Représentation et interaction des preuves en superdéduction modulo

Clément Houtmann

▶ **To cite this version:**

Clément Houtmann. Représentation et interaction des preuves en superdéduction modulo. Génie logiciel [cs.SE]. Université Henri Poincaré - Nancy I, 2010. Français. NNT : 2010NAN10026 . tel-00553219

HAL Id: tel-00553219
https://theses.hal.science/tel-00553219

Submitted on 6 Jan 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Nancy-Université

UFR STMIA – École Doctorale IAEM

DFD informatique

Représentation et interaction des preuves en superdéduction modulo

THÈSE

présentée et soutenue publiquement le 12 mars 2010

pour l'obtention du

Doctorat de l'Université Henri Poincaré – Nancy 1

spécialité informatique

par

Clément Houtmann

devant le jury composé de

Rapporteurs :

Gilles Dowek ..... Professeur à l'École Polytechnique, Palaiseau, France

Delia Kesner ..... Professeur à l'Université Denis Diderot, Paris, France

Examinateurs :

Adam Cichon ..... Professeur à l'Université Henri Poincaré, Nancy, France

Claude Kirchner (directeur) ..... Directeur de recherche à l'INRIA, Bordeaux, France

Luigi Liquori ..... Chargé de recherche à l'INRIA, Nice, France

Dale Miller (président du jury) ..... Directeur de recherche à l'INRIA, Palaiseau, France

Christian Urban .... Chargé de recherche à la Technische Universität, Munich, Allemagne

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

À mon père.

Remerciements

Il me tient à cœur de remercier toutes les personnes qui m'ont aidé dans mon travail de thèse et plus généralement qui m'ont fourni l'éducation et les enseignements qui sont maintenant les miens. La première de ces personnes est bien sûr Claude Kirchner, mon directeur de thèse, qui m'a accueilli dans son équipe de recherche il y a maintenant cinq années et qui m'a donné depuis le formidable enseignement que représente cette thèse. Même l'éloignement physique des dernières années de thèse n'ont pu altérer sa disponibilité. Il a su me guider habilement dans ma reflexion sans jamais restreindre mon autonomie croissante. Je souhaite aussi remercier chaleureusement toutes les personnes qui m'ont fait l'honneur d'accepter de faire partie de mon jury de thèse. La lecture de leurs travaux et les échanges que nous avons pu avoir ont profondément marqué ma reflexion pendant ces dernières années. Je leur suis donc reconnaissant à la fois pour avoir examiné avec justesse mon travail mais aussi pour m'avoir fourni une inspiration considérable. En particulier, Delia Kesner et Gilles Dowek ont subi l'épreuve de la longueur de ce manuscrit. Je les remercie sincèrement pour l'intérêt et le temps qu'ils m'ont consacré. Enfin je souhaite remercier l'ensemble de mon jury pour m'avoir posé de si judicieuses questions. Celles-ci soulignent avec justesse les perspectives soulevées par mes travaux et j'espère pouvoir apporter dans le futur de nouveaux éléments de réponse.

Il va sans dire que le déroulement de mes études doctorales n'aurait pas été si plaisant sans le soutien et l'accueil des équipes Protheo puis Pareo et des groupes de travail Modulo et Corias. Je souhaite donc souligner ici combien je dois à leurs membres, notamment Pierre-Etienne, Horatiu, Hélène, Thérèse, Yves, Richard et aux nombreux étudiants avec qui j'ai pu partager les joies de l'enseignement doctoral. Paul fut notamment un camarade de thèse formidable. Nous avons travaillé ensemble sur les systèmes de superdéduction et j'ai considérablement bénéficié de la richesse de sa vision de l'informatique et des méthodes formelles. Je remercie aussi Guillaume, Émilie, Radu, Cody, Cláudia, Jean-Christophe, Tony, Lisa, Mathieu, Denis, Oana, Anderson, Florent, Colin, Antoine, Germain, Benjamin, Clara et tous les autres sans qui l'ambiance aurait été beaucoup moins studieuse.

Je souhaite aussi remercier et souligner le mérite de Dani qui a su partager les débordements inévitables du doctorat sur le domaine personnel. Son soutien inconditionnel et constant constitue l'un des premiers ingrédients de cette thèse. Enfin il va sans dire que je dois à ma famille et mes amis d'en être arrivé là : ils m'ont apporté un soutien ferme et solide tout au long de mes études. J'ai été particulièrement

honoré de voir tant de personnes qui me sont chères venir assister à ma soutenance de thèse.

Table des matières

Introduction

Contexte et objectifs de cette thèse

Le sujet de cette thèse est l'étude de divers cadres et systèmes qui sont les clés de voûte des ponts qui relient la théorie de la preuve, la logique mathématique et l'informatique. Nos points de départ sont les approches de Gentzen (1935) dans la formalisation de la logique, à savoir la déduction naturelle et le calcul des séquents. Le but de Gentzen était d'obtenir un formalisme (la déduction naturelle) aussi proche que possible du raisonnement réel. L'originalité de son approche, par rapport aux systèmes de déduction à la Hilbert, est de considérer les axiomes *non logiques* comme des paramètres locaux du mécanisme déductif alors que les *axiomes logiques* représentent la définition même de ce mécanisme. Le travail de Gentzen est le fondement de nombreux développements dans le domaine du raisonnement automatisé. Plusieurs méthodes de déduction automatique, comme la méthode des tableaux (Hähnle, 2001) ou la méthode inverse (Degtyarev et Voronkov, 2001), sont dérivées de son calcul des séquents. D'autre part, la déduction naturelle est l'un des piliers de la correspondance de Curry-De Bruijn-Howard (Howard, 1980), qui exprime le fait que les preuves en déduction naturelle sont des lambda-termes et que la bêta-réduction permet de transformer les preuves en preuves analytiques (analytique est synonyme de sans coupure) par le biais d'un processus de normalisation (cette normalisation étant représentée par le processus d'élimination des coupures dans le cas du calcul des séquents). Cette relation entre les systèmes de preuve et les modèles du calcul conduit à une large gamme de développements et de systèmes, tels que la théorie des types de Martin-Löf (1984) ou le calcul des constructions de Coquand et Huet (1988) qui est à la base par exemple de l'assistant de preuve Coq (Team, August 2009). La correspondance de Curry-De Bruijn-Howard peut être utilisée dans les deux sens : Typer des programmes peut fournir une preuve de leur correction et inversement les preuves peuvent être formalisées comme des programmes que l'utilisateur peut typer (vérifier) puis exécuter. Cette dernière orientation est rendue intéressante par la propriété du témoin vérifiée par le cadre intuitionniste : Toute preuve intuitionniste et analytique de l'existence d'un objet mathématique *contient* l'objet. Par conséquent, chaque fois que l'on a prouvé l'existence de certains objets dans la logique intuitionniste, en considérant la preuve comme un programme par le biais de la correspondance de Curry-De Bruijn-Howard, on peut exécuter le programme et obtenir une preuve analytique et intuitionniste de cette existence puis le

témoin de cette existence.

La correspondance de Curry-De Bruijn-Howard ne s'est longtemps appliquée qu'au cadre de la déduction naturelle intuitionniste jusqu'à ce que Griffin (1990) montre que des opérateurs de contrôle pouvaient être typés par la loi de Pierce. Ce travail fondateur a été suivi par une série d'études sur l'interprétation calculatoire de la logique classique d'une part, et du calcul des séquents d'autre part. Alors que la bêta-réduction est généralement considérée comme *la* procédure d'élimination des coupures de la déduction naturelle, un grand nombre de procédures d'élimination des coupures peuvent être imaginées pour le calcul des séquents (Gentzen, 1935; Curien et Herbelin, 2000; Urban, 2001; Urban et Bierman, 2001). Ces réductions peuvent ne pas vérifier la propriété de normalisation forte, voire de façon plus inquiétante la propriété de confluence. Cela nous mène à l'un des objectifs de cette thèse : un travail fondamental sur l'élimination des coupures dans le calcul des séquents visant à transposer à la fois les avantages de la correspondance de Curry-De Bruijn-Howard au calcul des séquents et les avantages du calcul des séquents classique au paradigme des assistants de preuve interactifs.

La déduction modulo (Dowek *et al.*, 2003) est une présentation récente de la logique des prédicats qui fait suite à la motivation de Gentzen de rapprocher les preuves formelles du raisonnement réel. Outre la présentation des preuves formelles d'une manière lisible, concise et compréhensible, elle vise également à renforcer les paradigmes de raisonnement automatique. La déduction modulo repose sur une dichotomie entre la déduction et le calcul qui a d'abord été soulignée par Poincaré (1902). En plus de la distinction présentée par Gentzen entre axiomes logiques et non-logiques, la déduction modulo spécialise les axiomes calculatoires de sorte qu'ils enrichissent le mécanisme déductif de base provenant des axiomes logiques au sens de Gentzen. Étant donné que le calcul est généralement *la partie la plus facile* de tout processus de raisonnement et peut être rejoué à tout moment sans avoir besoin d'aucune autre information que son point de départ, la déduction modulo propose de représenter le calcul contenu dans un raisonnement par le biais d'un système de réécriture puis d'utiliser les systèmes de déduction de Gentzen modulo la relation de réécriture correspondante. Les calculs ne figurent donc pas dans les preuves formelles, mais peuvent être vérifiées car la relation de réécriture est fournie avec le système de déduction. C'est exactement la façon dont les mathématiciens écrivent les preuves : ils écrivent seulement l'essentiel, c'est-à-dire les étapes de déduction qui sont nécessaires pour que le lecteur puisse reconstruire une preuve complète. Ils cachent les étapes de calcul afin que le lecteur ne se noie pas dans des détails de peu d'importance. Le lecteur peut alors refaire les calculs jusqu'à obtenir une preuve assez complète pour être tout à fait convaincu.

Remarquons que nous avons mis l'accent sur deux relations différentes entre logique et calcul : « le calcul comme un enrichissement de la logique » au travers du paradigme de la déduction modulo ainsi que « la logique comme modèle de calcul » au travers de la correspondance de Curry-De Bruijn-Howard. Habituellement, chacune de ces deux relations utilise une représentation du calcul qui lui est spécifique : la réécriture ou le lambda-calcul respectivement. Cependant, les mélanger est

parfois utile, comme lorsqu'on veut encoder des types dépendants en logique du premier ordre à travers le principe de la déduction modulo. Plusieurs techniques tissent des liens entre lambda-calcul et réécriture. Le lambda-calcul est d'abord lui-même un système de réécriture d'ordre supérieur. On peut néanmoins aussi l'encoder sous forme de systèmes de réécriture du premier ordre en utilisant par exemple des combinateurs (Curry et Feys, 1958; Barendregt, 1984) ou bien des indices de De Bruijn (1972) et des substitutions explicites (Abadi *et al.*, 1991; Bloo et Rose, 1995; Benaissa *et al.*, 1996). Enfin notons qu'un formalisme propose de mêler lambda-calcul et réécriture : le calcul de réécriture ou rho-calcul (Cirstea et Kirchner, 2001; Cirstea *et al.*, 2003; Barthe *et al.*, 2003) est une extension du lambda-calcul qui remplace l'abstraction et les fonctions du lambda-calcul par le filtrage par motifs et les règles de réécriture en tant qu'entités de première classe.

Les objectifs de cette thèse sont de proposer de nouveaux cadres pour représenter et manipuler des preuves formelles de manière précise et confortable. En particulier, une preuve est un message et nous garderons toujours à l'esprit que sa représentation doit rester *compréhensible, vérifiable et concise*. Si le destinataire de ce message est une machine, il est facile de caractériser ce que compréhensible et vérifiable veulent dire en terme de calculabilité et de complexité. Si le destinataire est humain, alors ces deux notions seront liées à la présentation elle-même de la preuve. En particulier pour le lecteur humain, la verbosité est l'inconvénient habituel des preuves construites par ou pour les machines. Puisque les systèmes de Gentzen sont notre point de départ, notre travail comprend tout d'abord une analyse en amont de ces systèmes. Nous nous intéressons plus particulièrement au calcul des séquents classique en analysant de façon précise les éléments qui constituent les preuves dans ce système. Plus exactement, nous proposons des moyens de représenter des preuves du calcul des séquents classiques de manière précise, concise et efficace en ce qui concerne l'élimination des coupures. Ensuite et en nous basant sur l'idée que la déduction modulo est un pas en avant vers notre objectif principal, nous proposons et étudions d'autres cadres qui se dirigent vers le même but. Introduite il y a dix ans, la déduction modulo permet d'utiliser la partie calculatoire d'une théorie $\mathcal{T}h$ dans de vrais calculs modulo lesquels la déduction opère. En focalisant sur le calcul des séquents, nous présentons et étudions le concept dual pour lequel la partie déductive d'une théorie est utilisée afin d'enrichir le système de déduction de manière systématique, correcte et complète. Ces nouveaux systèmes de déduction sont appelés « superdéduction ». Les preuves dans un système de superdéduction sont beaucoup plus proches de l'intuition et de la pratique humaine. Nous le démontrons à l'aide d'une série d'exemples incluant l'égalité et l'induction noethérienne. Nous introduisons aussi un langage de termes de preuve ainsi qu'une procédure d'élimination des coupures tous deux basés sur le travail de Christian Urban sur le calcul des séquents classique (Urban, 2000, 2001; Urban et Bierman, 2001). Nous étudions plusieurs propriétés de ces systèmes. Nous prouvons la normalisation forte du système sous des hypothèses appropriées et naturelles, ce qui implique alors la cohérence de la théorie sous-jacente et du système de déduction. En utilisant des permutations des règles d'inférence, nous prouvons l'équivalence de deux systèmes déductifs :

la superdéduction modulo sans coupure et la déduction modulo sans coupure. La littérature contient déjà de nombreux théorèmes d'élimination des coupures pour la déduction modulo, proposant chacun des hypothèses différentes (Dowek et Werner, 2003; Hermant, 2005; Burel et Kirchner, 2007). Ces hypothèses impliquent toutes l'élimination des coupures pour la déduction modulo, et notre théorème prouve qu'elles impliquent aussi l'élimination des coupures pour la superdéduction modulo. Nous proposons aussi une méthode des tableaux pour la recherche de preuves en superdéduction modulo, inspirée de la méthode des tableaux en déduction modulo proposée par Bonichon (2004); Bonichon et Hermant (2006a). L'élimination des coupures de la superdéduction modulo implique la complétude de cette méthode. L'approche qui consiste à considérer des permutations des règles d'inférence nous amène aussi à comparer la superdéduction au focusing (Andreoli, 1992; Andreoli et Maieli, 1999; Andreoli, 2001), approche qui consiste à forcer l'application des règles d'inférence permutables dans un certain ordre, et qui permet alors d'obtenir des formes canoniques représentant les preuves. Alors nous proposons un système déductif basé sur cette approche et expliquons comment le focusing, la superdéduction et la déduction modulo en sont tous trois des instances. Enfin nous montrons comment la superdéduction associée à la déduction modulo peuvent former la fondation formelle d'une nouvelle génération d'assistants de preuve en décrivant un prototype d'implémentation de la superdéduction modulo, Lemuridæ.

