe une famille de formules* $(P_j)_{j\in\mathbb{N}}$ *telle que*
- *il existe une borne* $k \in \mathbb{N}$ *telle que pour tout* j *il existe une démonstration de* P_j *dans* A_{i+1} *en au plus* k *étapes ;*
- *pour tout* j *il existe une démonstration de* P_j *dans* A_i *;*
- *il n'existe pas de* k *tel que pour tout* j *il existe une démonstration de* P_j *dans* A_i *en au plus* k *étapes.*

Nous chercherons donc à voir dans quelle mesure ce théorème est invalidé si on simule l'arithmétique d'ordre supérieur dans un système du premier ordre modulo.

5.2 Réduction de taille en déduction modulo

5.2.1 Exemple simple

En déduction modulo, une partie de l'aspect calculatoire des démonstrations peut être passée dans les conditions des règles d'inférence. Si on considère que la taille d'une démonstration est son nombre d'applications de règles d'inférence, alors on voit que le fait de travailler en déduction modulo doit pouvoir réduire la taille des démonstrations. Néanmoins, comme indiqué ci-dessus, si on veut rester dans le cadre formel définit par Stephen Cook et Robert Reckhow (1979), il faut pouvoir vérifier les démonstrations en temps polynomial. Si on autorisait n'importe quel système de réécriture, et comme la réécriture a la puissance des machines de Turing, il va de soi qu'on pourrait faire diminuer la taille des démonstrations, en utilisant par exemple un système de réécriture

qui réécrit toutes les tautologies en $\top$. Dans ce cas, les démonstrations ne seraient par contre pas vérifiables en temps polynomial, puisqu'il faudrait calculer pour chaque règle d'inférence si les conditions sont respectées, ce qui peut même être indécidable. Nous allons montrer que l'on peut obtenir des réductions de la taille des démonstrations y compris dans les cas où on est capable de vérifier les démonstrations en temps polynomial.

Pour cela, il suffit de considérer un système de réécriture encodant l'addition par un prédicat ternaire :

$$\mathcal{A}dd \stackrel{\text{déf}}{=} \begin{cases} Add(\mathsf{O}, y, y) \to \top \\ Add(s(x), y, s(z)) \to Add(x, y, z) \end{cases}$$

On encode un entier naturel $n \in \mathbb{N}$ de façon standard par $\underline{n} \stackrel{\text{déf}}{=} \underbrace{s(\cdots s(}_{n \text{ fois}} \mathsf{O}))$. Pour ce système, $\stackrel{*}{\longleftrightarrow}$ est calculable polynomialement. En effet, ce système est confluent, et il termine en un nombre d'étapes linéaire par rapport à la taille des termes. Pour vérifier que deux formules sont congruentes, il suffit donc de comparer leur forme normale. Pourtant, il existe une réduction de taille arbitraire entre le calcul des séquences modulo $\mathcal{A}dd$ et le calcul des séquences étendu par toute présentation finie compatible avec $\mathcal{A}dd$.

Proposition 5.3. *Il existe une famille de formules $(a_i)_{i\in\mathbb{N}}$ telle que pour toute présentation finie Γ compatible avec $\mathcal{A}dd$,*

- *pour tout i, on peut démontrer $\vdash a_i$ en calcul des séquences modulo $\mathcal{A}dd$ avec une seule règle d'inférence ;*
- *il n'existe pas de borne au nombre d'applications de règles d'inférence nécessaires aux démonstrations de $\Gamma \vdash a_i$ en calcul des séquences sans modulo.*

Le fait d'utiliser une présentation compatible nous dit que l'on a une démonstration de $\Gamma \vdash a_i$ pour tout i, donc on peut utiliser la proposition 5.1 pour obtenir une réduction de taille arbitraire comme corollaire de cette proposition.

Démonstration. On prend comme famille de formules $(Add(\underline{i}, \underline{i}, \underline{2i}))_{i\in\mathbb{N}}$. Comme $Add(\underline{i}, \underline{i}, \underline{2i}) \xrightarrow{*} Add(\mathsf{O}, \underline{i}, \underline{i}) \longrightarrow \top$, on a la démonstration en calcul des séquences modulo

$$\dfrac{\vdash \top}{\vdash Add(\underline{i}, \underline{i}, \underline{2i})}\ Add(\underline{i}, \underline{i}, \underline{2i}) \xrightarrow{*} \top$$

Considérant maintenant une présentation finie Γ compatible avec $\mathcal{A}dd$. Les démonstrations de $\Gamma \vdash Add(\underline{i}, \underline{i}, \underline{2i})$ vont devoir détailler les calculs. Γ est constituée uniquement de formules vérifiées par l'addition. Par exemple on peut avoir des formules $\forall x\ y,\ Add(x, x, y) \Rightarrow Add(s^m(x), s^m(x), s^{2m}(y))$ pour un m fixé qui permet de diviser le nombre d'étapes de calculs par m. Néanmoins, comme on a un nombre fini de formules dans la présentation, cela ne suffit pas à borner la taille des démonstrations de $\Gamma \vdash Add(\underline{i}, \underline{i}, \underline{2i})$. □

Remarque 5.1 *:* Ce n'est plus vrai si on considère des présentations compatibles infinies. Il suffit pour s'en convaincre de prendre une présentation compatible contenant toutes les formules a_i.

Remarque 5.2 *:* Nous aurions pu utiliser l'encodage de l'addition par un symbole de fonction (infixe) + plutôt qu'un prédicat ternaire comme ci-dessus, avec le système sur les termes

$$\begin{aligned} \mathsf{O} + y &\to y \\ s(x) + y &\to s(x + y) \end{aligned} \quad .$$

Dans ce cas il aurait fallu fixer la signature utilisée pour être sûr de pouvoir obtenir une présentation compatible finie. Une autre solution serait de travailler avec un prédicat d'égalité vérifiant l'axiome de réflexivité (2.23) et le schéma d'axiome de Leibniz (2.24). On considère alors des présentations compatibles qui sont finies si on ne compte pas ce schéma.

5.2.2 Calcul des structures en déduction modulo

On peut être satisfait par le résultat précédant, qui montre que la déduction modulo peut permettre de réduire la taille des démonstrations de façon non bornée. Toutefois, si la taille des démonstrations est réduite, le temps qu'il faut pour les construire est le même, puisqu'il faut alors détailler toutes les étapes du calcul. En utilisant les résultats de Paola Bruscoli et Alessio Guglielmi (2009) en inférence profonde, nous démontrons que la déduction modulo permet de réduire la taille des démonstrations y compris si on y compte le nombre d'étapes de réécriture. Pour cela, on intègre le système KS décrit dans la figure 5.1 dans un système de déduction modulo.

Les règles d'inférence de KS peuvent être vue comme un système de réécriture modulo associativité, commutativité et propriétés de $\top$ et $\bot$. L'idée est alors de travailler modulo ce système. Néanmoins, en déduction modulo, on ne réécrit que des propositions atomiques, ce qui empêche de faire ceci directement. Nous allons d'abord encoder les formules par des termes, puis appliquer les règles de KS sur ces termes avant de les décoder. Pour cela, on utilise une astuce assez courante en déduction modulo, qui consiste à mimer la signature des prédicats à l'aide d'une signature pour les termes. Pour chaque symbole de prédicat A on considère un symbole de fonction $\dot{A}$ de même arité. Ici, comme on se place en logique propositionnelle, on peut se contenter de prédicats nullaires et de constantes. On encode également les connecteurs logiques par des symboles de fonctions. Ici, on se contente de $\top$, $\bot$, $\neg$, $\wedge$ et $\vee$ qui sont les connecteurs utilisé dans KS. On a donc deux constantes $\dot{\top}$ et $\emptyset$, un symbole de fonction unaire $\dot{\neg}$ ainsi que deux symboles de fonction binaires $\cap$ et $\cup$. On ajoute ensuite un prédicat unaire ϵ. On cherche à ce que $\epsilon(t)$ soit équivalent à la formule encodée par t. On utilise donc le système de réécriture

suivant :

$$\begin{aligned}
\epsilon(\dot{A}) &\to A && (A \text{ prédicat nullaire})\\
\epsilon(\dot{\top}) &\to \top\\
\epsilon(\emptyset) &\to \bot\\
\epsilon(\dot{\neg} x) &\to \neg\epsilon(x)\\
\epsilon(x \cap y) &\to \epsilon(x) \wedge \epsilon(y)\\
\epsilon(x \cup y) &\to \epsilon(x) \vee \epsilon(y)
\end{aligned}$$

On utilise également un prédicat nullaire spécial C qui permet de lancer la réduction.

$$C \to^{-} \epsilon(\dot{\top}) \ .$$

Pour les règles de KS, on va les voir comme des règles de réécriture allant de haut en bas. Comme ces règles d'inférence correspondent à des implications vue de haut en bas, et non des équivalences, on utilise des règles polarisées. On considère donc le système de réécriture sur les termes suivant :

$$\begin{aligned}
\dot{\top} &\to^{-} \dot{A} \cup \dot{\neg}\dot{A} && (A \text{ prédicat nullaire})\\
\emptyset &\to^{-} \dot{A} && ''\\
\dot{A} \cup \dot{A} &\to^{-} \dot{A} && ''\\
x \cap (y \cup z) &\to^{-} (x \cap y) \cup z\\
(x \cap y) \cup (z \cap w) &\to^{-} (x \cup z) \cap (y \cup w)
\end{aligned}$$

On notera $\mathcal{KS}$ la réunion de ces deux systèmes. Il est à noté que l'on doit alors appliquer ce système de réécriture modulo associativité, commutativité de $\wedge$ et $\vee$, et modulo propriétés de $\top$ et $\bot$. Rigoureusement, ce n'est pas possible en déduction modulo telle que nous l'avons présentée. Néanmoins, la formulation originelle par Gilles Dowek et collaborateurs (2003) utilise en fait une congruence basée sur un système de réécriture et une théorie équationnelle qui pourrait comprendre l'associativité, etc.

On peut montrer que travailler modulo ce système de réécriture n'ajoute rien d'un point de vue logique.

Proposition 5.4. *Si on peut démontrer une séquence en logique propositionnelle qui ne comporte pas de ϵ en calcul des séquences polarisé modulo $\mathcal{KS}$ alors on peut la démontrer en calcul des séquences sans modulo.*

Démonstration. Il suffit de remarquer que puisque les règles de KS sont des implications valides, si $\epsilon(p) \xrightarrow[\mathcal{KS}\ -]{*} \epsilon(q)$ et que $\epsilon(p) \xrightarrow[\mathcal{KS}]{*} P$ et $\epsilon(p) \xrightarrow[\mathcal{KS}]{*} Q$ pour des formules P et Q sans ϵ, alors on peut démontrer $P \Rightarrow Q$ dans le calcul des séquences non modulo. Il suffit alors d'utiliser une coupure avec cette démonstration pour passer de P à Q. De même, si on a $C \xrightarrow[\mathcal{KS}\ -]{} \epsilon(\dot{\top})$ on peut le remplacer par une coupure et une démonstration de $C \Rightarrow \top$. □

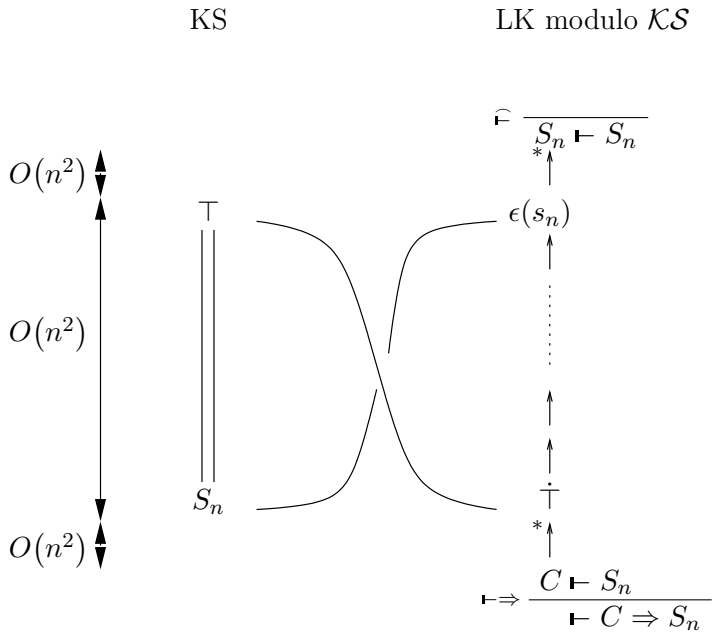


FIG. 5.2 : Simulation d'une démonstration dans KS en déduction modulo

On peut alors utiliser la famille de formules de Paola BRUSCOLI et Alessio GUGLIELMI (2009) pour montrer que le calcul des séquences sans $\vdash\!\!\!\smile$ modulo $\mathcal{KS}$ réduit la taille des démonstrations du calcul des séquences sans $\vdash\!\!\!\smile$ exponentiellement, y compris si on compte les pas de réécriture dans la taille de la démonstration. On considère donc les formules de STATMAN définies récursivement :

Définition 5.3 (Formules de STATMAN)**.** *On considère une signature infinie composée de prédicats nullaires C_i et D_i pour tout $i \in \mathbb{N}$. On définit alors récursivement les formules suivantes ($k < n$) :*

- $P_k^n \stackrel{\text{déf}}{=} C_k \wedge (\neg C_{k+1} \vee \neg D_{k+1}) \wedge \cdots \wedge (\neg C_n \vee \neg D_n)$;
- $Q_k^n \stackrel{\text{déf}}{=} D_k \wedge (\neg C_{k+1} \vee \neg D_{k+1}) \wedge \cdots \wedge (\neg C_n \vee \neg D_n)$;
- $S_n \stackrel{\text{déf}}{=} (C_n \wedge D_n) \vee (P_{n-1}^n \wedge Q_{n-1}^n) \vee \cdots \vee (P_0^n \wedge Q_0^n) \vee \neg C_0 \vee \neg D_0$.

Proposition 5.5. *Le calcul des séquences polarisé modulo $\mathcal{KS}$ sans coupure réduit exponentiellement la taille des démonstration du calcul des séquences sans coupure, y compris en comptant les étapes de conversion dans la taille de la démonstration.*

Démonstration. La figure 5.2 illustre cette démonstration. Le nombre de proposition atomiques de S_n est $2n^2 + 2$. Comme les négations sont toutes au niveau des atomes, le nombre de symbole y apparaissant est donc au plus de $2\times(2n^2+2)+2n^2+1 = 6n^2+5$. On peut donc encoder S_n au niveau des termes en un terme s_n tel que $\epsilon(s_n) \xrightarrow[\mathcal{KS}]{*} S_n$ en au plus

$6n^2 + 5$ étapes. De plus, Paola Bruscoli et Alessio Guglielmi (2009, Théorème 3.12) nous disent que $\dot{\top} \xrightarrow[\mathcal{KS}\ -]{*} s_n$ en $O(n^2)$ étapes. On a donc $C \xrightarrow[\mathcal{KS}\ -]{*} S_n$ en $O(n^2)$ étapes. On peut donc démontrer les formules $C \Rightarrow S_n$ sans coupure en $O(n^2)$ étapes. Ce n'est pas possible en calcul des séquences sans coupure d'après Richard Statman (1978). □

Le principal défaut de $\mathcal{KS}$ est son absence de confluence. La réduction du non-déterminisme du calcul des structures est d'ailleurs un sujet actif de recherche, principalement étudié par Ozan Kahramanoğullari (2006). De plus, si on a une signature infinie, on aura également un système de réécriture infini, du fait de la première règle permettant d'encoder les propositions atomiques. On peut bien entendu limiter la signature à celle des formules que l'on cherche à démontrer.

5.3 Simulation de l'arithmétique d'ordre supérieure en arithmétique du premier ordre modulo

Comme il est indiqué plus haut, l'arithmétique d'ordre deux réduit arbitrairement la taille des démonstrations de l'arithmétique d'ordre un. On peut même démontrer une telle réduction entre l'ordre i et l'ordre $i - 1$ pour tout $i > 1$. Nous nous intéressons à la façon d'utiliser la déduction modulo pour rester au premier ordre sans augmenter la taille des démonstrations.

5.3.1 Système schématique pour l'arithmétique d'ordre i

Nous donnons maintenant un système schématique particulier pour l'arithmétique d'ordre i. Il s'agit de le fixer pour la suite, mais nous aurions très bien pu en choisir un autre, Samuel Buss (1994) ayant démontré le théorème 5.2 pour n'importe quel système schématique, et même en fait pour des généralisations de systèmes schématiques dans lesquelles il est possible d'utiliser n'importe quelle tautologie comme axiome. Il est à noter que l'arithmétique d'ordre i peut être vue comme une théorie en logique du premier ordre, dans le même sens que la théorie des ensembles de Zermelo est une théorie du premier ordre. Néanmoins, Czesław Ryll-Nardzewski (1952) a démontré qu'il n'en existe pas de présentation finie, si on reste dans la même signature.

L'arithmétique d'ordre i est une théorie multisortée avec pour sorte $1, \ldots, \mathrm{i}$ et pour signature

$$\begin{array}{lll} \mathsf{O} : 1 & + : [1;1] \to 1 & = : [1;1] \\ s : [1] \to 1 & \times : [1;1] \to 1 & \in^j : [\mathrm{j};\mathrm{j}+1] \end{array}$$

pour $1 \leq j < i$.

Nous avons besoin de règles d'inférence pour la logique classique du premier ordre avec égalité. Pour cela, on prend les règles d'inférence (2.4) à (2.24) (cf. section 2.3.2). On a besoin ensuite d'un certain nombre d'axiomes pour définir les symboles de fonction

de la signature. Ces axiomes sont parfois appelés les axiomes de Robinson (cf. Andrzej Mostowski, Raphael Robinson et Alfred Tarski, 1953).

$$\forall x^1,\ \neg\ \mathsf{O} = s(x^1) \tag{5.1}$$
$$\forall x^1 y^1,\ s(x^1) = s(y^1) \Rightarrow x^1 = y^1 \tag{5.2}$$
$$\forall x^1,\ (\neg\ x^1 = \mathsf{O}) \Rightarrow \exists y^1,\ x^1 = s(y^1) \tag{5.3}$$
$$\forall x^1,\ x^1 + \mathsf{O} = x^1 \tag{5.4}$$
$$\forall x^1 y^1,\ x^1 + s(y^1) = s(x^1 + y^1) \tag{5.5}$$
$$\forall x^1,\ x^1 \times \mathsf{O} = \mathsf{O} \tag{5.6}$$
$$\forall x^1 y^1,\ x^1 \times s(y^1) = x^1 \times y^1 + x^1 \tag{5.7}$$

Il y a ensuite le schéma d'induction

$$A(\mathsf{O}) \Rightarrow \big(\forall y^1,\ A(y^1) \Rightarrow A(s(y^1))\big) \Rightarrow \forall x^1,\ A(x^1) \tag{5.8}$$

et entre chaque ordre un schéma de compréhension (cf. section 2.2.2)

$$\exists x^{\mathsf{j}+1},\ \forall y^{\mathsf{j}},\ y^{\mathsf{j}} \in^{\mathsf{j}} x^{\mathsf{j}+1} \Leftrightarrow A(y^{\mathsf{j}}) \qquad (x^{\mathsf{j}+1} \notin FV(A(y^{\mathsf{j}}))) \tag{5.9}$$

pour tout $1 \leq j < i$.

Remarque 5.3 *:* (5.3) est redondant à partir du moment où on a le schéma d'induction, ce qui fait qu'on pourra ne pas le prendre en compte dans la suite.

Remarque 5.4 *:* Suivant qu'on considère ou non le schéma d'axiome du tiers exclu (2.20), c'est-à-dire suivant qu'on se place en logique classique ou intuitionniste, on parle d'arithmétique de Peano ou de Heyting. Les résultats de ce chapitre sont valables dans les deux cas. Si on se place en logique classique, il faudra par exemple considérer le tiers exclu (2.20) comme règle d'inférence de la déduction naturelle.

À partir de maintenant on notera
- par $\mathrm{A}_i \vdash^{\mathsf{S}}_{k} P$ le fait qu'il existe une démonstration de P dans ce système schématique avec au plus k applications de règles d'inférence ;
- par $\mathrm{A}_i \vdash^{\mathsf{N}}_{k} P$ le fait qu'il existe un ensemble fini Γ d'instances des schémas d'axiomes (5.1) à (5.9) plus (2.23) et (2.24) tel que $\Gamma \vdash P$ peut être démontré en déduction naturelle en au plus k applications de règles d'inférence ;
- par $\mathrm{A}_i \vdash^{\mathsf{N}}_{k\,\mathcal{R}} P$ la même notion mais en déduction naturelle modulo $\mathcal{R}$.

5.3.2 Traduction entre système schématique et déduction naturelle

Nous n'avons pas défini de système schématique modulo, bien que cela aurait été possible. Néanmoins, le théorème 5.2 est démontré pour les systèmes schématiques, alors

que nous allons travailler en déduction naturelle modulo. Pour résoudre cela, il faut vérifier qu'il existe des traductions entre systèmes schématiques et déduction naturelle qui ne font pas trop grossir les démonstrations, pour être sûr que les réductions de taille que nous présenterons ne proviennent pas de la différence entre les systèmes d'inférence. Si on se contentait de travailler en logique propositionnelle, il n'y aurait pas de problème puisque Stephen Cook et Robert Reckhow (1979) ont démontré que les systèmes schématiques et la déduction naturelle se simulent polynomialement l'un l'autre. Ici, pour la logique prédicative, on donne des traductions explicites, qui ne sont d'ailleurs pas toujours polynomiale, mais qui sont suffisantes pour la suite.

Traduction système schématique vers déduction naturelle

Cette traduction est donnée par Gerhard Gentzen (1934) pour démontrer la correction et la complétude de ses systèmes de déduction par rapport au système schématique composé des règles (2.4) à (2.22). Il s'agit de démontrer que chacune des règles d'inférence de ce système est dérivable en hauteur bornée en déduction naturelle. Par exemple, au schéma d'axiome (2.8) correspond la démonstration

$$\dfrac{\dfrac{\dfrac{\dfrac{[X]_{(\mathrm{iii})} \qquad [X \Rightarrow Y \Rightarrow Z]_{(\mathrm{i})}}{Y \Rightarrow Z}\,\Rightarrow_{\vdash}}{Z}\,\Rightarrow_{\vdash}}{X \Rightarrow Z}\,(\mathrm{iii})\,\vdash_{\Rightarrow}}{Y \Rightarrow X \Rightarrow Z}\,(\mathrm{ii})\,\vdash_{\Rightarrow}}{(X \Rightarrow Y \Rightarrow Z) \Rightarrow Y \Rightarrow X \Rightarrow Z}\,(\mathrm{i})\,\vdash_{\Rightarrow}$$

et la règle d'inférence (2.22) à la dérivation

$$\dfrac{\dfrac{\dfrac{[\exists x^{\mathsf{j}},\ Y(x^{\mathsf{j}})]_{(\mathrm{i})} \qquad \dfrac{[Y(y^{\mathsf{j}})]_{(\mathrm{ii})} \qquad Y(y^{\mathsf{j}}) \Rightarrow X}{X}\,\Rightarrow_{\vdash}}{X}\,(\mathrm{ii})\,\exists_{\vdash}}{\exists x^{\mathsf{j}},\ Y(x^{\mathsf{j}}) \Rightarrow X}\,(\mathrm{i})\,\vdash_{\Rightarrow}$$

Il est à noter que la condition « y n'est pas libre dans $(\exists x,\ Y(x)) \Rightarrow X$ » permet de s'assurer qu'on peut trouver un « y [qui] n'est pas libre dans $Y(x), X$ » pour $\exists_{\vdash}$.

Les instances des schémas d'axiomes (5.1) à (5.9) plus (2.23) et (2.24) sont quant à elles laissées telles quelles.

Proposition 5.6. *Il est possible de traduire une démonstration de longueur k dans le système schématique pour l'arithmétique d'ordre i en une démonstration en déduction naturelle utilisant un nombre fini d'instances de* (5.1) *à* (5.9) *plus* (2.23) *et* (2.24) *comme hypothèses de longueur linéaire par rapport à k.*

$$\mathrm{A}_i \vdash^{\mathsf{S}}_{k} P \quad \rightsquigarrow \quad \mathrm{A}_i \vdash^{\mathsf{N}}_{O(k)} P$$

Traduction déduction naturelle vers système schématique

Pour la traduction en sens inverse, on s'inspire de la traduction d'un λ-terme en terme de la logique combinatoire (cf. Haskell Curry, Robert Feys et William Craig, 1958). On définit de façon mutuellement récursive deux fonctions de traduction : T traduit une démonstration en déduction naturelle de A utilisant les hypothèses Γ en une démonstration de A sous les hypothèses Γ dans le système schématique constitué des règles d'inférence (2.4) à (2.22), tandis que T_A traduit une démonstration en déduction naturelle de de B utilisant les hypothèses Γ en une démonstration de B sous les hypothèses Γ, A dans ce même système schématique.

$$\mathrm{T}\left(\begin{array}{c}\overset{[A]}{\vdots\,\pi}\\ \overset{\vdash}{\Rightarrow}\ \dfrac{B}{A\Rightarrow B}\end{array}\right) \overset{\text{déf}}{=} \mathrm{T}_A\left(\begin{array}{c}A\\ \vdots\,\pi\\ B\end{array}\right)$$

$$\mathrm{T}\left(\begin{array}{c}\pi_1\quad\pi_2\\ \overset{\Rightarrow}{\vdash}\ \dfrac{A\quad A\Rightarrow B}{B}\end{array}\right) \overset{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi_1)\quad \mathrm{T}(\pi_2)}{\dfrac{A\quad A\Rightarrow B}{B}}$$

$$\mathrm{T}\left(\begin{array}{c}\pi_1\quad\pi_2\\ \overset{\vdash}{\wedge}\ \dfrac{A\quad B}{A\wedge B}\end{array}\right) \overset{\text{déf}}{=}$$

$$(2.4)\ \dfrac{\mathrm{T}(\pi_1)\qquad (2.4)\ \dfrac{\mathrm{T}(\pi_2)\qquad [B\Rightarrow A\Rightarrow B]_{(2.6)}}{A\Rightarrow B}\qquad (2.4)\ \dfrac{[A\Rightarrow A]_{(2.5)}\qquad [\cdots]_{(2.12)}}{(A\Rightarrow B)\Rightarrow A\Rightarrow (A\wedge B)}}{A\Rightarrow (A\wedge B)}}{A\wedge B}$$

$$\mathrm{T}\left(\begin{array}{c}\pi\\ \overset{\wedge}{\vdash}\ \dfrac{A\wedge B}{A}\end{array}\right) \overset{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi)\quad A\wedge B\qquad [A\wedge B\Rightarrow A]_{(2.10)}}{A}$$

et pareil avec (2.11) pour l'autre côté.

$$\mathrm{T}\left(\begin{array}{c}\pi\\ \overset{\vdash}{\vee}\ \dfrac{A}{A\vee B}\end{array}\right) \overset{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi)\quad A\qquad [A\Rightarrow (A\vee B)]_{(2.13)}}{A\vee B}$$

et pareil avec (2.14) pour l'autre côté.

$$\mathrm{T}\left(\begin{array}{c}\overset{[A]}{\vdots\,\pi_2}\qquad\overset{[B]}{\vdots\,\pi_3}\\ \overset{\vee}{\vdash}\ \dfrac{\pi_1\quad A\vee B\qquad C\qquad C}{C}\end{array}\right) \overset{\text{déf}}{=}$$

$$(2.4)\ \dfrac{\mathrm{T}(\pi_1)\qquad (2.4)\ \dfrac{\mathrm{T}_B(\pi_3)\quad B\Rightarrow C\qquad (2.4)\ \dfrac{\mathrm{T}_A(\pi_2)\quad A\Rightarrow C\qquad [(A\Rightarrow C)\Rightarrow (B\Rightarrow C)\Rightarrow (A\vee B)\Rightarrow C]_{(2.15)}}{(B\Rightarrow C)\Rightarrow (A\vee B)\Rightarrow C}}{(A\vee B)\Rightarrow C}}{C}$$

$$\mathrm{T}\left(\begin{array}{c}\pi\\ \vdash_{\forall}\ \dfrac{\{y/x\}A}{\forall x,\ A}\end{array}\right) \stackrel{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi) \qquad \{y/x\}A \qquad (2.21)\ \dfrac{[\{y/x\}A \Rightarrow \{y/x\}A]_{(2.5)}}{\{y/x\}A \Rightarrow \forall x,\ A}}{\forall x,\ A}$$

Les conditions sont bien vérifiées

$$\mathrm{T}\left(\begin{array}{c}\pi\\ \vdash_{\forall}\ \dfrac{\forall x,\ A}{\{t/x\}A}\end{array}\right) \stackrel{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi) \qquad \forall x,\ A \qquad [\forall x,\ A \Rightarrow \{t/x\}A]_{(2.18)}}{\{t/x\}A}$$

$$\mathrm{T}\left(\begin{array}{c}\pi\\ \vdash_{\exists}\ \dfrac{\{t/x\}A}{\exists x,\ A}\end{array}\right) \stackrel{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi) \qquad \{t/x\}A \qquad [\{t/x\}A \Rightarrow \exists x,\ A]_{(2.19)}}{\exists x,\ A}$$

$$\mathrm{T}\left(\begin{array}{c}\qquad\qquad [\{y/x\}A]\\ \qquad\qquad\ \vdots\ \pi_2\\ \pi_1 \qquad\quad B\\ \vdash_{\exists}\ \dfrac{\exists x,\ A}{B}\end{array}\right) \stackrel{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi_1) \qquad \exists x,\ A \qquad (2.22)\ \dfrac{\mathrm{T}_A(\pi_2) \quad \{y/x\}A \Rightarrow B}{(\exists x,\ A) \Rightarrow B}}{B}$$

Les conditions sont bien vérifiées.

$$\mathrm{T}\left(\begin{array}{c}\pi\\ \vdash_{\perp}\ \dfrac{\perp}{A}\end{array}\right) \stackrel{\text{déf}}{=}$$

$$(2.4)\ \dfrac{[A \Rightarrow A]_{(2.5)} \qquad (2.4)\ \dfrac{\mathrm{T}(\pi) \quad \perp \qquad [\perp \Rightarrow (A \Rightarrow A) \Rightarrow \perp]_{(2.6)}}{(A \Rightarrow A) \Rightarrow \perp} \qquad [((A \Rightarrow A) \Rightarrow \perp) \Rightarrow (A \Rightarrow A) \Rightarrow A]_{(2.17)}}{(A \Rightarrow A) \Rightarrow A}}{A}$$

$$\mathrm{T}(\mathrm{A}) \stackrel{\text{déf}}{=} \mathrm{A}$$

$$\mathrm{T}_A\left(\begin{array}{c}\qquad [B]\\ \qquad\ \vdots\ \pi\\ \vdash_{\Rightarrow}\ \dfrac{C}{B \Rightarrow C}\end{array}\right) \stackrel{\text{déf}}{=} \mathrm{T}_A\left(\begin{array}{c}\mathrm{T}_B(\pi)\\ B \Rightarrow C\end{array}\right)$$

$$\mathrm{T}_A\left(\begin{array}{c}A \qquad\quad A\\ \vdots\ \pi_1 \qquad \vdots\ \pi_2\\ \vdash_{\Rightarrow}\ \dfrac{B \qquad B \Rightarrow C}{C}\end{array}\right) \stackrel{\text{déf}}{=}$$

$$(2.4)\ \dfrac{\mathrm{T}_A(\pi_2) \quad A \Rightarrow B \Rightarrow C \qquad [\cdots]_{(2.8)}}{B \Rightarrow A \Rightarrow C} \qquad (2.4)\ \dfrac{\mathrm{T}_A(\pi_1) \quad A \Rightarrow B \qquad [\cdots]_{(2.9)}}{(B \Rightarrow A \Rightarrow C) \Rightarrow A \Rightarrow A \Rightarrow C}}{A \Rightarrow A \Rightarrow C} \qquad [\cdots]_{(2.7)}}{A \Rightarrow C}$$

$$\mathrm{T}_A\left(\begin{array}{c} A \quad A \\ \vdots\,\pi_1 \quad \vdots\,\pi_2 \\ \vdash_\wedge \dfrac{B \qquad C}{B \wedge C} \end{array}\right) \overset{\text{déf}}{=}$$

$$(2.4)\ \frac{\mathrm{T}_A(\pi_2)\ A \Rightarrow C \qquad (2.4)\ \dfrac{\mathrm{T}_A(\pi_1)\ A \Rightarrow B \qquad [(A \Rightarrow B) \Rightarrow (A \Rightarrow C) \Rightarrow A \Rightarrow (B \wedge C)]_{(2.12)}}{(A \Rightarrow C) \Rightarrow A \Rightarrow (B \wedge C)}}{A \Rightarrow (B \wedge C)}$$

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ \vdash_\wedge \dfrac{B \wedge C}{B} \end{array}\right) \overset{\text{déf}}{=} (2.4)\ \frac{[(B \wedge C) \Rightarrow B]_{(2.10)} \qquad (2.4)\ \dfrac{\mathrm{T}_A(\pi)\ A \Rightarrow (B \wedge C) \qquad [\cdots]_{(2.9)}}{((B \wedge C) \Rightarrow B) \Rightarrow A \Rightarrow B}}{A \Rightarrow B}$$

et pareil avec (2.11) pour l'autre côté.

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ \vdash_\vee \dfrac{B}{B \vee C} \end{array}\right) \overset{\text{déf}}{=} (2.4)\ \frac{[B \Rightarrow (B \vee C)]_{(2.13)} \qquad (2.4)\ \dfrac{\mathrm{T}_A(\pi)\ A \Rightarrow B \qquad [\cdots]_{(2.9)}}{(B \Rightarrow (B \vee C)) \Rightarrow A \Rightarrow (B \vee C)}}{A \Rightarrow (B \vee C)}$$

et pareil avec (2.14) pour l'autre côté.

$$\mathrm{T}_A\left(\begin{array}{c} A \quad A,[B] \quad A,[C] \\ \vdots\,\pi_1 \quad \vdots\,\pi_2 \quad \vdots\,\pi_3 \\ \vdash_\vee \dfrac{B \vee C \qquad D \qquad D}{D} \end{array}\right) \overset{\text{déf}}{=}$$

$$(2.4)\ \frac{\mathrm{T}_C\begin{pmatrix}\mathrm{T}_A(\pi_3)\\ A \Rightarrow D\end{pmatrix}\ C \Rightarrow A \Rightarrow D \qquad (2.4)\ \dfrac{\mathrm{T}_B\begin{pmatrix}\mathrm{T}_A(\pi_2)\\ A \Rightarrow D\end{pmatrix}\ B \Rightarrow A \Rightarrow D \qquad [\cdots]_{(2.15)}}{(C \Rightarrow A \Rightarrow D) \Rightarrow (B \vee C) \Rightarrow A \Rightarrow D}}{(B \vee C) \Rightarrow A \Rightarrow D} \qquad (2.4)\ \dfrac{\mathrm{T}_A(\pi_1)\ A \Rightarrow (B \vee C) \qquad [\cdots]_{(2.9)}}{((B \vee C) \Rightarrow A \Rightarrow D) \Rightarrow A \Rightarrow A \Rightarrow D}$$

$$(2.4)\ \frac{\vdots \qquad [(A \Rightarrow A \Rightarrow D) \Rightarrow A \Rightarrow D]_{(2.7)}}{A \Rightarrow D}$$

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ \vdash_\forall \dfrac{\{y/x\}B}{\forall x,\ B} \end{array}\right) \overset{\text{déf}}{=} (2.21)\ \frac{\mathrm{T}_A(\pi)\ A \Rightarrow \{y/x\}B}{A \Rightarrow \forall x,\ B}$$

Les conditions sont bien vérifiées.

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ \vdash_\forall \dfrac{\forall x,\ B}{\{t/x\}B} \end{array}\right) \overset{\text{déf}}{=} (2.4)\ \frac{[(\forall x,\ B) \Rightarrow \{t/x\}B]_{(2.18)} \qquad (2.4)\ \dfrac{\mathrm{T}_A(\pi)\ A \Rightarrow \forall x,\ B \qquad [\cdots]_{(2.9)}}{((\forall x,\ B) \Rightarrow \{t/x\}B) \Rightarrow A \Rightarrow \{t/x\}B}}{A \Rightarrow \{t/x\}B}$$

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ \overset{\exists}{\vdash}\ \dfrac{\{t/x\}B}{\exists x,\ B} \end{array}\right) \overset{\text{déf}}{=} (2.4)\ \dfrac{[\{t/x\}B \Rightarrow \exists x,\ B]_{(2.19)} \quad (2.4)\ \dfrac{\mathrm{T}_A(\pi) \quad A \Rightarrow \{t/x\}B \qquad [\cdots]_{(2.9)}}{(\{t/x\}B \Rightarrow \exists x,\ B) \Rightarrow A \Rightarrow \exists x,\ B}}{A \Rightarrow \exists x,\ B}$$

$$\mathrm{T}_A\left(\begin{array}{ccc} A & & A, [\{y/x\}B] \\ \vdots\,\pi_1 & & \vdots\,\pi_2 \\ \overset{\exists}{\vdash}\ \dfrac{\exists x,\ B}{C} & & C \end{array}\right) \overset{\text{déf}}{=}$$

$$(2.4)\ \dfrac{(2.22)\ \dfrac{\mathrm{T}_B\left(\begin{array}{c}\mathrm{T}_A(\pi_2)\\ A \Rightarrow C\end{array}\right)}{\{y/x\}B \Rightarrow A \Rightarrow C} \quad (2.4)\ \dfrac{\mathrm{T}_A(\pi_1) \quad A \Rightarrow \exists x,\ B \qquad [\ldots]_{(2.9)}}{(\exists x,\ B \Rightarrow A \Rightarrow C) \Rightarrow A \Rightarrow A \Rightarrow C}}{\exists x,\ B \Rightarrow A \Rightarrow C} \quad [\cdots]_{(2.7)}}{A \Rightarrow A \Rightarrow C}}{A \Rightarrow C}$$

Les conditions sont bien vérifiées.

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ \overset{\bot}{\vdash}\ \dfrac{\bot}{B} \end{array}\right) \overset{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}_A(\pi) \quad A \Rightarrow \bot \qquad [(A \Rightarrow \bot) \Rightarrow A \Rightarrow B]_{(2.17)}}{A \Rightarrow B}$$

$$\mathrm{T}_A(A) \overset{\text{déf}}{=} [A \Rightarrow A]_{(2.5)}$$

$$\mathrm{T}_A\left(\begin{array}{c}\pi\\ B\end{array}\right) \overset{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}(\pi) \quad B \qquad [B \Rightarrow A \Rightarrow B]_{(2.6)}}{A \Rightarrow B}$$

si on n'utilise pas l'hypothèse A dans π.

La définition de T_A pour $\overset{\vdash}{\Rightarrow}$ ne boucle pas, car il n'y a plus de $\overset{\vdash}{\Rightarrow}$ dans $\mathrm{T}_B(\pi)$. Néanmoins, il faut alors définir ce que donne T_A sur des démonstrations contenant des règles d'inférence du système schématique. Comme (2.4) correspond à $\overset{\Rightarrow}{\vdash}$ la traduction est déjà définie. Pour les schémas d'axiomes on utilise la dernière définition. Il reste donc le cas de (2.21) et (2.22).

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ (2.21)\ \dfrac{B \Rightarrow C(t)}{B \Rightarrow \forall x,\ C(x)} \end{array}\right) \overset{\text{déf}}{=} (2.4)\ \dfrac{\mathrm{T}_A(\pi) \quad A \Rightarrow B \Rightarrow C(t) \qquad \varpi_1}{(2.21)\ \dfrac{(A \Rightarrow B \Rightarrow C(t)) \Rightarrow (A \wedge B) \Rightarrow C(t)}{(A \wedge B) \Rightarrow C(t)} \quad \varpi_2 \atop (2.4)\ \dfrac{(A \wedge B) \Rightarrow \forall x,\ C(x)}{A \Rightarrow B \Rightarrow \forall x,\ C(x)}}$$

où ϖ_1 est une démonstration quelconque de $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \wedge B) \Rightarrow C$, et ϖ_2 de $((A \wedge B) \Rightarrow C) \Rightarrow A \Rightarrow B \Rightarrow C$, qui utilisent les schémas d'axiomes (2.5) à (2.12) et la règle d'inférence (2.4).

$$\mathrm{T}_A\left(\begin{array}{c} A \\ \vdots\,\pi \\ (2.22)\ \dfrac{B(t) \Rightarrow C}{(\exists x,\ B(x)) \Rightarrow C} \end{array}\right) \overset{\text{déf}}{=}$$

$$\dfrac{\dfrac{\dfrac{\mathrm{T}_A(\pi)}{A \Rightarrow B(t) \Rightarrow C}\qquad [(A \Rightarrow B(t) \Rightarrow C) \Rightarrow B(t) \Rightarrow A \Rightarrow C]_{(2.8)}}{B(t) \Rightarrow A \Rightarrow C}\,(2.4)}{\exists x,\ B(x) \Rightarrow A \Rightarrow C}\,(2.22)\qquad \dfrac{}{A \Rightarrow \exists x,\ B(x) \Rightarrow C}\,(2.4)$$

La traduction ainsi obtenue n'est pas linéaire, mais ce sera suffisant pour la suite.

Proposition 5.7. *Il est possible de traduire une démonstration de longueur k en déduction naturelle utilisant un nombre fini d'instances de* (5.1) *à* (5.9) *plus* (2.23) *et* (2.24) *comme hypothèses en une démonstration dans le système schématique pour l'arithmétique d'ordre i de longueur exponentielle par rapport à k.*

$$\mathrm{A}_i \vdash^{\mathsf{N}}_{k} P \quad \rightsquigarrow \quad \mathrm{A}_i \vdash^{\mathsf{S}}_{O\left(K^k\right)} P$$

avec K dépendant uniquement du système schématique choisi pour l'arithmétique.

Démonstration. Posons K le nombre maximum d'étapes qui sont ajoutées en plus des appels récursifs dans chacun des cas de la définition de T_A.

Montrons tout d'abord que dans les cas où une démonstration ϖ ne contient pas de règles $\overset{\vdash}{\underset{\Rightarrow}{}}$, $\overset{\vee}{\underset{\vdash}{}}$ ou $\overset{\exists}{\underset{\vdash}{}}$, alors $|\mathrm{T}_A(\varpi)| \leq K|\varpi|$, qu'elle que soit la formule A. Pour cela, on regarde les différents cas de la définition de T_A, par induction sur ϖ. Nous ne détaillons que celui de $\overset{\Rightarrow}{\underset{\vdash}{}}$, les autres étant similaires : en reprenant les notations de la définition de T_A ci-dessus, on a

$$\begin{aligned}
|\mathrm{T}_A(\varpi)| &= |\mathrm{T}_A(\pi_1)| + |\mathrm{T}_A(\pi_2)| + 7 \\
&\leq K|\pi_1| + K|\pi_2| + K && \text{par hypothèse d'induction et par définition de } K \\
&\leq K(|\pi_1| + |\pi_2| + 1) \\
&\leq K|\varpi|
\end{aligned}$$

Montrons maintenant que dans tous les cas, $|\mathrm{T}_A(\varpi)| \leq K^{|\varpi|}$, toujours par induction sur ϖ. Dans le cas de $\overset{\Rightarrow}{\underset{\vdash}{}}$, on a

$$\begin{aligned}
|\mathrm{T}_A(\varpi)| &= |\mathrm{T}_A(\pi_1)| + |\mathrm{T}_A(\pi_2)| + 7 \\
&\leq K^{|\pi_1|} + K^{|\pi_2|} + K && \text{par hypothèse d'induction et par définition de } K \\
&\leq K^{|\pi_1|+|\pi_2|+1} \\
&\leq K^{|\varpi|}
\end{aligned}$$

Dans le cas de $\overset{\vdash}{\underset{\Rightarrow}{}}$, on a $|\mathrm{T}_A(\varpi)| = |\mathrm{T}_A(\mathrm{T}_B(\pi))|$. Par hypothèse d'induction, $|\mathrm{T}_B(\pi)| \leq K^{|\pi|}$. De plus, $\mathrm{T}_B(\pi)$ ne contient pas $\overset{\vdash}{\underset{\Rightarrow}{}}$, $\overset{\vee}{\underset{\vdash}{}}$ ou $\overset{\exists}{\underset{\vdash}{}}$, donc on peut utiliser le résultat précédent et avoir $|\mathrm{T}_A(\varpi)| \leq K|\mathrm{T}_B(\pi)| \leq K \times K^{|\pi|} \leq K^{|\pi|+1} \leq K^{|\varpi|}$.

Dans le cas $\overset{\vee}{\vdash}$, on a

$$\begin{aligned}
|\mathrm{T}_A(\varpi)| &= |\mathrm{T}_C(\mathrm{T}_A(\pi_3))| + |\mathrm{T}_B(\mathrm{T}_A(\pi_2))| + |\mathrm{T}_A(\pi_1)| + 8 \\
&\leq K|\mathrm{T}_A(\pi_3)| + K|\mathrm{T}_A(\pi_2)| + |\mathrm{T}_A(\pi_3)| + 8 \\
&\qquad\qquad \text{car } \mathrm{T}_A(\pi_3) \text{ et } \mathrm{T}_A(\pi_2) \text{ ne contiennent pas } \overset{\vdash}{\Rightarrow}, \overset{\vee}{\vdash}, \overset{\exists}{\vdash} \\
&\leq K \times K^{|\pi_3|} + K \times K^{|\pi_2|} + K^{|\pi_1|} + K \\
&\qquad\qquad \text{par hypothèse d'induction et par définition de } K \\
&\leq K^{|\pi_3|+|\pi_2|+|\pi_1|+1} \\
&\leq K^{|\varpi|} \ .
\end{aligned}$$

Le cas $\overset{\exists}{\vdash}$ est similaire, les cas restants sont similaires à celui de $\overset{\Rightarrow}{\vdash}$.

On peut alors montrer par induction sur ϖ, en suivant la définition de T, que $|\mathrm{T}(\varpi)| \leq K^{|\varpi|}$. □

5.3.3 Encodage de l'ordre supérieur en déduction modulo

Passer de schémas d'axiomes à un nombre fini d'axiomes

Czesław Ryll-Nardzewski (1952) a démontré qu'il n'est pas possible d'avoir une présentation finie de l'arithmétique du premier ordre. Néanmoins, il est possible de se placer dans une signature plus étendue pour obtenir une présentation finie d'une extension conservative de cette théorie. Comme l'a démontré Stephen Kleene (1952*a*), il est toujours possible d'étendre la signature d'une théorie présentée sous forme de schémas d'axiomes pour obtenir une extension conservative présentée par un ensemble fini de d'axiomes. Plus récemment, Florent Kirchner (2006) a démontré un résultat similaire et obtient des présentations finies qui sont de plus aisément orientables en des règles de réécriture utilisables en déduction modulo. Sa technique se fonde sur l'utilisation d'un schéma de compréhension qui peut ensuite être décomposé en un nombre d'axiome fini à partir du moment où la signature est finie.

L'idée est d'utiliser le même genre de traduction des formules que dans la section 5.2.2. Néanmoins, on cherche à traduire des formules de la logique prédicative du premier ordre et non plus seulement propositionnelle. Le prédicat ϵ sera donc binaire, et prendra en premier argument une liste de termes à substituer aux variables libres de la formule encodée. Un calcul de substitutions explicites très simple permettra de la décoder. Étant donnée une signature multisortée, on l'étend donc avec deux nouvelles sortes ℓ pour les listes de termes et c pour les classes qui encodent les formules, et on ajoute des nouveaux

symboles de fonction et de prédicat

$$\dot{A} : [\mathrm{s}_1; \ldots; \mathrm{s}_n] \to \mathrm{c} \qquad \text{pour tout symbole de prédicat } A \text{ de sorte } [\mathrm{s}_1; \ldots; \mathrm{s}_n]$$

$$1^{\mathrm{s}} : \mathrm{s} \qquad S^{\mathrm{s}} : [\mathrm{s}] \to \mathrm{s} \qquad \cdot[\cdot]^{\mathrm{s}} : [\mathrm{s}; \ell] \to \mathrm{s} \qquad \mathsf{nil} : \ell$$

$$::^{\mathrm{s}} : [\mathrm{s}; \ell] \to \ell \qquad \cup : [\mathrm{c}; \mathrm{c}] \to \mathrm{c} \qquad \cap : [\mathrm{c}; \mathrm{c}] \to \mathrm{c} \qquad \supset : [\mathrm{c}; \mathrm{c}] \to \mathrm{c}$$

$$\emptyset : \mathrm{c} \qquad \mathcal{P}^{\mathrm{s}} : [\mathrm{c}] \to \mathrm{c} \qquad \mathcal{C}^{\mathrm{s}} : [\mathrm{c}] \to \mathrm{c} \qquad \in : [\ell; \mathrm{c}]$$

pour toute sorte de la signature de départ s. L'idée est alors de remplacer les variables propositionnelles $X(t_1, \ldots, t_n)$ dans les schémas d'axiomes par une proposition atomique $(t_1 ::^{\mathrm{s}_1} \cdots ::^{\mathrm{s}_{n-1}} t_n ::^{\mathrm{s}_n} \mathsf{nil}) \in x^c$ avec s_i la sorte de t_i. Pour simplifier, on notera $\langle t_1, \ldots, t_n \rangle$ la liste $(t_1 ::^{\mathrm{s}_1} \cdots ::^{\mathrm{s}_{n-1}} t_n ::^{\mathrm{s}_n} \mathsf{nil})$ pour les bonnes sortes. En appliquant cette idée sur les schémas d'axiomes de l'arithmétique d'ordre i, c'est-à-dire les schémas d'axiomes (2.24), (5.8) et (5.9), on obtient les nouveaux axiomes suivants

$$\forall z^{\mathrm{c}}\ x^1\ y^1,\ x^1 = y^1 \Rightarrow \langle x^1 \rangle \in z^{\mathrm{c}} \Rightarrow \langle y^1 \rangle \in z^{\mathrm{c}} \tag{5.10}$$

$$\forall z^{\mathrm{c}},\ \langle \mathsf{O} \rangle \in z^{\mathrm{c}} \Rightarrow \big(\forall y^1,\ \langle y^1 \rangle \in z^{\mathrm{c}} \Rightarrow \langle s(y^1) \rangle \in z^{\mathrm{c}}\big) \Rightarrow \forall x^1,\ \langle x^1 \rangle \in z^{\mathrm{c}} \tag{5.11}$$

$$\forall z^{\mathrm{c}},\ \exists x^{\mathrm{j}+1},\ \forall y^{\mathrm{j}},\ y^{\mathrm{j}} \in^{\mathrm{j}} x^{\mathrm{j}+1} \Leftrightarrow \langle y^{\mathrm{j}} \rangle \in z^{\mathrm{c}} \tag{5.12}$$

Il faut également des axiomes pour encoder les formules dans des termes de sorte c. Dans le cas où la signature est finie, ce nombre d'axiomes sera fini.

$$\forall x^{\mathrm{j}},\ \ x^{\mathrm{j}}[\mathsf{nil}]^{\mathrm{j}} = x^{\mathrm{j}} \tag{5.13}$$

$$\forall x^{\mathrm{j}},\ \forall l^{\ell},\ \ 1^{\mathrm{j}}[x^{\mathrm{j}} ::^{\mathrm{j}} l^{\ell}]^{\mathrm{j}} = x^{\mathrm{j}} \tag{5.14}$$

$$\forall x^{\mathrm{j}},\ \forall y^{k},\ \forall l^{\ell},\ \ S^{\mathrm{j}}(x^{\mathrm{j}})[y^{k} ::^{k} l^{\ell}]^{\mathrm{j}} = x^{\mathrm{j}}[l^{\ell}]^{\mathrm{j}} \tag{5.15}$$

$$\forall x_1^{s_1},\ \ldots,\ \forall x_n^{s_n},\ \forall l^{\ell},\ \ f(x_1^{s_1}, \ldots, x_n^{s_n})[l^{\ell}] = f(x_1^{s_1}[l^{\ell}], \ldots, x_n^{s_n}[l^{\ell}]) \tag{5.16}$$

pour tout symbole de fonction f d'arité $[s_1; \ldots; s_n]$

$$\forall x_1^{s_1},\ \ldots,\ \forall x_n^{s_n},\ \forall l^{\ell},\ \ l^{\ell} \in \dot{A}(x_1^{s_1}, \ldots, x_n^{s_n}) \Leftrightarrow A(x_1^{s_1}[l^{\ell}]^{s_1}, \ldots, x_n^{s_n}[l^{\ell}]^{s_n}) \tag{5.17}$$

pour tout symbole de prédicat A de sorte $[s_1; \ldots; s_n]$

$$\forall x^{\mathrm{c}},\ \forall y^{\mathrm{c}},\ \forall l^{\ell},\ \ l^{\ell} \in x^{\mathrm{c}} \cup y^{\mathrm{c}} \Leftrightarrow l^{\ell} \in x^{\mathrm{c}} \vee l^{\ell} \in y^{\mathrm{c}} \tag{5.18}$$

$$\forall x^{\mathrm{c}},\ \forall y^{\mathrm{c}},\ \forall l^{\ell},\ \ l^{\ell} \in x^{\mathrm{c}} \cap y^{\mathrm{c}} \Leftrightarrow l^{\ell} \in x^{\mathrm{c}} \wedge l^{\ell} \in y^{\mathrm{c}} \tag{5.19}$$

$$\forall x^{\mathrm{c}},\ \forall y^{\mathrm{c}},\ \forall l^{\ell},\ \ l^{\ell} \in x^{\mathrm{c}} \supset y^{\mathrm{c}} \Leftrightarrow l^{\ell} \in x^{\mathrm{c}} \Rightarrow l^{\ell} \in y^{\mathrm{c}} \tag{5.20}$$

$$\forall l^{\ell},\ \ l^{\ell} \in \emptyset \Leftrightarrow \bot \tag{5.21}$$

$$\forall x^{\mathrm{c}},\ \forall l^{\ell},\ \ (l^{\ell} \in \mathcal{P}^{\mathrm{j}}(x^{\mathrm{c}}) \Leftrightarrow \exists y^{\mathrm{j}},\ y^{\mathrm{j}} ::^{\mathrm{j}} l^{\ell} \in x^{\mathrm{c}}) \tag{5.22}$$

$$\forall x^{\mathrm{c}},\ \forall l^{\ell},\ \ (l^{\ell} \in \mathcal{C}^{\mathrm{j}}(x^{\mathrm{c}}) \Leftrightarrow \forall y^{\mathrm{j}},\ y^{\mathrm{j}} ::^{\mathrm{j}} l^{\ell} \in x^{\mathrm{c}}) \tag{5.23}$$

Pour l'arithmétique d'ordre i, on a donc trois axiomes pour (5.16) :

$$\forall x^1,\ \forall l^\ell,\ \ s(x^1)[l^\ell]^1 = s(x^1[l^\ell]^1)$$
$$\forall x^1,\ \forall y^1,\ \forall l^\ell,\ \ (x^1 + y^1)[l^\ell]^1 = x^1[l^\ell]^1 + y^1[l^\ell]^1$$
$$\forall x^1,\ \forall y^1,\ \forall l^\ell,\ \ (x^1 \times y^1)[l^\ell]^1 = x^1[l^\ell]^1 \times y^1[l^\ell]^1$$

et i axiomes pour (5.17) :

$$\forall x^1,\ \forall y^1,\ \forall l^\ell,\ \ l^\ell\ \epsilon\ \doteq (x^1, y^1) \Leftrightarrow x^1[l^\ell]^1 = y^1[l^\ell]^1$$
$$\forall x^{\mathrm{j}},\ \forall y^{\mathrm{j}+1},\ \forall l^\ell,\ \ l^\ell\ \epsilon\ \dot\in^{\mathrm{j}}(x^{\mathrm{j}}, y^{\mathrm{j}+1}) \Leftrightarrow x^{\mathrm{j}}[l^\ell]^{\mathrm{j}} \in^{\mathrm{j}} y^{\mathrm{j}+1}[l^\ell]^{\mathrm{j}+1}$$

pour $1 \leq j < i$.

Florent Kirchner (2006, Proposition 4) montre que l'on obtient une extension conservative de la théorie de départ. En particulier, si on note A_i^{ws} la théorie présentée par les axiomes (2.23) et (5.1) à (5.7), plus (5.10) à (5.23), alors A_i^{ws} est une extension conservative de l'arithmétique d'ordre i.

Un système de réécriture pour encoder l'ordre supérieur

Comme on le voit, les axiomes (5.13) à (5.23) s'orientent facilement en des règles de réécritures sur les termes et sur les propositions. Nous cherchons ici à encoder tout ce qui concerne l'ordre supérieur dans des règles de réécriture, pour se ramener à des formules de l'arithmétique du premier ordre. Il nous faut donc essentiellement également transformer les axiomes (5.12) pour $1 \leq j < i$ en règle de réécriture propositionnelle. Pour cela, nous les skolémisons afin d'obtenir une quantification uniquement universelle de l'équivalence. On rajoute donc des symboles de fonction $comp^{\mathrm{j}}$ de sorte $[\mathrm{j}] \to c$ pour $2 \leq j \leq i$ et on obtient les axiomes

$$\forall z^{\mathrm{c}},\ \forall y^{\mathrm{j}},\ y^{\mathrm{j}} \in^{\mathrm{j}} comp^{\mathrm{j}+1}(z^{\mathrm{c}}) \Leftrightarrow \langle y^{\mathrm{j}} \rangle\ \epsilon\ z^{\mathrm{c}} \tag{5.24}$$

On note alors A_i^{sk} la théorie dont une présentation est celle de A_i^{ws} dans laquelle on a remplacé (5.12) par (5.24).

Proposition 5.8. *A_i^{sk} est une extension conservative de l'arithmétique d'ordre i.*

Démonstration. D'après Dirk van Dalen (1989, corollaire 3.4.5), la théorie présentée par $\mathrm{A}_i^{ws} \cup \{(5.24)\}$ est une extension conservative de A_i^{ws}. Or $Th\,(\mathrm{A}_i^{ws} \cup \{(5.24)\}) = Th\,(\mathrm{A}_i^{sk} \cup \{(5.12)\}) = Th\,\mathrm{A}_i^{sk}$ car (5.12) est démontrable à partir de A_i^{sk}. □

Il y a deux façons d'orienter (5.24). Si on oriente dans le sens $\langle y \rangle\ \epsilon\ z \to y \in^{\mathrm{j}} comp^{\mathrm{j}+1}(z)$ alors on va perdre la confluence avec les autres règles issues des axiomes (5.13) à (5.23).

$$
\begin{array}{rlrl}
x[\mathsf{nil}]^{\mathsf{j}} & \to x & l \in \dot{\in}^{\mathsf{j}'}(x_1, x_2) & \to x_1[l]^{\mathsf{j}'} \in^{\mathsf{j}'} x_2[l]^{\mathsf{j}'+1} \\
1^{\mathsf{j}}[x ::^{\mathsf{j}} l]^{\mathsf{j}} & \to x & l \in y \cup z & \to l \in y \vee l \in z \\
S^{\mathsf{j}}(x)[y ::^{\mathsf{k}} l]^{\mathsf{j}} & \to x[l]^{\mathsf{j}} & l \in y \cap z & \to l \in y \wedge l \in z \\
s(x)[l]^1 & \to s(x[l]^1) & l \in y \supset z & \to l \in y \Rightarrow l \in z \\
(x_1 + x_2)[l]^1 & \to x_1[l]^1 + x_2[l]^1 & l \in \emptyset & \to \bot \\
(x_1 \times x_2)[l]^1 & \to x_1[l]^1 \times x_2[l]^1 & l \in \mathcal{P}^{\mathsf{j}}(y) & \to \exists x.\ x ::^{\mathsf{j}} l \in y \\
l \in \doteq(x_1, x_2) & \to x_1[l]^1 = x_2[l]^1 & l \in \mathcal{C}^{\mathsf{j}}(y) & \to \forall x.\ x ::^{\mathsf{j}} l \in y
\end{array}
$$

$$x \in^{\mathsf{j}'} comp^{\mathsf{j}'+1}(y) \to x ::^{\mathsf{j}'} \mathsf{nil} \in y$$

pour $1 \leq j, k \leq i$ et $1 \leq j' < i$.

FIG. 5.3 : Système de réécriture encodant l'ordre supérieur $\mathcal{HO}_i$

On pourrait alors compléter le système obtenu, mais le système final serait inutilement compliqué. On oriente donc dans le sens $y \in^{\mathsf{j}} comp^{\mathsf{j}+1}(z) \to \langle y \rangle \in z$. On considère donc, pour un ordre i donné, le système $\mathcal{HO}_i$ de la figure 5.3.

Ce système a les propriétés suivantes :

- Il est fini, pour un i donné.
- Il est localement confluent : on peut facilement joindre les seules paires critiques qui sont de la forme $f(t_1, \ldots, t_n) \underset{\mathcal{HO}_i}{\longleftarrow} f(t_1, \ldots, t_n)[\mathsf{nil}] \underset{\mathcal{HO}_i}{\longrightarrow} f(t_1[\mathsf{nil}], \ldots, t_n[\mathsf{nil}])$ pour $f \in \{+; \times; s\}$.
- Il est linéaire à gauche, c'est-à-dire que les variables n'apparaissent qu'une seule fois dans les membres gauches.
- Il termine pour $i = 2$, c'est-à-dire si on est intéressé par la réduction de taille entre l'arithmétique d'ordre 2 et d'ordre 1. Plus généralement, si on ne prend en compte la dernière règle que pour $j' = i - 1$, alors on peut aussi montrer la terminaison. Dans ce cas, on considérera la différence entre l'arithmétique d'ordre $i-1$ et l'arithmétique d'ordre i (pour laquelle il existe une réduction de taille arbitraire). Nous conjecturons néanmoins que le système termine y compris quand on considère la dernière règle pour tout $1 \leq j' < i$.

Proposition 5.9. *Le système de réécriture $\mathcal{HO}_2$ termine.*

Démonstration. On peut par exemple le démontrer en utilisant l'outil AProVE. On obtient alors la démonstration donnée dans l'annexe A. □

On peut se demander si, à partir de cette démonstration de terminaison et à l'aide des quasi-interprétations de Guillaume BONFANTE, Jean-Yves MARION et Jean-Yves MOYEN (2005), il ne serait pas possible de borner la complexité de la congruence engendrée par $\mathcal{HO}_2$.

Conjecture 5.1.
Pour tout $i > 2$, le système de réécriture $\mathcal{HO}_i$ termine.

On sait que A_i^{sk} est une extension conservative de l'arithmétique d'ordre i. Comme les axiomes (5.13) à (5.24) sont compatibles avec $\mathcal{HO}_i$, on sait donc qu'une proposition du langage de l'arithmétique d'ordre i est démontrable dans le système schématique pour A_i ssi elle est démontrable sous les hypothèses (2.23) et (5.1) à (5.7) plus (5.10) et (5.11) en déduction naturelle modulo $\mathcal{HO}_i$. Toutefois, nous sommes également intéressé par la taille des démonstrations, donc nous allons montrer comment passer d'une démonstration dans le système schématique pour A_i en une démonstration en déduction naturelle modulo $\mathcal{HO}_i$ sans accroître la taille.

Florent Kirchner (2006, proposition 2) montre qu'il est possible d'encoder toutes les formules du langage de départ, c'est à dire que pour chaque formule P du langage de l'arithmétique d'ordre i et pour tout liste finie de variables $x_1^{\mathrm{s}_1}, \ldots, x_n^{\mathrm{s}_n}$, il est possible de démontrer

$$\exists e^{\mathrm{c}},\ \forall x_1^{\mathrm{s}_1},\ \ldots,\ \forall x_n^{\mathrm{s}_n},\ \langle x_1^{\mathrm{s}_1}, \ldots, x_n^{\mathrm{s}_n} \rangle\ \epsilon\ e^{\mathrm{c}} \Leftrightarrow P\ .$$

De plus, la démonstration de cette proposition est constructive et permet de définir de manière précise quel est le terme qui peut instancier e^{c} pour démontrer la formule ci-dessus. Cela donne la définition inductive suivante :

$$\begin{aligned}
|x^{\mathrm{s}}|^{\emptyset} &\stackrel{\text{déf}}{=} x^{\mathrm{s}} \\
|x_1^{\mathrm{s}_1}|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} 1 \\
|x^{\mathrm{s}}|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} S(|x^{\mathrm{s}}|^{x_2^{\mathrm{s}_2},\ldots,x_n^{\mathrm{s}_n}}) \qquad \text{si } x^{\mathrm{s}} \neq x_1^{\mathrm{s}_1} \\
|\mathsf{O}|^{\emptyset} &\stackrel{\text{déf}}{=} \mathsf{O} \\
|\mathsf{O}|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} S^1(|\mathsf{O}|^{x_2^{\mathrm{s}_2},\ldots,x_n^{\mathrm{s}_n}}) \\
|s(t)|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} s(|t|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}) \\
|t_1 + t_2|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} |t_1|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} + |t_2|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} \\
|t_1 \times t_2|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} |t_1|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} \times |t_2|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} \\
e_{t_1=t_2}^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} \doteq (|t_1|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}, |t_2|^{x_1,\ldots,x_n}) \\
e_{t_1 \in^{\mathrm{j}} t_2}^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} \dot{\in}^{\mathrm{j}}(|t_1|^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}, |t_2|^{x_1,\ldots,x_n}) \\
e_{\bot}^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} \emptyset \\
e_{P \Rightarrow Q}^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} \supset (e_P^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}, e_q^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}) \\
e_{P \wedge Q}^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} \cap (e_P^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}, e_q^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}) \\
e_{P \vee Q}^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} &\stackrel{\text{déf}}{=} \cup (e_P^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}, e_q^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}})
\end{aligned}$$

$$e^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}_{\forall x^{\mathrm{s}},\, P} \stackrel{\text{déf}}{=} \mathcal{C}^{\mathrm{s}}(e^{x^{\mathrm{s}},x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}_{P}) \qquad \text{si } x^{\mathrm{s}} \notin \{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}\}$$
$$e^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}_{\exists x^{\mathrm{s}},\, P} \stackrel{\text{déf}}{=} \mathcal{P}^{\mathrm{s}}(e^{x^{\mathrm{s}},x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}_{P}) \qquad \text{si } x^{\mathrm{s}} \notin \{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}\}$$

Pour les deux dernier cas, on fait une α-conversion si jamais $x^{\mathrm{s}} \in \{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}\}$.

Lemme 5.10. *Pour toute proposition P, toutes variables $x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}$ et tous termes $t_1,\ldots,t_n$ de sorte $\mathrm{s}_1,\ldots,\mathrm{s}_n$,*

$$\langle t_1,\ldots,t_n\rangle \in e_P^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}} \xrightarrow[\mathcal{HO}_i]{+} \{t_1/x_1^{\mathrm{s}_1},\ldots,t_n/x_n^{\mathrm{s}_n}\}P \ .$$

Démonstration. Par induction sur la définition de $e_P^{x_1^{\mathrm{s}_1},\ldots,x_n^{\mathrm{s}_n}}$. □

Exemple 5.1 *:* Pour $P \stackrel{\text{déf}}{=} x^1 = \mathsf{O} \vee \exists y^2,\ x \in^1 y$, on a $e_p^{x^1} = (\doteq (1, S(\mathsf{O}))) \cup \mathcal{P}^2\left(\dot\in^1(S(1),1)\right)$ et $\langle t\rangle \in e_p^{x^1} \xrightarrow[\mathcal{HO}_i]{+} t = \mathsf{O} \vee \exists x^2,\ t \in^1 x^2$.

Soit *f*A la présentation finie constituée des axiomes (2.23) et (5.1) à (5.7) plus (5.10) et (5.11). On notera $f\mathrm{A} \vdash^{\mathsf{N}}_{k\ \mathcal{R}} P$ s'il existe une démonstration de *f*A $\vdash P$ en k étapes en déduction naturelle modulo le système de réécriture $\mathcal{R}$.

Proposition 5.11. *Il est possible de traduire une démonstration de longueur k d'une formule P dans le système schématique pour l'arithmétique d'ordre i en une démonstration de *f*A $\vdash P$ en déduction naturelle modulo $\mathcal{HO}_i$ de longueur linéaire par rapport à k.*

$$\mathrm{A}_i \vdash^{\mathsf{S}}_{k} P \quad \rightsquigarrow \quad f\mathrm{A} \vdash^{\mathsf{N}}_{O(k)\ \mathcal{HO}_i} P$$

Démonstration. On utilise la traduction de la section 5.3.2 pour traduire les règles d'inférences (2.4) à (2.22) en dérivation de la déduction naturelle. Il reste ensuite à traduire les instances des règles du système schématique qui ne sont pas dans *f*A, c'est-à-dire les schémas d'axiomes (2.24), (5.8) et (5.9). On propose les démonstrations suivantes

$$\overset{\forall}{\vdash} \frac{[\forall z^{\mathrm{c}},\ \forall x^1 y^1,\ x^1 = y^1 \Rightarrow \langle x^1\rangle \in z^{\mathrm{c}} \Rightarrow \langle y^1\rangle \in z^{\mathrm{c}}]_{(5.10)}}{\forall x^1 y^1,\ x^1 = y^1 \Rightarrow P(x^1) \Rightarrow P(y^1)}$$

(on instancie z^{c} par $e^{x^1}_{P(x^1)}$ et alors $\langle x^1\rangle \in e^{x^1}_{P(x^1)} \Rightarrow \langle y^1\rangle \in e^{x^1}_{P(x^1)} \xrightarrow{*} P(x^1) \Rightarrow P(y^1)$)

$$\overset{\forall}{\vdash} \frac{[\forall z^{\mathrm{c}}.\langle \mathsf{O}\rangle \in z^{\mathrm{c}} \Rightarrow \left(\forall y^1,\ \langle y^1\rangle \in z^{\mathrm{c}} \Rightarrow \langle s(y^1)\rangle \in z^{\mathrm{c}}\right) \Rightarrow \forall x^1,\ \langle x^1\rangle \in z^{\mathrm{c}}]_{(5.11)}}{P(\mathsf{O}) \Rightarrow \left(\forall y^1,\ P(y^1) \Rightarrow P(s(y^1))\right) \Rightarrow \forall x^1,\ P(x^1)}$$

(on instancie z^{c} par $e^{x^1}_{P(x^1)}$ et pour tout t, $\langle t\rangle \in e^{x}_{P(x)} \xrightarrow{*} P(t)$)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})}) \Rightarrow y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})})}\,(\Rightarrow)_{\text{(ii)}}}{y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})}) \Rightarrow y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})})}\,(\Rightarrow)_{\text{(i)}}}{y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})}) \Leftrightarrow y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})})}\,(\wedge)}{\forall y^{\mathsf{j}},\ y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})}) \Leftrightarrow y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})})}\,(\forall)}{\exists x^{\mathsf{j}+1},\ \forall y^{\mathsf{j}},\ y^{\mathsf{j}} \in^{\mathsf{j}} x^{\mathsf{j}+1} \Leftrightarrow P(y^{\mathsf{j}})}\,(\exists)$$

$$\vdots \qquad [y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})})]_{\text{(ii)}}$$

$$\vdots \qquad [y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})})]_{\text{(i)}}$$

$$y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})}) \longrightarrow \langle y^{\mathsf{j}} \rangle \in e^{x^{\mathsf{j}}}_{P(x^{\mathsf{j}})} \xrightarrow{*} P(y^{\mathsf{j}})$$

On voit que la taille de ces démonstrations ne dépend pas de l'instance du schéma d'axiomes, c'est-à-dire de $P(x)$, ce qui justifie le fait que la traduction est linéaire. □

On peut également formuler le même résultat entièrement en déduction naturelle sans passer par les systèmes schématiques :

Théorème 5.12.
Il est possible de traduire une démonstration de longueur k en déduction naturelle utilisant un nombre fini d'instances de (5.1) *à* (5.9) *plus* (2.23) *et* (2.24) *comme hypothèses en une démonstration de $f\mathrm{A} \vdash P$ en déduction naturelle modulo $\mathcal{HO}_i$ de longueur linéaire par rapport à k.*

$$\mathrm{A}_i \vdash^{\mathsf{N}}_{k} P \quad \rightsquigarrow \quad f\mathrm{A} \vdash^{\mathsf{N}}_{\mathcal{O}(k)\ \mathcal{HO}_i} P$$

Démonstration. Ici, on a juste à traduire les instances des schémas d'axiomes (2.24), (5.8) et (5.9) comme dans la démonstration ci-dessus. □

5.3.4 Présentation purement calculatoire de l'arithmétique d'ordre supérieur

Lisa Allali (2007) différencie trois façons de présenter une théorie : via un ensemble d'axiomes, via un système de réécriture modulo lequel les démonstrations sont effectuées, ou une combinaison des deux. Elle dit alors suivant les cas que la présentation est *axiomatique*, *purement calculatoire* ou *modulo*. Ci-dessus, nous avons d'abord donnée une présentation axiomatique de l'arithmétique d'ordre supérieur, quand nous avons défini son système schématique, puis nous en avons donné une présentation modulo avec $f\mathrm{A}$ modulo $\mathcal{HO}_i$. Gilles Dowek et Benjamin Werner (2005) ont quant à eux présentée l'arithmétique du premier ordre de façon purement calculatoire. Nous nous intéressons donc ici à trouver une présentation purement calculatoire de l'arithmétique d'ordre supérieur, en prenant soin toutefois de ne pas faire exploser la taille des démonstrations.

Notons tout d'abord que Sergei Vorobyov (1988) a démontré qu'il n'est pas possible d'axiomatiser ne serait-ce que l'arithmétique de Presburger sans quantificateur (qui est pourtant décidable) à l'aide d'un système de réécriture convergent. Ce que nous allons démontrer n'est pas en contradiction avec ceci, d'une part parce que le système de réécriture que nous allons utiliser n'est pas terminant, d'autre part parce que la déduction modulo permet d'avoir des étapes de déduction en plus des étapes de réécriture.