Context and Objectives of this Thesis

The subject of this thesis is the study of various keystone frameworks in the bridges connecting proof theory, mathematical logic and computer science. Our starting points are the approaches of Gentzen (1935) toward the formalization of logic, namely natural deduction and sequent calculus. Gentzen's goal was to build a formalism (natural deduction) as close as possible to real deduction. Compared to deduction systems *à la* Hilbert, the novelty of his approach was to consider *non-logical* axioms as local parameters to the deduction mechanism while *logical* axioms represent the definition itself of this mechanism. Gentzen's frameworks are the foundation of various developments in automated reasoning. Several methods for automated deduction such as the tableaux method (Hähnle, 2001) or the inverse method (Degtyarev et Voronkov, 2001) are derived from his sequent calculus. Furthermore natural deduction is one of the pillars of the Curry-De Bruijn-Howard correspondence (Howard, 1980) : proofs in natural deduction are lambda-terms which are transformed into cut-free proofs through beta-reduction (this normalization is represented by cut-elimination procedures in the sequent calculus case). This relation between proof systems and computation models leads to a wide range of developments and systems such as Coquand et Huet's (1988) calculus of constructions, which is the foundation of the proof assistant Coq (Team, August 2009), or Martin-Löf's (1984) type theory. The correspondence may be used in both ways : 1. Typing programs can provide a proof of their correctness ; 2. proofs can be formalized as programs, which

the user can type (check) and run. The latter orientation is particularly interesting with respect to the witness property verified by the intuitionistic case : This property states that any intuitionistic cut-free proof of the existence of a mathematical object actually contains the object. Therefore each time the existence of an object is proved in intuitionistic logic, running the proof as a program through the Curry-De Bruijn-Howard correspondence returns an intuitionistic cut-free proof of this existence and consequently an actual witness of this existence.

For a long time the Curry-De Bruijn-Howard correspondence was relevant only to intuitionistic natural deduction until Griffin (1990) demonstrated that control operators can be typed by Pierce's law. This foundational work was then followed by a series of studies in computational interpretations of both classical logic and sequent calculus. While beta-reduction is usually considered as *the* cut-elimination procedure for natural deduction, a wide range of cut-elimination procedures can be imagined for sequent calculus (Gentzen, 1935; Curien et Herbelin, 2000; Urban, 2001; Urban et Bierman, 2001). Strong normalization or confluence may not hold for these reduction procedures. This leads to one of the objectives of this thesis : a foundational work on cut-elimination in sequent calculus that aims to transpose the advantages of the Curry-De Bruijn-Howard correspondence to sequent calculus as well as the advantages of classical sequent calculus to the interactive theorem proving paradigm.

Deduction modulo (Dowek *et al.*, 2003) is a recent presentation of logic which follows Gentzen's idea that formal proofs must remain close to actual reasoning. In addition to a readable and understandable presentation of formal proofs, it also aims to enhance the automated reasoning paradigms. Deduction modulo stands on a dichotomy between deduction and computation which was first underlined by Poincaré (1902). Besides Gentzen's distinction between logical and non-logical axioms, deduction modulo puts aside computational axioms so that they enrich the basic deduction mechanism provided by the logical axioms. Deduction modulo proposes to use a rewriting in order to represent computation steps occurring in a reasoning process. Gentzen's deduction systems are then used modulo the corresponding rewrite relation. Computation is generally *the easy part* of any reasoning process and can be replayed anytime without any other information than its starting point. Therefore computations are erased from formal proofs but can be replayed during a proof checking process since the rewrite system is provided along with the deduction system. This is exactly how mathematicians write proofs : They just write the essential parts, that is to say the steps that are strictly necessary for the reader to rebuild the complete proof. They hide the computation steps so that the reader does not drown into details of small importance. He can then replay the computations until he obtains a proof that is completed enough to convince him.

Let us remark we have underlined two distinct relations between computation and logic so far : *computation as an extension of logic* through the deduction modulo paradigm as well as *logic as a computational model* through the Curry-De Bruijn-Howard correspondence. Usually each of these relations uses a specific representation of computations : rewriting and lambda-calculus respectively. Nevertheless,

mixing these representations can be useful for example to encode dependent types in first-order logic through the deduction modulo principle. Several techniques build bridges between lambda-calculus and rewriting. Lambda-calculus is itself a higher-order rewrite system. It can also be encoded as a first-order rewrite system using devices such as combinators (Curry et Feys, 1958; Barendregt, 1984) or De Bruijn (1972) indices and explicit substitutions (Abadi *et al.*, 1991; Bloo et Rose, 1995; Benaissa *et al.*, 1996). Finally a formalism mixes lambda-calculus and rewriting : The rewriting calculus (a.k.a. rho-calculus) (Cirstea et Kirchner, 2001; Cirstea *et al.*, 2003; Barthe *et al.*, 2003) is an extension of the lambda-calculus which replaces respectively the lambda-abstraction and the lambda-calculus functions by pattern matching and rewrite rules which are then first-class objects of the language.

The objective of this thesis is to propose new frameworks to represent and handle formal proofs in a precise and comfortable way. In particular a proof is a message whose representation must remain *understandable, checkable and concise*. When the recipient of this message is a machine, it is easy to define the meaning of understandable and checkable. When it is a human, then these two notions are related to the presentation of the proof. In particular for the human reader, verboseness is the usual disadvantage of proofs designed by or for machines. Since Gentzen's systems are our starting points, our work will first consist in an upstream analysis of these systems. We are specifically interested in classical sequent calculus and analyze precisely which elements are essential to proof construction. We propose representations of proofs in classical sequent calculus that are precise, concise as well as efficient with respect to cut-elimination. Then basing our analysis on the idea that deduction modulo is a step forward toward our main objective, we propose and study new frameworks that aim for the same goal. Introduced ten years ago, deduction modulo allows the reasoning mechanism to be performed modulo real computations. If $\mathcal{T}h$ is some theory, deduction modulo proposes to use its computational part to enrich the deduction process with the calculations that this part represents. We present and study the dual concept in which the deductive part of a theory is used to enrich the deduction system in a systematic, sound and complete way. These new systems are called *superdeduction*. Proofs in a superdeduction system are much closer to human intuition and practice. We demonstrate this fact with several examples such as equality and noetherian induction. We also introduce a proofterm language and a cut-elimination procedure both based on Urban's work on classical sequent calculus (Urban, 2000, 2001; Urban et Bierman, 2001). We study several properties of these systems : First, we prove strong normalization under appropriate and natural hypotheses. It implies the consistency of both the superdeduction system and the underlying theory. Furthermore using permutations of inference rules, we prove the equivalence of cut-free superdeduction modulo and cut-free deduction modulo. Various cut-elimination theorems have already been proved for deduction modulo. Each one proposes its own set of hypotheses (Dowek et Werner, 2003; Hermant, 2005; Burel et Kirchner, 2007). Each set of hypotheses implies cut-elimination for deduction modulo. Our theorem then proves they also imply cut-elimination for superdeduction modulo. We also design a tableaux method for proof search in super-

deduction modulo inspired by the tableaux method for deduction modulo designed by Bonichon (2004); Bonichon et Hermant (2006a). Cut-elimination of the corresponding superdeduction system implies then the completeness of this method. We consider permutations of inference rules and compare of superdeduction with focusing (Andreoli, 1992; Andreoli et Maieli, 1999; Andreoli, 2001). Focusing consists in specifying an order for applying inference rules depending on their permutability. It allows to find canonical forms representing proofs. We describe a deduction system based on this approach and explain how focusing, superdeduction and deduction modulo are instances of this system. Finally we describe an implementation prototype for superdeduction modulo called Lemuridæ and show how superdeduction modulo can be the formal foundation of a new generation of proof assistants.

Plan

Première partie

En première partie, je présente une série d'outils qui peuvent être combinés de différentes manières pour obtenir de tels cadres.

Au chapitre 1, je présente la réécriture, le lambda-calcul et le calcul de réécriture qui permettent tous les trois de modéliser le calcul.

Au chapitre 2, je présente les systèmes déductifs de Gentzen : la déduction naturelle et le calcul des séquents.

Au chapitre 3, J'explique comment ces paradigmes se combinent dans la correspondance de Curry-De Bruijn-Howard. J'y présente un état de l'art détaillé des systèmes de typages pour la déduction naturelle et le calcul des séquents pour la logique intuitionniste ou classique.

Au chapitre 4, j'étudie la problématique de canonicité des preuves et je décris comment l'approche des réseaux de preuve permet d'apporter des éléments de réponse. J'expose une méthode originale de visualisation des séquents et des réseaux de preuve respectivement dans des espaces à deux et trois dimensions mettant en valeur la structure logique de ces objets.

Deuxième partie

En deuxième partie, j'étudie plusieurs paradigmes permettant de modifier les systèmes déductifs de Gentzen afin de construire des systèmes spécialisés pour une théorie donnée telle que l'arithmétique ou la géométrie dans le plan.

Au chapitre 5, je présente la déduction modulo permettant de rajouter à un système déductif un pouvoir calculatoire en utilisant par exemple le paradigme de réécriture. Après avoir exposé quelques contre-exemples à la normalisation et à l'élimination des coupures, je décris des techniques permettant de les assurer pour certaines classes de théories : j'expose l'approche de Gilles Dowek et la notion de superconsistance, l'approche sémantique de Olivier Hermant

ainsi que l'approche par complétion de Guillaume Burel et Claude Kirchner. Je décris aussi les méthodes de démonstration automatique de résolution et narrowing ENAR et des tableaux TaMeD basées sur la déduction modulo en expliquant pourquoi l'admissibilité des coupures est essentielle pour la complétude de ces méthodes.

Au chapitre 6, je présente le paradigme de superdéduction personnifié par les systèmes de superdéduction naturelle ainsi que par les systèmes de superdéduction. Je montre que se posent les mêmes problèmes d'admissibilité des coupures et de normalisation que dans le cas de la déduction modulo. À l'aide d'un langage de termes de preuve pour la superdéduction ainsi que d'une procédure d'élimination des coupures tous deux inspirés de la thèse de Christian Urban, je prouve un résultat de normalisation forte pour la superdéduction. J'expose aussi comment combiner déduction modulo et superdéduction puis je présente une méthode des tableaux pour les systèmes de superdéduction modulo obtenus basée sur la méthode TaMeD de Richard Bonichon pour la déduction modulo. Enfin pour démontrer les bienfaits des systèmes de superdéduction appliqués à la conception de systèmes de preuve interactifs, je décrit le prototype d'assistant à la preuve Lemuridæ que j'ai programmé avec Paul Brauner.

Au chapitre 7 et en me basant sur la permutabilité des inférences de LK, je montre comment une hypothèse de synchronicité permet de traduire toute preuve en déduction modulo sans coupure en une preuve en superdéduction modulo sans coupure et inversement. J'obtiens ainsi un résultat d'équivalence entre ces deux systèmes déductifs : en particulier sous cette même hypothèse de synchronicité, l'admissibilité des coupures dans un système implique l'admissibilité des coupures dans l'autre. Enfin en comparant superdéduction et focusing, je décris un traitement des axiomes similaire à celui de la superdéduction basé sur un paradigme de multifocusing pour la logique classique. Je montre alors que la complétude des systèmes obtenus n'est plus conditionnée par les hypothèses de synchronicité conditionnant la complétude des systèmes de superdéduction.

Contributions

Parmi les développements exposés dans cette thèse, voici mes contributions :

1. J'ai démontré la confluence du rho-calcul en appel par valeur (propriété 1.3.2) en sous-section 1.3.3 et j'ai mis au point un encodage des systèmes de réécriture de termes sans restriction de convergence dans le calcul de réécriture présenté en sous-section 1.3.5 avec le concours de Horatiu Cirstea et Benjamin Wack (Cirstea *et al.*, 2006a).
2. Les représentations des séquents en deux dimensions et des réseaux de preuve en trois dimensions exposées en section 4.4 font aussi partie de mes contributions personnelles. Je les ai exposées dans un article en cours (Houtmann, 2009).

3. Avec Paul Brauner et Claude Kirchner (Brauner *et al.*, 2007a,b), nous avons défini les systèmes de superdéduction pour le calcul des séquents classique présentés en sous-section 6.1.2, nous avons montré leur correction et complétude (propriétés 6.1.4 et 6.1.5) et nous avons développé le langage de termes de preuve exposé en sous-section 6.2.2 et la preuve de normalisation forte (propriété 6.2.6).
4. J'ai montré en section 7.1 comment des hypothèses de synchronicité permettent d'écrire une traduction des dérivations sans coupure en déduction modulo vers les dérivations sans coupure en superdéduction modulo et inversement (Houtmann, 2008a) (propriété 7.1.9). Cela permet ensuite de transférer les nombreux résultats d'admissibilité des coupures existants pour la déduction modulo au cadre des systèmes de superdéduction modulo. J'ai ainsi étendu la méthode des tableaux TaMeD de Richard Bonichon en une méthode des tableaux pour la superdéduction modulo exposée en section 6.4 et dont la complétude est impliquée par l'admissibilité des coupures du système de superdéduction modulo correspondant.
5. Les résultats de complétude, de normalisation forte ou d'admissibilité des coupures (en particulier les propriétés 6.1.5, 6.2.6 et 7.1.9) sont conditionnées par des hypothèses de synchronicité qui requièrent que les membres droits des règles de prédicats soient des monopôles ou des hyperpôles. Paul Brauner, Claude Kirchner et moi-même (Brauner *et al.*, 2007a) avons proposé les procédures de monopolarisation et d'hyperpolarisation exposées en section 7.2 permettant de transformer tout ensemble de règles de prédicats en un ensemble équivalent vérifiant ces hypothèses.
6. Avec Paul Brauner, nous avons programmé un prototype d'assistant à la preuve que nous avons appelé Lemuridæ. Je le présente en section 6.5. Basé sur la superdéduction modulo, celui-ci propose d'étendre naturellement le calcul des séquents classique en utilisant des règles de réécriture. Ce prototype permet le développement interactif de preuves puis leur vérification. Il est aussi capable de traduire des types inductifs en superdéduction, de définir des tactiques de démonstration complexes grâce au langage de stratégies de Tom (le langage d'implantation de Lemuridae), ou encore d'appliquer la procédure d'hyperpolarisation garantissant ainsi la complétude du système.
7. En section 7.3, j'ai poursuivi la comparaison du paradigme de superdéduction aux connecteurs synthétiques définis par le focusing que j'avais amorcée dans l'article publié dans les post-proceedings de TYPES'08 (Houtmann, 2008a). Cela m'a amené à définir un système de multifocusing permettant de traiter des axiomes en utilisant les connecteurs synthétiques du focusing et dont la complétude n'est plus conditionnée par les hypothèses de synchronicité conditionnant la complétude des systèmes de superdéduction.