Si on part de fA modulo $\mathcal{HO}_i$, il nous reste donc à orienter l'axiome de réflexivité (2.23), les axiomes de Robinson (5.1) à (5.7) (moins (5.3), cf. remarque 5.3) et les deux axiomes restants avec les classes (5.10) et (5.11)

Orienter les axiomes définissant l'addition et la multiplication ne pose de problème. Pour orienter l'axiome correspondant au schéma d'axiome de Leibniz et celui de la réflexivité, on utilise l'axiome suivant qui équivaut à leur conjonction :

$$\forall x^1\ y^1,\ x^1 = y^1 \Leftrightarrow (\forall z^c,\ \langle x^1\rangle\ \epsilon\ z^c \Rightarrow \langle y^1\rangle\ \epsilon\ z^c) \tag{5.25}$$

Pour les axiomes (5.1) et (5.2), au premier ordre, ont été proposées essentiellement deux méthodes. Gilles Dowek et Benjamin Werner (2005) étendent la signature avec un symbole de fonction *pred* et un prédicat unaire *Null* et ajoute les axiomes suivant permettant de les définir :

$$pred(\mathsf{O}) = \mathsf{O} \tag{5.26}$$

$$\forall x^1,\ pred(s(x^1)) = x^1 \tag{5.27}$$

$$Null(\mathsf{O}) \tag{5.28}$$

$$\forall x^1,\ \neg Null(s(x^1)) \tag{5.29}$$

Il est alors possible de démontrer (5.1) et (5.2) en utilisant ces quatre axiomes et ceux pour l'égalité. L'intérêt d'utiliser ces nouveaux symbole et prédicat est que l'égalité est alors uniquement définie par la réflexivité et le schéma de Leibniz, ce qui simplifie la construction d'un pré-modèle comme le font Gilles Dowek et Benjamin Werner (2005). Lisa Allali (2007) a une autre technique, qui permet même de se passer de ces axiomes pour l'égalité. Elle utilise les quatre axiomes :

$$\mathsf{O} = \mathsf{O} \tag{5.30}$$

$$\forall x^1,\ \neg \mathsf{O} = s(x^1) \tag{5.31}$$

$$\forall x^1,\ \neg s(x^1) = \mathsf{O} \tag{5.32}$$

$$\forall x^1\ y^1,\ s(x^1) = s(y^1) \Leftrightarrow x^1 = y^1 \tag{5.33}$$

Elle montre alors que l'arithmétique du premier ordre peut être présentée par ces quatre axiomes, les axiomes (5.4) à (5.7) et le schéma d'induction. Nous ne savons pas dans quelle mesure ce résultat peut-être étendu à l'arithmétique d'ordre supérieur. En effet, le schéma de Leibniz peut alors être instancié par des formules comportant des $\in^j$, et

il nous semble qu'il n'est alors pas possible de s'en passer. Une solution consisterait à garder le schéma d'axiomes de Leibniz et d'y ajouter une orientation des axiomes (5.30), (5.31) et (5.33) (sachant que (5.32) est alors redondant). Il serait alors difficile de trouver une interprétation de $=$ dans un pré-modèle ou dans une algèbre ordonnée de valeurs de vérité. Par la suite, nous utiliserons donc la technique de Gilles Dowek et Benjamin Werner (2005) avec le symbole de fonction *pred* et le prédicat *Null*.

Il reste à trouver un système de réécriture correspondant au schéma d'axiomes d'induction. Gilles Dowek et Benjamin Werner (2005) utilisent la technique suivante : ils enrichissent la signature avec un symbole de prédicat unaire N dont la sémantique correspondra informellement au fait d'être un entier naturel. L'axiome pour l'induction devient alors

$$\forall x^1,\ N(x^1) \Leftrightarrow \big(\forall z^{\mathrm{c}},\ \langle \mathsf{O} \rangle\ \epsilon\ z^{\mathrm{c}} \Rightarrow (\forall y^1,\ N(y^1) \Rightarrow \langle y^1 \rangle\ \epsilon\ z^{\mathrm{c}} \Rightarrow \langle s(y^1) \rangle\ \epsilon\ z^{\mathrm{c}}) \Rightarrow \langle x^1 \rangle\ \epsilon\ z^{\mathrm{c}}\big)$$

Remarque 5.5 *:* Gilles Dowek et Benjamin Werner (2005) ne travaillent en fait pas exactement sur les mêmes classes que Florent Kirchner (2006). À la place, ils utilisent un schéma de compréhension sur la signature originale (ce qui, rappelons le, donne une extension conservative) puis ils en skolémisent chacune des ses instances, ce qui donne pour toute formule P du langage de l'arithmétique du premier ordre et toutes variables $x^1, y_1^1, \ldots, y_n^1$ une constante $f_P^{x^1, y_1^1, \ldots, y_n^1}$ de sorte c telle que

$$\forall x^1, y_1^1, \ldots, y_n^1,\ x^1 \in f_P^{x^1}(y_1^1, \ldots, y_n^1) \Leftrightarrow P$$

et un schéma d'induction

$$\forall x^1,\ N(x^1) \Leftrightarrow \forall z^{\mathrm{c}},\ \mathsf{O} \in z^{\mathrm{c}} \Rightarrow (\forall y^1,\ N(y^1) \Rightarrow y^1 \in z^{\mathrm{c}} \Rightarrow s(y^1) \in z^{\mathrm{c}}) \Rightarrow x^1 \in z^{\mathrm{c}}\ .$$

Le travail de Florent Kirchner (2006) consiste alors à transformer l'infinité d'axiomes $\forall x^1, y_1^1, \ldots, y_n^1,\ x^1 \in f_P^{x^1}(y_1^1, \ldots, y_n^1) \Leftrightarrow P$ en une nombre fini d'axiomes en s'appuyant sur le fait que les formules sont définies inductivement. Dans la suite, nous ferons comme si Gilles Dowek et Benjamin Werner (2005) avait utilisée la solution de Florent Kirchner (2006), ce qui est équivalent.

Gilles Dowek et Benjamin Werner (2005) utilisent donc le système de réécriture $\mathcal{HA}$ suivant :

$$\begin{aligned}
pred(\mathsf{O}) &\to \mathsf{O} \\
pred(s(x)) &\to x \\
\mathsf{O} + y &\to y \\
s(x) + y &\to s(x + y) \\
\mathsf{O} \times y &\to y \\
s(x) \times y &\to x \times y + y
\end{aligned}$$

$$\begin{aligned} Null(\mathsf{O}) &\to \top \\ Null(s(x)) &\to \bot \\ N(x) &\to \forall z^{\mathrm{c}},\ \langle \mathsf{O} \rangle \in z^{\mathrm{c}} \Rightarrow (\forall y^1,\ N(y^1) \Rightarrow \langle y^1 \rangle \in z^{\mathrm{c}} \Rightarrow \langle s(y^1) \rangle \in z^{\mathrm{c}}) \Rightarrow \langle x \rangle \in z^{\mathrm{c}} \end{aligned}$$

On obtient bien ainsi présentation purement calculatoire d'une extension de l'arithmétique du premier ordre. Pour montrer qu'elle est conservative, il faut définir une traduction des formules $|\cdot|$ définie inductivement telle que $|\forall x^1,\ P| \stackrel{\text{déf}}{=} \forall x^1,\ N(x^1) \Rightarrow |P|$ et $|\exists x^1,\ P| \stackrel{\text{déf}}{=} \exists x^1,\ N(x^1) \wedge |P|$. Néanmoins, cette présentation ne préserve pas la taille des démonstrations.

Proposition 5.13. *Il existe une famille $(P_i)_{i\in\mathbb{N}}$ de formules de l'arithmétique du premier ordre telles que :*

- *il existe une borne $k \in \mathbb{N}$ telle que pour tout i on a $\mathrm{A}_1 \vdash^{\mathsf{N}}_{k} P_i$;*
- *pour tout i, on a $\vdash^{\mathsf{N}}_{\mathcal{HA}} |P_i|$;*
- *il n'existe pas de borne k telle que pour tout i on ait $\vdash^{\mathsf{N}}_{k\,\mathcal{HA}} |P_i|$.*

Démonstration. On prend comme formules $(\exists x^1,\ x^1 = \underline{i})_{i\in\mathbb{N}}$. On a alors les démonstrations

$$\dfrac{\dfrac{\dfrac{[\forall x^1,\ \forall y^1,\ x^1 = y^1 \Rightarrow x^1 = x^1 \Rightarrow y^1 = y^1]_{(2.24)}}{\underline{i} = \underline{i} \Rightarrow \underline{i} = \underline{i} \Rightarrow \underline{i} = \underline{i}}\,\forall \qquad \dfrac{[\underline{i} = \underline{i}]_{(\mathrm{i})}}{\underline{i} = \underline{i} \Rightarrow \underline{i} = \underline{i}}\,\Rightarrow}{\underline{i} = \underline{i}}\,\Rightarrow}{\exists x^1,\ x^1 = \underline{i}}\,\exists\ (\mathrm{i})$$

donc $\mathrm{A}_1 \vdash^{\mathsf{N}}_{7} P_i$ pour tout $i \in \mathbb{N}$.

$|P_i| = \exists x^1,\ N(x^1) \wedge x^1 = \underline{i}$. La seule façon de démontrer ceci est d'instancier x^1 par $\underline{i}$. Il faut donc démontrer $N(\underline{i})$ ce qui est possible mais nécessite au moins i étapes. □

Nous allons donc chercher un autre système de réécriture compatible avec le schéma d'induction. Pour cela, on utilise l'axiome (5.11) correspondant à ce schéma et on applique la fonction *Rew* définie dans la section 4.3.1 sur la séquence $\vdash \forall z^{\mathrm{c}},\ \langle \mathsf{O} \rangle \in z^{\mathrm{c}} \Rightarrow \big(\forall y^1,\ \langle y^1 \rangle \in z^{\mathrm{c}} \Rightarrow \langle s(y^1) \rangle \in z^{\mathrm{c}}\big) \Rightarrow \forall x^1,\ \langle x^1 \rangle \in z^{\mathrm{c}}$. On obtient donc la règle de réécriture $\langle x \rangle \in z \to^{+} \langle \mathsf{O} \rangle \in z \wedge \big(\forall y^1,\ \langle y^1 \rangle \in z \Rightarrow \langle s(y^1) \rangle \in z\big)$ ou, si l'on ne veut pas utiliser de règles polarisées, par le théorème 3.10, la règle

$$\langle x \rangle \in z \to \langle x \rangle \in z \vee \big(\langle \mathsf{O} \rangle \in z \wedge \big(\forall y^1,\ \langle y^1 \rangle \in z \Rightarrow \langle s(y^1) \rangle \in z\big)\big)\ .$$

Au final, on considère donc le système de réécriture $\mathcal{HHA}_i^{mod}$ défini dans la figure 5.4.

Cette fois, la taille des démonstrations n'explose pas.

Règles arithmétiques :

$$\begin{array}{rl} pred(0) & \to 0 \\ pred(s(x)) & \to x \\ 0 + y & \to y \\ s(x) + y & \to s(x + y) \end{array} \qquad \begin{array}{rl} 0 \times y & \to y \\ s(x) \times y & \to x \times y + y \\ Null(0) & \to \top \\ Null(s(x)) & \to \bot \end{array}$$

Règles de substitutions :

$$\begin{array}{rl} x[\mathsf{nil}]^{\mathsf{j}} & \to x \\ 1^{\mathsf{j}}[x ::^{\mathsf{j}} l]^{\mathsf{j}} & \to x \\ S^{\mathsf{j}}(x)[y ::^{k} l]^{\mathsf{j}} & \to x[l]^{\mathsf{j}} \\ s(x)[l]^1 & \to s(x[l]^1) \\ (x_1 + x_2)[l]^1 & \to x_1[l]^1 + x_2[l]^1 \\ (x_1 \times x_2)[l]^1 & \to x_1[l]^1 \times x_2[l]^1 \\ pred(n)[l]^1 & \to pred(n[l]^1) \\ \ell \in \dot{Null}(t) & \to Null(t[\ell]^1) \end{array} \qquad \begin{array}{rl} l \in \doteq (x_1, x_2) & \to x_1[l]^1 = x_2[l]^1 \\ l \in \dot{\in}^{\mathsf{j}'}(x_1, x_2) & \to x_1[l]^{\mathsf{j}'} \in^{\mathsf{j}'} x_2[l]^{\mathsf{j}'+1} \\ l \in y \cup z & \to l \in y \vee l \in z \\ l \in y \cap z & \to l \in y \wedge l \in z \\ l \in y \supset z & \to l \in y \Rightarrow l \in z \\ l \in \emptyset & \to \bot \\ l \in \mathcal{P}^{\mathsf{j}}(y) & \to \exists x,\ x ::^{\mathsf{j}} l \in y \\ l \in \mathcal{C}^{\mathsf{j}}(y) & \to \forall x,\ x ::^{\mathsf{j}} l \in y \end{array}$$

Règles pour les schémas d'axiomes :

$$\begin{array}{rl} x = y & \to \forall z^{\mathsf{c}},\ \langle x \rangle \in z \Rightarrow \langle y \rangle \in z \\ x ::^{0} \mathsf{nil} \in p & \to \langle x \rangle \in p \vee (\langle 0 \rangle \in p \wedge \forall y,\ \langle y \rangle \in p \Rightarrow \langle s(y) \rangle \in p) \end{array} \qquad \begin{array}{rl} x \in^{\mathsf{j}} comp^{\mathsf{j}+1}(y) & \to x ::^{\mathsf{j}} \mathsf{nil} \in y \end{array}$$

FIG. 5.4 : Présentation purement calculatoire de l'arithmétique de HEYTING d'ordre i $\mathcal{HHA}_i^{mod}$

Théorème 5.14.
Il est possible de traduire une démonstration de longueur k d'une formule P en déduction naturelle utilisant un nombre fini d'instances de (5.1) *à* (5.9) *plus* (2.23) *et* (2.24) *comme hypothèses en une démonstration de $\vdash P$ en déduction naturelle modulo $\mathcal{HHA}_i^{mod}$ de longueur linéaire par rapport à k.*

$$\mathrm{A}_i \vdash^{\mathsf{N}}_{k} P \quad \rightsquigarrow \quad \vdash^{\mathsf{N}}_{\mathcal{O}(k)\,\mathcal{HHA}_i^{mod}} P$$

Démonstration. Il suffit de montrer que toutes les instances de (5.1) à (5.9) plus (2.24) possèdent des démonstrations de taille bornée.

Les instances de (5.1) à (5.7) sont assez facilement démontrables en utilisant les règles arithmétiques et la règle pour $=$. Par exemple, (5.1) possède la démonstration

$$\dfrac{\dfrac{\dfrac{\dfrac{[\mathsf{O} = s(x^1)]_{(\mathrm{i})}}{Null(\mathsf{O}) \Rightarrow Null(s(x^1))}\,a \qquad \vdash_{\top} \dfrac{}{Null(\mathsf{O})}}{Null(s(x^1))}\; Null(\mathsf{O}) \longrightarrow \top}{\vdash_{\neg} \dfrac{\bot}{\neg \mathsf{O} = s(x^1)}\,(\mathrm{i})}\; Null(s(x^1)) \longrightarrow \bot}{\vdash_{\forall} \dfrac{}{\forall x^1,\ \neg \mathsf{O} = s(x^1)}}$$

[a] $\mathsf{O} = s(x^1) \longrightarrow \forall z^c,\ \langle \mathsf{O} \rangle \in z^c \Rightarrow \langle s(x^1) \rangle \in z^c$
$t \in \dot{Null}(1^1) \xrightarrow{*} Null(t)$

(2.23) possède la démonstration

$$\dfrac{\dfrac{\dfrac{[\langle x^1 \rangle \in z^c]_{(\mathrm{i})}}{\langle x^1 \rangle \in z^c \Rightarrow \langle x^1 \rangle \in z^c}\,(\mathrm{i})}{\vdash_{\forall} \dfrac{x^1 = x^1}{\forall x^1,\ x^1 = x^1}}\; x^1 = x^1 \longrightarrow \forall z^c,\ \langle x^1 \rangle \in z^c \Rightarrow \langle x^1 \rangle \in z^c$$

Les instances de (2.24) possèdent la démonstration

$$\dfrac{\dfrac{\dfrac{[x^1 = y^1]_{(\mathrm{i})}}{\langle x^1 \rangle \in e^x_{P(x)} \Rightarrow \langle y^1 \rangle \in e^x_{P(x)}}}{x^1 = y^1 \Rightarrow P(x^1) \Rightarrow P(y^1)}\,(\mathrm{i})}{\forall x^1,\ \forall y^1,\ x^1 = y^1 \Rightarrow P(x^1) \Rightarrow P(y^1)}$$

Les instances de (5.8) possèdent la démonstration

$$\dfrac{\dfrac{\dfrac{\dfrac{[P(\mathsf{O})]_{(\mathrm{i})} \qquad [\forall y^1,\ P(y^1) \Rightarrow P(s(y^1))]_{(\mathrm{ii})}}{\langle \mathsf{O} \rangle \in e_P^x \wedge \forall y^1,\ \langle y^1 \rangle \in e_P^x \Rightarrow \langle s(y^1) \rangle \in e_P^x}\wedge}{\langle x^1 \rangle \in e_P^x}\vee_2}{\forall x^1,\ P(x^1)}\forall}{P(\mathsf{O}) \Rightarrow (\forall y^1,\ P(y^1) \Rightarrow P(s(y^1))) \Rightarrow \forall x^1,\ P(x^1)}\Rightarrow \text{(i),(ii)}$$

Les instances de (5.9) possèdent la démonstration

$$\dfrac{\dfrac{\dfrac{\dfrac{[y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e_P^x)]_{(\mathrm{i})}}{y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e_P^x) \Rightarrow \langle y^{\mathsf{j}} \rangle \in e_P^x}\Rightarrow \text{(i)} \qquad \dfrac{[\langle y^{\mathsf{j}} \rangle \in e_P^x]_{(\mathrm{ii})}}{y^{\mathsf{j}} \in e_P^x \Rightarrow \langle y^{\mathsf{j}} \rangle \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e_P^x)}\Rightarrow \text{(ii)}}{y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e_P^x) \Leftrightarrow \langle y^{\mathsf{j}} \rangle \in e_P^x}\wedge}{\forall y^{\mathsf{j}},\ y^{\mathsf{j}} \in^{\mathsf{j}} comp^{\mathsf{j}+1}(e_P^x) \Leftrightarrow P(y^{\mathsf{j}})}\forall}{\exists x^{\mathsf{j}+1},\ \forall y^{\mathsf{j}},\ y^{\mathsf{j}} \in^{\mathsf{j}} x^{\mathsf{j}+1} \Leftrightarrow P(y^{\mathsf{j}})}\exists$$

□

On peut également vérifier qu'on a bien une extension conservative :

Théorème 5.15.
Il existe une démonstration d'une formule arithmétique P en arithmétique d'ordre i ssi il existe une démonstration de P en déduction naturelle modulo $\mathcal{HHA}_i^{mod}$.

Démonstration. Tout d'abord, on peut montrer, comme le font Gilles Dowek et Benjamin Werner (2005), que rajouter *pred* et les axiomes (5.26) et (5.27) fournit une extension conservative. Il s'agit essentiellement de skolémiser la formule $\forall x^1,\ \exists y^1,\ (x^1 = \mathsf{O} \Rightarrow y^1 = \mathsf{O}) \wedge (\forall z^1,\ x^1 = s(z^1) \Rightarrow y^1 = z^1)$ qui est démontrable en arithmétique d'ordre 1.

On montre ensuite que si on ajoute *Null* et les axiomes (5.28) et (5.29) et si on retire les axiomes (5.1) à (5.3), on a également une extension conservative. Pour cela, comme dans le démonstration de la proposition 10 de Gilles Dowek et Benjamin Werner (2005), il suffit de montrer comment transformer un modèle de la théorie précédente en un modèle de cette théorie : on interprète *Null* par l'ensemble $\{e :\ (e, \hat{\mathsf{O}}) \in \hat{=}\}$.

On applique ensuite la technique de Florent Kirchner (2006) et on obtient ainsi une extension conservative. Puis on skolémise les axiomes de compréhension (5.12) et on produit à nouveau une extension conservative. Il suffit ensuite de montrer que (2.23) et (5.10) sont équivalents à (5.25), et que (5.11) est équivalent à

$$\langle x \rangle \in z \Leftrightarrow \big(\langle x \rangle \in z \vee \big(\langle \mathsf{O} \rangle \in z \wedge \big(\forall y^1,\ \langle y^1 \rangle \in z \Rightarrow \langle s(y^1) \rangle \in z\big)\big)\big) \quad .$$

Il est alors trivial de montrer que la présentation obtenue, qui est donc une extension conservative de l'arithmétique d'ordre i, est compatible avec $\mathcal{HHA}_i^{mod}$. □

Le principal intérêt du système $\mathcal{HA}$ introduit par Gilles Dowek et Benjamin Werner (2005) est le fait que la déduction naturelle modulo $\mathcal{HA}$ normalise. La démonstration de cette normalisation forte à l'aide de pré-modèle ne semble pas s'étendre facilement à $\mathcal{HHA}_i^{mod}$, car il semble difficile de définir un candidat de réductibilité pour ϵ. On pourrait utiliser la procédure de complétion présentée au chapitre 4 pour être sûr d'avoir un système admettant les coupures, toutefois ceci ne serait possible que dans le cas classique. Pour le moment, nous ne savons pas si les systèmes de déduction modulo $\mathcal{HHA}^{mod}$ admettent les coupures ou normalisent fortement.

5.3.5 Application à la réduction de taille des démonstrations

Les résultats de la section précédente impliquent qu'on peut obtenir des réductions de la taille des démonstrations en utilisant la déduction modulo. Comme les démonstrations en arithmétique d'ordre i, celles dans *f*A modulo $\mathcal{HO}_i$ et celles modulo $\mathcal{HHA}_i^{mod}$ sont de la même taille, et que l'arithmétique d'ordre i réduit arbitrairement la taille des démonstrations en arithmétique d'ordre $i-1$, travailler dans *f*A modulo $\mathcal{HO}_i$ ou modulo $\mathcal{HHA}_i^{mod}$ réduit arbitrairement la taille des démonstrations en arithmétique d'ordre $i-1$. On peut décomposer cette réduction de la manière qui suit.

Réduction de taille dans les présentations compatibles

Tout d'abord, comme dans la section 5.2.1, on peut montrer que la déduction naturelle dans *f*A modulo $\mathcal{HO}_i$ ou modulo $\mathcal{HHA}_i^{mod}$ réduit la taille des démonstrations en déduction naturelle dans une présentation compatible finie.

Proposition 5.16. *Pour tout $i \geq 2$ il existe une famille $(P_j)_{j\in\mathbb{N}}$ de formules telles que pour toute présentation finie Γ_i compatible avec $\mathcal{HO}_i$,*

- *il existe une borne $k \in \mathbb{N}$ telle que pour tout j on a $f\mathrm{A} \vdash^{\mathsf{N}}_{k\,\mathcal{HO}_i} P_j$;*
- *pour tout j, on a $f\mathrm{A}, \Gamma_i \vdash^{\mathsf{N}} P_j$;*
- *il n'existe pas de borne k telle que pour tout j on ait $f\mathrm{A}, \Gamma_i \vdash^{\mathsf{N}}_k P_j$.*

Démonstration. On prend comme formules toutes les instances du schéma de compréhension (5.9) pour $j = i-1$. Comme l'arithmétique d'ordre i est plus puissante que *f*A, nous sommes obligés d'utiliser les axiomes de Γ_i ou les règles de $\mathcal{HO}_i$ pour démontrer ces formules. On sera donc amené à instancier une variable de sorte c pour un des axiomes (5.10) ou (5.11), ou encore d'utiliser la règle pour $comp^{\mathsf{j}}$ ou un axiome correspondant. Le terme instancié ne pourra être de taille bornée, et il faudra donc un nombre d'étapes non borné pour le décoder dans la présentation compatible, alors que ça sera immédiat en déduction modulo. □

Réduction de taille par une présentation compatible

Il existe également une réduction de taille par rapport à l'arithmétique d'ordre $i-1$ si on se place dans $f\mathrm{A}$ plus une présentation compatible avec $\mathcal{HO}_i$.

Proposition 5.17. *Pour tout $i \geq 2$, il existe une famille $(P_j)_{j\in\mathbb{N}}$ de formules telles que pour toute présentation finie Γ_i compatible avec $\mathcal{HO}_i$*
- *il existe une borne $k \in \mathbb{N}$ telle que pour tout j on a $f\mathrm{A}, \Gamma_i \vdash^{\mathsf{N}}_k P_j$;*
- *pour tout j, on a $\mathrm{A}_{i-1} \vdash^{\mathsf{N}} P_j$;*
- *il n'existe pas de borne k telle que pour tout j on ait $\mathrm{A}_{i-1} \vdash^{\mathsf{N}}_k P_j$.*

Démonstration. Pour démontrer ceci il faut se ramener à la démonstration du théorème 5.2 par Samuel Buss (1994). Grâce à un argument diagonal, il obtient une formule $P(n)$ telle que
- à l'ordre $i-1$, pour tout $n \in \mathbb{N}$, on peut démontrer $P(\underline{n})$ mais avec une démonstration de taille forcément plus grande que n ;
- à l'ordre i on peut démontrer $\forall x^1,\ P(x^1)$ et en instanciant ceci on obtient des démonstrations de $P(\underline{n})$ de la même taille pour tout $n \in \mathbb{N}$.

Pour pouvoir démontrer la proposition, il suffit donc de montrer que $\forall x^1,\ P(x^1)$ est démontrable en déduction naturelle sous les hypothèses $f\mathrm{A}, \Gamma_i$ pour pouvoir utiliser la même argumentation. C'est le cas, grâce à la proposition 5.11 et le fait que Γ_i est compatible avec $\mathcal{HO}_i$. □

Remarque 5.6 *:* On peut démontrer le même genre de propositions que les deux précédente avec $\mathcal{HHA}_i^{mod}$ au lieu de $f\mathrm{A}$ modulo $\mathcal{HO}_i$. Néanmoins, la congruence engendrée par $\mathcal{HHA}_i^{mod}$ est moins simple à vérifier que celle pour $\mathcal{HO}_i$, ce qui rend les résultats pour cette dernière plus intéressants d'un point de vue complexité.

On peut alors résumer les réductions de taille des démonstration en arithmétique d'ordre supérieur grâce au diagramme de la figure 5.5.

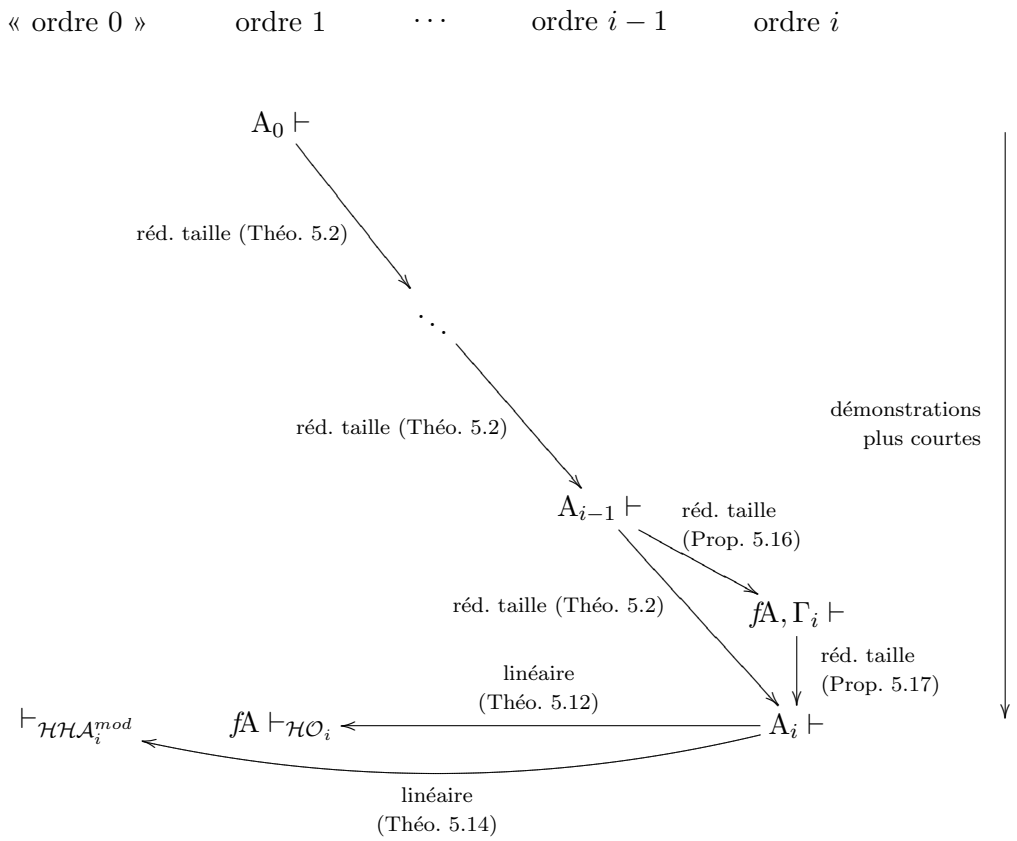
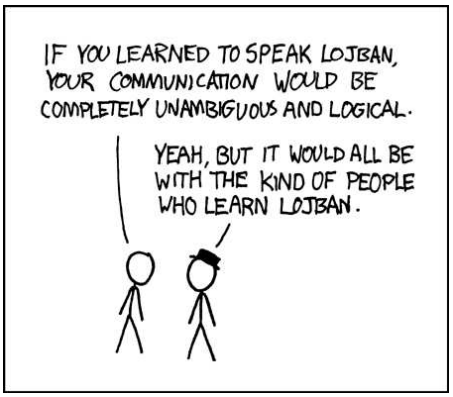


FIG. 5.5 : Réduction de taille en arithmétique d'ordre supérieur et en déduction modulo

Chapitre 6

La surdéduction comme cadre logique



Randall Munroe, http://xkcd.com/

Nous avons vu que la déduction modulo, ou de façon similaire la surdéduction, permet d'encoder des systèmes d'ordre supérieur comme la logique d'ordre supérieur basée sur la théorie simple des types de Church, ou encore l'arithmétique d'ordre supérieur. Si on regarde le calcul des séquences extensible correspondant à la logique d'ordre supérieur de Church, on voit que les nouvelles règles d'inférence correspondent exactement à celle du système HOL-λ pour cette logique. Ceci fait penser à un lemme d'adéquation dans un cadre logique. Un cadre logique peut-être défini comme un langage permettant de spécifier des systèmes de déduction. L'adéquation avec un système de déduction correspond au fait que l'encodage de ce système dans le cadre logique se comporte de la même façon que le système lui-même. Nous allons donc voir dans quelle mesure la surdéduction peut être considérée comme un cadre logique, notamment en montrant comment tous les systèmes de type purs fonctionnels peuvent être encodés dedans.

6.1 Méthodologie

6.1.1 Cadres logiques

Tel que défini par Frank Pfenning (1996), et comme indiqué ci-dessus, un *cadre logique* est un langage pour spécifier des systèmes de déduction. En général, on encodera avant tout des systèmes de déduction représentés sous forme de système d'inférence. Le plus connu des cadres logiques est le système LF introduit par Robert Harper et collaborateurs (1993). Il est basé sur le système de type pur $\lambda\Pi$ (cf. section 3.1.3). Il fait suite à une lignée d'outils de démonstration interactive ayant pour origine le système Automath de Nicolaas de Bruijn (1970). D'autres systèmes, parmi lesquels λProlog (Dale Miller, 1991) et Isabelle (Lawrence Paulson, 1990), sont plutôt fondés sur une sous-classe des logiques du premier ordre et d'ordre supérieur, les formules héréditaires de Harrop, pour lesquelles on connaît des méthodes de recherche de démonstration complètes et efficaces, et sur l'unification d'ordre supérieur.

Pour encoder un système de déduction représenté sous forme de règles d'inférence, il faut d'abord encoder la syntaxe des formules et des jugements de ce système dans la syntaxe des expressions du cadre logique. Il faut ensuite définir une théorie dans le cadre logique qui permet de simuler les règles d'inférence du système de déduction. Reste à démontrer que l'encodage est adéquat, c'est-à-dire que les démonstrations obtenues dans le cadre logique correspondent plus ou moins aux démonstrations dans le système d'inférence de départ.

Exemple 6.1 *:* Nous allons rappeler comment la déduction naturelle pour la logique du premier ordre est encodée dans LF, comme indiqué par Robert Harper et collaborateurs (1993).

On traduit tout d'abord la syntaxe des formules de la logique du premier ordre. On introduit deux variable $\iota : *$ et $o : *$ qui seront les types des termes et des propositions. Pour chaque symbole de fonction de la signature on ajoute une variable de type adéquat, par exemple $\dot{+} : \iota \Rightarrow \iota \Rightarrow \iota$. On fait de même pour les symboles de prédicat, par exemple $\dot{=}: \iota \Rightarrow \iota \Rightarrow o$. On introduit également des variables pour les connecteurs, par exemple $\dot{\Rightarrow} : o \Rightarrow o \Rightarrow o$ et $\dot{\forall} : (\iota \Rightarrow o) \Rightarrow o$. On voit ici comment la λ-abstraction dans LF est utilisée pour encoder les lieurs de la logique du premier ordre. Par exemple, $\forall x,\ A(x)$ devient $\dot{\forall}\ (\lambda x,\ \dot{A}\ x)$. Pour traduire les jugements, on utilise une nouvelle variable $vrai : o \Rightarrow *$. Une démonstration d'une formule P sera alors un terme de type $vrai\ |P|$ où $|P|$ est la traduction des formules que nous venons d'esquisser.

Il reste maintenant à encoder les règles d'inférence elles-mêmes. Nous ne les traitons pas toutes, on se contentera des exemples de l'implication et de la quantification universelle. On ajoute de nouvelles variables correspondant à chacune des règles d'inférence :

$$\dot{\vdash}_{\Rightarrow} : \Pi p : o,\ \Pi q : o,\ (vrai\ p \Rightarrow vrai\ q) \Rightarrow vrai\ (\dot{\Rightarrow}\ p\ q)$$
$$\dot{\Rightarrow}_{\vdash} : \Pi p : o,\ \Pi q : o,\ vrai\ (\dot{\Rightarrow}\ p\ q) \Rightarrow vrai\ p \Rightarrow vrai\ q$$

$$\dot{\forall}^{+} : \Pi p : \iota \Rightarrow o,\ (\Pi x : \iota,\ vrai\ (p\ x)) \Rightarrow vrai\ (\dot{\forall}\ (\lambda x,\ p\ x))$$
$$\dot{\forall}^{-} : \Pi p : \iota \Rightarrow o,\ \Pi x : \iota,\ vrai\ (\dot{\forall}\ (\lambda x,\ p\ x)) \Rightarrow vrai\ (p\ x)$$

On travaille alors dans le contexte formé de toutes ces variables.

Robert Harper et collaborateurs (1993) démontrent alors l'adéquation pour la syntaxe et les règles d'inférence : ils montrent que les termes de la logique du premier ordre sont en bijection avec les formes normales des termes de LF de type ι, de même pour les formules avec les forme normales de termes de LF de type o et pour les démonstrations d'une formule P avec les formes normales de termes de type $vrai\ |P|$.

6.1.2 Encodages en surdéduction

Nous allons définir une méthodologie, assez générale et donc assez floue, pour encoder un système d'inférence en surdéduction. Premièrement, il faut traduire la syntaxe des formules et des jugements. Il s'agit donc de l'exprimer à l'aide de termes du premier ordre. Pour ensuite indiquer que ces termes correspondent à des formules, on utilisera un prédicat ϵ tel que $\epsilon(|P|)$, où $|P|$ est la traduction de la formule P en tant que terme du premier ordre, aura informellement la même sémantique que P. Cette sémantique sera définie à l'aide de règle de réécriture que seront définies de telle façon que les surrègles associées correspondent auz règles d'inférence du système de déduction que l'on cherche à spécifier. Les deux principales difficultés apparaissent si jamais les formules sont définies modulo une théorie, ou quand il y a des lieurs (ce qui implique à la fois des formules définies modulo α et la possibilité d'effectuer des substitutions).

Formules modulo une théorie

Si les formules sont définies modulo une théorie, on peut distinguer trois marches à suivre. La première consiste à travailler modulo la théorie par le biais de la déduction modulo. Pour cela, il faut que la théorie soit définissable par le biais d'un système de réécriture, ce qui n'est au premier abord par exemple pas le cas si on considère des formules définies modulo β-conversion. On verra dans la section suivante que dans ce cas on peut utiliser les substitutions explicites. La deuxième méthode consiste à isoler des formes canoniques pour chaque classe de formules, et de n'utiliser que ces formes normales. Par exemple, si on considère des formules définies modulo η-réduction dans un cadre typé, on peut se contenter des formules en forme η-normales. La dernière approche, un peu similaire, consiste à utiliser une syntaxe alternative. Par exemple, dans le cas où on a des formules définies modulo α-conversion, on peut utiliser les indices de Nicolaas de Bruijn (1972). Il s'agit de représenter une variable par un encodage $\underline{i}$ du nombre i de λ qu'il faut remonter pour trouver l'endroit où la variable est abstraite, avec une convention pour les variables libres. Par exemple, $\lambda x,\ x\ (\lambda y,\ z\ y\ x)$ est représenté par $\lambda\underline{1}\ (\lambda\underline{5}\ \underline{1}\ \underline{2})$.

L'inconvénient de cette dernière technique est que les variables libres sont alors traduites par des indices sans véritable signification, sachant qu'il faut choisir un ordre arbitraire sur celles-ci pour déterminer à quels indices elles correspondent. Pour avoir des termes un peu plus significatifs, il faut séparer les variables libres et les variables liées dans les termes. On parle alors de *variables localement sans noms*. Ce mécanisme a été étudié entre autres par Brian Aydemir, Arthur Charguéraud, Benjamin Pierce, Randy Pollack et Stephanie Weirich (2008). On peut soit utiliser deux sortes différentes pour distinguer les variables, ou alors employer la technique dite de la *précuisson*, telle qu'on peut la retrouver chez Gilles Dowek et collaborateurs (2001). Nous allons l'illustrer sur le cas du λ-calcul que l'on veut considérer modulo α-conversion. Il s'agit de traduire les variables liées dans un terme par des indices de de Bruijn mais de laisser les variables liées telles quelles. Pour les protéger à l'intérieur des termes, on leur ajoute alors une opération shift (comme on la retrouve chez Nicolaas de Bruijn (1972) sous le nom de τ_1, et dans les calculs à substitutions explicites). On considère alors une traduction qui prend en argument le contexte de variables liées dans lequel on évolue :

$$
\begin{array}{rcll}
|x|_{\ell_1::x::\ell_2} & \stackrel{\text{déf}}{=} & \underline{\overline{\ell_1} + 1} & \text{si } x \notin \ell_1 \\
|x|_{\ell} & \stackrel{\text{déf}}{=} & x\,[\mathsf{shift}]^{\overline{\ell}} & \text{si } x \notin \ell \\
|\lambda x,\ t|_{\ell} & \stackrel{\text{déf}}{=} & \lambda\,|t|_{x::\ell} & \\
|t\ u|_{\ell} & \stackrel{\text{déf}}{=} & |t|_{\ell}\,|u|_{\ell} &
\end{array}
$$

où $\underline{i}$ désigne l'indice de de Bruijn numéro i et $\overline{\ell}$ la longueur de la liste de variables ℓ. Au départ, le contexte est vide. Par exemple, $\lambda x,\ x\ (\lambda y,\ z\ y\ x)$ est traduite par $\lambda\underline{1}\ (\lambda z\,[\mathsf{shift}]^2\ \underline{1}\ \underline{2})$. On remarque que les variables libres sont bien traduites par des noms et les variables liées par des indices.