Première partie

Calcul et déduction

Chapitre 1

Systèmes de calcul formel

Représenter le calcul est un enjeu majeur pour les informaticiens : les langages de programmation sont destinés à répondre à cette nécessité pratique qui implique cependant généralement un besoin sémantique. En effet, les logiciels ne devraient être considérés comme dignes de confiance qu'une fois que l'on a *formellement* défini la manière dont le langage de programmation agit. Ce besoin sémantique peut être satisfait au moyen d'un système formel (qui peut comprendre du calcul formel) qui constitue alors la *sémantique* du langage de programmation. Dans le cadre de la confiance que nous accordons aux machines effectuant nos calculs, proposer un modèle de calcul n'est généralement pas suffisant. En effet il faut aussi prouver un théorème mettant en adéquation le calcul effectif et son modèle. On peut par exemple rechercher à démontrer que certaines bonnes propriétés démontrables pour le modèle s'appliquent aussi au cas pratique. Ces propriétés peuvent être prouvées une bonne fois pour toutes dans le cadre de la définition de la compilation et de l'exécution des programmes. Une approche différente est de générer une propriété d'adéquation propre au programme en question à chaque compilation. Il faut noter qu'au cours du XX^{e} siècle, la croissance de la complexité des machines de calcul s'est accompagnée d'une croissance de la complexité de la programmation de ces machines. L'analyse, au moyen des modèles de calcul dont il est ici question, des programmes qui sont alors produits et la preuve de propriétés d'adéquation telles que nous venons de les décrire sont des tâches qui deviennent aussi d'une complexité considérable. Par conséquent il s'est révélé très efficace de reléguer ces tâches au moins partiellement aux machines de calcul elles-mêmes grâce à des prouveurs automatiques ou interactifs. Finalement la confiance que nous accordons aux programmes certifiés se réduit à

- la confiance que nous accordons à la mise en oeuvre du mécanisme de *vérification* des preuves ;
- la confiance que nous accordons à la logique sous-jacente en supposant sa cohérence.

Remarquons que les avancées dans le domaine des techniques de conception des processeurs a donc directement impliqué un besoin du développement des domaines

de l'automatisation du raisonnement (prouveurs automatiques ou interactifs) ainsi que de la logique formelle. Une application de cette croissance de l'informatique au domaine des mathématiques pures est la production à l'aide de techniques de preuve assistée par ordinateur de preuves de théorèmes purement mathématiques (qui ne sont pas motivés par un besoin de formalisation des machines de calcul) restés à l'état de conjectures avant l'avènement du calcul automatisé à grande échelle comme par exemple le célèbre théorème des quatres couleurs (Gonthier, 2007).

Notons que la formalisation du calcul, bien qu'étant un pilier de la certification de programmes informatiques, est un effort qui fut initié par les mathématiciens et les logiciens avant que ce besoin de certification n'apparaisse. En particulier c'est en 1932 que Church (1933-1934) introduisit le lambda-calcul. Son but était d'apporter un élément de réponse aux questions posées par la crise des fondements que traversaient alors les mathématiques et il présenta le lambda-calcul comme un ensemble de postulats pouvant servir de fondation aux mathématiques. C'est, entre autres, cette présentation du lambda-calcul qui mena plus tard à la formalisation de la théorie plus générale de la réécriture. Cette dernière théorie est plus générale dans le sens bien connu que le lambda-calcul est en fait un système de réécriture d'ordre supérieur, mais aussi dans le sens que c'est l'étude du lambda-calcul, parmi d'autres théories équationnelles, qui amena à la définition de système de réécriture comme un cadre d'étude de cette sorte de formalisme. Notons par exemple que la propriété de Church-Rosser en théorie de la réécriture s'appelle ainsi car elle provient directement du théorème de Church et Rosser (1936) prouvé initialement pour le lambda-calcul.

La section 1.1 du présent chapitre est donc dédiée à la théorie générale de la réécriture. La section 1.2 contient une présentation du lambda-calcul pur (c'est-à-dire non typé). Les sections suivantes présentent des extensions intéressantes de ces systèmes : la section 1.3 présente le calcul de réécriture (aussi appelé rho-calcul) qui est une extension directe du lambda-calcul proposant de remplacer l'abstraction simple (sur des variables) par une abstraction sur des motifs arbitraires.

1.1 Systèmes de réécriture de termes

1.1.1 Notions préliminaires sur les relations binaires

Nous présentons ici quelques notions préliminaires sur les relations binaires. Habituellement ces notions sont présentées dans le cadre général des *systèmes de réécriture abstraits* (Terese, 2003). Dans notre cas, la notion de relation binaire sera amplement suffisante pour les introduire. Commençons donc tout d'abord par quelques notations.

Notation 1.1.1. *Si $\rightarrow$ est une relation binaire sur un ensemble $\mathcal{A}$ (c'est-à-dire un sous-ensemble de $\mathcal{A} \times \mathcal{A}$), alors nous écrirons*

- *$a \rightarrow b$ pour $(a, b) \in \rightarrow$;*
- *$\leftarrow$ pour la relation $\{(a, b)/b \rightarrow a\}$;*

– $\leftrightarrow$ *pour* $\rightarrow \cup \leftarrow$ *;*
– $\rightarrow^+$ *pour la clôture transitive de* $\rightarrow$ *;*
– $\rightarrow^{0,1}$ *pour la clôture réflexive de* $\rightarrow$ *;*
– $\rightarrow^*$ *pour la clôture transitive et réflexive de* $\rightarrow$*.*

Nous dirons que a *se réduit en* b *(par* $\rightarrow$*) si* $a \rightarrow^* b$*. Si* $\rightarrow_1$ *et* $\rightarrow_2$ *sont deux relations binaires sur un même ensemble, alors la relation binaire* $\rightarrow_1 \circ \rightarrow_2$ *est définie par «* $a \rightarrow_1 \circ \rightarrow_2 b$ *s'il existe* c *tel que* $a \rightarrow_1 c \rightarrow_2 b$ *». Une séquence finie de réduction pour* $\rightarrow$ *est une suite finie* $(a_i)_{0 \leqslant i \leqslant n}$ *telle que pour tout* $0 \leqslant i \leqslant n-1$*,* $a_i \rightarrow a_{i+1}$*. Nous dirons que* a *admet une séquence de réduction de taille* n *quand il existe une telle suite de taille* $n+1$ *avec* $a_0 = a$*. Une séquence infinie de réduction pour* $\rightarrow$ *est une suite infinie* $(a_i)_{i \in \mathbb{N}}$ *telle que pour tout* $i \in \mathbb{N}$*,* $a_i \rightarrow a_{i+1}$*. Nous dirons que* a *admet une séquence de réduction infinie quand il existe une telle séquence infinie avec* $a_0 = a$*.*

Il est pratique d'utiliser des relations binaires afin de représenter le calcul :
– $a \rightarrow b$ se traduit par « une étape de calcul transforme a en b » ;
– $a \rightarrow^* b$ se traduit par « des étapes de calcul transforment a en b » .

Dans ce cas, la notion de forme normale représente les résultats *terminaux* opposés aux résultats *intermédiaires*.

Définition 1.1.1 (Forme normale). *Soit* $\rightarrow$ *une relation binaire sur un ensemble* $\mathcal{A}$*. Une forme normale (pour* $\rightarrow$*) est un élément* a *de* $\mathcal{A}$ *tel qu'il n'existe aucun* $b \in \mathcal{A}$ *tel que* $a \rightarrow b$*. De plus si* c *se réduit en* a*, nous dirons que* a *est une forme normale de* c*.*

Toujours dans l'optique de représenter du calcul par des relations binaires,
– la notion de normalisation faible représente l'existence d'un calcul menant à un résultat terminal ;
– la notion de normalisation forte représente le fait que tous les calculs mènent à un résultat terminal.

Définition 1.1.2 (Normalisation faible). *Soit* $\rightarrow$ *une relation binaire sur un ensemble* $\mathcal{A}$*.*

i *Un élément* $a \in \mathcal{A}$ *est faiblement normalisant pour* $\rightarrow$ *s'il se réduit en une forme normale.*

ii *La relation* $\rightarrow$ *est faiblement normalisante si tous les éléments de* $\mathcal{A}$ *sont faiblement normalisants pour* $\rightarrow$*.*

Habituellement, les éléments fortement normalisants sont définis comme les éléments n'admettant aucune séquence de réduction infinie. Afin d'éviter la double négation « *aucune* réduction *in*finie », on peut choisir de les définir comme les éléments pour lesquels *toute séquence de réduction* est finie. Néanmoins cette formulation a l'inélégance d'utiliser une quantification sur les séquences de réduction *finies ou infinies*. Deux autres formulations sont possibles : l'une est inductive, l'autre est la définition usuelle de relation bien fondée.

Définition 1.1.3 (Normalisation forte). *Soit $\to$ une relation binaire sur un ensemble $\mathcal{A}$. Soit un élément $a \in \mathcal{A}$. Alors les propositions suivantes sont équivalentes.*

i a *n'admet aucune séquence de réduction infinie.*

ii *Toutes les séquences de réduction de a sont finies, c'est-à-dire si $(a_i)_{i\in\mathbb{N}}$ est une suite d'éléments de $\mathcal{A}$ avec $a_0 = a$ et $a_i \to^{0,1} a_{i+1}$ pour tout $i \in \mathbb{N}$, alors l'ensemble $\{i / a_i \to a_{i+1}\}$ est fini.*

iii *a est dans le plus petit ensemble $\mathcal{SN}$ tel que si pour tout c tel que $b \to c$, alors $c \in \mathcal{SN}$, alors $b \in \mathcal{SN}$.*

iv *La relation $\to$ est bien fondée sur l'ensemble $(a)^* = \{b / a \to^* b\}$, c'est-à-dire pour tout sous-ensemble non vide A de $(a)^*$, il existe un élément b de a tel que pour tout $c \in A$, $b \not\to c$.*

Un élément $a \in \mathcal{A}$ est fortement normalisant s'il vérifie ces propriétés. La relation $\to$ est fortement normalisante si tous les éléments de $\mathcal{A}$ sont fortement normalisants pour $\to$.

Remarquons que nos trois dernières formulations sont plus fortes que la première en logique intuitionniste car elles évitent la double négation « *aucune* réduction *in*finie ». Lorsque l'on suppose l'axiome du choix dépendant, on peut par exemple démontrer que la formulation i implique la formulation iii.

Démonstration de i $\Rightarrow$ iii. Considérons l'ensemble $\mathcal{SN}'$ des éléments a de $\mathcal{A}$ pour lesquels il n'existe pas de telle suite infinie. Considérons l'ensemble $\mathcal{SN}$ défini comme étant le plus petit ensemble tel que pour tout $b \in \mathcal{A}$

$$\text{si} \quad \text{pour tout } c \text{ tel que } b \to c, \text{ alors } c \in \mathcal{SN} \quad \text{, alors} \quad b \in \mathcal{SN}.$$

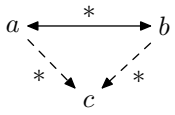
Montrons que $\mathcal{SN}' \subseteq \mathcal{SN}$ par contraposée. Soit donc $a \notin \mathcal{SN}$. On définit alors une suite infinie $(a_i)_{i\in\mathbb{N}}$ telle que pour tout $i \in \mathbb{N}$, $a_i \notin \mathcal{SN}$. Cette suite est définie inductivement comme suit : $a_0 = a \notin \mathcal{SN}$; si $a_i \notin \mathcal{SN}$, alors par définition de $\mathcal{SN}$ il existe $b \in \mathcal{A}$ tel que $a_i \to b$ et $b \notin \mathcal{SN}$ et nous choisissons $a_{i+1} = b$ (nous utilisons ici l'axiome du choix dépendant). Nous obtenons ainsi une suite infinie $(a_i)_{i\in\mathbb{N}}$ telle que $a_0 = a$ et pour tout $i \in \mathbb{N}$, $a_i \to a_{i+1}$. Donc $a \notin \mathcal{SN}'$. □

Lorsqu'il n'y a aucune ambiguïté sur la relation $\to$, nous dirons « faiblement normalisant » et « fortement normalisant » au lieu de « faiblement normalisant pour $\to$ » et « fortement normalisant pour $\to$ ». Nous noterons respectivement $\mathcal{SN}_\to$ et $\mathcal{WN}_\to$ ($\mathcal{SN}$ et $\mathcal{WN}$ lorsqu'il n'y a aucune ambiguïté) les éléments fortement normalisants et faiblement normalisants pour $\to$. Remarquons que tout élément fortement normalisant est faiblement normalisant et que toute relation fortement normalisante est faiblement normalisante.

Les propriétés de Church-Rosser, de confluence locale et de confluence sont trois autres concepts importants lorsqu'une relation binaire représente une notion de calcul. La première est définie comme une généralisation de la propriété que Church et Rosser (1936) ont prouvée pour le lambda-calcul . Cette propriété peut se formuler de la manière suivante.

Définition 1.1.4 (Propriété de Church-Rosser). *Une relation binaire $\to$ sur un ensemble $\mathcal{A}$ a la propriété de Church-Rosser si pour tous a et $b \in \mathcal{A}$ tels que $a \leftrightarrow^* b$, il existe $c \in \mathcal{A}$ tel que $a \to^* c$ et $b \to^* c$.*

La propriété de Church-Rosser est habituellement représentée par le diagramme



où l'arête pleine représente le lien universellement quantifié $a \leftrightarrow^* b$ alors que les arêtes pointillées représentent les liens existentiellement quantifiés $a \to^* c$ et $b \to^* c$. On peut aussi définir cette propriété par $\leftrightarrow^* \subseteq \to^* \circ \leftarrow^*$. Les notions de confluence et de confluence locale s'expriment de la manière suivante.

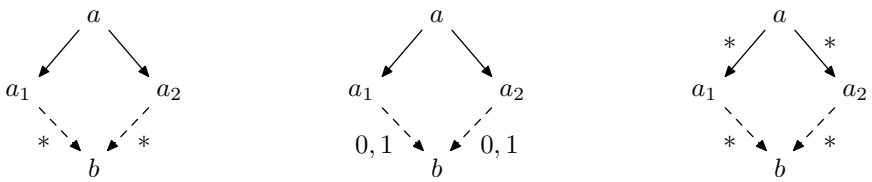
Définition 1.1.5 (Confluence). *Soit $\to$ une relation binaire sur un ensemble $\mathcal{A}$.*

i Elle est localement confluente si pour tous a, a_1 et $a_2 \in \mathcal{A}$ tels que $a \to a_1$ et $a \to a_2$, il existe $b \in \mathcal{A}$ tel que $a_1 \to^* b$ et $a_2 \to^* b$.

ii Elle est fortement confluente si pour tous a, a_1 et $a_2 \in \mathcal{A}$ tels que $a \to a_1$ et $a \to a_2$, il existe $b \in \mathcal{A}$ tel que $a_1 \to^{0,1} b$ et $a_2 \to^{0,1} b$.

iii Elle est confluente si pour tous a, a_1 et $a_2 \in \mathcal{A}$ tels que $a \to^* a_1$ et $a \to^* a_2$, il existe $b \in \mathcal{A}$ tel que $a_1 \to^* b$ et $a_2 \to^* b$.

Puisque $(\to^*)^* = \to^*$, la relation $\to$ est confluente si et seulement si $\to^*$ est localement confluente si et seulement si $\to^*$ est fortement confluente si et seulement si $\to^*$ est confluente. D'une manière similaire à la propriété de Church-Rosser, la confluence locale, la confluence forte et la confluence peuvent être représentées par les diagrammes suivants.

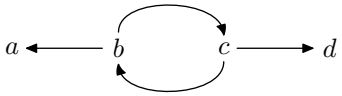


Remarquons que ces trois notions sont respectivement équivalentes à

$$\leftarrow \circ \to \subseteq \to^* \circ \leftarrow^* \quad , \qquad \leftarrow \circ \to \subseteq \to^{0,1} \circ \leftarrow^{0,1}$$

$$\text{et} \qquad \leftarrow^* \circ \to^* \subseteq \to^* \circ \leftarrow^* \quad .$$

Plus précisément la définition 1.1.5 contient les formes prenexes de ces trois formules. Il est nécessaire de remarquer que la confluence locale seule n'implique pas la confluence, comme le démontre le contre-exemple suivant.



Par contre, la confluence forte implique la confluence comme démontré par Newman (1942).

Propriété 1.1.1. *Une relation fortement confluente est toujours confluente.*

La propriété de Church-Rosser et la confluence sont deux propriétés totalement équivalentes.

Propriété 1.1.2. *Une relation admet la propriété de Church-Rosser si et seulement si elle est confluente.*

Dans le cas où la relation binaire représente un calcul, la confluence locale exprime le fait que deux *étapes* de calcul partant d'un même objet a peuvent toujours être respectivement complétées afin d'obtenir un même résultat (intermédiaire ou terminal). De même, la confluence exprime le fait que deux *séquences* d'étapes de calcul partant d'un même objet a peuvent toujours être respectivement complétées afin d'obtenir un même résultat. Il faut remarquer qu'une étape (ou une séquence d'étapes) qui se termine sur une forme normale ne peut être complétée que par une séquence de calcul *vide* : si a est une forme normale et $a \to^* b$, alors $a = b$. Alors la confluence d'une relation binaire a les conséquences suivantes :

i tout élément faiblement normalisant admet une unique forme normale ;

ii si la relation est faiblement normalisante, alors tout élément a admet une unique forme normale que nous noterons $\downarrow a$;

Une autre conséquence intéressante de la confluence est la suivante : montrer la cohérence de $\leftrightarrow^*$ en tant que théorie équationnelle se réduit à trouver deux formes normales distinctes $a \neq b$. En effet, la cohérence d'une théorie équationnelle est définie comme l'existence d'un modèle *non trivial* de cette théorie, ce qui est équivalent à l'existence de a et b tels que $a \not\leftrightarrow^* b$ (comme défini par exemple pour le lambda-calcul par Barendregt (1984)). Deux formes normales distinctes $a \neq b$ satisfont obligatoirement $a \not\leftrightarrow^* b$. Sinon par confluence de $\to$, $a \to^* \circ \leftarrow^* b$ et donc comme a et b sont des formes normales, $a = b$.

Nous avons déjà montré que la confluence locale n'impliquait pas la confluence dans le cas général. Néanmoins, cette implication est valide dans le cas d'une relation fortement normalisante, comme l'a démontré Newman (1942) puis Huet (1980) par une preuve simplifiée.

Propriété 1.1.3 (Lemme de Newman). *Si une relation est à la fois fortement normalisante et localement confluente, alors elle est confluente.*

Nous dirons qu'une relation est convergente si elle est à la fois fortement normalisante et confluente. L'intérêt de la convergence d'une relation réside dans le fait que tout élément admet une unique forme normale qu'on peut calculer simplement par réductions successives : la normalisation forte nous assure que ce processus termine et la confluence nous assure qu'on obtiendra le même résultat quelques soient les réductions que l'on choisit. C'est aussi une notion clé lorsque l'on s'intéresse au

problème du mot : pour une certaine théorie équationnelle E, on cherche à trouver un algorithme permettant de décider $t =_E u$ pour tous termes t et u. Trouver une relation convergente $\to$ telle que $t =_E u$ si et seulement si $t \leftrightarrow^* u$ est une façon de résoudre ce problème : il suffit alors de calculer $\downarrow t$ et $\downarrow u$ (ce qui est rendu possible par la convergence de $\to$). Alors $t =_E u$ si et seulement si ces deux formes normales sont syntaxiquement égales. Notons tout de même que trouver une telle relation convergente est un problème indécidable en général. En outre, il existe des théories équationnelles pour lesquelles le problème du mot est décidable mais qui n'admettent pas de telle relation convergente (Terese, 2003, exercice 2.6.2.).

Enfin nous utiliserons aussi le lemme de Kőnig (où plutôt une application au cadre de la réécriture).

Définition 1.1.6 (Branchements finis). *Nous dirons qu'une relation $\to$ sur un ensemble $\mathcal{A}$ est à branchements finis si pour tout élément $a \in \mathcal{A}$, l'ensemble $\{b \in \mathcal{A} / a \to b\}$ des réduits en un pas de a est fini.*

Propriété 1.1.4 (Lemme de Kőnig). *Si une relation $\to$ sur un ensemble $\mathcal{A}$ est à branchement fini et si $a \in \mathcal{A}$ admet un nombre infini de séquences finies de réduction, alors a admet une séquence infinie de réduction.*

Démonstration. Nous définissons une suite infinie $(a_i)_{i\in\mathbb{N}}$ d'éléments de $\mathcal{A}$ admettant tous un nombre infini de séquences finies par récurrence comme suit.

- $a_0 = a$. Notons que a admet bien un nombre infini de séquences finies.
- Supposons que nous avons déjà construits les a_k pour $k \leqslant i$. Puisque l'ensemble des réduits en un pas de a_i est fini mais que l'ensemble des séquences finies de réduction de a_i est infini, alors il existe b un réduit en un pas de a_i admettant un nombre infini de séquences finies de réduction. Nous choisissons alors $a_{i+1} = b$ (nous utilisons ici l'axiome du choix dépendant).

Nous avons alors montré que a admet une séquence infinie de réduction. □

1.1.2 Algèbres de termes

Nous allons à présent décrire de manière formelle les univers qui vont représenter les objets sur lesquels la réécriture opère dans le cadre des systèmes de réécriture de termes. Ces objets sont donc la représentation des données que les processus de calcul vont manipuler. À l'instar de la représentation des données dans une machine de Turing ou sur un disque dur, nous utiliserons ici des suites finies de symboles. Soit donc $(\mathcal{S}^*, \cdot)$ un monoïde libre généré par un ensemble de symboles $\mathcal{S}$. Pour tous éléments a et b de $\mathcal{S}^*$, nous écrirons habituellement ab à la place de $a \cdot b$. Similairement pour tout suite finie $(a_i)_{1\leqslant i\leqslant n}$, nous noterons $a_1 a_1 \dots a_n$ à la place de $a_1 \cdot a_2 \dots a_n$. Nous supposerons que $\mathcal{S}$ contient trois symboles réservés (dans le sens que ces symboles n'appartiendront à aucun sous-ensemble de $\mathcal{S}$ que nous utiliserons plus tard) « (», «) » et « , ». Dans ce contexte, une *signature* correspond à la déclaration d'un ensemble de symboles de fonction ainsi que de leurs arités respectives.

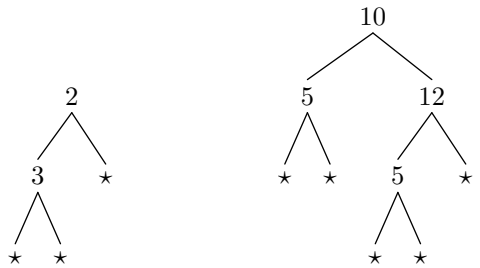
Définition 1.1.7 (Signature). *Une signature est une paire (F, n) contenant un sous-ensemble F de $\mathcal{S}$ ainsi qu'une fonction n de F vers $\mathbb{N}$ appelée* fonction d'arité.

Pour tout symbole $f \in F$, $n(f)$ sera donc appelé l'arité de f. Pour tout entier naturel k, les éléments de $n^{-1}(k)$ sont donc les symboles de fonction d'arité k. Les symboles de fonction d'arité 0 sont aussi appelés des symboles de constante ou plus simplement des constantes, les symboles de fonction d'arité $1, 2, 3\ldots$ sont respectivement appelés des symboles unaires, binaires, ternaires et ainsi de suite. Par abus de langage, nous confondrons généralement une signature et l'ensemble de ses symboles de fonctions. La notion d'algèbre de termes correspond alors aux objets que l'on a le droit de construire en utilisant ces symboles de fonctions. En particulier, ces objets se doivent de respecter l'arité des symboles de fonction.

Définition 1.1.8 (Algèbre de termes). *Soit une signature $\alpha = (F, n)$ et X un sous-ensemble de $\mathcal{S}$ (les éléments de X seront appelés des* variables*) tels que $X \cap F = \emptyset$. L'algèbre de termes $T_\alpha(X)$ est le plus petit ensemble vérifiant la condition suivante : pour tout $f \in F$, pour toute suite finie $(a_i)_{1 \leqslant i \leqslant n(f)}$ telle que pour tout i, $a_i \in T_\alpha(X)$, $f(a_1, \ldots, a_{n(f)}) \in T_\alpha(X)$.*

Les éléments d'une algèbre de termes $T_\alpha(X)$ sont appelés habituellement des *termes*. Si f est un symbole d'arité nulle ($n(f) = 0$), la notation $f(a_1, \ldots, a_{n(f)})$ utilisée dans la définition représente en fait $f()$ car la suite $(a_i)_{1 \leqslant i \leqslant n(f)}$ est en fait la suite vide. Alors pour toute constante a de α, $a()$ est un élément de $T_\alpha(X)$ que confondrons généralement avec a. Nous utiliserons les lettres $a, b, c\ldots$ pour représenter des constantes, les lettres $f, g, h\ldots$ pour représenter des symboles de fonction d'arité $n > 0$, les lettres $t, u, v\ldots$ pour représenter des termes et les lettres $x, y, z\ldots$ pour représenter des variables. Nous utiliserons parfois une notation infixe en écrivant par exemple $2 + 3$ au lieu de $+(2, 3)$. La notion d'algèbre de termes est proche de celle de *type de donnée algébrique* : la signature correspond à la déclaration du type algébrique et l'algèbre de termes peut être vue comme son modèle concrêt. Les arbres binaires sont l'exemple paradigmatique du type de donnée algébrique. On peut aisément les représenter en utilisant une algèbre de termes.

Exemple 1.1.1 (Arbres binaires sur les entiers naturels). *Soit F l'union disjointe $\{\star\} \cup \mathbb{N}$ et $n : F \to \mathbb{N}$ tel que $n(\star) = 0$ et $n(k) = 2$ pour tout $k \in \mathbb{N}$. Alors l'algèbre de termes $T_{(F,n)}(\emptyset)$ est l'ensemble des arbres binaires sur les entiers naturels. Les termes $2(3(\star, \star), \star)$ et $10(5(\star, \star), 12(5(\star, \star), \star))$ représentent respectivement les arbres suivants.*



Remarquons que toute algèbre de termes $T_\sigma(X)$ peut être décrite par la grammaire hors contexte suivante :

- les symboles terminaux sont les symboles de notre monoïde libre des symboles ;
- le seul symbole non terminal est S ;
- les règles de production sont

$$\begin{array}{ll} S \to x & \text{si } x \in X \\ S \to f(\underbrace{S, \ldots, S}_{n \text{ fois}}) & \text{si } f \text{ est un symbole de fonction de } \sigma \text{ avec arité } n\ . \end{array}$$

Soit à présent une algèbre de termes quelconque $T_\alpha(X)$.

Les symboles de variable, eux-mêmes éléments de l'algèbre de termes, représentent des objets ayant une signification particulière. Comme leur nom l'indique, ces symboles sont des termes auquels on peut attribuer différentes *valeurs* prises dans un certain ensemble. Ce point de vue oppose bien évidemment les symboles de variable aux symboles de fonction. L'opposition est soulignée par l'utilisation des mots « variable » et « constante » qui sont antonymes dans le langage courant. Il serait alors logique de décréter qu'une *valeur* est un terme construit sans aucune variable, uniquement avec des symboles de fonction. Afin de ne pas semer la confusion (le mot « valeur » pourrait très bien être utilisé pour désigner les formes normales, résultats finaux de l'*évaluation* du calcul), on appelle les termes sans symbole de variable des *termes clos*. Nous allons à présent les définir formellement.

L'ensemble des variables d'un terme est défini de la manière suivante.

Définition 1.1.9 (Ensemble des variables). *L'ensemble des variables $\mathcal{V}(t)$ d'un terme $t \in T_\alpha(X)$ est défini inductivement par*

$$\begin{array}{ll} \mathcal{V}(t) = \{x\} & \text{si } t = x \in X \\ \mathcal{V}(t) = \bigcup_{1 \leqslant i \leqslant n} \mathcal{V}(t_i) & \text{si } t = f(t_1, \ldots, t_n) \end{array}$$

L'ensemble des termes clos est défini de la façon suivante.

Définition 1.1.10 (Termes clos). *Les* termes clos *sont les éléments de l'ensemble $\{a \in T_\alpha(X) / \mathcal{V}(a) = \emptyset\}$ qui correspond d'ailleurs à l'algèbre de termes $T_\alpha(\emptyset)$.*

Nous allons maintenant introduire diverses notions qui nous apporterons une certaine flexibilité pour décrire des opérations sur les termes. Tout d'abord pour tout terme t, l'ensemble des sous-termes de t est défini comme suit.