Encodage des lieurs

Dans le cas où on a des lieurs, on peut considérer deux approches, les deux faisant au final appel à des calculs de substitutions explicites. La première repose sur la définition par Gilles Dowek, Thérèse Hardin et Claude Kirchner (2002) d'une logique des lieurs dans laquelle les symboles de fonction et les prédicats peuvent lier des variables, ce qui permet d'exprimer simplement la plupart des lieurs existants, comme la λ-abstraction ou la dérivée partielle ($\frac{\partial}{\partial x}$). Gilles Dowek et collaborateurs (2002) montrent alors que cette logique peut être traduite en logique prédicative du premier ordre modulo un calcul de substitutions explicites.

La deuxième approche s'appuie à la fois sur les syntaxes abstraites d'ordre supérieur et sur les substitutions explicites. Comme indiqué dans la section 2.2.2, Frank Pfenning et Conal Elliot (1988) préconisent, par le biais de ce qu'ils appellent les syntaxes

abstraites d'ordre supérieur, l'utilisation de la λ-abstraction pour simuler toute forme de lieur. Nous en avons vu deux exemples avec l'encodage des quantificateurs en logique d'ordre supérieur basée sur la théorie simple des types et celui de la logique du premier ordre dans LF. Malgré tout, le λ-calcul, ou plus exactement la β-réduction, ne peut pas s'exprimer en logique du premier ordre. C'est la raison pour laquelle on va utiliser un λ-calcul avec substitutions explicites. Il existe plusieurs versions de tels calculs. Pour rester le plus général possible, nous allons juste spécifier quelques bonnes propriétés que doit respecter le calcul. Ces propriétés ont été définies par Delia Kesner (2000), qui les utilisaient initialement pour avoir une démonstration générique de la confluence des λ-calculs avec substitutions explicites.

Définition 6.1 (Calcul de substitution)**.** *Étant donnée une signature de la logique de premier ordre multisortée avec deux sortes* t *et* s *et contenant au moins une constante* $\mathsf{shift} : \mathrm{s}$ *et trois symboles de fonction* $\lambda : [\mathrm{t}] \to \mathrm{t}$, $@ : [\mathrm{t}, \mathrm{t}] \to \mathrm{t}$ *et subst* $: [\mathrm{t}, \mathrm{s}] \to \mathrm{t}$*, en notant* $s\ t$ *pour* $@(s, t)$ *et* $t\,[s]$ *pour* $subst(t, s)$*, un* calcul de substitution *est la définition de deux fonctions lift* $: [\mathrm{s}] \to \mathrm{s}$ *et cons* $: [\mathrm{t}] \to \mathrm{s}$ *ainsi que la donnée d'un système de réécriture sur les termes qui contient au moins les deux règles :*

$$(ab)\,[s] \to a\,[s]\ \ b\,[s] \qquad \text{(App)}$$
$$(\lambda a)\,[s] \to \lambda\,(a\,[\mathit{lift}(s)]) \quad . \qquad \text{(Lambda)}$$

Un calcul de substitution permet de propager les subsitutions dans les termes, mais il ne simule pas la β-réduction proprement dite. Pour cela, on rajoute une règle qui déclenche la β-réduction :

$$(\lambda a)b \to a\,[\mathit{cons}(b)] \qquad \text{(Beta)}$$

mais qui ne fait pas partie du calcul de substitutions. Étant donné un calcul de substitution W on notera λ_W le système constitué du système de réécriture du calcul de substitution et de (Beta). Par abus de langage on désignera le système de réécriture d'un calcul de substitution W par W et on notera $W(a)$ la formale normale de a pour ce système. On note $\mathit{lift}_n(s)$ pour $\underbrace{\mathit{lift}(\ldots(\mathit{lift}(}_{n \text{ fois}}\, s))\ldots)$, et $t[s]^n$ pour $t\,\underbrace{[s]\ldots[s]}_{n \text{ fois}}$.

Exemple 6.2 *:* L'un des premiers λ-calculs avec substitutions explicites est λ_σ de Martin Abadi, Lucas Cardelli, Pierre-Louis Curien et Jean-Jacques Lévy (1991). La signature comporte en plus des symboles de fonction décrits ci-dessus les symboles

$$\mathit{id} : \mathrm{s} \qquad \cdot : [\mathrm{t}, \mathrm{s}] \to \mathrm{s} \qquad \mathsf{1} : \mathrm{t} \qquad \circ : [\mathrm{s}, \mathrm{s}] \to \mathrm{s}\ .$$

$\mathit{cons}(t)$ est défini par $t \cdot \mathit{id}$, $\mathit{lift}(s)$ par $\mathsf{1} \cdot (s \circ \mathsf{shift})$. Le système de réécriture σ contient

en plus de (App) et (Lambda) les règles

$$\begin{aligned}
\mathsf{1}[a \cdot s] &\to a \\
a[id] &\to a \\
a[s][t] &\to a[s \circ t] \\
id \circ s &\to s \\
\mathsf{shift} \circ (a \cdot s) &\to s \\
(s_1 \circ s_2) \circ s_3 &\to s_1 \circ (s_2 \circ s_3) \\
(a \cdot s) \circ t &\to a[t] \cdot (s \circ t) \\
s \circ id &\to s \\
\mathsf{1} \cdot \mathsf{shift} &\to id \\
\mathsf{1}[s] \cdot (\mathsf{shift} \circ s) &\to s
\end{aligned}$$

Delia Kesner (2000) définit ce qu'elle appelle un schéma, qui consiste en un ensemble de bonnes propriétés que doit vérifier un calcul de substitutions.

Définition 6.2 (Schéma)**.** *Un calcul de substitution vérifie le* schéma *si :*

1. *W est confluent ;*
2. *W est fortement normalisant ;*
3. *les formes normales de W ne contiennent pas de substitutions ;*
4. *$W(a\ b) = W(a)W(b)$ et $W(\lambda a) = \lambda W(a)$;*
5. *pour toute substitution $s : \mathrm{s}$ dans W, $\underline{1}\,[\mathit{lift}(s)] \overset{*}{\underset{W}{\longleftrightarrow}} \underline{1}$*
6. *pour toute substitution $s : \mathrm{s}$ dans W et pour $m \geq 1$, $\underline{m+1}\,[\mathit{lift}(s)] \overset{*}{\underset{W}{\longleftrightarrow}} \underline{m}\,[s]\,[\mathsf{shift}]$*
7. *pour tout terme $k : \mathrm{t}$ et pour $m \geq 1$, $\underline{m+1}\,[\mathit{cons}(k)] \overset{*}{\underset{W}{\longleftrightarrow}} \underline{m}$*
8. *pour tout terme $k : \mathrm{t}$, $\underline{1}\,[\mathit{cons}(k)] \overset{*}{\underset{W}{\longleftrightarrow}} k$*
9. *pour tout m, $\underline{m}\,[\mathsf{shift}] \overset{*}{\underset{W}{\longleftrightarrow}} \underline{m+1}$*
10. *pour toute variable x, toute substitution $s : \mathrm{s}$, et $n \geq 0$,*
    $x\,[\mathsf{shift}]^{n+1}\,[\mathit{lift}(s)] \overset{*}{\underset{W}{\longleftrightarrow}} x\,[\mathsf{shift}]^{n}\,[s]\,[\mathsf{shift}]$
11. *pour toute variable x, terme $k : \mathrm{t}$, et $n \geq 0$, $x\,[\mathsf{shift}]^{n+1}\,[\mathit{cons}(k)] \overset{*}{\underset{W}{\longleftrightarrow}} x\,[\mathsf{shift}]^{n}$*
12. *pour tout symbole de fonction $\xi : [\mathrm{k}_1; \cdots; \mathrm{k}_q] \to \mathrm{s}$ et pour $m \geq 1$, une des deux conditions suivantes est vérifiée*
    - *il existe un entier n, des indices distincts $i_1, \ldots, i_p$ dans $\{1; \cdots; q\}$ et des substitutions $u_1, \ldots, u_k$ tels que pour tout $s_1, \ldots, s_q$ on a*
      $\underline{m}\,[\xi(q_1, \ldots, q_n s)] \overset{*}{\underset{W}{\longleftrightarrow}} \underline{n}\,[s_{i_1}] \ldots \left[s_{i_p}\right]\,[u_1] \ldots [u_k]$;

– *il existe* $i \in \{1; \cdots; q\}$ *tel que pour tout* $s_1, \ldots, s_q$ *on ait* $\underline{m}\,[\xi(q_1, \ldots, q_n s)] \xleftrightarrow[W]{*} s_i$.

Delia Kesner (2000) montre qu'un grand nombre de calcul de substitution entre dans ce schéma, y compris λ_σ.

Remarque 6.1 *:* Le schéma proposé par Delia Kesner (2000) est légèrement différent, car il est donné pour des calculs avec uniquement des indices de de Bruijn et pas de précuisson. Toutefois, il est facile de l'étendre en considérant également la possibilité d'utiliser des variables libres correctement précuites par des shift. C'est la raison pour laquelle nous avons deux conditions supplémentaires, à savoir 10 et 11, dans notre définition.

Les calculs de substitution qui suivent le schéma vérifient un certain nombre de bonnes propriétés, comme la confluence, qui vont nous être utiles par la suite. On considère un calcul de substitution W vérifiant le schéma.

Lemme 6.1 (Delia Kesner, 2000, corollaire 3.2)**.** *Pour tout* $m \geq 1$*,* $n \geq 0$*, tout terme* $k : \mathrm{t}$*,*

$W(\underline{m}\,[\mathit{lift}_n(\mathsf{shift})]) = \underline{m}$ *si* $m \leq n$ *;*

$$W(\underline{m}\,[\mathit{lift}_n(\mathit{cons}(k))]) = \begin{cases} \underline{m} & \text{si } m < n+1, \\ W(k\,[\mathsf{shift}]^n) & \text{si } m = n+1. \end{cases}$$

Lemme 6.2 (Delia Kesner, 2000, lemme 4.5)**.** *Pour tout terme sans substitutions* a*, tout terme* $k : \mathrm{t}$*, et tout* $n \geq 0$*,*

$a\,[\mathit{lift}_n(\mathsf{shift})]\,[\mathit{lift}_n(\mathit{cons}(k))] \xrightarrow[W]{*} a$ *;*

$a\,\big[\mathit{lift}_{n+1}(\mathsf{shift})\big]\,[\mathit{lift}_n(\mathit{cons}(1))] \xrightarrow[W]{*} a$.

Lemme 6.3 (Delia Kesner, 2000, corollaire 5.7)**.** *Pour tout terme sans substitutions* a*, toute substitution* $s : \mathrm{s}$ *et tout* $n \geq 1$ *, on a* $W(a\,[\mathsf{shift}]^n\,[\mathit{lift}_n(\mathsf{shift})]) = W(a\,[s]\,[\mathsf{shift}]^n)$.

Lemme 6.4 (Delia Kesner, 2000, corollaire 5.9)**.** *Pour tous termes* a *et* b *de sorte* t*, on a* $W(a\,[\mathit{cons}(b)]\,[s]) = W(a\,[\mathit{lift}(s)]\,[\mathit{cons}(b\,[s])])$.

Lemme 6.5 (Confluence, Delia Kesner, 2000, théorème 5.18)**.** *Le système de réécriture* λ_W *est confluent.*

Encodage des règles d'inférence

Ensuite, il faut traduire les règles d'inférence du système. Suivant les cas, le système d'inférence ressemblera plutôt à un système de type déduction naturelle, avec des règles introduisant et éliminant des constructions, ou à un système de type calcul des séquences, avec des règles s'appliquant aux hypothèses et aux conclusions. Ceci est grandement facilité par le fait que la plupart des systèmes d'inférence sont des extensions ou des

modifications de la déduction naturelle ou du calcul des séquences. On aura pris le soin d'encoder les formules par un prédicat ϵ. Il s'agit alors de trouver des règles de réécriture autour de ϵ telles que les surrègles associées correspondent au règles d'inférence du système. Il n'y a là pas de recette fonctionnant à tous les coups. On trouvera un exemple dans la suite pour l'encodage des systèmes de type purs.

Pour démontrer l'adéquation, on pourra alors s'appuyer sur un théorème tel que le théorème 3.30 qui dit que seules les surrègles sont nécessaires pour démontrer une proposition atomique. Si on a encodé les jugements par des propositions atomiques, et que le système de réécriture est atomisant, il suffit alors de montrer que les règles d'inférence du système que l'on encode correspondent bien aux surrègles.

Exemple 6.3 *:* Pour illustrer cette section, nous allons l'appliquer au cas de la logique d'ordre supérieure basée sur la théorie des types. Plus particulièrement nous allons nous intéresser au calcul des séquences qui y est associé, HOL-λ, qui est définit par exemple par Gilles Dowek et collaborateurs (2001). Les règles de ce système de déduction sont données dans la figure 6.1.

Il faut tout d'abord encoder les formules par des termes du premier ordre. Pour cela, il s'agit essentiellement de traduire un λ-calcul comportant des constantes. On utilise donc une traduction avec des indices de de Bruijn, avec précuisson, et avec des substitutions explicites, et on travaille modulo par exemple λ_σ.

Il reste ensuite à trouver les règles de réécriture propositionnelle qui permettront d'obtenir un système de surdéduction convenable. Pour cela il paraît évident qu'on va plutôt utiliser le calcul des séquences extensible. Si on prend l'exemple de la règle $\dot{\wedge}\vdash$

$$\dot{\wedge}\vdash \frac{\Gamma, X, Y \vdash \Delta}{\Gamma, \dot{\wedge}\ X\ Y \vdash \Delta}$$

on voit que l'on va avoir besoin de la traduire par $\epsilon(\dot{\wedge}\ x\ y) \to \epsilon(x) \wedge \epsilon(y)$ ce qui donnera la surrègle gauche

$$\frac{\Gamma, \epsilon(x), \epsilon(y) \vdash \Delta}{\Gamma, \epsilon(\dot{\wedge}\ x\ y) \vdash \Delta}$$

Au passage la surrègle droite correspond alors à la règle $\vdash\dot{\wedge}$, ce qui nous évite à avoir à l'encoder. Si on s'intéresse maintenant à la règle

$$\dot{\forall}\vdash \frac{\Gamma, (X\ t)\downarrow \vdash \Delta}{\Gamma, \dot{\forall}\ X \vdash \Delta}$$

comme on doit obtenir un terme t quelconque à gauche, on va utiliser un quantificateur universel, et on va utiliser la règle suivante $\epsilon(\dot{\forall} f) \to \forall x,\ \epsilon(f\ x)$. La surrègle gauche est alors

$$\overset{\frown}{\vdash}\ \frac{}{X \vdash X} \qquad \overset{\vdash}{\smile}\ \frac{\Gamma, X \vdash \Delta \qquad \Gamma \vdash X, \Delta}{\Gamma \vdash \Delta}$$

$$\dot{\cdot}\vdash\ \frac{\Gamma, X, X \vdash \Delta}{\Gamma, X \vdash \Delta} \qquad \vdash\dot{\cdot}\ \frac{\Gamma \vdash X, X, \Delta}{\Gamma \vdash X, \Delta}$$

$$\dot{\cdot}\vdash\ \frac{\Gamma \vdash \Delta}{\Gamma, X \vdash \Delta} \qquad \vdash\dot{\cdot}\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash X, \Delta}$$

$$\dot{\Rightarrow}\vdash\ \frac{\Gamma \vdash X, \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, \dot{\Rightarrow}\ X\ Y \vdash \Delta} \qquad \vdash\dot{\Rightarrow}\ \frac{\Gamma, X \vdash Y, \Delta}{\Gamma \vdash \dot{\Rightarrow}\ X\ Y, \Delta}$$

$$\dot{\wedge}\vdash\ \frac{\Gamma, X, Y \vdash \Delta}{\Gamma, \dot{\wedge}\ X\ Y \vdash \Delta} \qquad \vdash\dot{\wedge}\ \frac{\Gamma \vdash X, \Delta \qquad \Gamma \vdash Y, \Delta}{\Gamma \vdash \dot{\wedge}\ X\ Y, \Delta}$$

$$\dot{\vee}\vdash\ \frac{\Gamma, X \vdash \Delta \qquad \Gamma, Y \vdash \Delta}{\Gamma, \dot{\vee}\ X\ Y \vdash \Delta} \qquad \vdash\dot{\vee}\ \frac{\Gamma \vdash X, \Delta}{\Gamma \vdash \dot{\vee}\ X\ Y, \Delta}$$

$$\dot{\forall}\vdash\ \frac{\Gamma, (X\ t)\downarrow \vdash \Delta}{\Gamma, \dot{\forall}\ X \vdash \Delta} \qquad \vdash\dot{\forall}\ \frac{\Gamma \vdash (X\ y)\downarrow, \Delta}{\Gamma \vdash \dot{\forall}\ X, \Delta}\ \begin{array}{l} y \text{ non libre dans} \\ \Gamma, \Delta \end{array}$$

$$\dot{\exists}\vdash\ \frac{\Gamma, (X\ y)\downarrow \vdash \Delta}{\Gamma, \dot{\exists}\ X \vdash \Delta}\ \begin{array}{l} y \text{ non libre dans} \\ \Gamma, X(x), \Delta \end{array} \qquad \vdash\dot{\exists}\ \frac{\Gamma \vdash (X\ t)\downarrow, \Delta}{\Gamma \vdash \dot{\exists}\ X, \Delta}$$

$$\dot{\bot}\vdash\ \frac{}{\Gamma, \dot{\bot} \vdash \Delta}$$

$$\dot{\neg}\vdash\ \frac{\Gamma \vdash X, \Delta}{\Gamma, \dot{\neg}\ X \vdash \Delta} \qquad \vdash\dot{\neg}\ \frac{\Gamma, X \vdash \Delta}{\Gamma \vdash \dot{\neg}\ X, \Delta}$$

$P\downarrow$ désigne la forme $\beta\eta$-normale de P

FIG. 6.1 : Règles d'inférence du système HOL-λ pour la logique d'ordre supérieure basée sur la théorie simple des types

$$\frac{\Gamma, \epsilon(f\ t) \vdash \Delta}{\Gamma, \epsilon(\dot{\forall}\ f) \vdash \Delta}$$

Comme on travaille modulo λ_σ, on peut effectivement simuler la règle de HOL-λ. De plus, la surrègle gauche, à savoir

$$\frac{\Gamma \vdash \epsilon(f\ y), \Delta}{\Gamma \vdash \epsilon(\dot{\forall}\ f), \Delta}\ \begin{array}{l} y \text{ non libre dans} \\ \Gamma, \Delta \end{array}$$

permet ici aussi de simuler $\vdash\dot{\forall}$. Le reste de l'encodage pour les autres règles est similaire, et au final on voit qu'on obtient le système de réécriture HOL-$\lambda\sigma$ défini par Gilles Dowek et collaborateurs (2001). Pour la démonstration que la traduction que nous venons de donner est bien adéquate, on se reportera donc au théorème 6.1 de Gilles Dowek et collaborateurs (2001). Au passage, remarquons que la présentation dans le calcul des séquences extensible, bien que théoriquement équivalente à celle en déduction modulo, est plus proche du système de déduction HOL-λ, puisqu'on a vraiment correspondance entre les règles d'inférence.

6.2 Systèmes de type purs fonctionnels en déduction surnaturelle

On cherche à encoder un système de type pur fonctionnel défini par les ensembles de sortes $\boldsymbol{S}$, d'axiomes $\boldsymbol{A}$ et de règles $\boldsymbol{R}$ en déduction surnaturelle.

6.2.1 Systèmes de type purs en $\lambda\Pi$ modulo

Il y a deux façon d'étendre le système de type pur $\lambda\Pi$. La première est de rajouter des règles au système de type pur pour obtenir par exemple le calcul des constructions. L'autre est d'enrichir la règle de conversion et de travailler modulo un système de réécriture en plus de la β-réduction. Denis Cousineau et Gilles Dowek (2007) se sont donc intéressés à la question de savoir si une de ces techniques est plus générale que l'autre, et ils ont démontré que tous les systèmes de type purs fonctionnels peuvent être encodés en $\lambda\Pi$ modulo.

Pour cela, ils définissent un contexte contenant pour chaque sorte s deux variables $u_s : * \text{ et } \epsilon_s : u_s \Rightarrow *$, pour chaque axiome (s, s') une variable $\dot{s} : u_{s'}$ et pour chaque règle (s_1, s_2, s_3) une variable $\dot{\pi}_{s_1,s_2,s_3} : \Pi x : u_{s_1},\ (\epsilon_{s_1}\ x) \Rightarrow u_{s_2} \Rightarrow u_{s_3}$.

Ils définissent également un système de réécriture

$$\epsilon_{s_2}\ \dot{s_1} \to u_{s_1}$$
$$\epsilon_{s_3}\ (\dot{\pi}_{s_1,s_2,s_3}\ x\ y) \to \Pi z : \epsilon_{s_1}\ x,\ \epsilon_{s_2}\ (y\ x)$$

Ils donnent alors une traduction simple des termes du système de type pur *bien typés* dans un contexte Γ :

$$
\begin{array}{rcll}
|x| & \stackrel{\text{déf}}{=} & x & \\
|s| & \stackrel{\text{déf}}{=} & \dot{s} & \\
|\Pi x : A,\ B| & \stackrel{\text{déf}}{=} & \dot{\pi}_{s_1,s_2,s_3}|A|(\lambda|x : \epsilon_{s_1}\ |A|,\ |B|) & s_1 \text{ type de } |A| \\
 & & & s_2 \text{ type de } |B| \\
 & & & s_3 \text{ type de } |\Pi x : A,\ B| \\
|\lambda x : A,\ T| & \stackrel{\text{déf}}{=} & \lambda x : \epsilon_{s_1}\ |A|,\ |T| & s_1 \text{ type de } |A| \\
|T\ U| & \stackrel{\text{déf}}{=} & |T|\ |U| &
\end{array}
$$

La traduction dépend donc du contexte. Ils donnent alors une traduction double qui est égale à $||A|| \stackrel{\text{déf}}{=} \epsilon_s\ |A|$ pour un terme A de type s et à $||s|| \stackrel{\text{déf}}{=} u_s$ pour une sorte non typées (par exemple $\Box$ dans le calcul des constructions). Ils étendent cette traduction sur les contextes de façon triviale.

Denis Cousineau et Gilles Dowek (2007) démontrent alors que la traduction est correcte, c'est-à-dire que si $\Gamma \vdash T : A$ est dérivable dans le système de type pur, alors $||\Gamma|| \vdash |T| : ||A||$ est dérivable en $\lambda\Pi$ modulo le système donné ci-dessus. Puis ils démontrent que la traduction est conservative, c'est-à-dire que si $||\Gamma|| \vdash T : ||A||$ est dérivable en $\lambda\Pi$ modulo le système donné ci-dessus, alors il existe un terme U tel que $||U|| \underset{\eta}{\overset{*}{\longleftrightarrow}} T$ et $\Gamma \vdash U : A$ est dérivable dans le système de type pur.

$\lambda\Pi$ est un système du premier ordre, dans le sens où on ne peut pas quantifier sur des formules. Néanmoins, on travaille modulo β, qui est un mécanisme d'ordre supérieur. La question s'est donc posée de savoir s'il est possible de trouver un encodage en logique du premier ordre modulo, comme cela a été fait par Gilles Dowek et collaborateurs (2001) pour la logique d'ordre supérieur fondé sur la théorie simple des types. C'est ce à quoi nous nous attelons par la suite. Notons tout de suite qu'en trouvant un encodage en déduction surnaturelle, nous avons donc par le biais des présentations compatibles une expression des systèmes de type purs comme théorie de la logique du premier ordre. Il ne s'agit pas de la première représentation des systèmes de type purs en logique du premier ordre, puisque par exemple Mark-Oliver Stehr et José Meseguer (2004) les ont exprimés en logique équationnelle d'appartenance (José Meseguer, 1998) qui peut être vue comme une sous-logique de la logique du premier ordre. Néanmoins notre traduction permet une plus grande similitude entre les dérivations de type et les démonstrations en logique du premier ordre (en déduction surnaturelle plus exactement), notamment en ce qui concerne leur normalisation. En effet, dans l'approche de Mark-Oliver Stehr et José Meseguer (2004), on a un symbole de prédicat[1] *Derivable* qui s'applique à des

[1] Si on se place effectivement en logique équationnelle d'appartenance, il s'agit en fait d'une sorte et non

jugements et est défini pour correspondre exactement aux règles d'inférence du système de type pur.

6.2.2 Traduction des termes

On cherche à exprimer la syntaxe des termes du système de type pur par des termes du premier ordre. Tout d'abord, on considère un λ-calcul avec substitutions explicites λ_W qui rentre dans le schéma défini par Delia Kesner (2000). On rajoute à ce calcul des constantes $\dot{s}$ pour chaque sorte $s \in \boldsymbol{S}$ du système de type pur, ainsi que des symboles de fonction binaires $\dot{\pi}_{s_1,s_2,s_3}$ pour chaque règle $(s_1, s_2, s_3) \in \boldsymbol{R}$, pour encoder les lieurs Π comme dans le cas de $\lambda\Pi$ modulo ci-dessus.

Il faut alors étendre le calcul de substitutions explicites pour prendre en compte ces nouveaux symboles :

$$\dot{s}\,[t] \to \dot{s}$$
$$\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a,b)\,[s] \to \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a\,[s]\,,b\,[\mathit{lift}(s)])$$

On pourrait se passer de la première règle en protégeant les constantes $\dot{s}$ par précuisson. La règle permet d'alléger la traduction des termes. La deuxième règle permet de faire passer les substitutions sous les $\dot{\pi}$. Comme le $\dot{\pi}$ lie une variable dans sa seconde composante, on retrouve un *lift* comme dans la cas de la règle (Lambda).

On peut alors traduire les termes du système de type pur en termes du premier ordre.

Définition 6.3. *Étant donnés un contexte* Γ *on définit une traduction des termes du système de type pur bien typés dans le contexte* Γ *vers les terme du premier ordre :*

$$\begin{aligned}
|x|^{\Gamma}_{\ell_1::x::\ell_2} &\stackrel{\text{déf}}{=} \underline{\overline{\ell_1}+1} && \textit{si } x \notin \ell_1\\
|x|^{\Gamma}_{\ell} &\stackrel{\text{déf}}{=} x\,[\mathsf{shift}]^{\overline{\ell}} && \textit{si } x \notin \ell\\
|s|^{\Gamma}_{\ell} &\stackrel{\text{déf}}{=} \dot{s} && (s \in \boldsymbol{S})\\
|\lambda x : A,\ T|^{\Gamma}_{\ell} &\stackrel{\text{déf}}{=} \lambda\,|T|^{\Gamma,x:A}_{x::\ell}\\
|A\ B|^{\Gamma}_{\ell} &\stackrel{\text{déf}}{=} |A|^{\Gamma}_{\ell}\,|B|^{\Gamma}_{\ell}\\
|\Pi x : A\ B|^{\Gamma}_{\ell} &\stackrel{\text{déf}}{=} \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|^{\Gamma}_{\ell}\,,|B|^{\Gamma,x:A}_{x::\ell}\right)
\end{aligned}$$

où s_1 *correspond au type de* A *dans* Γ*,* s_2 *au type de* B *dans* $\Gamma, x : A$*, et* $(s_1, s_2, s_3) \in \boldsymbol{R}$*. On notera* $|T|^{\Gamma}$ *pour* $|T|^{\Gamma}_{[]}$ *et on omettra le contexte* Γ *s'il est clair.*

Cette traduction est bien déterministe, car on a affaire à un système de type pur fonctionnel.

pas d'un symbole de prédicat. Toutefois, si on se place dans la logique du premier ordre équivalente, on aura bien un symbole de prédicat (cf. José Meseguer, 1998, section 3).

Remarque 6.2 : Si $\Gamma \subseteq \Gamma'$ alors si $|T|^{\Gamma}$ est bien défini alors $|T|^{\Gamma'}$ l'est aussi et lui est égal.

Pour traduire les jugements de type, on va utiliser un symbole de prédicat binaire ϵ. La sémantique informelle de $\epsilon(a, b)$ sera « a a le type b ».

On peut alors définir inductivement la traduction d'un contexte par $|[]| \overset{\text{déf}}{=} []$ et $|\Gamma, x : A| \overset{\text{déf}}{=} |\Gamma|, \epsilon\left(x, |A|_{[]}^{\Gamma}\right)$, et d'un jugement de type : $|\Gamma \vdash T : A| \overset{\text{déf}}{=} |\Gamma| \vdash \epsilon\left(|T|_{[]}^{\Gamma}, |A|_{[]}^{\Gamma}\right)$.

6.2.3 Traduction des règles d'inférence

On cherche à traduire les règles d'inférence d'un système de type pur. Comme un système de type pur est, via l'isomorphisme de Curry-Howard, une extension de la déduction naturelle, il semble plus judicieux de se placer dans le cadre de la déduction surnaturelle plutôt que dans celui du calcul des séquences extensible.

Dans la suite, on va supposer qu'on travaille toujours dans un contexte bien formé. On ne va donc pas chercher à traduire les règles d'inférence Vide et Déclaration.

On propose donc les règles de réécriture propositionnelles suivantes, qui trouveront leur justification quand on comparera les surrègles qu'elles produisent avec celle du système de type pur.

$$\epsilon(\dot{s}_1, \dot{s}_2) \to \top \qquad \text{pour tout } (s_1, s_2) \in \boldsymbol{A} \tag{6.1}$$

$$\epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a, b), \dot{s}_3\right) \to \epsilon(a, \dot{s}_1) \wedge \forall z.\ \epsilon(z, a) \Rightarrow \epsilon(b[cons(z)], \dot{s}_2) \tag{6.2}$$

$$\epsilon\left(t, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a, b)\right) \to \epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a, b), \dot{s}_3\right) \wedge \forall z.\ \epsilon(z, a) \Rightarrow \epsilon(t\ z, b[cons(z)]) \tag{6.3}$$

Ces règles de réécriture propositionnelles engendrent donc les surrègles en déduction surnaturelle présentées dans la figure 6.2.

On notera $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ le système de réécriture complet correspondant au système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$. La figure 6.3 représente ce système dans le cas où le λ-calcul avec substitutions explicite est λ_σ.

6.2.4 Adéquation

Propriétés de l'encodage

Pour démontrer l'adéquation, c'est à dire la correction et le conservatisme de l'encodage, nous avons besoin d'un certain nombre de lemmes énonçant des propriétés liant les réductions dans le système de type pur et celles dans l'encodage.

Pour alléger les notations, on pourra ne pas noter le système de réécriture $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ dans les étapes de réécriture. On notera donc $\longrightarrow$, $\overset{*}{\longleftrightarrow}$, etc. au lieu de $\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longrightarrow}$, $\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\overset{*}{\longleftrightarrow}}$, etc.

$$\overset{\vdash}{(6.1)}\ \frac{}{\Gamma \vdash \epsilon\left(\dot{s}_1, \dot{s}_2\right)}\ (s_1, s_2) \in \boldsymbol{A}$$

$$\overset{\vdash}{(6.2)}\ \frac{\Gamma \vdash \epsilon\left(a, \dot{s}_1\right) \qquad \Gamma, \epsilon\left(z, a\right) \vdash \epsilon\left(b\left[cons(z)\right], \dot{s}_2\right)}{\Gamma \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right)}\ z \notin FV(\Gamma, a, b),\ (s_1, s_2, s_3) \in \boldsymbol{R}$$

$$\overset{\vdash}{(6.3)}\ \frac{\Gamma \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right) \qquad \Gamma, \epsilon\left(z, a\right) \vdash \epsilon\left(t\ z, b\left[cons(z)\right]\right)}{\Gamma \vdash \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right)}\ \begin{array}{l} z \notin FV(\Gamma, a, b, t), \\ (s_1, s_2, s_3) \in \boldsymbol{R} \end{array}$$

$$\overset{(6.2)1}{\vdash}\ \frac{\Gamma \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right)}{\Gamma \vdash \epsilon\left(a, \dot{s}_1\right)}$$

$$\overset{(6.2)2}{\vdash}\ \frac{\Gamma \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right) \qquad \Gamma \vdash \epsilon\left(u, a\right)}{\Gamma \vdash \epsilon\left(b\left[cons(u)\right], \dot{s}_2\right)}$$

$$\overset{(6.3)1}{\vdash}\ \frac{\Gamma \vdash \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right)}{\Gamma \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right)}$$

$$\overset{(6.3)2}{\vdash}\ \frac{\Gamma \vdash \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right) \qquad \Gamma \vdash \epsilon\left(u, a\right)}{\Gamma \vdash \epsilon\left(t\ u, b\left[cons(u)\right]\right)}$$

FIG. 6.2 : Surrègles pour la déduction surnaturelle encodant un système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$

$$
\begin{aligned}
(\lambda a)\ b &\to a[b \cdot id] \\
(a\ b)[s] &\to a[s]\ b[s] \\
1[a \cdot s] &\to a \\
a[id] &\to a \\
\lambda a[s] &\to \lambda a[1 \cdot (s \circ \mathsf{shift})] \\
a[s][t] &\to a[s \circ t] \\
id \circ s &\to s \\
\mathsf{shift} \circ (a \cdot s) &\to s \\
(s_1 \circ s_2) \circ s_3 &\to s_1 \circ (s_2 \circ s_3) \\
(a \cdot s) \circ t &\to a[t] \cdot (s \circ t) \\
s \circ id &\to s \\
1 \cdot \mathsf{shift} &\to id \\
1[s] \cdot (\mathsf{shift} \circ s) &\to s \\
\dot{s}\,[t] &\to \dot{s} && \text{pour tout } s \in \boldsymbol{S} \\
\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a,b)\,[s] &\to \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a\,[s]\,, b\,[\mathit{lift}(s)]) && \text{pour tout } (s_1,s_2,s_3) \in \boldsymbol{R}
\end{aligned}
$$

$$
\begin{aligned}
\epsilon\,(\dot{s}_1, \dot{s}_2) &\to \top && \text{pour tout } (s_1,s_2) \in \boldsymbol{A} && (6.1)\\
\epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a,b)\,, \dot{s}_3\right) &\to \epsilon\,(a, \dot{s}_1) \wedge \forall z.\ \epsilon\,(z,a) \Rightarrow \epsilon\,(b\,[\mathit{cons}(z)]\,, \dot{s}_2) && && (6.2)\\
\epsilon\left(t, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a,b)\right) &\to \epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a,b)\,, \dot{s}_3\right) \wedge \forall z.\ \epsilon\,(z,a) \Rightarrow \epsilon\,(t\ z, b\,[\mathit{cons}(z)]) && && (6.3)
\end{aligned}
$$

FIG. 6.3 : Système de réécriture $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ permettant l'encodage d'un système de type pur, dans le cas où le λ-calcul avec substitutions explicites est λ_σ

Lemme 6.6 (Simulation de la β-réduction, Delia Kesner, 2000, lemme 5.17)**.** *Pour tous termes typés A et B du système de type pur, $A \xrightarrow[\beta]{*} B$ ssi $|A| \xrightarrow[\lambda_W]{*} |B|$.*

Lemme 6.7. *La traduction est compatible avec l'α-conversion : si $y \notin FV(B)$,*

$$|\lambda x : A\ B|_\ell = |\lambda y : A\ \{y/x\}B|_\ell$$

Démonstration. Ceci ce démontre par une simple induction sur le terme α-converti. □

Lemme 6.8. *Si $x \notin \ell$, alors*

$$|A|_{\ell::x::nil}\left[lift_{\overline{\ell}}(cons(|u|_{\ell::nil}))\right] \overset{*}{\longleftrightarrow} |\{u/x\}A|_{\ell::nil} \quad .$$

Démonstration. Par induction sur A :

– Si $A = x$, alors

$$\begin{aligned} |A|_{\ell::x::nil}\left[lift_{\overline{\ell}}(cons(|u|_{\ell::nil}))\right] &= \underline{\overline{\ell}+1}\left[lift_{\overline{\ell}}(cons(|u|_{\ell::nil}))\right] \\ &\overset{*}{\longleftrightarrow} |u|_{\ell::nil} && \text{(lemme 6.1)} \\ &= |\{u/x\}A|_{\ell::nil} \end{aligned}$$

– Si $A = y \in \ell$, alors

$$\begin{aligned} |A|_{\ell::x::nil}\left[lift_{\overline{\ell}}(cons(|u|_{\ell::nil}))\right] &= \underline{m}\left[lift_{\overline{\ell}}(cons(|u|_{\ell::nil}))\right] && \text{avec } m < \overline{\ell}+1 \\ &\overset{*}{\longleftrightarrow} \underline{m} && \text{(lemme 6.1)} \\ &= |y|_{\ell::nil} \\ &= |\{u/x\}A|_{\ell::nil} \end{aligned}$$

– Si $A = y \notin \ell :: x$, alors

$$\begin{aligned} |A|_{\ell::x::nil}\left[lift_{\overline{\ell}}(cons(|u|_{\ell::nil}))\right] &= y\,[\mathsf{shift}]^{\overline{\ell}+1}\left[lift_{\overline{\ell}}(cons(|u|_{\ell::nil}))\right] \\ &\overset{*}{\longleftrightarrow} y\,[\mathsf{shift}]\,[cons(|u|_{\ell::nil})]\,[\mathsf{shift}]^{\overline{\ell}} && \text{(lemme 6.3)} \\ &\overset{*}{\longleftrightarrow} y\,[\mathsf{shift}]^{\overline{\ell}} && \text{(lemme 6.2)} \\ &= |y|_{\ell::nil} \\ &= |\{u/x\}A|_{\ell::nil} \end{aligned}$$

– Le cas le plus intéressant est celui d'une abstraction de x, avec $y \notin FV(B, x)$:

$$
\begin{aligned}
&|\lambda x : A\ B|_{\ell::x::nil}\left[\mathit{lift}_{\overline{\ell}}(\mathit{cons}(|u|_{\ell::nil}))\right]\\
&= |\lambda y : A\ \{y/x\}B|_{\ell::x::nil}\left[\mathit{lift}_{\overline{\ell}}(\mathit{cons}(|u|_{\ell::nil}))\right]\\
&\overset{*}{\longleftrightarrow} \lambda\left(|\{y/x\}B|_{y::\ell::x::nil}\left[\mathit{lift}_{\overline{\ell}+1}(\mathit{cons}(|u|_{\ell::nil}))\right]\right)\\
&\overset{*}{\longleftrightarrow} \lambda\left(|\{u/x\}\{y/x\}B|_{y::\ell::nil}\right) && \text{(hypothèse d'induction)}\\
&= \lambda\left(|\{y/x\}B|_{y::\ell::nil}\right)\\
&= |\lambda y : A\ \{y/x\}B|_{\ell::nil}\\
&= |\lambda x : A\ B|_{\ell::nil}\\
&= |\{u/x\}\lambda x : A\ B|_{\ell::nil}
\end{aligned}
$$

□

Corollaire 6.9. $|A|_{x::nil}\,[\mathit{cons}(|u|)] \overset{*}{\longleftrightarrow} |\{u/x\}A|$.