Définition 1.1.11 (Sous-terme). *Soit t un terme de $T_\alpha(X)$. Alors l'ensemble des sous-termes de t, dénoté* sub(t)*, est défini récursivement par*

$$\begin{array}{ll} \text{sub}(t) = \{t\} & \text{si } t \text{ est une variable ou une constante ;} \\ \text{sub}(t) = \{t\} \cup \left(\bigcup_{1 \leqslant i \leqslant n} \text{sub}(t_i) \right) & \text{si } t = f(t_1, \ldots, t_n) \end{array}$$

La relation de sous-terme $\sqsubseteq$ est définie comme $\{(t, t') / t \in \text{sub}(t')\}$.
La relation de sous-terme $\sqsubset$ est définie comme $\{(t, t') / t \in \text{sub}(t') \text{ et } t \neq t'\}$.

Remarquons que la relation binaire $\sqsubseteq$ est une relation d'ordre partiel bien fondée. Un terme t est en fait un sous-terme d'un autre terme u s'il apparaît à l'intérieur de u. Ce concept est toutefois assez superficiel : en effet le sous-terme peut apparaître plusieurs fois. Afin de manipuler ces occurrences de sous-termes de manière plus précise, on peut utiliser différentes notions. Nous allons en introduire deux qui sont équivalentes : nous pourrons dire que t est un sous-terme de u à la position ν, ou encore que t est un sous-terme de u dans le contexte C. Commençons par définir la notion de contexte.

Définition 1.1.12 (Contexte). *Soit un symbole $\Box$ qui n'est ni un symbole de fonction, ni un symbole de variable. Alors un contexte est un terme de l'algèbre de termes $T_\alpha(X \cup \{\Box\})$.*

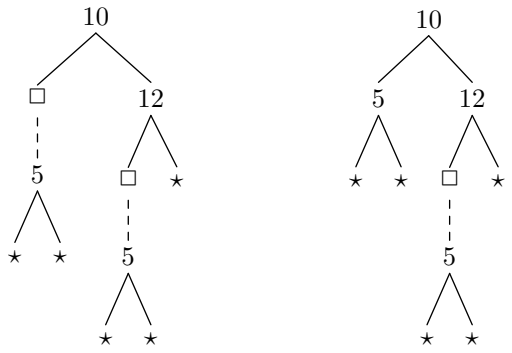
L'arité d'un contexte est le nombre de symboles $\Box$ qu'il contient. Elle peut être définie formellement comme suit : l'arité d'une variable de X ou d'une constante est 0 ; l'arité du symbole $\Box$ est 1 ; pour tout symbole de fonction d'arité $n > 0$, l'arité d'un contexte $f(C_1, \ldots, C_n)$ est la somme des arités des contextes C_i pour $1 \leqslant i \leqslant n$. Remarquons que tout terme (c'est-à-dire tout élément de $T_\alpha(X)$) est un contexte (c'est-à-dire un élément de $T_\alpha(X \cup \{\Box\})$) d'arité 0. Les contextes d'arité 1, 2, $3\ldots$ sont respectivement appelés des contextes unaires, binaires, ternaires et ainsi de suite. Si C est un contexte et t un terme, alors $C[t]$ est le remplacement du symbole $\Box$ par le terme t, défini comme suit :

si $C = \Box$	alors	$C[t] = t$;
si $C = f(C_1, \ldots, C_n)$	alors	$C[t] = f(C_1[t], \ldots, C_n[t])$;
sinon (dans ce cas, C est un terme)	alors	$C[t] = C$.

La notion d'occurrence dans un contexte est alors définie comme suit.

Définition 1.1.13 (Occurrence dans un contexte). *Nous dirons que u admet n occurrences de t dans le contexte C si C est un contexte d'arité $n > 0$ et $u = C[t]$.*

Les contextes unaires sont alors un cas particulier intéressant : ils nous permettent de souligner une occurrence précise d'un sous-terme. Par exemple $10(5(\star,\star), 12(5(\star,\star),\star))$ admet deux occurrences de $5(\star,\star)$ dans le contexte $10(\Box, 12(\Box,\star))$. On peut aussi dire que $10(5(\star,\star), 12(5(\star,\star),\star))$ admet une occurrence de $5(\star,\star)$ dans le contexte $10(5(\star,\star), 12(\Box,\star))$. Ces deux assertions sont illustrées par les diagrammes suivants.



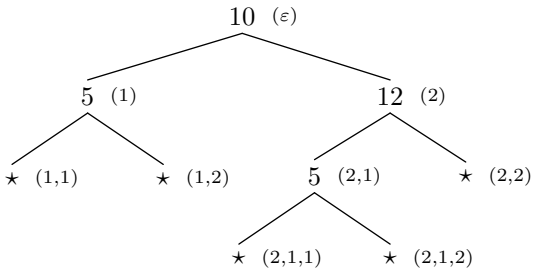
Nous dirons que C est un contexte de u s'il existe t tel que $C[t] = u$.

Une autre notion permet de parler d'occurrences avec la même précision : celle de position. Nous la définissons de la façon suivante.

Définition 1.1.14 (Position dans un terme). *Une position est une suite finie d'entiers naturels non nuls, potentiellement vide. Nous noterons ε la suite vide. Pour tout terme t dans l'algèbre de termes $T_\alpha(X)$ et toute position ν, le terme $t_{|\nu}$ est defini par*

$$\begin{aligned} t_{|\nu} &= t && \text{si } \nu = \varepsilon \ ; \\ t_{|\nu} &= t_{i|\nu'} && \text{si } t = f(t_1, \ldots, t_n) \text{ et } \nu = i, \nu' \text{ avec } 1 \leqslant i \leqslant n \ . \end{aligned}$$

Remarquons que $t_{|\nu}$ n'est pas défini pour tout terme t et toute position ν : par exemple si t est une constante ou une variable, la seule position ν telle que $t_{|\nu}$ est bien défini est la position vide ε. Les positions d'un terme t sont les positions ν pour lesquelles $t_{|\nu}$ est effectivement défini. Par exemple les positions du termes $10(5(\star,\star), 12(5(\star,\star),\star))$ de l'exemple 1.1.1 sont celles représentées sur le diagramme suivant.



On peut alors définir la notion d'occurrence à une position donnée comme suit.

Définition 1.1.15 (Occurrence à une position). *Nous dirons que u admet une occurrence de t à une position ν si $u_{|\nu} = t$.*

Par exemple $10(5(\star,\star), 12(5(\star,\star),\star))$ admet une occurrence de $5(\star,\star)$ à la position 1 ainsi qu'à la position $2,1$. Remarquons que raisonner sur les occurrences en termes de contextes est équivalent à raisonner sur les occurrences en termes de positions. On peut en fait, pour un terme donné u, définir un fonction γ_u traduisant tout contexte de u d'arité n en liste contenant n positions distinctes de u de la manière suivante : si $C = \Box$ alors $\gamma_u(C) = \varepsilon$; si $C = f(C_1 \ldots C_n)$ alors $u = f(u_1, \ldots, u_n)$ et $\gamma_u(C)$ est la concaténation des listes $\gamma_{u_i}(C_i)$ pour i de 1 a n ; sinon (dans ce cas, C est un terme), $\gamma_u(C)$ est la liste vide. La fonction γ_u renvoit en fait la liste des positions de $\Box$ dans le contexte en question. Alors le jugement « u admet n occurrences de t dans le contexte C » est équivalent au jugement « u admet une occurrence de t pour chacune des n positions de la liste $\gamma_u(C)$ ». En particulier si C est un contexte unaire, on obtient une liste contenant une seule position. Inversement on peut définir une fonction δ_u traduisant toute position ν de u en un contexte C unaire de la façon suivante : si $\nu = \varepsilon$, alors $\delta_u(\nu) = \Box$; si $\nu = k, \nu'$, alors $u = f(u_1, \ldots, u_n)$

avec $1 \leqslant k \leqslant n$ et $\delta_u(\nu) = f(u_1, \ldots, u_{k-1}, \delta_{u_k}(\nu'), u_{k+1}, \ldots, u_n)$. Alors le jugement « u admet une occurrence de t à la position ν » est équivalent au jugement « u admet une occurrence de t dans le contexte $\delta_u(\nu)$ ». Pour tout terme t et toute position ν de t, nous noterons $t[\,]_\nu$ le contexte $\delta_t(\nu)$.

Il est possible de redéfinir la relation d'ordre $\sqsubseteq$ en utilisant les notions de contexte ou de position : en effet les trois propositions suivantes sont équivalentes.

- $t \sqsubset u$
- il existe un contexte C différent de $\Box$ tel que $u = C[t]$;
- il existe une position ν différente de ε telle que $u_{|\nu} = t$.

Remarquons aussi que si ν et ν' sont deux positions d'un terme t telles que ν est un préfixe strict de ν', alors $t_{|\nu'} \sqsubset t_{|\nu}$. De manière similaire, on peut définir la notion de préfixe en termes de contextes : tout d'abord le remplacement par un terme t du symbole $\Box$ dans un contexte C peut être facilement étendu au remplacement par un contexte C', alors dénoté $C[C']$. Nous dirons que C est un préfixe de $C[C']$. En outre, si $C' \neq \Box$, nous parlerons de préfixe strict. En effet dans ce cas, $C \neq C[C']$. Alors si C_1 et C_2 sont deux contextes unaires, t est un terme admettant une occurrence de u_1 dans le contexte C_1 et une occurrence de u_2 dans le contexte C_2 et si C_1 est un préfixe strict de C_2, alors $u_1 \sqsubset u_2$.

Les termes représentent des données. En particulier une constante est une donnée atomique et un terme construit par un symbole de fonction d'arité n et une suite finie de n sous-termes est une donnée composée. Dans le but de modéliser des processus de calcul agissant sur ces données, la réécriture propose d'utiliser des relations binaires sur des algèbres de termes. Il est alors important de savoir formellement caractériser le fait que le calcul peut se produire à n'importe quelle position du terme, ou de façon équivalente dans n'importe quel contexte.

Définition 1.1.16 (Relation stable par contexte). *Une relation binaire $\rhd$ sur $T_\alpha(X)$ est stable par contexte si pour tous termes t et u, si $t \rhd u$, alors $C[t] \rhd C[u]$ pour tout contexte C* **unaire**.

Si $\rhd$ est une relation binaire sur $T_\alpha(X)$, la clôture par contexte de $\rhd$ est définie comme la plus petite relation stable par contexte contenant $\rhd$.

Définition 1.1.17 (Congruence). *Une congruence est une relation d'équivalence stable par contexte.*

La raison pour laquelle les éléments de X sont appelés des symboles de *variable* est la suivante : ils peuvent prendre différentes valeurs, c'est-à-dire, au cours d'un processus de calcul, être remplacés par un autre terme. Cette opération de remplacement s'appelle une substitution.

Définition 1.1.18 (Substitution). *Une substitution σ est une fonction de l'ensemble des variables X vers $T_\sigma(X)$ telle que $\{x \in X / \sigma(x) \neq x\}$ est fini. Cet ensemble est appelé le domaine de σ et dénoté* $\mathrm{dom}(\sigma)$. *L'ensemble $\sigma(\mathrm{dom}(\sigma))$ est appelé le codomaine de σ et est dénoté* $\mathrm{codom}(\sigma)$. *L'application d'une substitution σ à un*

terme $t \in T_\alpha(X)$ dénotée $t\sigma$ est définie inductivement par

$$\begin{array}{lll} t\sigma & = & \sigma(x) \qquad\qquad \textit{si } t \textit{ est une variable } x \textit{ ;} \\ t\sigma & = & f(t_1\sigma, \ldots, t_n\sigma) \quad \textit{si } t \textit{ est } f(t_1, \ldots, t_n)\textit{ .} \end{array}$$

Une substitution σ dont le domaine est $\{x_i/1 \leqslant i \leqslant n\}$ sera notée $[t_1/x_1, \ldots, t_n/x_n]$ où pour i entre 1 et n, t_i est le terme $\sigma(x_i)$.

Exemple 1.1.2. *Rappelons que les fonctions sur $T_\alpha(X)$ peuvent être vues comme des relations binaires qui relient un antécédent à son image. Un couple (t, u) appartient à la relation correspondant à la fonction f quand $f(t) = u$. La fonction f est stable par contexte lorsque pour tous t et u tels que $f(t) = u$, pour tout contexte unaire C, $(C[t], C[u])$ appartient à la relation, c'est-à-dire $f(C[t]) = C[u] = C[f(t)]$. En particulier puisque les substitutions sont des fonctions sur $T_\alpha(X)$, ce sont des relations binaires. Un substitution σ est donc stable par contexte si pour tout terme t et tout contexte unaire C, $(C[t])\sigma = C[t\sigma]$.*

i Si la signature α ne contient que des symboles d'arité $n \leqslant 1$, alors les substitutions sont stables par contexte.

ii Si la signature α contient au moins un symbole d'arité $n > 1$, alors la seule substitution stable par contexte est l'identité.

Dans le premier cas, tout contexte unaire C est de la forme

$$f_1(f_2(\ldots f_n(\Box)\ldots))\text{ .}$$

Dans ce cas, si t et u sont deux termes et σ une substitution tels que $t = u\sigma$, alors $C[t] = C[u\sigma] = (C[u])\sigma$. Cela démontre que les substitutions sont stables par contexte.

Dans le deuxième cas, il existe un symbole d'arité $n > 1$ que nous notons f. Soit une substitution σ stable par contexte. Soit t un terme. Démontrons à présent que $t\sigma = t$. Considérons le contexte $C = f(\Box, t, \ldots, t)$. Puisque σ est stable par contexte, $C[t\sigma] = (C[t])\sigma$. D'où $f(t\sigma, t, \ldots, t) = f(t\sigma, t\sigma, \ldots, t\sigma)$, ce qui démontre que $t\sigma = t$ (pour tout t).

En plus de la composition usuelle des fonctions qui est séquentielle, nous définissons à présent une manière parallèle de combiner deux substitutions : deux substitutions σ et σ' sont dites compatibles si leur union en tant que relations est une relation fonctionnelle. Cette compatibilité est réflexive et symétrique. Alors si σ et σ' sont deux substitutions compatibles, leur union en tant que relations est une substitution que nous noterons $\sigma \oplus \sigma'$. L'opération $\oplus$ est d'ailleurs associative, commutative et idempotente, ce qui nous permet alors de définir de manière unique la combination suivante de n substitutions compatibles deux à deux.

$$\bigoplus_{1 \leqslant i \leqslant n} \sigma_i = (\ldots(\sigma_1 \oplus \sigma_2) \cdots \oplus \sigma_n)$$

Enfin la notion de relation substitutive caractérise formellement les relations qui respectent la sémantique des symboles de variables apportée par les substitutions.

Définition 1.1.19 (Relation substitutive). *Une relation $\rhd$ est substitutive si pour tous t_1 et t_2 tels que $t_1 \rhd t_2$, alors pour toute substitution σ, $t_1\sigma \rhd t_2\sigma$.*

Lorsque la relation en question est aussi une congruence, nous parlerons de théorie équationnelle.

Définition 1.1.20 (Théorie équationnelle). *Une théorie équationnelle est une congruence substitutive. Si E est une relation binaire sur $T_\alpha(X)$, c'est-à-dire un ensemble de couples de termes, alors la théorie équationnelle engendrée par E, que nous noterons généralement $=_E$, est la plus petite théorie équationnelle contenant E.*

La notion de substitution est une notion tout à fait similaire à la notion de remplacement que nous avons définie pour les contextes. En fait la notation $C[t]$ est équivalente à $C[t/\Box]$. Nous avons vu que cette notion de remplacement induisait une relation d'ordre partielle : la relation de sous-terme $\sqsubseteq$. Quant à elles, les substitutions permettent de définir un préordre.

Définition 1.1.21 (Renommage). *Nous dirons qu'une substitution est un renommage si c'est en fait une fonction bijective de X vers X. S'il existe un renommage σ tel que $t = u\sigma$, alors nous dirons que t est un renommage de u. Nous le noterons $t \cong u$. La relation $\cong$ est une relation d'équivalence.*

Définition 1.1.22 (Subsomption). *S'il existe une substitution σ telle que $t = u\sigma$, alors nous dirons que u subsume t et que t est une instance de u. Nous le noterons $u \preccurlyeq t$. Enfin si $u \preccurlyeq t$ et $t \not\cong u$, alors nous noterons $u \prec t$. Pour les substitutions, nous dirons que σ_1 subsume σ_2 et nous le noterons $\sigma_1 \preccurlyeq \sigma_2$ s'il existe une troisième substitution σ_3 telle que $\sigma_3 \circ \sigma_1 = \sigma_2$.*

Sur les termes, la relation $\preccurlyeq$ est bel et bien un préordre. Remarquons que sur les classes d'équivalence de $\cong$, nous obtenons un inf-demi-treillis : Tout d'abord on peut prouver que $t \preccurlyeq u$ et $u \preccurlyeq t$ si et seulement si $t \cong u$. En outre si t et u sont deux termes, il existe toujours un terme v tel que t et u sont tous les deux des instances de v. Notons par ailleurs qu'il n'existe pas de suite $(t_i)_{i\in\mathbb{N}}$ telle que pour tout i, $t_{i+1} \prec t_i$. On peut alors démontrer que si t et u admettent une instance commune, alors ils admettent un infimum, c'est-à-dire une instance commune plus générale que toutes leurs autres instances. Cette instance commune plus générale est unique modulo renommage. Elle sera notée $\mathrm{pgci}(t, u)$. Voici quelques exemples.

- $\mathrm{succ}(x)$ et x admettent des instances communes : par exemple $\mathrm{succ}(\mathrm{succ}(y))$. Leur pgci est la classe d'équivalence modulo renommage de $\mathrm{succ}(x)$.
- $\mathrm{plus}(x, \mathrm{zero})$ et $\mathrm{plus}(x, \mathrm{succ}(y))$ n'admettent pas d'instance commune.
- $\mathrm{plus}(x, \mathrm{succ}(y))$ et $\mathrm{plus}(\mathrm{succ}(y), x)$ admettent des instances communes : par exemple $\mathrm{plus}(\mathrm{succ}(\mathrm{zero}), \mathrm{succ}(\mathrm{zero}))$ et $\mathrm{plus}(\mathrm{succ}(\mathrm{zero}), \mathrm{succ}(\mathrm{succ}(y)))$. Leur pgci est la classe d'équivalence modulo renommage de $\mathrm{plus}(\mathrm{succ}(x), \mathrm{succ}(y))$.

Notons que si t et u admettent une instance commune, c'est qu'il existe deux substitutions *potentiellement différentes* σ et σ' telles que $v = t\sigma = u\sigma'$. Lorsque

ces substitutions sont égales, on dit que t et u sont *unifiables* et que σ est un unificateur de t et u. Deux termes peuvent admettre une instance commune sans être unifiables : par exemple x et $f(x)$ admettent bel et bien une instance commune (par exemple $f(x)$) mais aucun unificateur. Si deux termes sont unifiables, il existe toujours un unificateur minimal pour $\preccurlyeq$ unique sur les classes d'équivalence modulo renommage. On l'appelle l'*unificateur le plus général.*

Un *problème de filtrage* pose la question « Est-ce que t est une instance de l ? ».

Définition 1.1.23 (Filtrage). *Un problème de filtrage est un couple de termes $l \overset{?}{\preccurlyeq} t$. Une solution d'un problème de filtrage $l \overset{?}{\preccurlyeq} t$ est une substitution σ telle que $l\sigma = t$.*

Bien entendu, le problème de filtrage $l \overset{?}{\preccurlyeq} t$ admet une solution si et seulement si $l \preccurlyeq t$. Le problème de décision qui consiste à répondre par l'affirmative ou la négative à la question « Est-ce que t est une instance de l ? » est algorithmiquement décidable. Alors lorsqu'elle existe, la solution au probème de filtrage $l \overset{?}{\preccurlyeq} t$ est calculable. En outre cette solution est unique.

Propriété 1.1.5 (Décidabilité et unicité du filtrage). *L'algorithme suivant prend comme argument un problème de filtrage $l \overset{?}{\preccurlyeq} t$ et retourne en temps fini une solution σ s'il en existe une,* FAIL *s'il n'en existe aucune.*

- *Si l est une variable x, alors retourner $\{t/x\}$.*
- *Si l est $f(l_1, \ldots, l_n)$, si t est $f(t_1, \ldots, t_n)$, alors pour tout $1 \leqslant i \leqslant n$, calculer une solution σ_i de $l_i \overset{?}{\preccurlyeq} t_i$. Si les σ_i sont compatibles deux à deux, alors retourner $\bigoplus_{1 \leqslant i \leqslant n}(\sigma_i)$, sinon retourner* FAIL.
- *Sinon retourner* FAIL.

En outre un telle solution est toujours unique.

Le filtrage tel que nous venons de le définir et qu'on appelle généralement *filtrage syntaxique* est un cas particulier du *filtrage équationnel* : la question que l'on se pose alors est « Est-ce que t est une instance de u dans la théorie équationnelle engendrée par E ? » . On peut reformuler cette question par « Est-ce qu'il existe une substitution σ telle que $t =_E u\sigma$? » . Le filtrage syntaxique est alors l'instance du filtrage équationnel pour la théorie équationnelle engendrée par la relation vide $\emptyset$: la relation $=_\emptyset$ représente alors l'égalité syntaxique. Notons que les propriétés d'unicité et de décidabilité du filtrage peuvent être perdues lorsqu'on étudie des problèmes de filtrage équationnel. Lorsqu'il n'existe toujours qu'une unique solution aux problèmes de filtrage, on dit que le filtrage est unitaire.

1.1.3 Systèmes de réécriture de termes

Comme l'explique la sous-section 1.1.1, nous souhaitons représenter le calcul par des relations binaires. Comme l'explique la sous-section 1.1.2, nous souhaitons représenter les données sur lesquelles le calcul opère par des termes d'une algèbre de termes $T_\alpha(X)$. Nous possédons donc les deux ingrédients nécessaires à la construction de systèmes de réécriture. Dans cette sous-section, nous supposons à nouveau donné une algèbre de termes $T_\alpha(X)$. Alors une relation de réécriture (sur $T_\alpha(X)$)

est une relation binaire qui est substitutive et stable par contexte. Commençons par définir les notions de règle de réécriture et de système de réécriture.

Définition 1.1.24 (Règle et système de réécriture). *Une règle de réécriture est un couple de termes $l \to r$ tel que $\mathcal{V}(r) \subseteq \mathcal{V}(l)$. Les termes l et r sont respectivement appelés la partie gauche et la partie droite de la règle. Un système de réécriture est un ensemble de règles de réécriture.*

La relation de réécriture induite par un système de réécriture est alors définie comme suit.

Définition 1.1.25 (Relation de réécriture). *Soit $\mathcal{R}$ un système de réécriture.*

- *La relation de réécriture en tête en un pas associée à $\mathcal{R}$, notée $\mapsto_{\mathcal{R}}$, est la plus petite relation substitutive qui contient $\mathcal{R}$, c'est-à-dire*

$$l\sigma \mapsto_{\mathcal{R}} r\sigma \quad \text{pour toute substitution } \sigma \text{ et toute règle } l \to r \in \mathcal{R} \ .$$

- *La relation de réécriture en un pas associée à $\mathcal{R}$, notée $\to_{\mathcal{R}}$, est la plus petite relation stable par contexte qui contient $\mapsto_{\mathcal{R}}$, c'est-à-dire*

$$C[l\sigma] \to_{\mathcal{R}} C[r\sigma]$$

  pour tout contexte unaire C, toute substitution σ et toute règle $l \to r \in \mathcal{R}$.
- *La relation de réécriture associée à $\mathcal{R}$ est la clôture réflexive et transitive de $\to_{\mathcal{R}}$, notée $\to_{\mathcal{R}}^*$.*

La restriction $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ de la définition 1.1.24 permet d'éviter des règles comme $l \to y$ avec $y \notin \mathcal{V}(l)$. En effet, par substitutivité, toute instance de l se réduit par cette règle en n'importe quel terme de $T_\alpha(X)$. Nous dirons qu'un système de réécriture $\mathcal{R}$ est respectivement Church-Rosser, confluent, localement confluent, fortement ou faiblement normalisant, convergent quand la relation $\to_{\mathcal{R}}$ l'est. La définition de l'addition sur les entiers de Peano constitue un exemple paradigmatique de relation de réécriture.

Exemple 1.1.3 (Arithmétique de Peano). *Considérons la signature contenant une constante* zero*, un symbole unaire* succ *ainsi qu'un symbole binaire* plus*. La représentation $\overline{n}$ d'un entier $n \in \mathbb{N}$ est définie inductivement par $\overline{0} =$ zero et $\overline{n+1} =$ succ$(\overline{n})$. Considérons alors le système de réécriture contenant les règles*

$$\begin{aligned} \text{plus}(x, \text{zero}) &\to x \\ \text{plus}(x, \text{succ}(y)) &\to \text{succ}(\text{plus}(x, y)) \end{aligned}$$

Ces règles correspondent bien sûr à une programmation récursive de la fonction d'addition. Celle-ci s'écrit par exemple en Caml comme suit.

```
let rec plus x = function
    0 -> x
  | y -> (plus x (y-1))+1;;
```

Le terme $\text{plus}(\text{succ}(\text{succ}(\text{zero})), \text{succ}(\text{zero}))$ *représente par exemple* $2+1$. *Il admet d'ailleurs* $\overline{3}$ *comme forme normale.*

$$\begin{aligned}\text{plus}(\text{succ}(\text{succ}(\text{zero})), \text{succ}(\text{zero})) &\to_{\mathcal{R}} \text{succ}(\text{plus}(\text{succ}(\text{succ}(\text{zero})), \text{zero})) \\ &\to_{\mathcal{R}} \text{succ}(\text{succ}(\text{succ}(\text{zero}))) \\ &= \overline{3}\end{aligned}$$

Notons que le système de réécriture $\mathcal{R}$ *est en fait convergent (sa confluence sera montrée par la propriété 1.1.6). L'ensemble des formes normales est* $\{\overline{n}/n \in \mathbb{N}\}$. *Enfin notons que si* $\downarrow t = \overline{n}$ *et* $\downarrow u = \overline{m}$, *alors* $\downarrow \text{plus}(t,u) = \overline{n+m}$.

Lorsque nous considérerons une règle de réécriture $l \to r$, nous dirons qu'une instance $l\sigma$ de l est un redex. Parfois, il sera pratique de nommer les règles de réécriture. Une règle de réécriture $l \to r$ nommée ρ sera alors notée $\rho : l \to r$. Un redex pour cette règle sera alors appelé un ρ-redex. Comme nous l'avons expliqué dans la section 1.1.1, un lien $t \to_{\mathcal{R}} u$ représente une étape de calcul. Cette étape de calcul peut néanmoins se décomposer de la façon suivante : trouver tout d'abord un redex, c'est-à-dire un contexte C tel que $t = C[t']$ et tel que le problème de filtrage $l \overset{?}{\preccurlyeq} t'$ admet une solution σ ; appliquer ensuite successivement cette solution σ puis le contexte C au terme r.

Certains critères syntaxiques assurent la confluence d'un système de réécriture. Nous allons définir à présent les notions de chevauchement et de paire critique.

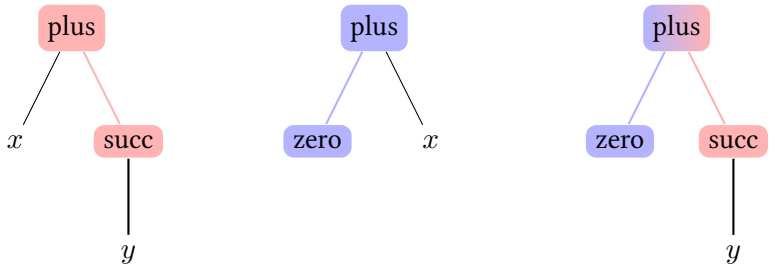
Définition 1.1.26 (Chevauchement). *Nous dirons qu'un terme* l_1 *en chevauche un deuxième terme* l_2 *s'il existe une position* ν *de* l_1 *telle que* $l_{1|\nu}$ *n'est pas une variable et* $l_{1|\nu}$ *et* l_2 *admettent une instance commune, c'est-à-dire qu'il existe deux substitutions* σ *et* σ' *tels que* $l_{1|\nu}\sigma = l_2\sigma'$. *Nous dirons que deux termes se chevauchent si l'un des deux chevauche l'autre. Nous dirons que deux règles de réécriture se chevauchent si leurs parties gauche se chevauchent.*

Remarquons que cette définition peut être aussi écrite en termes de contexte : l_1 chevauche l_2 si $l_1 = C[t]$ où t n'est pas une variable et t et l_2 admettent une instance commune.

Exemple 1.1.4 (Arithmétique de Peano 2). *Les deux règles de réécriture du système de l'exemple 1.1.3 ne se chevauchent pas : Les positions de* $\text{plus}(x, \text{zero})$ *qui ne pointent pas vers une variable sont* ε *et* 2 *et représentent respectivement les termes* $\text{plus}(x, \text{zero})$ *et* zero. *Clairement ils ne partagent pas d'instance avec* $\text{plus}(x, \text{succ}(y))$. *De même les positions de* $\text{plus}(x, \text{succ}(y))$ *qui ne pointent pas vers une variable sont* ε *et* 2 *et représentent les termes* $\text{plus}(x, \text{succ}(y))$ *et* $\text{succ}(y)$. *Ils ne partagent pas non plus d'instance avec* $\text{plus}(x, \text{zero})$.