Lemme 6.10. *Si* $|A| \overset{*}{\longrightarrow} a$ *alors il existe* A' *tel que* $a \overset{*}{\longrightarrow} |A'|$.

Démonstration. Par définition du schéma de Delia Kesner (2000), la forme W-normale de a ne contient pas de substitutions et peut donc être traduite inversement en un terme A' tel que $a \overset{*}{\longrightarrow}_W |A'|$. □

Lemme 6.11. *Si* z *n'apparaît pas dans* ℓ*, en notant informellement le replacement des occurrences de* $z\,[\mathsf{shift}]$ *par* $\underline{1}$ *dans* t *par* $\{\underline{1}/z\,[\mathsf{shift}]\}t$*, on a*

$$|A|_{\ell::z::nil} \overset{*}{\longleftrightarrow} \{\underline{1}/z\,[\mathsf{shift}]\}\left(|A|_{\ell::nil}\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right)$$

Démonstration. On procède par induction sur A :

– Si $A = z$, alors

$$
\begin{aligned}
\{\underline{1}/z\,[\mathsf{shift}]\}\left(|A|_{\ell::nil}\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right) &= \{\underline{1}/z\,[\mathsf{shift}]\}\left(z\,[\mathsf{shift}]^{\overline{\ell}}\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right)\\
&= \{\underline{1}/z\,[\mathsf{shift}]\}\left(z\,[\mathsf{shift}]^{\overline{\ell}+1}\right) && \text{(lemme 6.3)}\\
&= \underline{1}\,[\mathsf{shift}]^{\overline{\ell}}\\
&\overset{*}{\longleftrightarrow} \underline{\overline{\ell}+1} && \text{(définition 6.2)}\\
&= |A|_{\ell::z::nil}
\end{aligned}
$$

– Si $A = y \in \ell$, alors

$$
\begin{aligned}
\{\underline{1}/z\,[\mathsf{shift}]\}\left(|A|_{\ell::nil}\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right) &= \{\underline{1}/z\,[\mathsf{shift}]\}\left(\underline{m}\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right) && \text{(avec } m < \overline{\ell}\text{)}\\
&= \{\underline{1}/z\,[\mathsf{shift}]\}\underline{m} && \text{(lemme 6.1)}\\
&= \underline{m}\\
&= |A|_{\ell::z::nil}
\end{aligned}
$$

– Si $A = y \notin \ell :: z$, alors

$$\begin{aligned}
\{\underline{1}/z\,[\mathsf{shift}]\}\left(|A|_{\ell::nil}\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right) &= \{\underline{1}/z\,[\mathsf{shift}]\}\left(y\,[\mathsf{shift}]^{\overline{\ell}}\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right)\\
&= \{\underline{1}/z\,[\mathsf{shift}]\}\left(y\,[\mathsf{shift}]^{\overline{\ell}+1}\right) \qquad \text{(lemme 6.3)}\\
&= y\,[\mathsf{shift}]^{\overline{\ell}+1}\\
&= |A|_{\ell::z::nil}
\end{aligned}$$

– Le cas le plus intéressant est celui de l'abstraction de z avec $y \notin FV(B, z)$:

$$\begin{aligned}
&|\lambda z : A\ B|_{\ell::z::nil}\\
&=_\alpha |\lambda y : A\ \{y/z\}B|_{\ell::z::nil}\\
&= \lambda\left(|\{y/z\}B|_{y::\ell::z::nil}\right)\\
&\longleftrightarrow^{*} \lambda\left(\{\underline{1}/z\,[\mathsf{shift}]\}\left(|\{y/z\}B|_{y::\ell::nil}\left[\mathit{lift}_{\overline{\ell}+1}(\mathsf{shift})\right]\right)\right) \quad \text{(par hypothèse d'induction)}\\
&= \{\underline{1}/z\,[\mathsf{shift}]\}\lambda\left(|\{y/z\}B|_{y::\ell::nil}\left[\mathit{lift}_{\overline{\ell}+1}(\mathsf{shift})\right]\right)\\
&\longleftarrow \{\underline{1}/z\,[\mathsf{shift}]\}\left(\left(\lambda\,|\{y/z\}B|_{y::\ell::nil}\right)\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right)\\
&= \{\underline{1}/z\,[\mathsf{shift}]\}\left(\left(|\lambda y : A\ \{y/z\}B|_{\ell::nil}\right)\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right)\\
&=_\alpha \{\underline{1}/z\,[\mathsf{shift}]\}\left(\left(|\lambda z : A\ B|_{\ell::nil}\right)\left[\mathit{lift}_{\overline{\ell}}(\mathsf{shift})\right]\right)
\end{aligned}$$

□

Corollaire 6.12. $|A|_{z::nil} \overset{*}{\longleftrightarrow} \{\underline{1}/z\,[\mathsf{shift}]\}\,|A|_{nil}\,[\mathsf{shift}]$

Lemme 6.13 (Réduction sous Π)**.** – *Si* $\langle s_1, s_2, s_3\rangle \in \boldsymbol{R}$*, et si* $\Gamma \vdash A : s_1$ *et* $\Gamma[x : A] \vdash B : s_2$*, sont dérivable dans le système de type pur,* x *n'est pas libre dans* b*,* $a \overset{*}{\longrightarrow} |A|^{\Gamma}$ *et* $b\,[cons(x)] \longrightarrow |B|^{\Gamma[x:A]}$ *alors* $\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, b) \overset{*}{\longleftrightarrow} |\Pi x : A\ B|^{\Gamma}$*.*

– *Si* $\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, b) \overset{*}{\longrightarrow} |C|$ *alors il existe* A *et* B *tels que* $C = \Pi x : A\ B$*.*

Démonstration. Pour le premier point,

$$\begin{aligned}
|\Pi x : A\ B| &= \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, |B|_{x::nil})\\
&\overset{*}{\longleftarrow} \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, (\{\underline{1}/x\,[\mathsf{shift}]\}\,|B|_{nil}\,[\mathsf{shift}])) && \text{(corollaire 6.12)}\\
&\overset{*}{\longleftarrow} \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, (\{\underline{1}/x\,[\mathsf{shift}]\}b\,[cons(x)]\,[\mathsf{shift}]))\\
&\overset{*}{\longleftarrow} \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, (\{\underline{1}/x\,[\mathsf{shift}]\}b\,[\mathit{lift}(\mathsf{shift})]\,[cons(x\,[\mathsf{shift}])])) && \text{(lemme 6.4)}\\
&= \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, (b\,[\mathit{lift}(\mathsf{shift})]\,[cons(\underline{1})])) && (x \text{ n'est pas libre dans } b)\\
&\overset{*}{\longrightarrow} \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(a, b) && \text{(lemme 6.2)}
\end{aligned}$$

Pour le deuxième point, les réductions de $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ ne peuvent pas faire disparaître le $\dot{\pi}_{\langle s_1,s_2,s_3\rangle}$, donc $|C| = \dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a',b')$ pour certains termes a' et b'. Comme la traduction de C existe, a' et b' sont les traductions de certains termes A et B du système de type pur : $a' = |A|$ et $b' = |B|_y^{y:A}$ et $C = \Pi u : A,\ B$. Si $y \neq x$, on fait un α-conversion. □

Correction et conservatisme

Il nous faut vérifier que les surrègles d'inférence produites ne sont pas trop éloignées du système de type pur. Tout d'abord on montre que la traduction est correcte.

Théorème 6.14 (Correction)**.**
Pour tous termes T et A du système de type pur $(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})$ et pour tout contexte Γ, si $\Gamma \vdash T : A$ est dérivable dans le système de type pur alors on peut démontrer $|\Gamma| \vdash \epsilon\left(|T|^{\Gamma}, |A|^{\Gamma}\right)$ en déduction surnaturelle pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$.

Démonstration. Pour démontrer ceci, il suffit de vérifier que la traduction de chaque règle d'inférence est dérivable en déduction surnaturelle. On procède donc par induction sur la structure de la dérivation de type dans le système de type pur. Vide et Déclaration ne sont pas concernées, car on n'a pas besoin de vérifier que le contexte est bien formé en déduction surnaturelle. Néanmoins il est à noter que puisque $\Gamma \vdash T : A$ est dérivable, Γ est bien formé.

Dans la suite, on ajoutera parfois certaine règles de Conversion pour mettre en lumière les étapes modulo λ_W. On raisonne par cas suivant la dernière règle d'inférence. Si on a

$$\textsf{Sorte}\ \frac{\Gamma \text{ bien formé}}{\Gamma \vdash s_1 : s_2}\ (s_1,s_2) \in \boldsymbol{A}$$

alors on traduit par

$$\overset{\vdash}{(6.1)}\ \frac{}{\Gamma \vdash \epsilon\left(\dot{s}_1,\dot{s}_2\right)}\ (s_1,s_2) \in \boldsymbol{A}$$

Si on a

$$\textsf{Variable}\ \frac{\Gamma \text{ bien formé}}{\Gamma \vdash x : A}\ x : A \in \Gamma$$

alors par définition de la traduction $|x| = x$ et $\epsilon\left(x, |A|^{\Gamma'}\right)$ est dans $|\Gamma|$ pour $\Gamma' \subset \Gamma$. De plus, par la remarque 6.2, $|A|^{\Gamma'} = |A|^{\Gamma}$. On a donc

$$\overset{\frown}{\vdash}\ \frac{}{|\Gamma| \vdash \epsilon\left(x, |A|^{\Gamma}\right)}$$

Si on a

$$\textsf{Produit}\ \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A,\ B : s_3}\ (s_1, s_2, s_3) \in \boldsymbol{R}$$

alors par hypothèse d'induction on a des démonstrations π_1 de $|\Gamma| \vdash \epsilon\left(|A|, \dot{s}_1\right)$ et π_2 de $|\Gamma|, \epsilon\left(x, |A|\right) \vdash \epsilon\left(|B|, \dot{s}_2\right)$. De plus, comme $\Gamma, x : A$ est bien formé, $x \notin FV(\Gamma)$. Par le corollaire 6.9, $|B|_x\left[cons(x)\right] \underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\overset{*}{\longleftrightarrow}} |B|$ donc on peut construire la démonstration

$$\underset{(6.2)}{\vdash}\ \frac{\overset{\pi_1}{|\Gamma| \vdash \epsilon\left(|A|, \dot{s}_1\right)} \qquad \textsf{Conversion}\ \frac{\overset{\pi_2}{|\Gamma|, \epsilon\left(x, |A|\right) \vdash \epsilon\left(|B|, \dot{s}_2\right)}}{|\Gamma|, \epsilon\left(x, |A|\right) \vdash \epsilon\left(|B|_x\left[cons(x)\right], \dot{s}_2\right)}\ \text{Cor. 6.9}}{|\Gamma| \vdash \epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|, |B|_x\right), \dot{s}_3\right)}\ x \notin FV(|\Gamma|)$$

Si on a

$$\textsf{Abstraction}\ \frac{\Gamma \vdash \Pi x : A,\ B : s_3 \qquad \Gamma, x : A \vdash T : B}{\Gamma \vdash \lambda x : A,\ T : \Pi x : A,\ B}\ s_3 \in \boldsymbol{S}$$

$\Pi x : A,\ B$ est typé, il peut donc être traduit en un terme de la forme $\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|, |B|_x\right)$. Par hypothèse d'induction on a des démonstrations π_1 de $|\Gamma| \vdash \epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|, |B|_x\right), \dot{s}\right)$ et π_2 de $|\Gamma|, \epsilon\left(x, |A|\right) \vdash \epsilon\left(|T|, |B|\right)$. Comme $\Gamma, x : A$ est bien formé, $x \notin FV(\Gamma)$. De plus, par le corollaire 6.9, $\epsilon\left(|T|, |B|\right) \underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\overset{*}{\longleftrightarrow}} \epsilon\left((\lambda\, |T|_x)x, |B|_x\left[cons(x)\right]\right)$ On peut donc construire la démonstration suivante :

$$\underset{(6.3)}{\vdash}\ \frac{\overset{\pi_1}{|\Gamma| \vdash \epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|, |B|_x\right), \dot{s}_3\right)} \qquad \overset{\pi_2}{|\Gamma|, \epsilon\left(x, |A|\right) \vdash \epsilon\left((\lambda\, |T|_x)x, |B|_x\left[cons(x)\right]\right)}}{|\Gamma| \vdash \epsilon\left(\lambda |T|_x, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|, |B|_x\right)\right)}\ x \notin FV(|\Gamma|)$$

Si on a

$$\textsf{Application}\ \frac{\Gamma \vdash T : \Pi x : A,\ B \qquad \Gamma \vdash U : A}{\Gamma \vdash (T\ U) : \{U/x\}B}$$

alors par hypothèse d'induction on a des démonstrations π_1 de $|\Gamma| \vdash \epsilon\left(|T|, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|, |B|_x\right)\right)$ et π_2 de $|\Gamma| \vdash \epsilon\left(|U|, |A|\right)$. On peut donc construire la démonstration

$$\underset{(6.3)2}{\vdash}\ \frac{\overset{\pi_1}{|\Gamma| \vdash \epsilon\left(|T|, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(|A|, |B|_x\right)\right)} \qquad \overset{\pi_2}{|\Gamma| \vdash \epsilon\left(|U|, |A|\right)}}{\textsf{Conversion}\ \frac{|\Gamma| \vdash \epsilon\left(|T|\ \ |U|, |B|_x\left[cons(|U|)\right]\right)}{|\Gamma| \vdash \epsilon\left(|T|\ \ |U|, |\{U/x\}B|\right)}\ \text{Corollaire 6.9}}$$

Si on a

$$\textsf{Conversion}\ \frac{\Gamma \vdash T : A \qquad \Gamma \vdash A : s}{\Gamma \vdash T : B}\ s \in \boldsymbol{S} \text{ et } A \underset{\beta}{\overset{*}{\longleftrightarrow}} B$$

alors par hypothèse d'induction on a une démonstration π de $|\Gamma| \vdash \epsilon(|T|, |A|)$. D'après le lemme 6.6, comme $A \overset{*}{\underset{\beta}{\longleftrightarrow}} B$ on a $|A| \overset{*}{\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longleftrightarrow}} |B|$, et π est donc également une démonstration de $|\Gamma| \vdash \epsilon(|T|, |B|)$. □

Nous pouvons également démontrer que les nouvelles règles ne permettent pas de démontrer trop de résultats.

Théorème 6.15 (Conservatisme)**.**
Pour tout contexte bien formé Γ d'un système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$, pour tous termes du premier ordre a et b, si on peut démontrer $|\Gamma| \vdash \epsilon(a, b)$ en déduction surnaturelle pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$, alors il existe des termes A et B du système de type pur tels que $a \overset{*}{\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longrightarrow}} |A|$, $b \overset{*}{\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longrightarrow}} |B|$ et $\Gamma \vdash A : B$ est dérivable dans le système de type pur.

Démonstration. On procède par induction pour l'ordre lexicographique sur la structure de la démonstration de $|\Gamma| \vdash \epsilon(a, b)$ et le nombre d'étapes modulo. Comme les règles propositionnelles de $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ sont atomisantes, en utilisant le théorème 3.30, on n'a à considérer que les surrègles et $\overset{\frown}{\vdash}$.

Si la dernière règle d'inférence de la démonstration est appliquée modulo le système de réécriture sur les termes, on considère la démonstration où on applique cette dernière règle sans modulo, qui a donc pour conséquence $|\Gamma| \vdash \epsilon(a', b')$ avec $a \overset{*}{\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longleftrightarrow}} a'$ et $b \overset{*}{\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longleftrightarrow}} b'$. Par hypothèse d'induction on obtient deux termes A' et B' tels que $a' \overset{*}{\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longrightarrow}} |A'|$, $b' \overset{*}{\underset{\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}}{\longrightarrow}} |B'|$ et $\Gamma \vdash A' : B'$ est dérivable dans le système de type pur. Par confluence de λ_W, il existe a'' tel que $a \overset{*}{\longrightarrow} a'' \overset{*}{\longleftarrow} |A'|$. Par le lemme 6.10 il existe un A tel que $a'' \overset{*}{\longrightarrow}_W |A|$. Par réduction du sujet, $\Gamma \vdash A : B'$ possède une dérivation de type π. Par confluence de λ_W, il existe b'' tel que $b \overset{*}{\longrightarrow} b'' \overset{*}{\longleftarrow} |B'|$. Par le lemme 6.10 il existe un B tel que $b'' \overset{*}{\longrightarrow}_W |B|$. Par le lemme 3.4, on peut donc dériver $\Gamma \vdash A : B$. Nous pouvons par conséquent considérer par la suite que les règles d'inférence sont appliquées sans modulo.

Si on a

$$\overset{\frown}{\vdash} \frac{}{|\Gamma_1|, \epsilon(x, b), |\Gamma_2| \vdash \epsilon(x, b)}$$

Par définition de la traduction du contexte $|\Gamma| = |\Gamma_1|, \epsilon(x, b), |\Gamma_2|$ cela veut dire qu'il existe un terme B tel que $b = |B|^{\Gamma_1}$. De plus, le remarque 6.2 nous dit que $b = |B|^{\Gamma}$. Comme on suppose que Γ est bien formé on peut bien construire la dérivation de type

$$\mathsf{Variable}\ \frac{\Gamma \text{ bien formé}}{\Gamma_1, x : B, \Gamma_2 \vdash x : B}$$

Si on a

$$\overset{\vdash}{(6.1)}\ \frac{}{|\Gamma| \vdash \epsilon\left(\dot{s}_1, \dot{s}_2\right)}\ (s_1, s_2) \in \boldsymbol{A}$$

alors comme par hypothèse Γ est bien formé, on peut construire la dérivation de type

$$\mathsf{Sorte}\ \frac{\Gamma \text{ bien formée}}{\Gamma \vdash s_1 : s_2}\ (s_1, s_2) \in \boldsymbol{A}$$

Si on a

$$\overset{\vdash}{(6.2)}\ \frac{|\Gamma| \vdash \epsilon\left(a, \dot{s}_1\right) \qquad |\Gamma|, \epsilon\left(z, a\right) \vdash \epsilon\left(b\left[cons(z)\right], \dot{s}_2\right)}{|\Gamma| \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right)}\ z \notin FV(|\Gamma|, a, b),\ (s_1, s_2, s_3) \in \boldsymbol{R}$$

par hypothèse d'induction sur la sous-démonstration gauche il existe un terme A tel que $a \xrightarrow{*} |A|$ et il existe une dérivation de type π de $\Gamma \vdash A : s_1$. Comme z n'est pas libre dans $|\Gamma|$, il ne l'est pas dans Γ et donc $\Gamma, z : A$ est bien formé. On peut donc appliquer l'hypothèse d'induction sur la sous-démonstration droite, ce qui nous donne un terme B tel que $b\left[cons(z)\right] \xrightarrow{*} |B|$ et il existe une dérivation π' de $\Gamma, z : A \vdash B : s_2$. Par conséquent on peut construire la dérivation

$$\mathsf{Produit}\ \frac{\overset{\pi}{\Gamma \vdash A : s_1} \qquad \overset{\pi'}{\Gamma, z : A \vdash B : s_2}}{\Gamma \vdash \Pi z : A,\ B : s_3}$$

et par le lemme 6.13 $\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right) \xrightarrow{*} \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(|A|, |B|_z\right) = |\Pi z : A,\ B|$.

Si on a

$$\overset{\vdash}{(6.3)}\ \frac{|\Gamma| \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right) \qquad |\Gamma|, \epsilon\left(z, a\right) \vdash \epsilon\left(t\ z, b\left[cons(z)\right]\right)}{|\Gamma| \vdash \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right)}\ \begin{array}{l} z \notin FV(|\Gamma|, a, b, t), \\ (s_1, s_2, s_3) \in \boldsymbol{R} \end{array}$$

par hypothèse d'induction sur la sous-démonstration gauche il existe un terme P tel que $\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right) \xrightarrow{*} |P|$ et $\Gamma \vdash P : s_3$ est dérivable. Par le lemme 6.13, il existe des termes A et B_1 tels que $P = \Pi x : A,\ B_1$, et $\Gamma \vdash A : s_1$ et $\Gamma, x : A \vdash B_1 : s_2$ sont dérivables. Comme z n'est pas libre dans Γ, le contexte $\Gamma, z : A$ est bien formé et on peut appliquer l'hypothèse d'induction à la sous-démonstration droite. On obtient des termes T' et B_2 tels que $t\ z \xrightarrow{*} |T'|$, $b\left[cons(z)\right] \xrightarrow{*} |B_2|$ et on a une dérivation π de $\Gamma, z : A \vdash T' : B_2$. Par le corollaire 6.9, $|\{z/x\}B_1| \xleftrightarrow{*} b\left[cons(z)\right]$ donc par confluence de λ_W et le lemme 6.10 il existe un terme B tel que $\{z/x\}B_1 \underset{\beta}{\longrightarrow} B \underset{\beta}{\longleftarrow} B_2$. On peut donc utiliser l'α-conversion et la réduction du sujet pour obtenir une dérivations π_1 de $\Gamma \vdash \Pi z : A,\ B : s_3$. Par le lemme 3.4, à partir de π on peut produire une dérivation π_2 de $\Gamma, z : A \vdash T' : B$. On peut donc construire la dérivation

$$\mathsf{Abstraction}\ \frac{\overset{\pi_1}{\Gamma \vdash \Pi z : A,\ B : s_3} \qquad \overset{\pi_2}{\Gamma, z : A \vdash T' : B}}{\Gamma \vdash \lambda z : A,\ T' : \Pi z : A,\ B}$$

On considère alors la réduction $t\ z \xrightarrow{*} |T'|$ et on distingue deux cas :

– L'application en tête ne se réduit jamais, auquel cas T' est de la forme $T\ z$ avec $t \xrightarrow{*} |T|$. Dans ce cas, comme $z \notin FV(t)$, il suffit d'utiliser la réduction du sujet pour η (lemme 3.2) pour obtenir une dérivation de $\Gamma \vdash T : \Pi z : A,\ B$.
– L'application en tête se réduit, on a $t\ z \xrightarrow{*} (\lambda t_1)\ z \xrightarrow{*} |T'|$. On a

$$\begin{aligned}
\lambda(t\,[\mathsf{shift}]\ \underline{1}) &= \{\underline{1}/z\,[\mathsf{shift}]\}\lambda(t\,[\mathsf{shift}]\ z\,[\mathsf{shift}]) \\
&= \lambda\left(\{\underline{1}/z\,[\mathsf{shift}]\}((tz)\,[\mathsf{shift}])\right) \\
&\xrightarrow{*} \lambda\left(\{\underline{1}/z\,[\mathsf{shift}]\}(\left|T'\right|_{[]}\,[\mathsf{shift}])\right) \\
&\xleftrightarrow{*} \lambda\left|T'\right|_z && \text{(lemme 6.12)} \\
&= |\lambda z : A,\ T'|
\end{aligned}$$

On a aussi

$$\begin{aligned}
\lambda(t\,[\mathsf{shift}]\ \underline{1}) &\xrightarrow{*} \lambda((\lambda t_1)\,[\mathsf{shift}]\ \underline{1}) \\
&\xrightarrow{*} \lambda((\lambda t_1\,[\mathit{lift}(\mathsf{shift})])\underline{1}) \\
&\longrightarrow_{(Beta)} \lambda(t_1\,[\mathit{lift}(\mathsf{shift})]\,[\mathit{cons}(\underline{1})]) \\
&\xrightarrow{*} \lambda t_1 && \text{(lemme 6.2)} \\
&\xleftarrow{*} t
\end{aligned}$$

donc $t \xleftrightarrow{*} |\lambda z : A,\ T'|$ et par confluence de λ_W et par le lemme 6.10 il existe un terme T tel que $t \xrightarrow{*} |T|$ et $\lambda z : A,\ T' \xrightarrow[\beta]{*} T$. En utilisant la réduction du sujet on a bien une dérivation de $\Gamma \vdash T : \Pi z : A,\ B$.

Si on a

$$\overset{(6.2)1}{\vdash}\ \frac{|\Gamma| \vdash \epsilon\left(\dot\pi_{\langle s_1,s_2,s_3\rangle}\,(a,b)\,,\dot s_3\right)}{|\Gamma| \vdash \epsilon\,(a,\dot s_1)}$$

par hypothèse d'induction il existe un terme P tel que $\dot\pi_{\langle s_1,s_2,s_3\rangle}\,(a,b) \xrightarrow{*} |P|$ et $\Gamma \vdash P : s_3$ est dérivable. Par le lemme 6.13, il existe des termes A et B tels que $P = \Pi x : A,\ B$ et $a \xrightarrow{*} |A|$. Par définition de la traduction, $|P| = \dot\pi_{\langle s'_1,s'_2,s'_3\rangle}\,(a',b')$ avec s'_1 le type de A dans le contexte Γ, s'_2 le type de B dans le contexte $\Gamma, x : A$, et s'_3 le type de $\Pi x : A,\ B$ dans le contexte Γ. Comme $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ ne peut pas modifier les $\dot\pi$, on a forcément $s_i = s'_i$ pour $i \in \{1;2;3\}$. Par conséquent on a bien une dérivation de $\Gamma \vdash A : s_1$.

Si on a

$$\overset{(6.2)2}{\vdash}\ \frac{|\Gamma| \vdash \epsilon\left(\dot\pi_{\langle s_1,s_2,s_3\rangle}\,(a,b)\,,\dot s_3\right) \qquad |\Gamma| \vdash \epsilon\,(u,a)}{|\Gamma| \vdash \epsilon\,(b\,[\mathit{cons}(u)]\,,\dot s_2)}$$

par hypothèse d'induction sur la sous-démonstration gauche il existe un terme P tel que $\dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a,b) \xrightarrow{*} |P|$ et $\Gamma \vdash P : s_3$ est dérivable. Par le lemme 6.13, il existe des termes A_1 et B tels que $P = \Pi x : A_1,\ B$ et $b \xrightarrow{*} |B|_x$. Par définition de la traduction, $|P| = \dot{\pi}_{\langle s'_1,s'_2,s'_3\rangle}(a',b')$ avec s'_1 le type de A_1 dans le contexte Γ, s'_2 le type de B dans le contexte $\Gamma, x : A_1$, et s'_3 le type de $\Pi x : A_1,\ B$ dans le contexte Γ. Comme $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ ne peut pas modifier les $\dot{\pi}$, on a forcément $s_i = s'_i$ pour $i \in \{1;2;3\}$. Par conséquent on a une dérivation π de $\Gamma, x : A_1 \vdash B : s_2$. Par hypothèse d'induction sur la sous-démonstration droite il existe des termes U et A_2 tels que $u \xrightarrow{*} |U|$, $a \xrightarrow{*} |A_2|$ et $\Gamma \vdash U : A_2$ possède une dérivation π'. Par confluence de λ_W et le lemme 6.10 il existe un terme A tel que $A_1 \xrightarrow[\beta]{*} A \xleftarrow[\beta]{*} A_2$. En appliquant Conversion après chaque utilisation de Variable pour $x : A_1$ dans π on peut obtenir une dérivation de $\Gamma, x : A \vdash B : s_2$. En appliquant le lemme 3.4 sur π' on obtient une dérivation de $\Gamma \vdash U : A$. En utilisant le lemme de substitution 3.5 on a donc une dérivation de $\Gamma \vdash \{U/x\}B : s_2$.

Si on a

$$\overset{(6.3)1}{\vdash}\ \frac{|\Gamma| \vdash \epsilon\left(t, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a,b)\right)}{|\Gamma| \vdash \epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a,b), \dot{s}_3\right)}$$

par hypothèse d'induction il existe des termes T et P tel que $t \xrightarrow{*} |T|$, $\dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a,b) \xrightarrow{*} |P|$ et $\Gamma \vdash T : P$ est dérivable. Par le lemme 6.13, il existe des termes A et B tels que $P = \Pi x : A,\ B$. Par définition de la traduction, $|P| = \dot{\pi}_{\langle s'_1,s'_2,s'_3\rangle}(a',b')$ avec s'_1 le type de A dans le contexte Γ, s'_2 le type de B dans le contexte $\Gamma, x : A$, et s'_3 le type de $\Pi x : A,\ B$ dans le contexte Γ. Comme $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ ne peut pas modifier les $\dot{\pi}$, on a forcément $s_i = s'_i$ pour $i \in \{1;2;3\}$. Par conséquent on a bien une dérivation de $\Gamma \vdash \Pi x : A,\ B : s_3$.

Si on a

$$\overset{(6.3)2}{\vdash}\ \frac{|\Gamma| \vdash \epsilon\left(t, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a,b)\right) \qquad |\Gamma| \vdash \epsilon(u,a)}{|\Gamma| \vdash \epsilon(t\ u, b\,[cons(u)])}$$

par hypothèse d'induction sur la sous-démonstration gauche il existe un terme P tel que $\dot{\pi}_{\langle s_1,s_2,s_3\rangle}(a,b) \xrightarrow{*} |P|$ et $\Gamma \vdash T : P$ possède une dérivation π. Par le lemme 6.13, il existe des termes A_1 et B_1 tels que $P = \Pi x : A_1,\ B_1$, $a \xrightarrow{*} |A_1|$ et $b \xrightarrow{*} |B_1|_x$. Par hypothèse d'induction sur la sous-démonstration droite il existe des termes U et A_2 tels que $u \xrightarrow{*} |U|$, $a \xrightarrow{*} |A_2|$ et $\Gamma \vdash U : A_2$ possède une dérivation π'. Par confluence de λ_W et le lemme 6.10 il existe un terme A tel que $A_1 \xrightarrow[\beta]{*} A \xleftarrow[\beta]{*} A_2$. On peut appliquer le lemme 3.4 pour obtenir une dérivation ϖ de $\Gamma \vdash T : \Pi x : A,\ B_1$, de même pour π' pour obtenir une dérivation ϖ' de $\Gamma \vdash U : A$. On peut donc construire la dérivation

$$\textsf{Application}\ \frac{\overset{\varpi}{\Gamma \vdash T : \Pi x : A,\ B_1} \qquad \overset{\varpi'}{\Gamma \vdash U : A}}{\Gamma \vdash T\ U : \{U/x\}B_1}$$

or d'après le corollaire 6.9, $b\,[cons(u)] \overset{*}{\longleftrightarrow} \{U/x\}B_1$. Par confluence de λ_W et le lemme 6.10 il existe un terme B tel que $b\,[cons(u)] \overset{*}{\longrightarrow} |B|$ et $\{U/x\}B_1 \overset{*}{\underset{\beta}{\longrightarrow}} B$. On peut alors appliquer le lemme 3.4 à la dérivation ci-dessus pour obtenir une dérivation de $\Gamma \vdash T\ U : B$. □

Remarque 6.3 *:* On ne peut pas se passer du fait qu'on ait besoin de réécrire a et b pour obtenir la traduction de termes typables. En effet, par le fait que nous travaillons modulo λ_W, il est possible de démontrer $\epsilon\,(a, b)$ pour des termes a et b qui par exemple ne normalisent pas fortement. Ainsi, si on se place dans le système de type pur $\lambda\Pi$, il est possible de démontrer $\vdash \epsilon\,((\lambda\lambda\underline{1})(\lambda\underline{11})\dot{*}, \dot{\Box})$ dans la déduction surnaturelle correspondante, mais il n'est pas possible de trouver des types A, B et C tels que $\vdash (\lambda x : A,\ \lambda y : B,\ y)\ (\lambda x : C,\ x\ x) * : \Box$ soit dérivable dans $\lambda\Pi$.

Néanmoins, si le système de type pur est fortement normalisant et si on peut démontrer $|\Gamma| \vdash \epsilon\,(a, b)$ pour un contexte Γ bien formé, alors avec le théorème que l'on vient de démontrer a et b normalisent faiblement. De plus, quand un λ-terme normalise faiblement, on connaît une stratégie de réduction qui mène à sa forme normale : il s'agit de réduire d'abord les radicaux les plus à les moins profonds et les plus à gauche. On sait alors comment réduire a et b pour trouver leur forme normale et ainsi obtenir des A et B tels que $\Gamma \vdash A : B$ est dérivable dans le système de type pur.

Si on veut se passer de l'hypothèse que le contexte est bien formé, il faut encoder les jugements de type autrement. On peut par exemple considérer une constante nil et un symbole de fonction :: pour construire des listes et un nouveau prédicat ternaire C qui est comme ϵ mais prend un liste de terme comme contexte en plus. On peut alors donner la traduction d'un contexte : $|x : A, \Theta|_\ell^\Gamma \overset{\text{déf}}{=} |A|_\ell^\Gamma :: |\Theta|_\ell^{\Gamma, x:A}$ et $|[]| \overset{\text{déf}}{=} \mathsf{nil}$, puis celle d'un jugement de type : $|\Gamma \vdash A : B| = C(|\Gamma|\,, |A|^\Gamma_{x_1,\ldots,x_n}\,, |B|^\Gamma_{x_1,\ldots,x_n})$ où les $x_1, \ldots, x_n$ sont les variables de Γ. On ajoute alors deux nouvelles règles propositionnelles

$$C(t :: l, a, b) \to (\epsilon\,(t, \dot{s_1}) \vee \ldots \vee \epsilon\,(t, \dot{s_n})) \qquad (6.4)$$
$$\wedge\, \forall x,\ \epsilon\,(x, t) \Rightarrow C(l\,[cons(x)]\,, a\,[cons(x)]\,, b\,[cons(x)])$$
$$C(\mathsf{nil}, a, b) \to \epsilon\,(a, b) \qquad (6.5)$$

où $\{s_1, \ldots, s_n\} = \boldsymbol{S}$, et deux règles sur les termes pour propager les substitutions dans les listes

$$(t :: l)\,[s] \to t\,[s] :: l\,[\mathit{lift}(s)] \qquad (6.6)$$
$$\mathsf{nil}\,[s] \to \mathsf{nil} \qquad (6.7)$$

Un premier inconvénient est que la déduction surnaturelle n'a pas été définie pour la disjonction. Toutefois, ici, comme les disjonctions portent sur des propositions atomiques, on peut s'en sortir et calculer les surrègles suivantes :

$$\overset{\vdash}{}_{(6.4)i}\ \frac{\Gamma \vdash t : s_i \qquad \Gamma, \epsilon\,(x,t) \vdash C(l\,[cons(x)]\,, a\,[cons(x)]\,, b\,[cons(x)])}{\Gamma \vdash C(t :: l, a, b)}$$

$$\overset{\vdash}{}_{(6.5)}\ \frac{\Gamma \vdash \epsilon\,(a,b)}{\Gamma \vdash C(\mathsf{nil}, a, b)}$$

$$\overset{(6.4)}{\vdash}\ \frac{\Gamma \vdash C(t :: l, a, b) \qquad \Gamma, t : s_1 \vdash D \qquad \ldots \qquad \Gamma, t : s_n \vdash D}{\Gamma \vdash D}$$

$$\overset{(6.5)}{\vdash}\ \frac{\Gamma \vdash C(\mathsf{nil}, a, b)}{\Gamma \vdash \epsilon\,(a,b)}$$

Les règles $\overset{\vdash}{}_{(6.4)i}$ permettent de vérifier que la traduction du contexte du système de type pur est bien formée avant de l'ajouter dans le contexte de la déduction surnaturelle. On peut alors montrer sans trop de difficultés la correction de la traduction, c'est-à-dire que si $\Gamma \vdash A : B$ est dérivable dans le système alors $\vdash |\Gamma \vdash A : B|$ peut être démontré en déduction surnaturelle avec les nouvelles surrègles. Réciproquement, il est probablement possible de montrer un résultat de conservatisme disant que si $\vdash C(l, a, b)$ est démontrable en déduction surnaturelle, alors on peut trouver un jugement de type dérivable dans le système de type pur et qui est la traduction d'un réduit de $C(l, a, b)$. Pour cela, la seule règle problématique serait $\overset{(6.4)}{\vdash}$. Néanmoins, nous ne chercherons pas à démontrer ce théorème, car de toute façon les démonstrations n'ont plus la même structure que les dérivations de type, ce qui diminue l'intérêt de notre traduction par rapport aux travaux existants comme ceux de Mark-Oliver Stehr et José Meseguer (2004).

6.3 Applications

6.3.1 Un calcul des séquences pour la recherche automatique de démonstrations dans les systèmes de types purs

Grâce aux traductions données dans la section 3.3.2, on sait que le calcul des séquences extensibles pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ est équivalent à la déduction surnaturelle pour ce système. Il possède donc les mêmes propriétés de correction et de conservatisme par rapport au système de type pur.

Les règles du calcul des séquences extensible pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ sont représentées dans la figure 6.4. Ces règles s'appliquent modulo les règles sur les termes.

Si on cherche à construire une méthode de recherche de démonstrations automatique, on va plutôt privilégier une version asymétrique du calcul des séquences extensibles, où les règles sur les termes ne s'appliquent que de bas vers le haut. Dans le cas général, cela ne pose pas problème car λ_W est confluent. Dans le cas de λ_σ, on est en plus avantagé par le fait que le lemme 6.9 devient :

$$(6.1)\vdash \frac{\Gamma \vdash \Delta}{\Gamma, \epsilon\left(\dot{s}_1, \dot{s}_2\right) \vdash \Delta}\left(s_1, s_2\right) \in \boldsymbol{A}$$

$$\vdash(6.1) \frac{}{\Gamma \vdash \epsilon\left(\dot{s}_1, \dot{s}_2\right), \Delta}\left(s_1, s_2\right) \in \boldsymbol{A}$$

$$(6.2)\vdash \frac{\Gamma, \epsilon\left(a, \dot{s}_1\right), \epsilon\left(b\left[cons(u)\right], \dot{s}_2\right) \vdash \Delta \qquad \Gamma, \epsilon\left(a, \dot{s}_1\right) \vdash \epsilon\left(u, a\right), \Delta}{\Gamma, \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right) \vdash \Delta}$$

$$\vdash(6.2) \frac{\Gamma \vdash \epsilon\left(a, \dot{s}_1\right), \Delta \qquad \Gamma, \epsilon\left(z, a\right) \vdash \epsilon\left(b\left[cons(z)\right], \dot{s}_2\right)}{\Gamma \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right), \Delta}$$

$$(6.3)\vdash \frac{\Gamma, \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right), \epsilon\left(t\ u, b\left[cons(u)\right]\right) \vdash \Delta \qquad \Gamma, \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right) \vdash \epsilon\left(u, a\right), \Delta}{\Gamma, \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right) \vdash \Delta}$$

$$\vdash(6.3) \frac{\Gamma \vdash \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right), \Delta \qquad \Gamma, \epsilon\left(z, a\right) \vdash \epsilon\left(t\ z, b\left[cons(z)\right]\right)}{\Gamma \vdash \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right), \Delta}$$

FIG. 6.4 : Surrègles du calcul des séquences extensible encodant un système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$

Lemme 6.16. $|B|_x \left[|U|_{[]} \cdot id \right] \xrightarrow[\lambda_\sigma]{*} |\{U/x\}B|_{[]}$

ce qui fait que dans la traduction donnée par le théorème 6.14 on peut se contenter d'appliquer les règles sur les termes vers le haut après une introduction et vers le bas après une élimination. En traduisant vers le calcul des séquences extensibles, ces règles s'appliqueront donc vers le haut.

Pour avoir une méthode de recherche de démonstration automatique avec un espace de recherche limité, il faudrait que $\vdash_{\supset}$ soit admissible. Comme nous le montrons ci-dessous, ce n'est pas toujours le cas, en particulier pour le système $\lambda*$ qui n'est pas fortement normalisant. Néanmoins, comme nous le démontrons, si le système de type pur est faiblement normalisant alors le calcul des séquences extensible admet les coupures. Dans tous les cas, grâce à la proposition 3.29, on sait qu'on peut se restreindre à des coupures introduisant des propositions atomiques.

Un des problèmes pour le recherche de démonstration automatique réside dans le fait que u doit être deviné dans les règles $(6.2)\vdash$ et $(6.3)\vdash$, de la même façon que dans les règles $\forall\vdash$ et $\vdash\exists$. La technique classique pour résoudre cela est l'utilisation de méta-variables, comme par exemple cela est fait dans TaMed (cf. sections 3.2.4 et 4.4.2). Mais même en utilisant ceci, on va générer un grand nombre de méta-variables, et les contraintes d'unifications seront difficiles à résoudre.

Pour restreindre l'espace de recherche, on peut alors s'appuyer sur le fait qu'en général le u correspond à un argument qui est appliqué à une t dans les conclusions de la séquence, pour pouvoir au final utiliser $\vdash_{\supset}$. On peut alors proposer une instance de $(6.3)\vdash$qui sera plus appropriée.

$$\textsf{App.}\ \frac{\Gamma, \epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(a,b\right),\dot{s}_3\right),\epsilon\left(t\ u_1,b\left[cons(u_1)\right]\right)\vdash\epsilon\left(t\ u_1\ \ldots\ u_n,c\right),\Delta \qquad \vdots}{\Gamma,\epsilon\left(t,\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(a,b\right)\right)\vdash\epsilon\left(t\ u_1\ \ldots\ u_n,c\right),\Delta}$$

$$\Gamma,\epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(a,b\right),\dot{s}_3\right)\vdash\epsilon\left(u_1,a\right),\epsilon\left(t\ u_1\ \ldots\ u_n,c\right),\Delta$$

Ainsi, l'argument appliqué à t n'a pas à être deviné, mais il provient du reste de la séquence. On peut alors suivre une stratégie de recherche de démonstration qui applique cette nouvelle règle en prenant un t le plus gros possible de façon à minimiser le nombre d'arguments qui restent à appliquer. Ceci reste une heuristique, mais nous pensons que cette stratégie est complète pour la recherche de démonstrations dans les contextes bien formés.

Implémentation en Lemuridæ

Lemuridæ est un prototype d'assistant à la démonstration développé principalement par Paul Brauner et basé sur la surdéduction, plus particulièrement le calcul des séquences extensibles pour la logique classique du premier ordre (Paul Brauner et collaborateurs, 2007*b*). Il est écrit en TOM/Java.

Lemuridæ repose sur la logique classique. Ceci pose un problème car l'extension de la logique classique par la théorie compatible avec l'encodage d'un système de type pur n'est pas conservative par rapport à celle de la logique intuitionniste, y compris sans considérer les tautologies de la logique classique qui ne sont pas des tautologies de la logique intuitionniste, comme le montre l'exemple suivant :

Exemple 6.4 *:* Il est connu que la formule de Pierce $((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$ est un exemple de formule valide en logique classique mais pas en logique intuitionniste. Par conséquent, il est impossible de trouver un λ-terme T tel que $P : *, Q : * \vdash T : \Pi x : (\Pi y : (\Pi z : P,\ Q),\ P),\ P$ soit dérivable dans le système de type pur $\lambda\!\!\rightarrow$ correspondant au λ-calcul simplement typé. Par le théorème 6.15, il n'existe donc pas de terme t tel que $\epsilon\left(p, \dot{*}\right), \epsilon\left(q, \dot{*}\right) \vdash \epsilon\left(t, \dot{\pi}_{\langle \dot{*},\dot{*},\dot{*}\rangle}\left(\dot{\pi}_{\langle \dot{*},\dot{*},\dot{*}\rangle}\left(p, q\left[\mathsf{shift}\right]\right), p\left[\mathsf{shift}\right]\right), p\left[\mathsf{shift}\right]\right)\right)$ en déduction surnaturelle pour $\mathcal{PTS}_{\lambda\!\!\rightarrow}$. Comme $\lambda\!\!\rightarrow$ est fortement normalisant, nous montrerons ci-dessous que le calcul des séquences extensible pour $\mathcal{PTS}_{\lambda\!\!\rightarrow}$ admet les coupures, et possède donc la propriété du témoin. Par conséquent $\epsilon\left(p, \dot{*}\right), \epsilon\left(q, \dot{*}\right) \vdash \exists t,\ \epsilon\left(t, \dot{\pi}_{\langle \dot{*},\dot{*},\dot{*}\rangle}\left(\dot{\pi}_{\langle \dot{*},\dot{*},\dot{*}\rangle}\left(p, q\left[\mathsf{shift}\right]\right), p\left[\mathsf{shift}\right]\right), p\left[\mathsf{shift}\right]\right)\right)$ n'est pas démontrable (plus exactement $\vdash \exists t,\ (\epsilon\left(p, \dot{*}\right) \wedge \epsilon\left(q, \dot{*}\right)) \Rightarrow \epsilon\left(t, \dot{\pi}_{\langle \dot{*},\dot{*},\dot{*}\rangle}\left(\dot{\pi}_{\langle \dot{*},\dot{*},\dot{*}\rangle}\left(\dot{\pi}_{\langle \dot{*},\dot{*},\dot{*}\rangle}\left(p, q\left[\mathsf{shift}\right]\right), p\left[\mathsf{shift}\right]\right), p\left[\mathsf{shift}\right]\right)\right)$, ce qui est intuitionnistement équivalent). Néanmoins, cette séquence est démontrable en calcul des séquences extensible pour $\mathcal{PTS}_{\lambda\!\!\rightarrow}$ en logique classique, en instanciant tout d'abord t par $\lambda(\underline{1}\ d)$, ce qui permet, après l'application de plusieurs surrègles, d'avoir $\epsilon\left(z, p\right)$ dans le contexte pour un z frais. On peut alors réinstancier t par $\lambda(z\left[\mathsf{shift}\right])$ et fermer la démonstration.

Remarque 6.4 *:* On aurait pu démontrer ce résultat avec d'autres formules, en particulier avec des tautologies de la logique classique qui n'en sont pas pour la logique intuitionniste. Néanmoins, les séquences de la forme $|\Gamma| \vdash \exists t,\ \epsilon\left(t, |P|\right)$ pour un contexte Γ bien formé sont typiquement celles que l'on va chercher à démontrer pour faire de la recherche automatique de démonstration dans les systèmes de type purs, puisque l'on va chercher un terme t qui est la traduction d'un terme de démonstration de la proposition P. L'exemple ci-dessus montre que ce n'est pas envisageable si on se base sur la logique classique.

Pour pouvoir utiliser Lemuridæ de façon correcte, il faudrait donc le modifier pour le restreindre à la logique intuitionniste, ce que n'avons à l'heure actuelle pas réalisé. Nous avons toutefois ajoutée une tactique qui permet de simuler la règle App. décrite ci-dessus, afin de simplifier la construction des démonstrations dans Lemuridæ. En effet, sans cela, l'utilisateur est obligé de fournir le terme appliqué quand on utilise la surrègle $(6.3)\vdash$.

6.3.2 Normalisation

Nous allons maintenant étudier la relation entre la normalisation forte d'un système de type pur et celle de la déduction surnaturelle correspondante. Pour cela, intéressons nous aux termes de démonstrations de cette déduction surnaturelle, qui possèdent les règles

$$\overset{\vdash}{{}_{(6.1)}}\ \frac{}{\Gamma \vdash \mathsf{f}_{(6.1)} : \epsilon\left(\dot{s}_1, \dot{s}_2\right)}\ (s_1, s_2) \in \boldsymbol{A}$$

$$\overset{\vdash}{{}_{(6.2)}}\ \frac{\Gamma \vdash p : \epsilon\left(a, \dot{s}_1\right) \qquad \Gamma, x : \epsilon\left(z, a\right) \vdash q : \epsilon\left(b\left[cons(z)\right], \dot{s}_2\right)}{\Gamma \vdash \lambda \mathsf{f}^1_{(6.2)},\ p \wr \lambda f^2_{(6.2)}(z, x),\ q : \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right)}\ z \notin FV(\Gamma, a, b),\ (s_1, s_2, s_3) \in \boldsymbol{R}$$

$$\overset{\vdash}{{}_{(6.3)}}\ \frac{\Gamma \vdash p : \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right) \qquad \Gamma, x : \epsilon\left(z, a\right) \vdash q : \epsilon\left(t\ z, b\left[cons(z)\right]\right)}{\Gamma \vdash \lambda \mathsf{f}^1_{(6.3)},\ p \wr \lambda f^2_{(6.3)}(z, x),\ q : \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right)}\ \begin{array}{l} z \notin FV(\Gamma, a, b, t), \\ (s_1, s_2, s_3) \in \boldsymbol{R} \end{array}$$

$$\overset{(6.2)1}{\vdash}\ \frac{\Gamma \vdash p : \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right)}{\Gamma \vdash p\ \mathsf{f}^1_{(6.2)} : \epsilon\left(a, \dot{s}_1\right)}$$

$$\overset{(6.2)2}{\vdash}\ \frac{\Gamma \vdash p : \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right) \qquad \Gamma \vdash q : \epsilon\left(u, a\right)}{\Gamma \vdash p\ f^2_{(6.2)}(u, q) : \epsilon\left(b\left[cons(u)\right], \dot{s}_2\right)}$$

$$\overset{(6.3)1}{\vdash}\ \frac{\Gamma \vdash p : \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right)}{\Gamma \vdash p\ \mathsf{f}^1_{(6.3)} : \epsilon\left(\dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right), \dot{s}_3\right)}$$

$$\overset{(6.3)2}{\vdash}\ \frac{\Gamma \vdash p : \epsilon\left(t, \dot{\pi}_{\langle s_1, s_2, s_3\rangle}\left(a, b\right)\right) \qquad \Gamma \vdash q : \epsilon\left(u, a\right)}{\Gamma \vdash p\ f^2_{(6.3)}(u, q) : \epsilon\left(t\ u, b\left[cons(u)\right]\right)}$$

FIG. 6.5 : Typage des termes de démonstration de la déduction surnaturelle correspondant à un système de type pur

de typage indiquées dans la figure 6.5. Notons toutefois qu'il existe en fait un constante $\mathsf{f}_{(6.1)}$ et une règle de typage correspondante pour chaque axiome $(s_1, s_2) \in \boldsymbol{A}$, de même pour les symboles de fonction $f^i_{(6.2)}$ et $f^i_{(6.3)}$ $(i \in \{1; 2\})$ dont il existe un exemplaire pour chaque règle $(s_1, s_2, s_3) \in \boldsymbol{R}$.

Notons que la démonstration du théorème 6.14 fournit une traduction des termes bien typés dans le système de type pur vers les termes de démonstration de la déduction surnaturelle.

Définition 6.4. *Un terme bien typé d'un système de type pur* $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ *peut être traduit en un terme de démonstration de la déduction surnaturelle pour* $\mathcal{PTS}_{(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})}$ *de la façon*

suivante :

$$
\begin{array}{rcll}
(\!|s_1|\!)_{s_2} & \stackrel{\text{déf}}{=} & \mathsf{f}_{(6.1)} & \text{pour } (s_1, s_2) \in \boldsymbol{A} \\
(\!|x|\!) & \stackrel{\text{déf}}{=} & x & \\
(\!|\lambda x : A,\ T|\!)_P & \stackrel{\text{déf}}{=} & \left(\lambda \mathsf{f}^1_{(6.3)},\ (\!|P|\!)\right) \wr \left(\lambda f^2_{(6.3)}(x,x),\ (\!|T|\!)\right) & \text{où } P \text{ est le type de } \lambda z : A,\ T \\
(\!|\Pi x : A,\ B|\!) & \stackrel{\text{déf}}{=} & \left(\lambda \mathsf{f}^1_{(6.2)},\ (\!|A|\!)\right) \wr \left(\lambda f^2_{(6.2)}(x,x),\ (\!|B|\!)\right) & \\
(\!|T\ U|\!) & \stackrel{\text{déf}}{=} & (\!|T|\!)\ f^2_{(6.3)}((\!|U|\!), (\!|U|\!)) &
\end{array}
$$

Proposition 6.17 (Correction)**.** *T possède le type P dans le contexte $x_1 : P_1, \ldots, x_n : P_n$ pour le système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ ssi $(\!|T|\!)$ est un terme de démonstration pour $\epsilon\left(|T|^{x_1:P_1,\ldots,x_n:P_n}, |P|^{x_1:P_1,\ldots,x_n:P_n}\right)$ dans le contexte $x_1 : \epsilon\left(x_1, |P_1|\right), \ldots, x_n : \epsilon\left(x_n, |P_n|^{x_1:P_1,\ldots,x_{n-1}:P_{n-1}}\right)$ en déduction surnaturelle pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$.*

Démonstration. Il s'agit essentiellement de reprendre la démonstration du théorème 6.14. □

Proposition 6.18 (Simulation)**.** *Pour tout terme T bien typé dans le système de type pur, si $T \underset{\beta}{\longrightarrow} U$ alors $(\!|T|\!) \underset{dis}{\longrightarrow} (\!|U|\!)$.*

Démonstration. Premièrement, remarquons que puisque T est typé, par réduction du sujet U l'est aussi et $(\!|U|\!)$ est bien défini.

Si $T \underset{\beta}{\longrightarrow} U$ alors il y a un β-radical $(\lambda x : A,\ B)C$ dans T à une certaine position. Par définition de la traduction, il y a donc un sous-terme de $(\!|T|\!)$ égal à $\left(\lambda \mathsf{f}^1_{(6.3)},\ (\!|P|\!) \wr \lambda f^2_{(6.3)}(x,x),\ (\!|B|\!)\right)\ f^2_{(6.3)}((\!|C|\!), (\!|C|\!))$ qui se réduit en une étape pour *dis* en $\{(\!|C|\!)/x\}(\!|B|\!)$. Il reste donc à démontrer que pour tous termes T et U typés dans système de type pur (et dont le type de U est le même que celui de $T_{|\mathfrak{p}}$) et toute position $\mathfrak{p}$ de T, on a $(\!|T[U]_{\mathfrak{p}}|\!) = (\!|T|\!)[(\!|U|\!)]_{\mathfrak{p}}$, ce qui se montre par induction sur la position $\mathfrak{p}$. Si elle est vide, le résultat est trivial. Sinon, on analyse les cas suivant la structure de T. Si $T = \lambda x : A,\ B$, alors si la position pointe dans A, $(\!|T|\!)$ dépend du type de A (plus précisément du type de $\lambda x : A,\ B$) et non de A lui même. Le fait de remplacer un sous-terme de A par un sous-terme de même type ne changera donc pas le type de A et de $\lambda x : A,\ B$, et donc $(\!|T[U]_{\mathfrak{p}}|\!)_P = (\!|T|\!)_P = (\!|T|\!)_P[(\!|U|\!)]_{\mathfrak{p}}$. Si la position pointe dans B, on applique l'hypothèse d'induction sur B et on obtient que

$$
\begin{aligned}
(\!|(\lambda x : A,\ B)[U]_{2\mathfrak{p}}|\!)_P &= (\!|\lambda x : A,\ (B[U]_{\mathfrak{p}})|\!)_P \\
&= \lambda \mathsf{f}^1_{(6.3)},\ (\!|P|\!) \wr \lambda f^2_{(6.3)}(x,x),\ (\!|B[U]_{\mathfrak{p}}|\!) \\
&= \lambda \mathsf{f}^1_{(6.3)},\ (\!|P|\!) \wr \lambda f^2_{(6.3)}(x,x),\ (\!|B|\!)\,[(\!|U|\!)]_{\mathfrak{p}} \quad \text{(hypothèse d'induction)} \\
&= (\!|\lambda x : A,\ B|\!)_P[(\!|U|\!)]_{2\mathfrak{p}}
\end{aligned}
$$

Les autres cas sont similaires à ce dernier. □

Corollaire 6.19. *Si toutes les démonstrations en déduction surnaturelle pour* $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ *de conséquence de la forme* $|\Gamma| \vdash \epsilon\,(a, b)$ *avec* Γ *bien formé normalisent fortement, alors le système de type pur* $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ *est fortement normalisant.*

Ce corollaire associé au théorème 3.22 montre que l'on pourrait donc utiliser les critères de normalisation forte de la déduction modulo pour démontrer qu'un système de type pur est fortement normalisant. Néanmoins, comme nous allons le démontrer, la déduction surnaturelle associée à un système de type pur n'est jamais fortement normalisante. On pourrait essayer d'adapter les critères pour les restreindre aux démonstrations dans des contextes bien formés, mais il semble toutefois que ceci serait d'un intérêt assez faible :

- On ne peut pas appliquer le critère d'un ordre bien fondé qui inclus l'ordre sous-formule et la relation de réécriture, car $\epsilon\left(\dot{\pi}_{\langle s_1,s_2,s_3\rangle}\,(\dot{s}_3, b)\,, \dot{s}_3\right)$ est strictement plus petit pour l'ordre sous-formule que $\epsilon\,(\dot{s}_3, \dot{s}_1) \wedge \forall z.\ \epsilon\,(z, \dot{s}_3) \Rightarrow \epsilon\,(b\,[\mathit{cons}(z)]\,, \dot{s}_2)$ alors que le premier se réécrit en le second par (6.2).
- La technique principale pour montrer qu'un système de type pur est fortement normalisant repose sur les candidats de réductibilité. L'utilisation de pré-modèles, qui interprètent les formules par des candidats de réductibilité, ne semble donc pas a priori permettre de démontrer de nouveaux résultats de forte normalisation. Il serait toutefois intéressant de regarder s'il est possible d'utiliser les candidats de réductibilité démontrant la forte normalisation d'un système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ pour obtenir la normalisation forte de la déduction modulo $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$.
- La supercohérence est bien adaptée aux cas où on peut définir l'interprétation des formules en utilisant des points fixes. Ici, il semble difficile de déterminer la sémantique de ϵ grâce à un point fixe.

On peut chercher à démontrer un résultat similaire pour l'autre direction, c'est-à-dire montrer que la normalisation forte du système de type pur implique celle de la déduction surnaturelle associée. Ce résultat est faux dans le cas général, car on peut rajouter des axiomes dans le contexte qui permettront de construire des termes non normalisants.

Proposition 6.20. *Étant donné un système de type pur* $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ *avec* $\boldsymbol{R} \neq \emptyset$*, la déduction surnaturelle pour* $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ *n'est pas fortement normalisante, même si le système de type pur l'est.*

Démonstration. Le contre-exemple de Jean-Yves Girard (1972) nous dit qu'il existe un terme T qui ne normalise pas et est typé par $\Pi x : *,\ x$ dans le système de type pur $\lambda*$. Soit (s_1, s_2, s_3) une des règles du système de type pur. On considère le contexte en déduction surnaturelle suivant : $\Gamma \stackrel{\text{déf}}{=} \forall x,\ \epsilon\,(x, \dot{s}_1) \Leftrightarrow \epsilon\,(x, \dot{s}_2)\,, \forall x,\ \epsilon\,(x, \dot{s}_1) \Leftrightarrow \epsilon\,(x, \dot{s}_3)\,, \epsilon\,(\dot{s}_3, \dot{s}_3)$, qui n'est pas la traduction d'un contexte bien formé. On peut alors traduire toute dérivation de $\Theta \vdash A : B$ de $\lambda*$ par une démonstration de $\Gamma, \{\dot{s}_3/\dot{*}, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}/\dot{\pi}_{\langle *,*,*\rangle}\}|\Theta| \vdash \epsilon\left(\{\dot{s}_3/\dot{*}, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}/\dot{\pi}_{\langle *,*,*\rangle}\}\,|A|^{\Theta}\,, \{\dot{s}_3/\dot{*}, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}/\dot{\pi}_{\langle *,*,*\rangle}\}\,|B|^{\Theta}\right)$

en déduction surnaturelle pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ (les traduction étant par contre effectuées pour $\mathcal{PTS}_{\lambda *}$), en suivant la démonstration du théorème 6.14, en remplaçant les axiomes manquant par l'hypothèse $\epsilon\left(\dot{s}_3, \dot{s}_3\right)$ et en utilisant les équivalences dans les cas où une sorte ne convient pas. Comme pour la proposition 6.18, un β-radical est alors traduit par un ρ-radical, et la traduction du réduit est le réduit de la traduction. Par conséquent, comme T ne normalise pas, la démonstration de $\Gamma \vdash \epsilon\left(\left\{\dot{s}_3/\dot{*}, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}/\dot{\pi}_{\langle *,*,*\rangle}\right\}|T|\,, \dot{\pi}_{\langle s_1,s_2,s_3\rangle}\left(\dot{s}_3, 1\right)\right)$ ne normalise donc pas non plus, quelque soit la réduction. □

Comme on le voit, ce résultat repose sur le fait qu'il est possible de rajouter des axiomes pour retomber dans $\lambda *$. Dans la suite, nous allons donc nous restreindre aux démonstrations effectuées dans la traduction d'un contexte bien formé. Nous conjecturons qu'un système de type pur est fortement normalisant ssi toutes les démonstration en déduction surnaturelle pour un tel contexte normalisent fortement.

Conjecture 6.1.
Tous les λ-termes typés dans un système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ normalisent fortement ssi toutes les démonstrations en déduction surnaturelle pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ de séquences de la forme $|\Gamma| \vdash \epsilon\,(a, b)$ (où Γ est un contexte bien formé) normalisent fortement .

Le corollaire 6.19 montre la direction « si », car seuls les démonstrations dont la conséquence est de la forme ci-dessus correspondent à des traduction de dérivation dans le système de type pur.

Pour l'autre direction, remarquons tout d'abord que le théorème 6.15 fournit une traduction des termes de démonstration pour la déduction surnaturelle pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ vers des λ-termes bien typés dans le système de type pur $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$. Toutefois, dans cette traduction, tous les ρ-radicaux ne sont pas traduits par des β-radicaux. En particulier, les ρ-radicaux pour la règle (6.2) (de la forme $(\lambda \mathsf{f}^1_{(6.2)},\ t \wr \lambda f^2_{(6.2)}(z, x),\ u)\ f^i_{(6.2)}(\overrightarrow{t})$) sont en quelque sorte réduits avant d'être traduit. Par conséquent on ne peut pas démontrer un résultat analogue à celui de la proposition 6.18. Notons toutefois que les ρ-radicaux pour la règle (6.3) sont traduits par des β-radicaux, ce qui veut dire que si le système de type pur est fortement normalisant, alors la non-normalisation de la déduction surnaturelle devrait faire intervenir au moins une réduction pour (6.2).

Pour chercher un contre-exemple à cette conjecture, si on se place dans le calcul des constructions, $P \overset{\text{déf}}{=} \epsilon\left(\dot{\pi}_{\langle \square,*,*\rangle}\left(\dot{*}, 1\right), \dot{*}\right)$ semble être un bon candidat de formule pour trouver une démonstration non fortement normalisante de la forme attendue. En effet cette formule se réécrit en $\epsilon\left(\dot{*}, \square\right) \wedge \forall z,\ \epsilon\left(z, \dot{*}\right) \Rightarrow \epsilon\left(z, \dot{*}\right)$, qui est facilement démontrable. Malgré tout, P est une sous-formule de cette dernière, et on pourrait donc chercher à obtenir une démonstration non-normalisante à la manière de $A(\mathsf{r}) \longrightarrow \forall z,\ z = \mathsf{r} \Rightarrow A(z) \Rightarrow B$. Néanmoins, nous n'avons pas trouvé de moyen de forcer l'égalité de z avec une terme de la forme $\dot{\pi}_{\langle \square,*,*\rangle}\left(\dot{*}, u\right)$ de manière à pouvoir réappliquer la règle (6.2).

C'est l'absence apparente de contre-exemple qui nous pousse à croire que la normalisation du système de type pur entraîne celle de la déduction surnaturelle correspondante

pour les démonstrations dans la traduction d'un contexte bien formé.

En revanche, il est possible de démontrer que la normalisation d'un système de type pur entraîne que la déduction surnaturelle correspondante admet les coupures dans les démonstrations dans la traduction d'un contexte bien formé.

Proposition 6.21. *Si un système de type pur* $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ *est faiblement normalisant, alors pour tout contexte bien formé* Γ*, pour tous termes* a *et* b*, la séquence* $|\Gamma| \vdash \epsilon(a, b)$ *possède une démonstration en calcul des séquences extensible pour* $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ *ssi elle est possède une sans utiliser* $\overset{\frown}{\vdash}$*.*

Démonstration. Supposons que $|\Gamma| \vdash \epsilon(a, b)$ possède une démonstration en déduction surnaturelle pour $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$. Par le théorème 6.15, il existe des termes A et B tels que $\Gamma \vdash A : B$ est dérivable dans le système de type pur, $a \xrightarrow{*} |A|$ et $b \xrightarrow{*} |B|$. Comme le système de type pur est faiblement normalisant, A et B possèdent une forme normale A' et B'. Par réduction du sujet et en appliquant le lemme 3.4, $\Gamma \vdash A' : B'$ est dérivable dans le système de type pur. Par le théorème 6.14, $|\Gamma| \vdash \epsilon(|A'|, |B'|)$ possède une démonstration en déduction surnaturelle. De plus, cette démonstration contient exactement les mêmes radicaux que la dérivation dont elle est issue, par conséquent aucun. Comme $a \xrightarrow{*} |A'|$ et $b \xrightarrow{*} |B'|$, c'est également une démonstration de $|\Gamma| \vdash \epsilon(a, b)$. Si on traduit cette démonstration en calcul des séquences extensible, elle ne contiendra donc pas de $\overset{\frown}{\vdash}$. □

Réciproquement, dans le cas où le système de type pur n'est pas fortement normalisant, le corollaire 6.19 montre que la déduction surnaturelle ne normalise pas non plus. Toutefois, le calcul des séquences modulo $\mathcal{PTS}_{(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})}$ pourrait admettre les coupures tout de même. Ce n'est pas toujours le cas. Si on se place dans $\lambda*$, le contre-exemple de Jean-Yves Girard (1972), la proposition 3.24 et le théorème 6.14 montrent qu'il existe un terme t tel qu'on peut démontrer $\vdash \epsilon\left(t, \dot{\pi}_{\langle *,*,*\rangle}(\dot{*}, 1)\right)$ en calcul des séquences extensibles pour $\mathcal{PTS}_{\lambda*}$. Toutefois, si on cherche à obtenir une démonstration sans coupure on obtient la dérivation

$$\dfrac{\dfrac{\vdash_{(6.1)} \dfrac{}{\vdash \epsilon(\dot{*}, \dot{*})} \quad \overset{\frown}{\vdash} \dfrac{}{\epsilon(z, \dot{*}) \vdash \epsilon(1\,[cons(z)], \dot{*})}}{\vdash_{(6.2)} \vdash \epsilon\left(\dot{\pi}_{\langle *,*,*\rangle}(\dot{*}, 1), \dot{*}\right)} \quad \overset{?}{\epsilon(z, \dot{*}) \vdash \epsilon(t\ z, z)}}{\vdash_{(6.3)} \vdash \epsilon\left(t, \dot{\pi}_{\langle *,*,*\rangle}(\dot{*}, 1)\right)}$$

Or on ne peut appliquer aucune règle à part $\overset{\frown}{\vdash}$ à $\epsilon(z, \dot{*}) \vdash \epsilon(t\ z, z)$. Par conséquent il n'existe pas de démonstration sans coupure de $\vdash \epsilon\left(t, \dot{\pi}_{\langle *,*,*\rangle}(\dot{*}, 1)\right)$ et $\mathcal{PTS}_{\lambda*}$ n'admet pas les coupures. Il existe donc des cas où le système n'admet pas les coupures, y compris pour les démonstrations dans la traduction d'un contexte bien formé.

On peut résumer les relations entre système de type pur et déduction surnaturelle à propos de normalisation grâce au schéma de la figure 6.6.

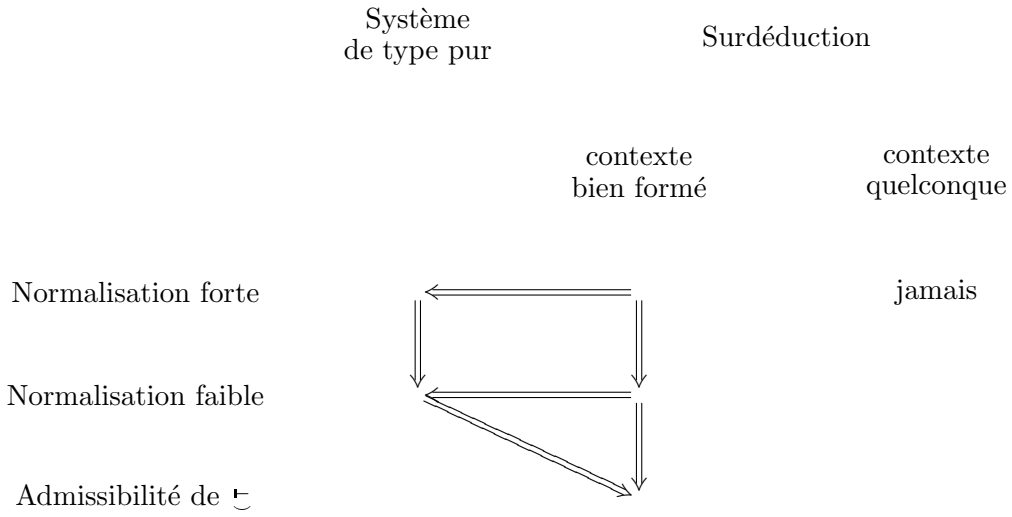


FIG. 6.6 : Schéma récapitulatif des relations entre normalisation dans un système de type pur et dans les systèmes de surdéduction correspondants

Chapitre 7

Conclusion

> S'il est quelquefois logique de s'en rapporter à l'apparence des phénomènes, ce premier chant finit ici.
>
> Comte de Lautréamont, *Les Chants de Maldoror*

Nous avons étudiés différents critères qui permettent d'obtenir des démonstrations plus simplement, en s'appuyant principalement sur la déduction modulo et le formalisme voisin de la surdéduction. Le premier critère est l'admissibilité des coupures, et nous avons vu que, même si cette propriété est indécidable en général, on peut utiliser une généralisation de la procédure de complétion standard de Donald Knuth et Peter Bendix (1970) pour modifier un système afin qu'au final il admette les coupures. Le deuxième critère est la longueur des démonstrations, et nous avons vu comment la déduction modulo permet de réduire celle-ci. Cette réduction permet de mieux comprendre quels phénomènes entraînent les réductions de longueurs des démonstrations quand on augmente l'ordre en arithmétique. Enfin nous avons cherché à savoir comment produire des démonstrations à la fois simples du point de vue de l'implémentation du système de déduction mais aussi simples pour son utilisateur, en regardant comment des logiques d'ordre supérieur comme les systèmes de type purs peuvent être encodés dans une logique du premier ordre. Ces recherches éclairent d'un jour nouveau la notion de bonne démonstrations et ouvrent des perspectives que nous relevons maintenant.

7.1 Vers d'autres critères de simplicité

Les critères de simplicité que nous avons étudiés ont été justifiés, mais ce ne sont pas les seuls qui peuvent être utiles. La première question qui vient à l'esprit est de savoir quels autres critères sont pertinents. À partir de l'admissibilité des coupures, on est naturellement amené à se pencher sur la normalisation. Certes, du point de vue de la démonstration automatique, la plupart des bonnes propriétés des systèmes de déduction, comme l'analycité ou la propriété du témoin, sont des conséquences de l'admissibilité des coupures et ne requièrent pas la normalisation. Néanmoins, la normalisation peut

être utile pour la vérification des démonstrations : il est souvent plus facile d'écrire un programme qui vérifie uniquement les démonstrations en forme normale. Alors, à partir d'une démonstration quelconque, il s'agit de la normaliser et de vérifier la démonstration ainsi obtenue. Si on n'a que l'admissibilité des coupures, cela n'est pas toujours possible, même pour des démonstrations correctes, et on est obligé de reconstruire une nouvelle démonstration à partir de zéro si on veut avoir une démonstration sans coupure. De plus, grâce à l'isomorphisme de Curry-Howard, la normalisation forte d'un système permet de garantir la terminaison des programmes qui y sont typés correctement. Il faudrait donc regarder dans quel mesure notre étude sur l'admissibilité des coupures peut être étendue à celle de la normalisation, en particulier, s'il est possible d'instancier les systèmes canoniques abstraits différemment de la façon proposée au chapitre 4 pour obtenir une procédure de complétion qui permet de transformer un système pour qu'au final il normalise fortement. Une autre approche pour résoudre ceci serait de rajouter des règles de simplifications à la procédure de complétion que nous avons proposée. En effet, si le système obtenu par complétion ne normalise pas, c'est en partie parce qu'il est toujours possible de construire les démonstrations non normalisantes présentent au départ, étant donné qu'on a uniquement ajouté des règles. Si on prend l'exemple du système de Crabbé, au final on a le système $A \to^- B \wedge \neg A; A \to^+ B \wedge \neg A; B \to^- \bot$. Il est donc toujours possible de construire une démonstration non normalisante de $B \vdash$ en coupant autour de A. La complétion standard de Donald Knuth et Peter Bendix (1970) contient la règle de simplification $\{s \to t; u[s]_\mathfrak{p} \to v\} \rightsquigarrow \{s \to t; u[t]_\mathfrak{p} = v\}$. Si on applique naïvement cette règle dans le système ci-dessus, on obtient $A \to^- \bot \wedge \neg A; A \to^+ B \wedge \neg A; B \to^- \bot$ et on pourrait envisager d'ajouter des règles de simplification « logiques » qui conduiraient au système $A \to^- \bot; A \to^+ B \wedge \neg A; B \to^- \bot$ dans lequel il ne serait plus possible de construire une démonstration non normalisante. Il nous faut donc comprendre dans quelle mesure ce cas particulier est une exception ou peut être généralisé.

Un autre critère important en ce qui concerne la déduction modulo est la décidabilité de la congruence engendrée par le système de réécriture modulo lequel on travaille. En effet, c'est une condition nécessaire pour la décidabilité de la vérification des démonstrations, à moins que l'on détaille à chaque étape de la démonstration les étapes de réécriture qui la justifient (par exemple via des règles de conversion étape par étape explicites), ce qui fait perdre l'intérêt de travailler modulo, en particulier en ce qui concerne la longueur des démonstrations. Une façon d'assurer cette décidabilité est d'utiliser un système de réécriture confluent et terminant. Le système produit par notre procédure de complétion pour retrouver l'admissibilité des coupures n'est en général ni confluent ni terminant, en particulier si on utilise le théorème 3.10. Il faudrait donc ici aussi chercher à s'assurer que la congruence soit décidable après que le système de réécriture a été complété. Si on veut rentrer dans le cadre de systèmes de déduction de Stephen Cook et Robert Reckhow (1979), il faut même que les démonstrations soient vérifiable en temps polynomial, ce qui implique que la congruence doit être décidable en temps polynomial aussi. Pour cela, même avec un système non confluent, on pourrait utiliser des techniques

étudiées à partir des années 1970 pour s'assurer de l'équivalence de deux termes. Une approche pourrait consister à utiliser des techniques de fermeture de congruence (*congruence closure*, Greg Nelson et Derek Oppen, 1980), dont certaines sont très efficaces (Rakesh Verma, 1993), ou de complétion close (Wayne Snyder, 1989).