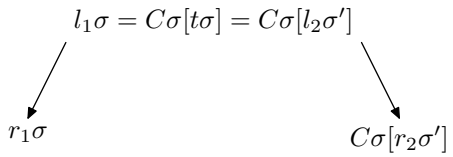
Si l'on rajoute à présent la règle $\text{plus}(\text{zero}, x) \to x$ *au système, nous obtenons les trois parties gauches* $\text{plus}(x, \text{zero})$, $\text{plus}(x, \text{succ}(y))$, $\text{plus}(\text{zero}, x)$. *Remarquons que la dernière partie gauche chevauche à la fois la première et la deuxième à la position* ε *: le premier chevauchement est obtenu avec l'instance* $\text{plus}(\text{zero}, \text{zero})$, *le*

deuxième avec l'instance plus(zero, succ(y)). *Le deuxième chevauchement est illustré par les arbres suivants, représentant respectivement les deux parties gauches puis l'instance commune. Notons que cette instance commune est en fait le* pgci.



Définition 1.1.27 (Paire critique). *Soit deux règles de réécriture $l_1 \to r_1$ et $l_2 \to r_2$ qui se chevauchent. Alors $l_1 = C[t]$ où t n'est pas une variable et t et l_2 admettent une instance commune et donc un* pgci. *Il existe donc deux substitutions σ et σ' telles que $t\sigma = l_2\sigma' = \text{pgci}(t, l_2)$. On peut d'ailleurs supposer que $\text{dom}(\sigma) = \mathcal{V}(t)$ et $\mathcal{V}(t\sigma) \cap \mathcal{V}(C) = \emptyset$. Alors la paire critique de ce chevauchement est la paire (modulo renommage) $(r_1\sigma, C\sigma[r_2\sigma'])$.*

Remarquons que les éléments $r_1\sigma$ et $C\sigma[r_2\sigma']$ de la paire critique d'un chevauchement entre $l_1 \to r_1$ et $l_2 \to r_2$ sont les résultats d'une étape de réécriture à partir de $l_1\sigma$ en utilisant respectivement la première règle dans le contexte vide et la deuxième règle dans le contexte $C\sigma$.



Les paires critiques d'un système de réécriture sont les paires critiques des chevauchements de ses règles de réécriture. Nous dirons qu'une paire critique (t, u) est convergente si t et u se réduisent vers un même terme. La convergence de toutes les paires critiques d'un système implique la confluence locale, comme l'ont démontré Knuth et Bendix (1970) puis Huet (1980).

Propriété 1.1.6 (Paires critiques). *Si toutes les paires critiques d'un système de réécriture sont convergentes, alors le système est localement confluent. Si le système est fortement normalisant, alors le système est confluent.*

La deuxième assertion est évidemment une conséquence directe de la première et du théorème 1.1.3. Pour tout système fortement normalisant, la convergence de toutes les paires critiques est donc équivalente à la confluence du système.

Exemple 1.1.5 (Arithmétique de Peano 3). *Reprenons les systèmes de réécriture présentés dans les exemples 1.1.3 et 1.1.4 :*

$$\begin{cases} \text{plus}(x, \text{zero}) & \to & x \\ \text{plus}(x, \text{succ}(y)) & \to & \text{succ}(\text{plus}(x, y)) \end{cases}$$

et

$$\begin{cases} \text{plus}(x, \text{zero}) & \to & x \\ \text{plus}(\text{zero}, x) & \to & x \\ \text{plus}(x, \text{succ}(y)) & \to & \text{succ}(\text{plus}(x, y)) \ . \end{cases}$$

Prouver que ces deux systèmes terminent est un exercice plutôt facile. Prouver qu'ils sont confluents l'est aussi grâce à la propriété 1.1.6. En effet le premier n'admet aucun chevauchement et donc aucune paire critique. Il est donc localement confluent, confluent par la propriété 1.1.3 et donc convergent. Quant au deuxième, il admet les paires critiques $(\text{zero}, \text{zero})$ *ainsi que* $(\text{succ}(y), \text{succ}(\text{plus}(\text{zero}, y)))$*. La première est trivialement convergente. La deuxième est aussi convergente car* $\text{plus}(\text{zero}, y) \to y$*. Le deuxième système est donc aussi localement confluent et confluent par la propriété 1.1.6 et convergent par la propriété 1.1.3.*

Dans le cas de systèmes qui ne sont pas fortement normalisants, la convergence de toutes les paires critiques n'implique pas forcément la confluence comme le montre le contre-exemple de Klop (1980) :

$$\begin{cases} A & \to & C(A) \\ C(x) & \to & D(x, C(x)) \\ D(x, x) & \to & E \ . \end{cases}$$

Ce système n'admet pas de paire critique. La propriété 1.1.6 montre qu'il est donc localement confluent. Néanmoins il n'est pas confluent puisque $C(A)$ se réduit à la fois en E et $C(E)$ et puisque ces deux termes n'admettent pas de réduit commun. Il nous faut remarquer que la partie gauche de la règle $D(x, x) \to E$ contient deux occurrences de la variable x. Une telle règle de réécriture est alors appelée non-linéaire à gauche.

Définition 1.1.28 (Règle de réécriture linéaire). *Une règle de réécriture est linéaire à gauche (respectivement à droite) si sa partie gauche (respectivement droite) ne contient pas deux occurrences distinctes d'une même variable.*

Nous dirons qu'un système de réécriture est linéaire (à gauche ou à droite) si toutes ses règles de réécriture le sont. On attribue généralement la cause de non-confluence du contre-exemple de Klop à sa non-linéarité à gauche. En effet ce système de réécriture n'admet aucun chevauchement (et donc aucune paire critique), et s'il avait été linéaire à gauche, il aurait été confluent, comme le montre la propriété 1.1.7 suivante. De tels systèmes sont d'ailleurs appelés des systèmes orthogonaux.

Définition 1.1.29 (Système de réécriture orthogonal). *Un système de réécriture est orthogonal s'il est linéaire à gauche et s'il n'admet aucun chevauchement.*

Propriété 1.1.7 (Huet, 1980). *Si un système est orthogonal, alors il est confluent.*

Une sous-classe des systèmes orthogonaux à laquelle nous nous intéresserons est la classe des schémas de programmes récursifs.

Définition 1.1.30 (Schémas de programmes récursifs (SPR)). *Un schéma de programme récursif est un système de réécriture $\mathcal{R}$ sur une algèbre de termes $T_\alpha(X)$ tel que (i) la signature α peut être partitionnée en deux ensembles disjoints $\{f_1, \ldots, f_n\}$ et $\{g_1, \ldots, g_m\}$; (ii) $\mathcal{R}$ contient exactement pour chaque symbole f_i une règle de la forme $f_i(x_1, \ldots, x_p) \rightarrow t$ où les variables x_k sont disctinctes deux à deux.*

Un tel système est orthogonal et donc confluent. En outre, la décidabilité de la normalisation faible et forte ont été démontrées par Khasidashvili (1993a,b).

Propriété 1.1.8. *Pour les SPR, la normalisation forte et la normalisation faible sont des propriétés décidables.*

1.2 Lambda-calcul

En tant que théorie générale des fonctions mathématiques, le lambda-calcul fut introduit par Church (1933-1934) dans le cadre de recherches des fondements des mathématiques. Bien que Kleene et Rosser (1936) ont démontré l'inconsistance du système original de Church , le lambda-calcul est une approche couronnée de succès en ce qui concerne un effort de formalisation légèrement moins ambitieux et qui avait lieu au même moment : la formalisation du calcul. Church (1936) avait déjà proposé la notion de lambda-définissable, basée sur le lambda-calcul, comme un candidat à la définition formelle de la notion d'*effectivement calculable*. Un peu plus tard quand Turing (1936, 1937) proposa sa propre analyse des machines de calcul par les notions universellement acceptées de *Machines de Turing* et de *Turing Calculable*, il prouva lui-même que cette dernière notion était équivalente à la notion de lambda-définissable.

Aujourd'hui le lambda-calcul en tant que modèle de calcul est un outil performant pour l'informaticien théorique. En effet et contrairement à l'approche *bas niveau* de Turing, le lambda-calcul se base sur des notions *haut niveau* qui sont des pierres d'angles de la plupart des langages de programmation : les variables représentant les données et les abstractions (fonctions) représentant des procédures. Par conséquent, il peut se révéler utile de construire la traduction d'un langage de programmation vers le lambda-calcul afin de donner une sémantique formelle au langage. C'est ce que proposait déjà Landin (1965) pour le langage ALGOL en 1965. Le succès du lambda-calcul par cette approche mena alors à la définition de langages de programmation fonctionnels comme OCaml (Leroy *et al.*, 2009) ou encore Haskell (Peyton-Jones, 2003) qui ne sont plus reliés à la couche matérielle qu'au travers de la compilation. Ces langages sont directement inspirés par la syntaxe et la sémantique du lambda-calcul et par conséquent bénéficient de son grand pouvoir expressif. Le pont construit par Turing entre le lambda-calcul et le calcul bas niveau n'a cependant pas comblé la brêche : c'est en effet toujours aujourd'hui un domaine de recherche actif qui a produit des techniques très efficaces comme les indices de De Bruijn (1972) ou les substitutions explicites (Abadi *et al.*, 1991; Bloo et Rose, 1995; Benaissa *et al.*, 1996).

À l'instar des informaticiens, les logiciens n'ont pas non plus abandonné le lambda-calcul. En effet De Bruijn (1968) proposa un système informatique appelé Automath dédié à la construction de preuves mathématiques formelles. Le méta-langage qu'il utilisa pour formaliser les preuves est très proche du lambda-calcul. Beaucoup plus tôt, Curry (1934) avait observé que les combinateurs de sa Logique Combinatoire (un système équivalent au lambda-calcul) correspondaient en fait aux schémas d'axiomes de la logique intuitionniste implicationnelle, puis (1958) qu'un fragment de la Logique Combinatoire coincidaient avec un fragment des systèmes déductifs à la Hilbert. Enfin Howard (1980) observa que la déduction naturelle intuitionniste de Gentzen peut être interprétée comme une variante typée du lambda-calcul : au travers d'un système de typage, les preuves en deduction naturelle sont mises en correspondance avec une certaine classe de lambda-termes (les termes simplement typés). Cette correspondance de Curry-De Bruijn-Howard admet alors certaines propriétés intéressantes :

Subject Reduction Les calculs effectués dans le lambda-calcul (bêta-réduction…) à partir de preuves valides en déduction naturelle renvoient des preuves valides en déduction naturelle.

Normalisation Forte Les calculs effectués dans le lambda-calcul à partir de preuves valides en déduction naturelle terminent. On obtient alors une preuve en forme normale, c'est-à-dire *analytique*.

L'intérêt du lambda-calcul en théorie de la preuve provient alors de certaines propriétés des preuves analytiques telles que par exemple la propriété du témoin.

Le sujet de cette section est d'introduire le lambda-calcul pur comme modèle formel du calcul. Les aspects logiques ainsi que la correspondance de Curry-De Bruijn-Howard seront traités dans le chapitre 3. Plus de détails concernant le lambda-calcul se trouvent dans l'ouvrage de Barendregt (1984).

1.2.1 Syntaxe du lambda-calcul

Soit X un ensemble infini de variables.

Définition 1.2.1 (Lambda-termes). *Soit $\sigma = (F \cup \{\text{app}\}, n)$ une signature telle que*
- *il existe une bijection $\mathfrak{n} : X \to F$;*
- *pour tout $x \in X$, $n(\mathfrak{n}(x)) = 1$;*
- $n(\text{app}) = 2$.

Alors l'ensemble des lambda-termes est défini comme l'algèbre de termes $T_\sigma(X)$.

Nous écrirons $\lambda x.t$ au lieu de $f(t)$ si $f = \mathfrak{n}(x)$. Nous écrirons aussi $t_1\ t_2$ au lieu de $\text{app}(t_1, t_2)$. En fait, l'ensemble des lambda-termes peut être aussi défini comme le langage hors contexte généré par la grammaire hors contexte dont le seul symbole non terminal est S et qui contient les règles suivantes.

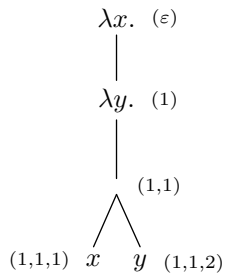
$$\begin{array}{lll} S & \to x & \text{si } x \in X \\ S & \to (\lambda x.S) & \text{si } x \in X \\ S & \to (S\ S) & \end{array}$$

Cette grammaire peut être écrite alternativement sous forme de Backus-Naur.

$$S ::= x \mid (\lambda x.S) \mid (S\ S)$$

Un terme de la forme $(\lambda x.t)$ est appelé une abstraction et nous dirons que t est le corps de l'abstraction. Un terme de la forme $(t_1\ t_2)$ est appelé une application. Nous omettrons les parenthèses lorsque celles-ci ne sont pas nécessaires à l'identification d'un lambda-terme. Par exemple nous écrirons $\lambda x.\lambda y.t$ au lieu de $\lambda x.(\lambda y.t)$. Nous prenons aussi les conventions suivantes : $\lambda x.t_1\ t_2$ représente $\lambda x.(t_1\ t_2)$ et non $(\lambda x.t_1)\ t_2$; $t_1\ t_2\ t_3$ représente $((t_1\ t_2)\ t_3)$ et non $t_1\ (t_2\ t_3)$; $\lambda x_1 x_2 \ldots x_n.t$ représente $\lambda x_1.\lambda x_2 \ldots \lambda x_n.t$.

Comme nous avons défini l'ensemble des lambda-termes comme une algèbre de termes, nous héritons directement des définitions que nous avons écrites en section 1.1.2 pour les algèbres de termes : ensemble de variables (définition 1.1.9), sous-termes (définition 1.1.11), contextes (définition 1.1.12), positions (définition 1.1.14), substitutions (définition 1.1.18). Par exemple les positions du lambda-terme $\lambda x.\lambda y.(x\ y)$ sont décrites par le schéma suivant.



Néanmoins nous verrons que certaines notions, comme les substitutions, ne sont pas totalement adaptées au lambda-calcul et qu'il faudra les modifier ou encore les manipuler avec des précautions supplémentaires.