7.2 Vers un cadre logique unique et mécanisé

7.2.1 Automatisation des théories

Au cours de la section 4.3.1 nous avons proposé un algorithme qui permet de transformer toute présentation d'une théorie en logique du premier ordre en un système de réécriture compatible avec cette théorie (corollaire 4.10). En appliquant ensuite la procédure de complétion du chapitre 4 sur ce système, ceci permet d'obtenir de façon automatique un système de démonstration analytique pour la théorie de départ. Jusqu'alors, les systèmes correspondant à des théories utilisés en déduction modulo, comme par exemple celui pour l'arithmétique (Gilles Dowek et Benjamin Werner, 2005) ou pour la théorie des ensembles (Gilles Dowek et Alexandre Miquel, 2007), ont été exhibés à la main. L'idée serait alors de développer une méthode automatique qui transforme une théorie en un système de réécriture compatible et ayant toutes les bonnes propriétés attendues, à savoir admissibilité des coupures voire normalisation forte, décidabilité de la congruence voire vérifiabilité des démonstrations en temps polynomial, etc. Outre l'extension de nos approches concernant l'admissibilité des coupures à ces autres propriétés, plusieurs tâches doivent être accomplies à cet effet.

Premièrement, la plupart des théories ne sont pas présentées sous la forme d'un nombre fini d'axiomes de la logique du premier ordre, mais via des schémas d'axiomes voire par l'utilisation de logiques d'ordre supérieur. Le travail de Florent Kirchner (2006) résout la présence de schéma d'axiomes, mais possède l'inconvénient de faire travailler dans une extension, certes conservative, de la théorie initiale, en particulier via l'utilisation d'une logique multisortée. L'impossibilité de trouver une axiomatisation finie de l'arithmétique du premier ordre sans utiliser d'extension nous montre toutefois qu'il ne semble pas possible de faire autrement. Notre travail apporte également un début de solution à l'utilisation de logiques d'ordre supérieur, mais ouvre de nouvelles perspectives.

Deuxièmement, l'algorithme que nous donnons à la section 4.3.1 est valable pour la logique *classique* du premier ordre, mais pas en logique intuitionniste, comme souligné dans la remarque 4.1. Cette remarque indique aussi que si on veut que le système ait la propriété du témoin ssi la théorie initiale l'a, alors en général on ne sera pas capable de trouver un système de réécriture compatible. Néanmoins, le simple fait d'utiliser des extensions, en particulier via la skolémisation, peut donner la propriété du témoin à une théorie qui ne l'avait pas. On peut donc se demander si le respect de la propriété du témoin est vraiment importante. De plus, il existe des théories pour lesquelles il est possible de trouver une présentation compatible, ne serait-ce que l'arithmétique. Il nous

faut alors chercher à caractériser quelles classes de théories peuvent être transformées en un système de réécriture compatible pour la logique intuitionniste.

7.2.2 Encodage et combinaison de systèmes de déduction

Par ailleurs, ce travail devra être étendu à d'autres objets que des théories. En particulier, dans le chapitre précédent, nous avons cherché à montrer comment des systèmes d'inférence peuvent être encodés en déduction modulo, ou plus précisément en surdéduction. La méthodologie que nous avons proposée est générale et peut être appliquée aussi bien au calcul des séquences pour la logique d'ordre supérieur basée sur la théorie simple des types qu'aux systèmes de type purs fonctionnels. Elle reste toutefois très informelle, et il nous faut chercher à la mécaniser le plus possible, ainsi qu'à garantir automatiquement que le système de surdéduction obtenu au final est bien adéquat avec le système de départ, y compris par rapport à des propriétés concernant la réduction des démonstrations (par exemple l'équivalence de la normalisation forte dans le cas des systèmes de type purs). Pour progresser dans cette approche, nous pourrons entre autres nous intéresser aux travaux d'Agata Ciabattoni, Nikolaos Galatos et Kazushige Terui (2008) qui ont montré, dans un cadre particulier, une relation entre systèmes d'inférence et théories. L'idée serait alors d'appliquer les méthodes développées pour l'encodage de théories, que nous avons discutées ci-dessus, à la théorie correspondant au système d'inférence.

L'utilisation de la déduction modulo et de la surdéduction comme cadre logique permet également d'envisager de les utiliser pour le développement de démonstrations hétérogènes. En effet, pour démontrer des spécifications complexes, on est amené à utiliser des méthodes de démonstrations différentes. Par exemple, si on cherche à vérifier le système informatique d'un avion, on utilisera par exemple des outils de vérification par modèles (*model checking*) pour certifier les parties matérielles, mais on peut envisager que certains aspects logiciels élaborés ait besoin d'être vérifiés à l'aide d'assistants interactifs à la démonstrations. La question est donc de savoir comment recoller logiquement ces différents éléments qui ne sont, pris individuellement, que des démonstrations partielles de la correction du tout. En outre, les méthodes de démonstrations automatiques font de plus en plus appel à des procédures spécialisées pour traiter certains aspects des démonstrations. On peut par exemple citer les procédures de satisfaisabilité modulo théorie. Pour démontrer des formules faisant intervenir par exemple certaines formules de l'arithmétique linéaire, on applique d'abord une procédure de satisfaisabilité pour la logique propositionnelle pour trouver un modèle de la formule en dehors de la théorie de l'arithmétique. Puis on utilise une procédure spécifique aux équations linéaires pour vérifier que le modèle produit est valide en arithmétique. Si ce n'est pas le cas, on réutilise la procédure de satisfaisabilité propositionnelle pour obtenir un autre modèle qu'on vérifie ensuite avec la procédure spécialisée, etc. La plupart des logiciels qui utilisent ces méthodes se contentent d'une réponse oui/non. À moins de fournir le modèle comme ré-

ponse, si on veut produire une démonstration, il va falloir combiner les différents aspects. Dans ces différents cas, on peut penser à utiliser la déduction modulo ou la surdéduction pour fournir le cadre dans lequel les différentes parties des démonstrations sont réunies. De ce fait, on pourrait combiner l'efficacité de l'utilisation de procédures spécialisées dans un aspect particulier de la démonstration et l'existence d'un formalisme unique pour vérifier les démonstrations ainsi obtenues. Il reste bien entendu à voir comment les différentes parties des démonstrations peuvent être combinées, autrement dit, il nous faut étudier comment obtenir de la modularité en déduction modulo.

Un bon exemple pour étudier ces aspects d'interaction entre différentes parties de démonstrations est l'utilisation de notre encodage des systèmes de types purs en surdéduction. Les assistants interactifs à la démonstration que nous avons cités au cours de ce travail ne représentent qu'un petit nombre de ceux existants. De nombreux développements ont été effectués pour chacun d'entre eux. On aimerait alors par exemple être en mesure d'utiliser la formalisation d'une théorie faite dans un assistant pour effectuer un développement dans un autre assistant. La première approche pour faire cela est de trouver des traductions entre les différents systèmes de démonstrations qui sont à la base des assistants. Cela nécessite la recherche d'une traduction pour chaque couple d'assistants. Comme indiqué dans l'introduction, une autre méthode a été proposée par Florent Kirchner et Claudio Sacerdoti Coen via leur logiciel Fellowship[1]. L'idée est d'effectuer les développements des théories que l'on veut partager entre les différents assistants dans un système très simple, à savoir le calcul des séquences pour la logique du premier ordre. Il est alors facile de traduire ces développements dans les différents assistants, ce qui est possible pour l'instant vers Coq et PVS. Le travail de Denis Cousineau et Gilles Dowek (2007) fournit une approche inverse : en effet, les développements sont alors effectués dans les différents assistants mais peuvent être vérifiés par un outil unique, appelé Europa, basé sur le système $\lambda\Pi$ modulo. L'utilisation de la surdéduction pour encoder les systèmes de type purs, comme nous l'avons montrée dans le chapitre précédent, permet de combiner ces deux aspects. En effet, en plus de fournir un système permettant de vérifier les dérivations de tous les systèmes de types pur, on peut, en utilisant une théorie compatible, transformer ces dérivations en démonstrations de la logique du premier ordre, et utiliser Fellowship pour les utiliser dans d'autres assistants à la démonstrations. La figure 7.1 illustre ces propos.

Un autre aspect à étudier concernant la coopération des assistants à la démonstration grâce à la surdéduction provient du fait que si on fait passer une démonstration d'un système de type pur, par exemple le calcul des constructions, à un autre, par exemple λHOL, en passant par la déduction surnaturelle et le calcul des séquences pour la logique du premier ordre, on a alors un encodage de la démonstration, qui démontre une formule contenant des prédicats ϵ, et non pas la démonstration elle-même. Il serait intéressant de regarder sous quelles conditions et de quelle manière il serait possible de proposer une

[1] http://www.lix.polytechnique.fr/Labo/Florent.Kirchner/fellowship/trunk/

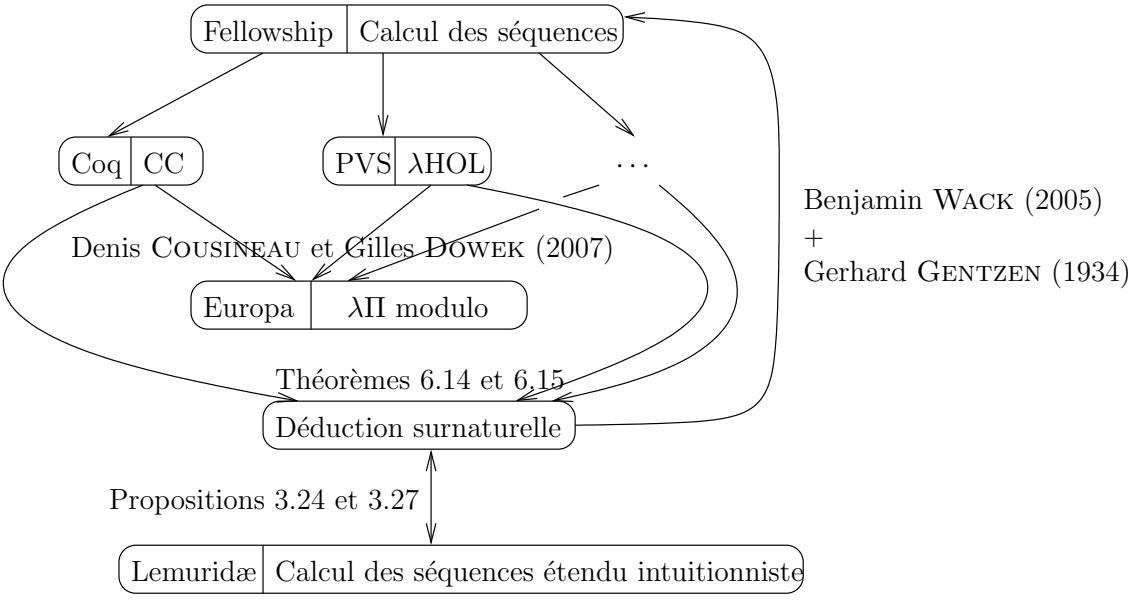


Fig. 7.1 : Schéma illustrant des approches pour faire coopérer différents assistants à la démonstration

traduction inverse correcte qui transforme les prédicats $\epsilon\,(t, p)$ en formules P du système de type pur, de façon à pouvoir se passer de l'encodage. Il faudrait également chercher dans quelle mesure il serait possible de donner un encodage plus léger des systèmes de types purs.

Toutefois, si les systèmes de type purs sont à la base des principaux assistants à la démonstration, leur encodage ne suffit pas, car les systèmes d'inférence des assistants sont en général plus riches que les assistants. Par exemple, Coq repose en fait sur le calcul des constructions *inductives*. Un des principaux objectifs est donc de rechercher comment encoder les types inductifs en déduction modulo. Comme la réécriture est dans un certain sens plus générale que les types inductifs, nous pensons donc que cela est possible. Frédéric Blanqui (2005) montre d'ailleurs que le calculs des constructions inductives peut être simulé par le calcul des constructions algébriques qui est, rappelons le, une extension du calcul des constructions avec une règle de conversion incluant des règles de réécriture en plus de la β-conversion. Toutefois, pour rentrer dans ce cadre (en particulier pour préserver la normalisation forte), Frédéric Blanqui (2005) utilise un nombre infini de règles de réécriture. Il est à noter que cette étude sera également utile pour l'automatisation de la transformation de théories en systèmes de réécriture compatibles car si on regarde l'arithmétique, c'est le schéma de récurrence qui est le plus dur à transformer, à partir du moment où on veut avoir la normalisation forte. On

pourra à ce propos s'intéresser aux travaux de Lisa Allali et Paul Brauner (2008) qui montrent comment les termes inductifs apparaissent naturellement si on applique la surdéduction à l'équation définissant un type inductif.

Un dernier aspect concernant l'utilisation de la déduction modulo ou de la surdéduction comme cadre logique a trait aux représentations de démonstrations qui ne sont pas basées sur des systèmes d'inférences. En effet, il apparaît de plus en plus qu'une des façons les plus intéressantes de représenter une démonstration est d'utiliser un graphe, voire un polygraphe (Albert Burroni, 1993), plutôt qu'un arbre d'inférence, de façon à limiter certains aspects dit de bureaucratie entre démonstrations, certaines démonstrations du calcul des séquences étant par exemple essentiellement les mêmes à une certaine permutation de règles près. C'est ainsi qu'ont été introduits les réseaux de démonstrations pour la logique linéaire (Jean-Yves Girard, 1987), qui ont été étendus à d'autres logiques et qui ont également été revisités par le calcul des structures (Lutz Strassburger, 2005*a*) et forment ainsi des polygraphes de dimension 3 (Yves Guiraud, 2006). On peut même trouver des formalismes qui proposent de construire des démonstrations sans syntaxe (Dominic Hughes, 2006). On peut donc se demander dans quelle mesure de tels systèmes de déduction pourraient être représentés en déduction modulo. Il nous semble que cette tâche devrait être plus facile que si on devait les encoder dans un cadre logique plus usuel comme ELF, puisqu'il sera possible d'utiliser le système de réécriture pour exprimer les réseaux comme des termes modulo. Notre encodage du calcul des structures dans la section 5.2.2 est d'ailleurs une première étape dans cette direction.

7.3 Un cadre théorique pour la complexité des démonstrations

Au cours de notre travail, nous avons rencontrés différents formalismes permettant de discuter de la complexité des démonstrations. Tout d'abord, il y a le cadre des systèmes canoniques abstraits, qui traduit cette complexité par un ordre bien fondé sur les démonstrations. Nous avons vu qu'il est bien adapté pour modéliser des systèmes de déduction comme ceux pour la logique équationnelle et la déduction modulo. D'autre part, il y a les travaux de Stephen Cook et Robert Reckhow (1979) qui s'intéressent uniquement à la longueur des démonstrations en logique propositionnelle du premier ordre mais qui relient celle-ci à la théorie de la complexité et permettent de comparer des systèmes de déduction entre eux. Il sera intéressant de formaliser un cadre abstrait qui combine ces deux aspects et qui fournisse des outils théoriques à des questions de complexité. Nous pensons notamment à un cadre qui soit l'équivalent vis-à-vis de la complexité des démonstrations de ce que sont les systèmes acceptables de programmation (voir Michael Machtey et Paul Young, 1978) pour la théorie de la complexité. De la même façon que les systèmes acceptables de programmation permettent d'unifier la notion de complexité d'un programme, on aurait alors une vision plus claire de ce qu'est la complexité d'une démonstration. Ce nouveau cadre permettrait peut-être également

d'envisager sous un angle nouveau les diverses conjectures apparues suite aux travaux de Stephen Cook et Robert Reckhow (1979).

Une de ces conjectures est qu'on peut réduire exponentiellement la taille de démonstration d'un système schématique si on ajoute ce qu'on appelle des extensions de Gregory Tseytin (1968). Il s'agit d'ajouter des axiomes de la formes $A_i \Leftrightarrow P_i$ où A_i n'apparaît ni dans la conclusion ni dans les hypothèses de la démonstration en cours, ni dans P_i ni dans les axiomes ajoutés pour $j < i$. Il a été démontré que ceci est polynomialement équivalent au fait de pouvoir substituer une formule atomique par une formule à un endroit quelconque dans la démonstration, néanmoins on ne sait toujours pas si on obtient ainsi des démonstrations vraiment plus courtes que dans les systèmes schématiques sans extensions. La forme des axiomes de Tseytin suggère que la déduction modulo pourrait être utilisée pour avoir un regard neuf sur ces questions. D'ailleurs, Lutz Strassburger (2008) a montré comment ces axiomes peuvent être transformés en des règles qui ressemblent beaucoup aux règles de conversion explicite de la déduction modulo.

D'autre part, notre recherche sur la longueur des démonstrations en arithmétique d'ordre supérieur suggère une nouvelle approche de la complexité inhérente à une formule arithmétique. De façon standard, cette complexité est définie comme le couple constitué de l'ordre du langage de la formule et du nombre d'alternance de quantificateurs qui apparaissent dans la formules. Si on prend une formule démontrable à l'ordre i, sa longueur ne pourra que diminuer au fur et à mesure qu'on augmente l'ordre, jusqu'à atteindre un palier. On pourrait alors définir des notions d'*ordre de complexité minimale* d'une formule, qui correspondrait à l'ordre auquel on doit se placer pour pouvoir construire une démonstration de la formule de longueur minimale, ainsi que la *complexité implicite* d'une formule, qui correspond à cette taille minimale. Il reste à déterminer dans quel mesure ces notions sont pertinentes, et si on peut les lier à d'autres notions de complexité implicites qui permettent de connaître la classe de complexité de la fonction calculée par un programme (et pas forcément celle du programme) en regardant son type qui correspond, par la correspondance de Curry-Howard, à une formule.

En outre, la démonstration que nous avons faite de l'indécidabilité de l'admissibilité des coupures en déduction modulo (théorème 3.14) suggère que ce problème est dans la même classe d'indécidabilité que celui de la validité en logique du premier ordre. Il conviendrait de démontrer cette affirmation et de caractériser les classes de systèmes de réécriture pour lesquelles l'admissibilité est décidable.

7.4 Mise en œuvre

Le prototype de méthode des tableaux modulo qui nous avons implémenté (cf. section 4.4.2) nous a conduit à nous interroger sur la conception d'outils de démonstration automatique efficaces basés sur la déduction modulo. En particulier, l'utilisation de ce prototype nous pousse à considérer l'utilisation de stratégies dans de tels outils. Ainsi,

si on cherche une démonstration modulo le système encodant un système de type pur, comme on utilise de la surréduction, un grand nombre de règles de réécriture est applicable à chaque étape. La méthode proposée par Richard Bonichon et Olivier Hermant (2006*b*) consiste à appliquer chacune de ces règles l'une après l'autre, ce qui possède l'avantage de pouvoir facilement montrer que la méthode est complète, mais ce qui oblige à considérer des cas inutiles. Plutôt que cela, il faudrait pouvoir déterminer quelle règle utiliser à quel moment tout en restant complet. Ceci est d'autant plus important dans le cas où on travaille modulo λ_σ puisque Paul-André Melliès (1995) a démontré que suivant la stratégie d'évaluation, la traduction de certains termes fortement normalisants peut être réduite indéfiniment. Il faudrait également déterminer dans quelle mesure il est possible de déterminer automatiquement une stratégie d'application des règles de réécriture efficace et par quels moyens il faut permettre à l'utilisateur de renseigner la stratégie qu'il veut employer, par exemple en utilisant une algèbre de stratégies comme dans ELAN[2], Stratego[3] ou TOM, ou bien en utilisant une monade comme l'ont proposé Florent Kirchner et César Muñoz (2006).

D'autre part, le calcul des séquences étendu offre naturellement une méthode des tableaux associée, comme l'a remarqué Clément Houtmann (2008). Néanmoins, ici aussi, il reste à déterminer la stratégie d'application des règles de cette méthode. En effet, dans le cas du calcul des séquences usuel, cette stratégie est déterminée par des règles de bon sens : on applique d'abord les règles qui ne dupliquent pas les branches, et qui ne créent pas de nouvelles métavariables. On applique donc dans l'ordre α puis δ, β et enfin γ (cf. section 3.2.4). Dans le cas de la surdéduction, il va falloir déterminer dans quel ordre les surrègles seront appliquées, en utilisant des heuristiques ou en se basant sur des résultats plus théoriques. Ces recherches pourront alors être appliquées en particulier pour développer des méthodes efficaces de recherche de démonstration dans les systèmes de type purs (via l'encodage que nous avons donné) qui pourront alors être appelées depuis les assistants à la démonstration existants.

Il nous faudra également étudier dans quelle mesure nos résultats concernant la longueur des démonstrations peuvent avoir des implications pratiques pour le développement de méthodes de recherche de démonstrations automatique efficaces.

7.5 Une vision plus lointaine

Pour finir et en guise d'ouverture, nous avons étudiées au cours de ce travail des méthodes pour simplifier des démonstrations contenant du calcul. Inversement, on peut se demander comment formaliser dans des démonstrations les simplifications qu'on est parfois amené à effectuer dans des calculs. Nous pensons en tout particulier aux approximations qui sont effectuées lors du développement de méthodes de simulations de

[2] http://elan.loria.fr/
[3] http://strategoxt.org/

phénomènes physiques et chimiques, en particulier dans le domaine de la mécanique quantique. En effet, le but principal des logiciels de chimie quantique est de résoudre l'équation de SCHRÖDINGER sur des systèmes bien particulier, ce qui n'est pas possible de façon analytique dans la plupart des cas. On recourt alors à des approximations, par exemple en découplant certaines variables ou en se projetant sur une base d'orbitales finie. En physique classique, on utilise aussi beaucoup des limitations à un certain ordre de développements de TAYLOR-YOUNG. Dans l'optique de pouvoir vérifier formellement de tels logiciels, il faudra prendre en compte d'une manière ou d'une autre de telles approximations. La façon la plus simple est de considérer que la spécification est celle qui est déjà approximée. Il faut alors faire confiance au théoricien qui a développée la méthode. L'autre approche serait de développer un cadre formel pour ces approximations. Par exemple, on pourrait considérer que la spécification initiale est l'équation de SCHRÖDINGER. Il faudrait alors être capable de démontrer que, vis-à-vis d'une certaine approximation, le programme à vérifier respecte la spécification. Il est à noter que si cette idée semble avoir des liens avec l'interprétation abstraite, il ne s'agit pas de la même chose, car un programme vérifiant une approximation de la spécification ne la vérifiera en général pas, alors qu'au contraire un programme vérifiant l'abstraction d'une spécification vérifie cette spécification.

Annexe A

Démonstration de la terminaison du système $\mathcal{HO}_2$

> The method of "postulating" what we want has many advantages; they are the same as the advantages of theft over honest toil.
>
> Bertrand Russell, *Introduction to Mathematical Philosophy*

Nous ajoutons ici le résultat obtenu à l'aide d'AProVE, que nous livrons tel quel. Sans entrer dans les détails, il s'agit de calculer les paires de dépendances du systèmes puis de trouver une interprétation polynomiale pour chacun des cycles du graphe de dépendance. Le lecteur intéressé pourra consulter la documentation d'AProVE pour avoir plus d'explications.

Termination proof with AProVE[1]

Termination of $\mathcal{HO}_2$ could successfully be *proven*:

Term Rewriting System $\mathcal{HO}_2$:

$$
\begin{aligned}
\mathsf{subst0}(t, \mathsf{nil}) &\to t \\
\mathsf{subst0}(\mathsf{un0}, \mathsf{cons0}(t, l)) &\to t \\
\mathsf{subst0}(\mathsf{S0}(n), \mathsf{cons0}(t, l)) &\to \mathsf{subst0}(n, l) \\
\mathsf{subst0}(\mathsf{S0}(n), \mathsf{cons1}(t, l)) &\to \mathsf{subst0}(n, l) \\
\mathsf{subst0}(\mathsf{s}(n), l) &\to \mathsf{s}(\mathsf{subst0}(n, l)) \\
\mathsf{subst0}(\mathsf{plus}(t1, t2), l) &\to \mathsf{plus}(\mathsf{subst0}(t1, l), \mathsf{subst0}(t2, l)) \\
\mathsf{subst0}(\mathsf{mult}(t1, t2), l) &\to \mathsf{mult}(\mathsf{subst0}(t1, l), \mathsf{subst0}(t2, l)) \\
\mathsf{subst1}(t, \mathsf{nil}) &\to t \\
\mathsf{subst1}(\mathsf{un1}, \mathsf{cons1}(t, l)) &\to t \\
\mathsf{subst1}(\mathsf{S1}(n), \mathsf{cons0}(t, l)) &\to \mathsf{subst1}(n, l)
\end{aligned}
$$

[1] `http://www-i2.informatik.rwth-aachen.de/AProVE`

$$
\begin{array}{rcl}
\mathsf{subst1}(\mathsf{S1}(n), \mathsf{cons1}(t, l)) & \to & \mathsf{subst1}(n, l) \\
\mathsf{inc}(l, \mathsf{doteq}(t1, t2)) & \to & \mathsf{eq}(\mathsf{subst0}(t1, l), \mathsf{subst0}(t2, l)) \\
\mathsf{inc}(l, \mathsf{dotin0}(t1, t2)) & \to & \mathsf{in0}(\mathsf{subst0}(t1, l), \mathsf{subst1}(t2, l)) \\
\mathsf{inc}(l, \mathsf{supset}(A, B)) & \to & \mathsf{imp}(\mathsf{inc}(l, A), \mathsf{inc}(l, B)) \\
\mathsf{inc}(l, \mathsf{cup}(A, B)) & \to & \mathsf{or}(\mathsf{inc}(l, A), \mathsf{inc}(l, B)) \\
\mathsf{inc}(l, \mathsf{cap}(A, B)) & \to & \mathsf{and}(\mathsf{inc}(l, A), \mathsf{inc}(l, B)) \\
\mathsf{inc}(l, \mathsf{empty}) & \to & \mathsf{bot} \\
\mathsf{inc}(l, \mathsf{P0}(A)) & \to & \mathsf{ex}(\mathsf{inc}(\mathsf{cons0}(\mathsf{fresh}, l), A)) \\
\mathsf{inc}(l, \mathsf{P1}(A)) & \to & \mathsf{ex}(\mathsf{inc}(\mathsf{cons1}(\mathsf{fresh}, l), A)) \\
\mathsf{inc}(l, \mathsf{C0}(A)) & \to & \mathsf{all}(\mathsf{inc}(\mathsf{cons0}(\mathsf{fresh}, l), A)) \\
\mathsf{inc}(l, \mathsf{C1}(A)) & \to & \mathsf{all}(\mathsf{inc}(\mathsf{cons1}(\mathsf{fresh}, l), A)) \\
\mathsf{in0}(x, \mathsf{comp1}(A)) & \to & \mathsf{inc}(\mathsf{cons0}(x, \mathsf{nil}), A)
\end{array}
$$

Termination of $\mathcal{HO}_2$ to be shown.

R
↪**Dependency Pair Analysis**

$\mathcal{HO}_2$ contains the following *Dependency Pairs:*

$$
\begin{array}{rcl}
\mathsf{SUBST0}(\mathsf{S0}(n), \mathsf{cons0}(t, l)) & \to & \mathsf{SUBST0}(n, l) \\
\mathsf{SUBST0}(\mathsf{S0}(n), \mathsf{cons1}(t, l)) & \to & \mathsf{SUBST0}(n, l) \\
\mathsf{SUBST0}(\mathsf{s}(n), l) & \to & \mathsf{SUBST0}(n, l) \\
\mathsf{SUBST0}(\mathsf{plus}(t1, t2), l) & \to & \mathsf{SUBST0}(t1, l) \\
\mathsf{SUBST0}(\mathsf{plus}(t1, t2), l) & \to & \mathsf{SUBST0}(t2, l) \\
\mathsf{SUBST0}(\mathsf{mult}(t1, t2), l) & \to & \mathsf{SUBST0}(t1, l) \\
\mathsf{SUBST0}(\mathsf{mult}(t1, t2), l) & \to & \mathsf{SUBST0}(t2, l) \\
\mathsf{SUBST1}(\mathsf{S1}(n), \mathsf{cons0}(t, l)) & \to & \mathsf{SUBST1}(n, l) \\
\mathsf{SUBST1}(\mathsf{S1}(n), \mathsf{cons1}(t, l)) & \to & \mathsf{SUBST1}(n, l) \\
\mathsf{INC}(l, \mathsf{doteq}(t1, t2)) & \to & \mathsf{SUBST0}(t1, l) \\
\mathsf{INC}(l, \mathsf{doteq}(t1, t2)) & \to & \mathsf{SUBST0}(t2, l) \\
\mathsf{INC}(l, \mathsf{dotin0}(t1, t2)) & \to & \mathsf{IN0}(\mathsf{subst0}(t1, l), \mathsf{subst1}(t2, l)) \\
\mathsf{INC}(l, \mathsf{dotin0}(t1, t2)) & \to & \mathsf{SUBST0}(t1, l) \\
\mathsf{INC}(l, \mathsf{dotin0}(t1, t2)) & \to & \mathsf{SUBST1}(t2, l) \\
\mathsf{INC}(l, \mathsf{supset}(A, B)) & \to & \mathsf{INC}(l, A) \\
\mathsf{INC}(l, \mathsf{supset}(A, B)) & \to & \mathsf{INC}(l, B) \\
\mathsf{INC}(l, \mathsf{cup}(A, B)) & \to & \mathsf{INC}(l, A) \\
\mathsf{INC}(l, \mathsf{cup}(A, B)) & \to & \mathsf{INC}(l, B) \\
\mathsf{INC}(l, \mathsf{cap}(A, B)) & \to & \mathsf{INC}(l, A) \\
\mathsf{INC}(l, \mathsf{cap}(A, B)) & \to & \mathsf{INC}(l, B)
\end{array}
$$

$$\begin{array}{rcl}
\mathsf{INC}(l, \mathsf{P0}(A)) & \to & \mathsf{INC}(\mathsf{cons0}(\mathsf{fresh}, l), A) \\
\mathsf{INC}(l, \mathsf{P1}(A)) & \to & \mathsf{INC}(\mathsf{cons1}(\mathsf{fresh}, l), A) \\
\mathsf{INC}(l, \mathsf{C0}(A)) & \to & \mathsf{INC}(\mathsf{cons0}(\mathsf{fresh}, l), A) \\
\mathsf{INC}(l, \mathsf{C1}(A)) & \to & \mathsf{INC}(\mathsf{cons1}(\mathsf{fresh}, l), A) \\
\mathsf{IN0}(x, \mathsf{comp1}(A)) & \to & \mathsf{INC}(\mathsf{cons0}(x, \mathsf{nil}), A)
\end{array}$$

Furthermore, $\mathcal{HO}_2$ contains three SCCs.

R
↪DPs
→DP Problem 1
↪**Polynomial Ordering**
→DP Problem 2
↪Polo
→DP Problem 3
↪Polo

The node numbers are specified as follows:

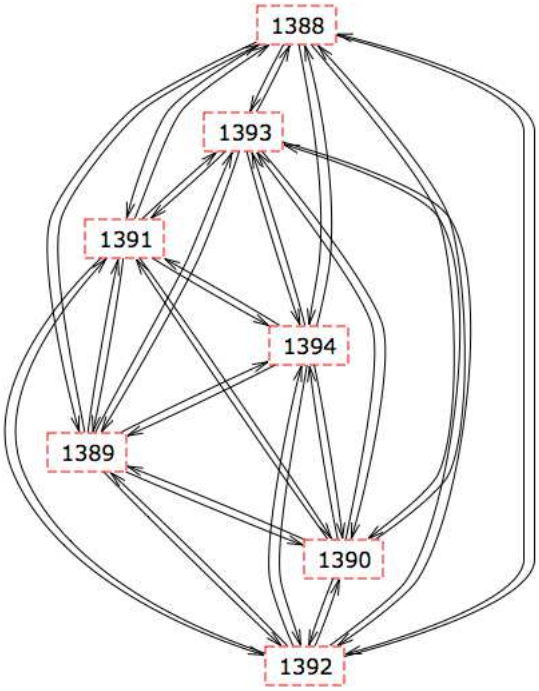
$$\begin{array}{rrcl}
1388: & \mathsf{SUBST0}(\mathsf{S0}(n), \mathsf{cons0}(t, l)) & \to & \mathsf{SUBST0}(n, l) \\
1389: & \mathsf{SUBST0}(\mathsf{S0}(n), \mathsf{cons1}(t, l)) & \to & \mathsf{SUBST0}(n, l) \\
1390: & \mathsf{SUBST0}(\mathsf{s}(n), l) & \to & \mathsf{SUBST0}(n, l) \\
1391: & \mathsf{SUBST0}(\mathsf{plus}(t1, t2), l) & \to & \mathsf{SUBST0}(t1, l) \\
1392: & \mathsf{SUBST0}(\mathsf{plus}(t1, t2), l) & \to & \mathsf{SUBST0}(t2, l) \\
1393: & \mathsf{SUBST0}(\mathsf{mult}(t1, t2), l) & \to & \mathsf{SUBST0}(t1, l) \\
1394: & \mathsf{SUBST0}(\mathsf{mult}(t1, t2), l) & \to & \mathsf{SUBST0}(t2, l)
\end{array}$$

The following dependency pairs can be strictly oriented:

$$\begin{array}{rcl}
\mathsf{SUBST0}(\mathsf{S0}(n), \mathsf{cons1}(t, l)) & \to & \mathsf{SUBST0}(n, l) \\
\mathsf{SUBST0}(\mathsf{S0}(n), \mathsf{cons0}(t, l)) & \to & \mathsf{SUBST0}(n, l) \\
\mathsf{SUBST0}(\mathsf{mult}(t1, t2), l) & \to & \mathsf{SUBST0}(t1, l) \\
\mathsf{SUBST0}(\mathsf{s}(n), l) & \to & \mathsf{SUBST0}(n, l) \\
\mathsf{SUBST0}(\mathsf{plus}(t1, t2), l) & \to & \mathsf{SUBST0}(t2, l) \\
\mathsf{SUBST0}(\mathsf{mult}(t1, t2), l) & \to & \mathsf{SUBST0}(t2, l) \\
\mathsf{SUBST0}(\mathsf{plus}(t1, t2), l) & \to & \mathsf{SUBST0}(t1, l)
\end{array}$$

There are no usable rules w.r.t. to the implicit AFS that need to be oriented. Used ordering: Polynomial ordering with Polynomial interpretation:

$$POL(\mathsf{mult}(x_1, x_2)) \quad = \quad 1 + x_1 + x_2$$



$$
\begin{aligned}
POL(\mathsf{plus}(x_1, x_2)) &= 1 + x_1 + x_2 \\
POL(\mathsf{cons1}(x_1, x_2)) &= 0 \\
POL(\mathsf{SUBST0}(x_1, x_2)) &= 1 + x_1 \\
POL(\mathsf{s}(x_1)) &= 1 + x_1 \\
POL(\mathsf{cons0}(x_1, x_2)) &= 0 \\
POL(\mathsf{S0}(x_1)) &= 1 + x_1
\end{aligned}
$$

resulting in one new DP problem.

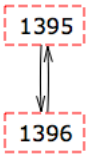
R
↪DPs
→DP Problem 1
↪Polo
→DP Problem 4
↪**Dependency Graph**

→DP Problem 2
↪Polo
→DP Problem 3
↪Polo

The node numbers are specified as follows:

Using the Dependency Graph resulted in no new DP problems.

R
↪DPs
→DP Problem 1
↪Polo
→DP Problem 2
↪**Polynomial Ordering**
→DP Problem 3
↪Polo



The node numbers are specified as follows:

$$\begin{array}{lrcl} 1395: & \mathsf{SUBST1}(\mathsf{S1}(n), \mathsf{cons0}(t, l)) & \to & \mathsf{SUBST1}(n, l) \\ 1396: & \mathsf{SUBST1}(\mathsf{S1}(n), \mathsf{cons1}(t, l)) & \to & \mathsf{SUBST1}(n, l) \end{array}$$

The following dependency pairs can be strictly oriented:

$$\mathsf{SUBST1}(\mathsf{S1}(n), \mathsf{cons1}(t, l)) \quad \to \quad \mathsf{SUBST1}(n, l)$$

$$\mathsf{SUBST1}(\mathsf{S1}(n), \mathsf{cons0}(t, l)) \quad \rightarrow \quad \mathsf{SUBST1}(n, l)$$

There are no usable rules w.r.t. to the implicit AFS that need to be oriented. Used ordering: Polynomial ordering with Polynomial interpretation:

$$\begin{aligned} POL(\mathsf{cons1}(x_1, x_2)) &= 0 \\ POL(\mathsf{S1}(x_1)) &= 1 + x_1 \\ POL(\mathsf{cons0}(x_1, x_2)) &= 0 \\ POL(\mathsf{SUBST1}(x_1, x_2)) &= 1 + x_1 \end{aligned}$$

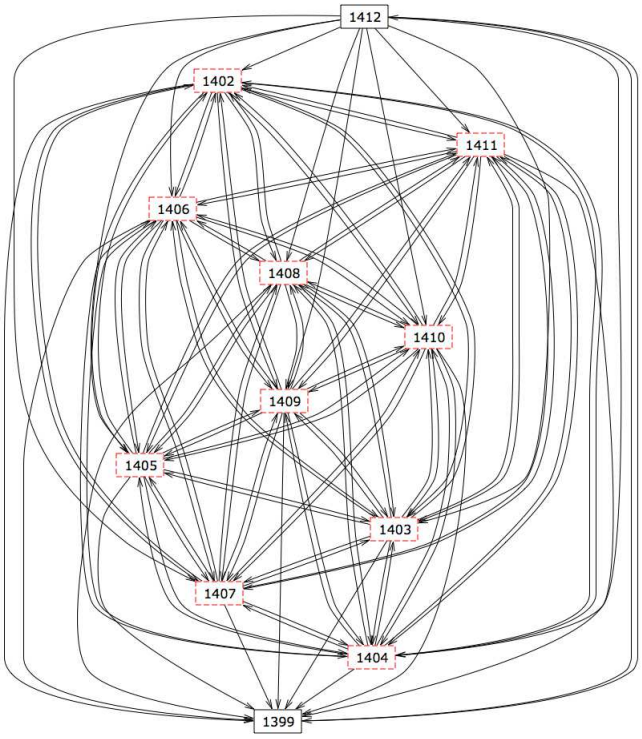
resulting in one new DP problem.

R
↪DPs
→DP Problem 1
↪Polo
→DP Problem 2
↪Polo
→DP Problem 5
↪**Dependency Graph**
→DP Problem 3
↪Polo

The node numbers are specified as follows:

Using the Dependency Graph resulted in no new DP problems.

R
↪DPs
→DP Problem 1
↪Polo
→DP Problem 2
↪Polo
→DP Problem 3
↪**Polynomial Ordering**



The node numbers are specified as follows:

$$
\begin{array}{lrcl}
1399: & \mathsf{INC}(l, \mathsf{dotin0}(t1, t2)) & \to & \mathsf{IN0}(\mathsf{subst0}(t1, l), \mathsf{subst1}(t2, l)) \\
1402: & \mathsf{INC}(l, \mathsf{supset}(A, B)) & \to & \mathsf{INC}(l, A) \\
1403: & \mathsf{INC}(l, \mathsf{supset}(A, B)) & \to & \mathsf{INC}(l, B) \\
1404: & \mathsf{INC}(l, \mathsf{cup}(A, B)) & \to & \mathsf{INC}(l, A) \\
1405: & \mathsf{INC}(l, \mathsf{cup}(A, B)) & \to & \mathsf{INC}(l, B) \\
1406: & \mathsf{INC}(l, \mathsf{cap}(A, B)) & \to & \mathsf{INC}(l, A) \\
1407: & \mathsf{INC}(l, \mathsf{cap}(A, B)) & \to & \mathsf{INC}(l, B) \\
1408: & \mathsf{INC}(l, \mathsf{P0}(A)) & \to & \mathsf{INC}(\mathsf{cons0}(\mathsf{fresh}, l), A) \\
1409: & \mathsf{INC}(l, \mathsf{P1}(A)) & \to & \mathsf{INC}(\mathsf{cons1}(\mathsf{fresh}, l), A) \\
1410: & \mathsf{INC}(l, \mathsf{C0}(A)) & \to & \mathsf{INC}(\mathsf{cons0}(\mathsf{fresh}, l), A) \\
1411: & \mathsf{INC}(l, \mathsf{C1}(A)) & \to & \mathsf{INC}(\mathsf{cons1}(\mathsf{fresh}, l), A) \\
1412: & \mathsf{IN0}(x, \mathsf{comp1}(A)) & \to & \mathsf{INC}(\mathsf{cons0}(x, \mathsf{nil}), A)
\end{array}
$$

The following dependency pairs can be strictly oriented:

$$\begin{aligned}
\mathsf{INC}(l, \mathsf{cup}(A, B)) &\to \mathsf{INC}(l, A)\\
\mathsf{INC}(l, \mathsf{P1}(A)) &\to \mathsf{INC}(\mathsf{cons1}(\mathsf{fresh}, l), A)\\
\mathsf{INC}(l, \mathsf{cap}(A, B)) &\to \mathsf{INC}(l, A)\\
\mathsf{INC}(l, \mathsf{C0}(A)) &\to \mathsf{INC}(\mathsf{cons0}(\mathsf{fresh}, l), A)\\
\mathsf{INC}(l, \mathsf{cup}(A, B)) &\to \mathsf{INC}(l, B)\\
\mathsf{INC}(l, \mathsf{supset}(A, B)) &\to \mathsf{INC}(l, B)\\
\mathsf{INC}(l, \mathsf{C1}(A)) &\to \mathsf{INC}(\mathsf{cons1}(\mathsf{fresh}, l), A)\\
\mathsf{INC}(l, \mathsf{dotin0}(t1, t2)) &\to \mathsf{IN0}(\mathsf{subst0}(t1, l), \mathsf{subst1}(t2, l))\\
\mathsf{IN0}(x, \mathsf{comp1}(A)) &\to \mathsf{INC}(\mathsf{cons0}(x, \mathsf{nil}), A)\\
\mathsf{INC}(l, \mathsf{cap}(A, B)) &\to \mathsf{INC}(l, B)\\
\mathsf{INC}(l, \mathsf{P0}(A)) &\to \mathsf{INC}(\mathsf{cons0}(\mathsf{fresh}, l), A)\\
\mathsf{INC}(l, \mathsf{supset}(A, B)) &\to \mathsf{INC}(l, A)
\end{aligned}$$

Additionally, the following usable rules w.r.t. to the implicit AFS can be oriented:

$$\begin{aligned}
\mathsf{subst1}(t, \mathsf{nil}) &\to t\\
\mathsf{subst1}(\mathsf{un1}, \mathsf{cons1}(t, l)) &\to t\\
\mathsf{subst1}(\mathsf{S1}(n), \mathsf{cons0}(t, l)) &\to \mathsf{subst1}(n, l)\\
\mathsf{subst1}(\mathsf{S1}(n), \mathsf{cons1}(t, l)) &\to \mathsf{subst1}(n, l)
\end{aligned}$$

Used ordering: Polynomial ordering with Polynomial interpretation:

$$\begin{aligned}
POL(\mathsf{mult}(x_1, x_2)) &= 0\\
POL(\mathsf{supset}(x_1, x_2)) &= 1 + x_1 + x_2\\
POL(\mathsf{plus}(x_1, x_2)) &= 0\\
POL(\mathsf{subst0}(x_1, x_2)) &= 0\\
POL(\mathsf{IN0}(x_1, x_2)) &= x_2\\
POL(\mathsf{cup}(x_1, x_2)) &= 1 + x_1 + x_2\\
POL(\mathsf{un1}) &= 0\\
POL(\mathsf{C1}(x_1)) &= 1 + x_1\\
POL(\mathsf{subst1}(x_1, x_2)) &= x_1 + x_2\\
POL(\mathsf{cons0}(x_1, x_2)) &= x_2\\
POL(\mathsf{P0}(x_1)) &= 1 + x_1\\
POL(\mathsf{S0}(x_1)) &= 0\\
POL(\mathsf{INC}(x_1, x_2)) &= x_1 + x_2\\
POL(\mathsf{comp1}(x_1)) &= 1 + x_1\\
POL(\mathsf{C0}(x_1)) &= 1 + x_1
\end{aligned}$$

$$
\begin{aligned}
POL(\mathsf{cons1}(x_1, x_2)) &= x_1 + x_2 \\
POL(\mathsf{P1}(x_1)) &= 1 + x_1 \\
POL(\mathsf{un0}) &= 0 \\
POL(\mathsf{cap}(x_1, x_2)) &= 1 + x_1 + x_2 \\
POL(\mathsf{S1}(x_1)) &= 1 + x_1 \\
POL(\mathsf{nil}) &= 0 \\
POL(\mathsf{s}(x_1)) &= 0 \\
POL(\mathsf{dotin0}(x_1, x_2)) &= 1 + x_2 \\
POL(\mathsf{fresh}) &= 0
\end{aligned}
$$

resulting in one new DP problem.

R
↪DPs
→DP Problem 1
↪Polo
→DP Problem 2
↪Polo
→DP Problem 3
↪Polo
→DP Problem 6
↪**Dependency Graph**

The node numbers are specified as follows:

Using the Dependency Graph resulted in no new DP problems.
Termination of $\mathcal{HO}_2$ successfully shown.

Index

Table des figures

Liste des tableaux

Bibliographie

Martin Abadi, Lucas Cardelli, Pierre-Louis Curien et Jean-Jacques Lévy, 1991. Explicit substitutions. *Journal of Functional Programming*, **1**, 4 : 375–416.

Marc Aiguier, Clément Boin et Delphine Longuet, 2005. On generalized theorems for normalization of proofs. Rapport technique, LaMI – CNRS et université d'Évry Val d'Essonne.

Marc Aiguier et Delphine Longuet, 2007. Notes on generalization of proof normalization. Soumis, voir aussi Marc Aiguier, Clément Boin et Delphine Longuet (2005).

Lisa Allali, 2007. Algorithmic equality in heyting arithmetic modulo. Dans : Marino Miculan, Ivan Scagnetto et Furio Honsell (dir.), *TYPES*, LNCS. Springer.

Lisa Allali et Paul Brauner, 2008. A semantic normalization proof of systems with recursors. Soumis.

Thorsten Altenkirch et Conor McBride (dir.), 2007. *Types for Proofs and Programs, International Workshop, TYPES 2006, Nottingham, UK, April 18-21, 2006, Revised Selected Papers*, *LNCS*, volume 4502. Springer. ISBN 978-3-540-74463-4.

Jean-Marc Andreoli, 1992. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, **2**, 3 : 297–347.

Kenneth Appel, Wolfgang Haken et John A. Koch, 1977. Every planar map is four-colorable. *Illinois Journal of Mathematics*, **21** : 429–567.

Brian E. Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack et Stephanie Weirich, 2008. Engineering formal metatheory. Dans : George C. Necula et Philip Wadler (dir.), *POPL*, 3–15. ACM. ISBN 978-1-59593-689-9.

Franz Baader et Tobias Nipkow, 1998. *Term* Re*writing and* all That. Cambridge University Press.

Leo Bachmair, 1987. *Proof methods for equational theories*. Thèse de doctorat, University of Illinois, Urbana-Champaign, (Ill., USA). Version révisée, août 1988.

———, 1991. *Canonical equational proofs*. Computer Science Logic, Progress in Theoretical Computer Science. Birkhäuser Verlag AG.

Leo Bachmair et Nachum Dershowitz, 1989. Completion for rewriting modulo a congruence. *Theoretical Computer Science*, **67**, 2-3 : 173–202.

———, 1994. Equational inference, canonical proofs, and proof orderings. *Journal of Association for Computing Machinery*, **41**, 2 : 236–276.

Leo Bachmair, Nachum Dershowitz et Jieh Hsiang, 1986. Orderings for equational proofs. Dans : *Proceedings 1st IEEE Symposium on Logic in Computer Science, Cambridge (Mass., USA)*, 346–357.

Leo Bachmair, Nachum Dershowitz et David Plaisted, 1989. Completion without failure. Dans : H. Aït-Kaci et M. Nivat (dir.), *Resolution of Equations in Algebraic Structures, Volume 2 : Rewriting Techniques*, 1–30. Academic Press inc.

Leo Bachmair et Harald Ganzinger, 1994. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, **4**, 3 : 217–247.

Emilie Balland, Claude Kirchner et Pierre-Etienne Moreau, 2006. Formal islands. Dans : Michael Johnson et Varmo Vene (dir.), *Proceedings of Algebraic Methodology and Software Technology - AMAST'06*, LNCS. sv.

Henk Barendregt, 1984. *The Lambda Calculus, its Syntax and Semantics*. Studies in Logic and the Foundation of Mathematics, 2e édition. Amsterdam : Elsevier Science Publishers B. V. (North-Holland).

———, 1992. *Handbook of logic in computer science*, volume 2, chapitre Lambda calculi with types, 117–309. Oxford University Press. ISBN 0-19-853761-1.

Karell Bertet et Mirabelle Nebut, 2004. Efficient algorithms on the family associated to an implicational system. *Discrete Mathematics & Theoretical Computer Science*, **6**, 2 : 315–338.

Frédéric Blanqui, 2005. Inductive types in the calculus of algebraic constructions. *Fundamenta Informaticae*, **65** : 61–86.

Frédéric Blanqui, Thérèse Hardin et Pierre Weis, 2007. On the implementation of construction functions for non-free concrete data types. Dans : Rocco De Nicola (2007), 95–109.

Frédéric Blanqui, Jean-Pierre Jouannaud et Mitsuhiro Okada, 1999. The calculus of algebraic constructions. Dans : P. Narendran et M. Rusinowitch (dir.), *Proceedings of the 10th International Conference on Rewriting Techniques and Applications (RTA-99)*, 301–316. Trento, Italy : Springer-Verlag LNCS 1631.

Maria Paola Bonacina et Nachum Dershowitz, 2007*a*. Abstract canonical inference. *ACM Transactions on Computational Logic*, **8**, 1. ISSN 1529-3785.

———, 2007*b*. Canonical ground horn theories. Research Report 49/2007, Dipartimento di Informatica, Università degli Studi di Verona.

———, 2008. Canonical inference for implicational systems. Dans : Alessandro Armando et Peter Baumgartner (dir.), *IJCAR*, *Lecture Notes in Artificial Intelligence*, volume 5195, 380–395. Springer.

Guillaume Bonfante, Jean-Yves Marion et Jean-Yves Moyen, 2005. Quasi-interpretations : a way to control resources. *Theoretical Computer Science*. To appear.

Richard Bonichon, 2004. TaMeD : A tableau method for deduction modulo. Dans : David A. Basin et Michaël Rusinowitch (dir.), *IJCAR*, *LNCS*, volume 3097, 445–459. Springer. ISBN 3-540-22345-2.

Richard Bonichon et Olivier Hermant, 2006*a*. On constructive cut admissibility in deduction modulo. Dans : Thorsten Altenkirch et Conor McBride (2007), 33–47.

———, 2006*b*. A semantic completeness proof for TaMed. Dans : Miki Hermann et Andrei Voronkov (2006), 167–181.

Samuel Boutin, 1997. Using reflection to build efficient and certified decision procedures. Dans : Martín Abadi et Takayasu Ito (dir.), *TACS*, *LNCS*, volume 1281, 515–529. Springer. ISBN 3-540-63388-X.

Paul Brauner, Gilles Dowek et Benjamin Wack, 2007*a*. Normalization in supernatural deduction and in deduction modulo. Disponible à l'adresse http://www.loria.fr/~brauner/submission_94.pdf.

Paul Brauner, Clément Houtmann et Claude Kirchner, 2007*b*. Principle of superdeduction. Dans : Luke Ong (dir.), *LICS*, 41–50. URL http://doi.ieeecomputersociety.org/10.1109/LICS.2007.37.

Kai Brünnler, 2003. *Deep Inference and Symmetry in Classical Proofs*. Thèse de doctorat, Technische Universität Dresden.

Paola Bruscoli et Alessio Guglielmi, 2009. On the proof complexity of deep inference. *ACM Transactions on Computational Logic*. À paraître.

Bruno Buchberger, 1985. *Multidimensional Systems Theory*, chapitre Gröbner bases : an algorithmic method in polynomial ideal theory, 184–232. Reidel, Bose, N.K. Ed.

Guillaume Burel, 2005. *Systèmes canoniques abstraits : application à la déduction naturelle et à la complétion*. Rapport de master, université Denis Diderot (Paris 7) & LORIA. URL http://hal.inria.fr/inria-00000773.

———, 2007. Unbounded proof-length speed-up in deduction modulo. Dans : Jacques Duparc et Thomas A. Henziger (dir.), *CSL 2007*, *LNCS*, volume 4646, 496–511. Springer.

———, 2008*a*. Efficiently simulating higher-order arithmetic by a first-order theory modulo. Disponible à l'adresse http://arxiv.org/abs/0805.1464, voir également Guillaume Burel (2007).

———, 2008*b*. A first-order representation of pure type systems using superdeduction. Dans : Frank Pfenning (2008), 253–263.

Guillaume Burel et Claude Kirchner, 2006. Completion is an instance of abstract canonical system inference. Dans : Kokichi Futatsugi (dir.), *Algebra, Meaning and Computation*, *LNCS*, volume 4060, 497–520. Springer.

———, 2007. Cut elimination in deduction modulo by abstract completion. Dans : Sergei N. Artemov et Anil Nerode (dir.), *LFCS*, *LNCS*, volume 4514, 115–131. Springer.

———, 2008. Regaining cut admissibility in deduction modulo using abstract completion. Disponible à l'adresse http://www.loria.fr/~burel/download/gencomp_ic.pdf, voir également Guillaume Burel et Claude Kirchner (2007).

Albert Burroni, 1993. Higher-dimensional word problems with applications to equational logic. *Theoretical Computer Science*, **115**, 1 : 43–62.

Samuel R. Buss, 1994. On Gödel's theorems on lengths of proofs I : Number of lines and speedup for arithmetics. *The Journal of Symbolic Logic*, **59**, 3 : 737–756.

Alonzo Church, 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic*, **5** : 56–68.

Agata Ciabattoni, Nikolaos Galatos et Kazushige Terui, 2008. From axioms to analytic rules in nonclassical logics. Dans : Frank Pfenning (2008).

Horatiu Cirstea, Clément Houtmann et Benjamin Wack, 2006. Distributive rho-calculus. Dans : *Proceedings of 6th International Workshop on Rewriting Logic and its Applications*, To appear in Electronic Notes in Theoretical Computer Science. Elsevier.

Horatiu Cirstea et Claude Kirchner, 2001. The rewriting calculus — Part I *and* II. *Logic Journal of the Interest Group in Pure and Applied Logics*, **9**, 3 : 427–498.

Stephen A. Cook et Robert A. Reckhow, 1974. On the lengths of proofs in the propositional calculus (preliminary version). Dans : *STOC '74 : Proceedings of the sixth annual ACM symposium on Theory of computing*, 135–148. New York, NY, USA : ACM.

———, 1979. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, **44**, 1 : 36–50.

Thierry Coquand, 1985. *Une théorie des constructions*. Thèse de doctorat, université Paris 7.

Thierry Coquand et Christine Paulin-Mohring, 1990. Inductively defined types. Dans : P. Martin-Löf et G. Mints (dir.), *Proceedings of Colog'88*, *Lecture Notes in Computer Science*, volume 417. Springer.

Denis Cousineau et Gilles Dowek, 2007. Embedding pure type systems in the lambda-pi-calculus modulo. Dans : Simona Ronchi Della Rocca (dir.), *TLCA*, *LNCS*, volume 4583, 102–117. Springer.

Patrick Cousot, 2007. Proving the absence of run-time errors in safety-critical avionics code. Dans : Christopher Kirsch et Reinhard Wilhelm (dir.), *Proceedings of the Seventh ACM & IEEE International Conference on Embedded Sofware, Embedded Systems Week, (EMSOFT 2007)*, 7–9. Salzburg, Austria : ACM press.

Marcel Crabbé, 1974. Non-normalisation de la théorie de Zermelo. Manuscript.

Haskell B. Curry, Robert Feys et William Craig, 1958. *Combinatory Logic*, volume 1. Amsterdam : Elsevier Science Publishers B. V. (North-Holland).

Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle et Joachim Posegga (dir.), 1999. *Handbook of Tableau Methods*. Boston : Kluwer Academic Publishers.

Nicolaas de Bruijn, 1970. The mathematical language AUTOMATH, its usage, and some of its extensions. Dans : *Symposium on Automatic Demonstration*, *Lecture Notes in Mathematics*, volume 125, 29 – 61. Versailles : Springer.

———, 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, **34**, 5 : 381–392.

Rocco De Nicola (dir.), 2007. *Programming Languages and Systems, 16th European Symposium on Programming, ESOP 2007, Held as Part of the Joint European Conferences on Theory and Practics of Software, ETAPS 2007, Braga, Portugal, March 24 - April 1, 2007, Proceedings*, *LNCS*, volume 4421. Springer. ISBN 978-3-540-71314-2.

Nachum Dershowitz, 1982. Orderings for term-rewriting systems. *Theoretical Computer Science*, **17** : 279–301.

———, 2003. Canonicity. Dans : Ingo Dahn et Laurent Vigneron (dir.), *FTP*, *Electronic Notes in Theoretical Computer Science*, volume 86, 147–158. Elsevier Science Publishers B. V. (North-Holland).

Nachum Dershowitz et Claude Kirchner, 2003. Abstract saturation-based inference. Dans : *LICS*, 65–74. IEEE Computer Society. ISBN 0-7695-1884-2.

———, 2006. Abstract canonical presentations. *Theoretical Computer Science*, **357** : 53–69.

Gilles Dowek, 1996. A type-free formalization of mathematics where proofs are objects. Dans : Eduardo Giménez et Christine Paulin-Mohring (dir.), *TYPES*, *Lecture Notes in Computer Science*, volume 1512, 88–111. Springer. ISBN 3-540-65137-3.

———, 2001. About folding-unfolding cuts and cuts modulo. *Journal of Logic and Computation*, **11**, 3 : 419–429.

———, 2002. What is a theory ? Dans : Helmut Alt et Afonso Ferreira (dir.), *STACS*, *LNCS*, volume 2285, 50–64. Springer. ISBN 3-540-43283-3.

———, 2003. Confluence as a cut elimination property. Dans : Robert Nieuwenhuis (dir.), *RTA*, *LNCS*, volume 2706, 2–13. Springer. ISBN 3-540-40254-3.

———, 2006. Truth values algebras and proof normalization. Dans : Thorsten Altenkirch et Conor McBride (2007), 110–124.

Gilles Dowek, Thérèse Hardin et Claude Kirchner, 2001. HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, **11**, 1 : 1–25.

———, 2002. Binding logic : proofs and models. Dans : Matthias Baaz et Andrei Voronkov (dir.), *Proceedings of the LPAR conference*, *LNAI*, volume 2514, 130–144. Tbilisi, Georgia : Springer.

———, 2003. Theorem proving modulo. *Journal of Automated Reasoning*, **31**, 1 : 33–72. ISSN 0168-7433.

Gilles Dowek et Alexandre Miquel, 2007. Cut elimination for zermelo's set theory. Disponible sur la page web des auteurs.

Gilles Dowek et Benjamin Werner, 2003. Proof normalization modulo. *The Journal of Symbolic Logic*, **68**, 4 : 1289–1316.

———, 2005. Arithmetic as a theory modulo. Dans : Jürgen Giesl (dir.), *RTA*, *LNCS*, volume 3467, 423–437. Springer. ISBN 3-540-25596-6.

Robert Feys et Jean Ladrière (dir.), 1965. *Recherches sur la déduction logique*. Paris : Presses universitaires de France. Traduction française des articles de Gerhard Gentzen (1934).

Frederic B. Fitch, 1952. *Symbolic Logic : An Introduction*. New York : The Ronald Press Company.

Gerhard Gentzen, 1934. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, **39** : 176–210, 405–431. Translated in Szabo, editor., *The Collected Papers of Gerhard Gentzen* as "Investigations into Logical Deduction".

Herman Geuvers, 1992. The Church-Rosser property for beta-eta-reduction in typed lambda-calculi. Dans : *LICS*, 453–460. IEEE Computer Society.

———, 1993. *Logics and type systems*. Thèse de doctorat, Nijmegen University.

Jean-Yves Girard, 1972. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse d'état, université Paris 7.

———, 1987. Linear logic. *Theoretical Computer Science*, **50** : 1–102.

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

Kurt Gödel, 1931. Über formal unentscheidbare Sätze der Principia Mathemathica und verwandter Systeme I. *Monatsh für Math. u Phys.*, **12**, XXXVIII : 173–198.

Kurt Gödel, 1936. Über die Länge von Beweisen. *Ergebnisse eines Mathematischen Kolloquiums*, **7** : 23–24. Traduction anglaise dans (Kurt Gödel, 1986).

———, 1986. On the length of proofs. Dans : Solomon Feferman (dir.), *Kurt Gödel : Collected Works*, volume 1, 396–399. Oxford : Oxford University Press.

Georges Gonthier, 2005. A computer-checked proof of the four colour theorem. Rapport technique, Microsoft Research Cambridge.

Alessio Guglielmi, 2007. A system of interaction and structure. *ACM Trans. Comput. Logic*, **8**, 1. ISSN 1529-3785.

Yves Guiraud, 2006. The three dimensions of proofs. *Annals of Pure and Applied Logic*, **141**, 1-2 : 266–295.

Robert Harper, Furio Honsell et Gordon Plotkin, 1993. A framework for defining logics. *Journal of the ACM*, **40**, 1 : 143–184.

Hugo Herbelin, 1995. *Séquents qu'on calcule : de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Thèse de Doctorat d'Université, université Paris 7.

Miki Hermann et Andrei Voronkov (dir.), 2006. *Logic for Programming, Artificial Intelligence, and Reasoning, 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006, Proceedings*, *LNCS*, volume 4246. Springer. ISBN 3-540-48281-4.

Olivier Hermant, 2005. *Méthodes Sémantiques en Déduction Modulo*. Thèse de doctorat, École Polytechnique.

David Hilbert, 1901. Mathematische Probleme. *Archiv der Mathematik und Physik*, **1**, 3 : 44–63 ;213–237.

Clément Houtmann, 2008. Axiom directed focusing. Soumis, disponible sur la page web de l'auteur.

Jieh Hsiang et Michaël Rusinowitch, 1991. Proving refutational completeness of theorem proving strategies : The transfinite semantic tree method. *Journal of the ACM*, **38**, 3 : 559–587.

Dominic J.D. Hughes, 2006. Proofs without syntax. *Annals of Mathematics*, **164** : 1065–1076.

Stanisław Jaśkowski, 1934. On the rules of suppositions in formal logic. *Studia Logica*, **1** : 5–32. Reprinted in S. McCall (1967) *Polish Logic 1920–1939*, Oxford University Press, pp. 232–258.

Jean-Pierre Jouannaud et Hélène Kirchner, 1986. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, **15**, 4 : 1155–1194.

Ozan Kahramanoğullari, 2006. Reducing nondeterminism in the calculus of structures. Dans : Miki Hermann et Andrei Voronkov (2006), 272–286.

Stig Kanger, 1957. *Provability in Logic*, *Studies in Philosophy*, volume 1. Stockholm : Almqvist and Wicksell.

Delia Kesner, 2000. Confluence of extensional and non-extensional λ-calculi with explicit substitutions. *Theoretical Computer Science*, **238**, 1–2 : 183–220.

Claude Kirchner et Hélène Kirchner, 2001. Rewriting, solving, proving. Version préliminaire.

Claude Kirchner, Radu Kopetz et Pierre-Etienne Moreau, 2007. Anti-pattern matching. Dans : Rocco De Nicola (2007), 110–124.

Florent Kirchner, 2006. A finite first-order theory of classes. Dans : Thorsten Altenkirch et Conor McBride (2007), 188–202.

Florent Kirchner et César Muñoz, 2006. PVS# : Streamlined tacticals for PVS. *Electronic Notes in Theoretical Computer Science*, **174**, 11 : 47–58. ISSN 1571-0661. URL http://www.sciencedirect.com/science/article/B75H1-4P1PR1M-4/2/d7c0369c9b01102ca2b34eda7c0743b1.

Stephen C. Kleene, 1952*a*. Finite axiomatizability of theories in the predicate calculus using additional predicate symbols. Dans : Stephen Kleene (1952*d*), 27–68.

———, 1952*b*. *Introduction to Metamathematics*. The University series in higher mathematics. Princeton, USA : Van Nostrand.

———, 1952*c*. Permutability of inferences in Gentzen's calculi LK and LJ. Dans : Stephen Kleene (1952*d*), 1–26.

———, 1952*d*. *Two papers on the predicate calculus*, *Memoirs of the american mathematical society*, volume 10. Providence, USA : American Mathematical Society.

———, 1967. *Mathematical Logic*. New York, USA : John Wiley.

Donald E. Knuth et Peter B. Bendix, 1970. Simple word problems in universal algebras. Dans : J. Leech (dir.), *Computational Problems in Abstract Algebra*, 263–297. Oxford : Pergamon Press.

Jan Krajíček, 1989. On the number of steps in proofs. *Annals of Pure and Applied Logic*, **41**, 2 : 153–178.

Dallas S. Lankford, 1975. Canonical inference. Rapport technique, Louisiana Tech. University.

Jean Largeault (dir.), 1971. *Logique Mathématique*. Paris : Armand Colin. Traduction française de l'ouvrage de Stephen Kleene (1967).

Michael Machtey et Paul Young, 1978. *An Introduction to the General Theory of Algorithms*. New York, NY, USA : Elsevier Science Inc. ISBN 044400226X.

Shôji Maehara, 1954. Eine Darstellung der intuitionistischen Logik in der klassischen. *Nagoya Mathematical Journal*, **7** : 45–64.

Per Martin-Löf, 1971. A theory of types. Rapport technique 71–3, University of Stockholm.

Paul-André Melliès, 1995. Typed λ-calculi with explicit substitutions may not terminate. Dans : M. Dezani (dir.), *Int. Conf. on Typed Lambda Calculus and Applications*, *LNCS*, volume 902, 328–334. Springer.

José Meseguer, 1989. General logics. Dans : H.-D. Ebbinghaus (dir.), *Logic Colloquium'87*, 275–329. North-Holland : Elsevier Science Publisher.

José Meseguer, 1998. Membership algebra as a logical framework for equational specification. Dans : *Recent Trends in Algebraic Development Techniques*, *LNCS*, volume 1376, 18–61. Springer.

Dale Miller, 1991. A logic programming language with lambda-abstraction, function variables, and simple unification. Dans : Peter Schroeder-Heister (dir.), *Extensions of Logic Programming : International Workshop, Tübingen, Germany, December 1989*, *LNCS*, volume 475, 253–281. Springer.

Grigori Mints, 2000. *A Short Introduction to Intuitionistic Logic*. The University series in mathematics. Kluwer Academic Publishers.

Andrzej Mostowski, Raphael M. Robinson et Alfred Tarski, 1953. *Undecidable Theories*. Studies in Logic and the Foundations of Mathematics. Amsterdam : North-Holland.

Greg Nelson et Derek C. Oppen, 1980. Fast decision procedures based on congruence closure. *Journal of the ACM*, **27**, 2 : 356–364.

Rohit J. Parikh, 1973. Some results on the length of proofs. *Transactions of the ACM*, **177** : 29–36.

Christine Paulin-Mohring, 1992. Inductive definitions in the system Coq – rules and properties. Rapport de recherche 92-49, LIP, École normale supérieure de Lyon. Également dans les actes de TLCA'93.

Lawrence C. Paulson, 1990. Isabelle : the next 700 theorem provers. Dans : P. Odifreddi (dir.), *Logic in Computer Science*. Academic Press.

Gerald Peterson et Mark E. Stickel, 1981. Complete sets of reductions for some equational theories. *Journal of the ACM*, **28** : 233–264.

Frank Pfenning, 1996. The practice of logical frameworks. Dans : *CAAP*, *LNCS*, volume 1059, 119–134. Springer. ISBN 3-540-61064-2.

———, 2000. Structural cut elimination I. Intuitionistic and classical logic. *Information and Computation*, **157** : 84–141.

Frank Pfenning (dir.), 2008. *LICS*. Pittsburgh, USA : IEEE Computer Society.

Frank Pfenning et Conal Elliot, 1988. Higher-order abstract syntax. *SIGPLAN Not.*, **23**, 7 : 199–208. ISSN 0362-1340.

Henri Poincaré, 1902. *La Science et l'Hypothèse*. Flammarion.

Dag Prawitz, 1965. *Natural Deduction : A Proof Theoretical Study*. Almquist & Wiksell.

Allan Robinson, 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, **12** : 23–41.

Czesław Ryll-Nardzewski, 1952. The role of the axiom of induction in elementary arithmetic. *Fundamenta Mathematicae*, **39** : 239–263.

Peter Schroeder-Heister, 1990. Cut elimination for logics with definitional reflection. Dans : *Nonclassical Logics and Information Processing*, *Lecture Notes in Computer Science*, volume 619, 146–171. Springer.

Wayne Snyder, 1989. Efficient ground completion : An $O(n \log(n))$ algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations E. Dans : N. Dershowitz (dir.), *Proceedings 3rd Conference on Rewriting Techniques and Applications, Chapel Hill (N.C., USA)*, *LNCS*, volume 355, 419–433. Springer.

Richard Statman, 1978. Bounds for proof search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, **15** : 225–287.

Mark-Oliver Stehr et José Meseguer, 2004. Pure type systems in rewriting logic : Specifying typed higher-order languages in a first-order logical framework. Dans : Olaf Owe, Stein Krogdahl et Tom Lyche (dir.), *Essays in Memory of Ole-Johan Dahl*, *LNCS*, volume 2635, 334–375. Springer. ISBN 3-540-21366-X.

Lutz Strassburger, 2005*a*. From deep inference to proof nets. Dans : Paola Bruscoli, François Lamarche et Charles Stewart (dir.), *Structures and Deduction*, 2–18. Technische Universität Dresden. ICALP Workshop. ISSN 1430-211X. http://www.lix.polytechnique.fr/~lutz/papers/deepnet.ps.

———, 2005*b*. What is a logic, and what is a proof ? Dans : Jean-Yves Beziau (dir.), *Logica Universalis*, 135–145. Basel : Birkhäuser. Updated version at http://www.lix.polytechnique.fr/~lutz/papers/WhatLogicProof.pdf.

———, 2008. Extension without cut. http://www.lix.polytechnique.fr/~lutz/papers/psppp.pdf.

Rüdiger Thiele, 2003. Hilbert's twenty-fourth problem. *The American Mathematical Monthly*, **110**, 1 : 1–24.

Gregory S. Tseytin, 1968. On the complexity of derivation in propositional calculus. Dans : *Studies in Constructive Mathematics and Mathematical Logic II*, *Zapiski Nauchnykh Seminarov LOMI*, volume 8, 235–259. Leningrad : Nauka. In Russian.

Dirk van Dalen, 1989. *Logic and Structure*. Universitext, 2e édition. Springer.

Rakesh M. Verma, 1993. Smaran : A congruence-closure based system for equational computations. Dans : Claude Kirchner (dir.), *RTA*, *Lecture Notes in Computer Science*, volume 690, 457–461. Springer. ISBN 3-540-56868-9.

Sergei G. Vorobyov, 1988. On the arithmetic inexpressiveness of term rewriting systems. Dans : *LICS*, 212–217. IEEE Computer Society.

Arild Waaler et Lincoln Wallen, 1999. *Handbook of Tableau Methods*, chapitre Tableaux for Intutionistic Logics. Dans : Marcello D'Agostino et collaborateurs (1999).

Benjamin Wack, 2005. *Typage et Déduction dans le Calcul de Réécriture*. Thèse de doctorat, université Henri Poincaré – Nancy 1.

———, 2006. Supernatural deduction. Manuscript, available at http://www.loria.fr/~wack/papers/supernatural.ps.gz.

Benjamin Werner, 1994. *Une Théorie des Constructions Inductives*. Thèse de doctorat, université Paris 7.

Résumé

Cette thèse étudie comment l'intégration du calcul dans les démonstrations peut les simplifier. Nous nous intéressons pour cela à la déduction modulo et à la surdéduction, deux formalismes proches dans lesquels le calcul est incorporé dans les démonstrations via un système de réécriture. Pour améliorer la recherche mécanisée de démonstration, nous considérons trois critères de simplicité.

L'admissibilité des coupures permet de restreindre l'espace de recherche des démonstrations, mais elle n'est pas toujours assurée en déduction modulo. Nous définissons une procédure qui complète le système de réécriture pour, au final, admettre les coupures. Au passage, nous montrons comment transformer toute théorie pour l'intégrer à la partie calculatoire des démonstrations.

Nous montrons ensuite comment la déduction modulo permet de réduire arbitrairement la taille des démonstrations, en transférant des étapes de déduction dans le calcul. En particulier, nous appliquons ceci à l'arithmétique d'ordre supérieur pour démontrer que les réductions de taille qui sont possibles en augmentant l'ordre dans lequel on se place disparaissent si on travaille en déduction modulo.

Suite à ce dernier résultat, nous avons recherchés quels sont les systèmes d'ordre supérieur pouvant être simulés au premier ordre, en déduction modulo. Nous nous sommes intéressés aux systèmes de type purs et nous montrons comment ils peuvent être encodés en surdéduction, ce qui offre de nouvelles perspectives concernant leur normalisation et la recherche de démonstration dans ceux-ci. Nous développons également une méthodologie qui permet d'utiliser la surdéduction pour spécifier des systèmes de déduction.

Mots clefs : Théorie de la preuve, démonstration automatique de théorèmes, systèmes de réécriture (informatique), ordre sur les démonstrations

---

Abstract

This thesis studies how computations may simplify proofs and aims to make mechanized proof search better. We are particularly interested in deduction modulo and superdeduction, two close formalisms allowing the integration of computations into proofs through a rewrite system. We consider three simplicity criteria related to proof search.

Cut admissibility makes it possible to restrain the proof-search space but does not always hold in deduction modulo. We define a completion method transforming a rewrite system representing computations so that at the end cut admissibility holds. By the way, we show how to transform any first-order theory to integrate it into the computational part of proofs.

We show then how deduction modulo unboundedly reduces proof lengths, by transferring deduction steps into computations. In particular, we apply this to higher-order arithmetic to show that proof-length speedups between ith- and i+1st-order arithmetic disappear when working in deduction modulo, making it possible to work in first-order logic modulo without increasing proof lengths.

Following this last result, we investigate which higher-order systems can be simulated in first order using deduction modulo. We are interested by pure type systems, which are generic type systems for the λ-calculus with dependent types. We show how these systems can be encoded in superdeduction. This offers new perspectives on their normalization and on proof search within them. We also develop a methodology to describe how superdeduction can be used to specify deductive systems.

Keywords: Proof theory, automated theorem proving, rewrite systems, proof ordering