1.2.2 Sémantique du lambda-calcul : alpha

Le lambda-calcul a originellement été proposé comme une formalisation de la théorie des fonctions mathématiques. En l'occurrence, les lambda-termes symbolisent des fonctions. Par exemple le lambda-terme $\lambda x.x$ représente la fonction identité qui associe à n'importe quel objet M le même objet M. Dans l'expression $\lambda x.x$, le symbole x est une variable représentant « n'importe quel objet M » qui pourrait être l'argument de la fonction identité. Nous aurions pu d'ailleurs choisir un autre symbole de variable : par exemple si nous avions choisi y, nous aurions obtenu $\lambda y.y$. C'est d'ailleurs une autre représentation syntaxique de la fonction identité. Lorsqu'on définit une fonction, les symboles de variable utilisés pour représenter des arguments potentiels n'ont donc de valeur que localement, dans le contexte $\lambda x.\Box$. Autrement dit ces variables ont comme portée le corps de l'abstraction. Nous dirons qu'une occurrence d'une variable x dans le contexte $\lambda x.\Box$ est *liée*. Le symbole de fonction $\lambda(x)$ ou par abus de langage le symbole λ est appelé un *lieur*. Enfin les occurrences de variable qui ne sont pas liées sont appelées des occurrences *libres*.

Définition 1.2.2 (Occurrence libre et liée d'une variable). *L'occurrence d'une variable x à une position ν dans un lambda-terme t est liée s'il existe un préfixe ν' de ν tel que $t_{|\nu'}$ est de la forme $\lambda x.t'$. Sinon, l'occurrence est libre.*

Si t est un lambda-terme, alors l'ensemble des variables libres de t et l'ensemble des variables liées de t sont respectivement définis par

$$\mathcal{FV}(t) = \{x / \exists\ \nu \text{ occurrence libre de } x \text{ dans } t\}$$

et

$$\mathcal{BV}(t) = \{x / \lambda(x) \text{ apparaît dans } t\}\ .$$

Remarquons qu'une variable $x \in \mathcal{BV}(t)$ peut n'admettre aucune occurrence liée dans t. Par contre, l'inverse est vrai : si elle admet une occurrence liée dans t, alors $x \in \mathcal{BV}(t)$. Habituellement, ces ensembles sont définis récursivement de la façon suivante.

Définition 1.2.3 (Variables libres et liées). *Soit un lambda-terme t. L'ensemble des variables liées dans t, noté $\mathcal{BV}(t)$, et l'ensemble des variables libres dans t, noté $\mathcal{FV}(t)$, sont définis par*

$$\begin{array}{llll}
\mathcal{BV}(t) = \emptyset & \text{et} & \mathcal{FV}(t) = \{x\} & \text{si } t = x\ ; \\
\mathcal{BV}(t) = \mathcal{BV}(t') \cup \{x\} & \text{et} & \mathcal{FV}(t) = \mathcal{FV}(t') \backslash \{x\} & \text{si } t = \lambda x.t'\ ; \\
\mathcal{BV}(t) = \mathcal{BV}(t_1) \cup \mathcal{BV}(t_2) & \text{et} & \mathcal{FV}(t) = \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) & \text{si } t = t_1\ t_2\ .
\end{array}$$

Comme nous l'avons expliqué, les variables liées représentent les arguments des fonctions et leur représentation syntaxique par une variable x ou y n'est pas pertinente : les mathématiciens utilisent de manière indifférente les notations $x \mapsto x^2$ et $y \mapsto y^2$ pour représenter la fonction carré. Évidemment, le choix du nom de la variable n'importe pas. Similairement, les lambda-termes doivent être identifiés modulo renommage de variables liées. Cette opération de renommage est appelée *alpha-conversion* et est définie comme suit.

Définition 1.2.4 (Alpha-conversion). *L'alpha-conversion, notée $\equiv_\alpha$, est la plus petite congruence sur les lambda-termes telle que $\lambda x.t \equiv_\alpha \lambda y.(t[y/x])$ dès que y, $\lambda(y)$ et $\lambda(x)$ n'apparaissent pas dans t.*

Remarquons que les variables libres ne sont pas modifiées par l'alpha-conversion.

Propriété 1.2.1. *Si $t \equiv_\alpha t'$, alors $\mathcal{FV}(t) = \mathcal{FV}(t')$. Plus précisément, si une variable admet une occurrence libre à la position ν dans t, alors elle admet aussi une occurrence libre à la même position dans t'.*

Remarquons que la relation $\equiv_\alpha$ n'est pas substitutive : il existe des lambda-termes t, t' et une substitution σ tels que $t \equiv_\alpha t'$ et $t\sigma \not\equiv_\alpha t'\sigma$:

$$\lambda x.z \equiv_\alpha \lambda y.z \quad \text{mais} \quad (\lambda x.z)[x/z] = \lambda x.x \not\equiv_\alpha \lambda y.x = (\lambda y.z)[x/z]\ .$$

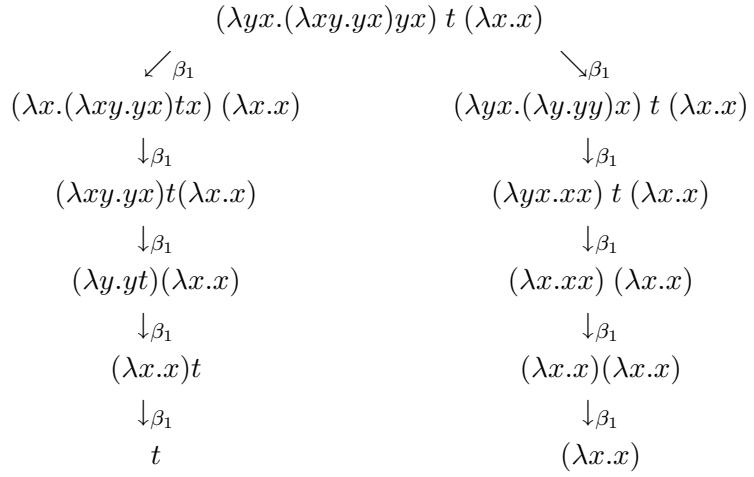


FIG. 1.1 – Incohérence de β_1

Lors de l'application de la substitution $[x/z]$ au terme $\lambda x.z$, l'occurrence libre de la variable z s'est transformée en une occurrence liée de la variable x. On dit dans ce cas que le lieur λx a capturé la variable x. L'inverse est aussi possible :

$$\lambda x.x \equiv_\alpha \lambda y.y \quad \text{mais} \quad (\lambda x.x)[z/x] = \lambda x.z \not\equiv_\alpha \lambda y.y = (\lambda y.y)[z/x]\ .$$

Ici c'est la substitution qui a capturé la variable x qui apparaissait liée dans $\lambda x.x$.

1.2.3 Sémantique du lambda-calcul : bêta

Le mécanisme le plus important du lambda-calcul est appelé la bêta-réduction. C'est le mécanisme qui permet d'évaluer une fonction $\lambda x.t$ appliquée à un certain argument t' : le résultat est le remplacement de toutes les occurrences de x dans t par t'. Ce remplacement est bien évidemment défini par une notion de substitution. Si l'on utilise les substitutions comme nous les avons définies en section 1.1, alors nous obtenons un système inconsistant : soit en effet $\equiv_{\beta_1}$ la plus petite relation de congruence telle que pour tous termes t et t', $(\lambda x.t)t' \equiv_{\beta 1} t[t'/x]$. Alors pour tous termes t et t', $t \equiv_{\beta_1} t'$, comme démontré dans l'ouvrage de Barendregt (1984). En effet pour tout lambda-terme t, la figure 1.1 démontre que $t \equiv_{\beta_1} (\lambda x.x)$ et donc si t et t' sont deux lambda-termes quelconques, alors $t \equiv_{\beta_1} (\lambda x.x) \equiv_{\beta_1} t'$. L'incohérence provient du fait que les substitutions de la section 1.1 n'empêchent aucunement les captures de variables. Il y a bien une capture lors de l'étape de β_1-réduction $(\lambda xy.yx)y \rightarrow_{\beta_1} \lambda y.yy$ du coin en haut à droite de la figure 1.1. Il est possible de corriger cela en caractérisant quand ces captures ont lieu : lorsqu'on applique une substitution σ à un terme t, il ne peut pas y avoir de capture si $\mathcal{BV}(t) \cap (\text{dom}(\sigma) \cup \mathcal{FV}(\text{codom}(\sigma))) = \emptyset$. Nous dirons qu'une substitution σ évite les captures dans un lambda-terme t quand cette condition est vérifée. Comme nous souhaitons à présent éviter à tout prix les captures, nous n'appliquerons plus sur des lambda-termes que des substitutions qui

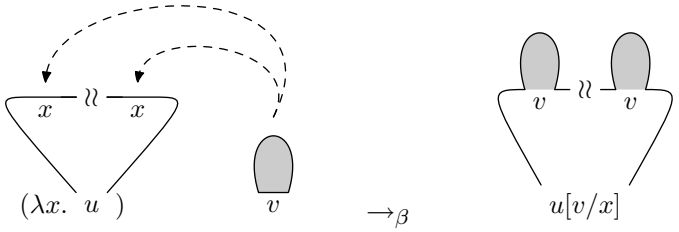


FIG. 1.2 – Bêta-réduction

évitent les captures. Notons par exemple que la substitution utilisée dans la définition 1.2.4 (alpha-conversion) satisfait cette condition. Il n'est donc pas nécessaire de redéfinir l'alpha-conversion. La notion de substitutivité est par contre redéfinie sur les lambda-termes comme suit.

Définition 1.2.5 (Relation lambda-substitutive). *Une relation $\rhd$ sur les lambda-termes est lambda-substitutive si pour tous lambda-termes t_1 et t_2 tels que $t_1 \rhd t_2$, alors pour toute substitution σ évitant les captures dans t_1 et dans t_2, $t_1\sigma \rhd t_2\sigma$.*

Alors l'alpha-équivalence est lambda-substitutive. La bêta-réduction est définie comme suit.

Définition 1.2.6 (Bêta-réduction). *La bêta-réduction en tête, notée $\mapsto_\beta$, est définie comme la relation $\equiv_\alpha \circ \beta \circ \equiv_\alpha$ où β est la relation définie par $(\lambda x.t)\ t' \ \beta\ t[t'/x]$ si $[t'/x]$ évite les captures dans t. La bêta-réduction, notée $\rightarrow_\beta$ est alors définie comme $\equiv_\alpha \circ \bar{\beta} \circ \equiv_\alpha$ où $\bar{\beta}$ est la clôture par contexte de $\mapsto_\beta$. Finalement la bêta-équivalence, notée $\leftrightarrow^*_\beta$ est définie comme la clôture refléxive, symétrique et transitive de $\rightarrow_\beta$.*

La bêta-réduction est illustrée par la figure 1.2. La bêta-réduction est lambda-substitutive. Nous verrons aussi que contrairement à β_1, elle est consistante. Remarquons par ailleurs que tout lambda-terme $(\lambda x.t)\ t'$ se réduit par bêta-réduction en tête, c'est-à-dire qu'il existe toujours un terme t'' tel que $(\lambda x.t)\ t' \ \mapsto_\beta \ t''$. Le cas problématique est bien entendu le cas où la substitution $[t'/x]$ n'évite pas les captures dans t, c'est-à-dire lorsque $\mathcal{BV}(t) \cap (\{x\} \cup \mathcal{FV}(t')) \neq \emptyset$. Dans ce cas, il existe toujours un terme u tel que $t \equiv_\alpha u$ et $\mathcal{BV}(u) \cap (\{x\} \cup \mathcal{FV}(t')) = \emptyset$, comme le démontre le lemme de rafraichissement 1.2.2.

Propriété 1.2.2 (Lemme de rafraichissement). *Si Y est un ensemble fini de variables et t un lambda-terme, alors il existe toujours u tel que $t \equiv_\alpha u$ et $\mathcal{BV}(u) \cap Y = \emptyset$.*

Démonstration. Par induction sur t et en utilisant le fait que l'ensemble des variables X est infini. □

Un lambda-terme $(\lambda x.t)\ t'$ se réduit donc toujours par bêta-réduction. Nous appellerons donc les occurrences de lambda-termes de cette forme dans un lambda-

terme quelconque des bêta-redex. Même si les étapes de bêta-réduction peuvent inclure des étapes d'alpha-conversion, notons que la relation $\leftrightarrow_\beta^*$ ne contient pas la relation $\equiv_\alpha$. Nous noterons donc $\equiv_{\alpha\beta}$ la plus petite congruence contenant à la fois $\leftrightarrow_\beta^*$ et $\equiv_\alpha$. Remarquons aussi que la bêta-réduction ne peut pas être définie comme une relation de réécriture : tout d'abord la « règle » de bêta-réduction $(\lambda x.t)t' \to t[t'/x]$ ne représente pas une seule règle de réécriture mais plutôt l'ensemble de règles

$$\left\{ (\lambda x.t)t' \to t[t'/x] \;\middle/\; \left\{ \begin{array}{l} t \text{ et } t' \text{ lambda-termes et} \\ x \in X \text{ tels que } \mathcal{BV}(t) \cap (\{x\} \cup \mathcal{FV}(t')) = \emptyset \end{array} \right\} \right.$$

Remarquons aussi que la bêta-réduction n'est pas une relation de réécriture : elle n'est pas substitutive, mais lambda-substitutive.

Malgré la simplicité de sa syntaxe, le lambda-calcul bénéficie d'une forte expressivité. Les entiers de Church constituent un premier exemple intéressant.

Exemple 1.2.1 (Entiers de Church). *Si n est un entier naturel, f et x sont deux variables de X, le lambda-terme $f^n x$ est défini inductivement par $f^0 x = x$ et $f^{n+1} x = f(f^n x)$. Alors pour tout entier naturel n, l'entier de Church $\overline{n}$ est défini par $\overline{n} = \lambda f x.f^n x$. Par exemple $\overline{0}$ est la fonction $\lambda f x.x$. On peut alors définir l'addition et la multiplication comme suit.*

$$\begin{array}{rcl} \text{plus} & = & \lambda n m f x.(nf)((mf)x) \\ \text{times} & = & \lambda n m f.n(mf) \end{array}$$

Alors pour tous entiers naturels n et m, $\text{plus}\ \overline{n}\ \overline{m} \to_\beta^* \overline{n+m}$ *et* $\text{times}\ \overline{n}\ \overline{m} \to_\beta^* \overline{n \times m}$.

La bêta-réduction n'est pas faiblement normalisante, comme le démontre le contre-exemple suivant.

Exemple 1.2.2 (Oméga). *Soit ω le lambda-terme $\lambda x.xx$ et Ω le lambda-terme $\omega\omega$. Alors $\Omega \to_\beta \Omega$. Le terme Ω étant d'ailleurs lui-même son unique réduit, il n'est pas faiblement normalisant. La relation $\to_\beta$ n'est donc pas faiblement normalisante.*

N'étant pas faiblement normalisante, elle n'est pas fortement normalisante.

1.2.4 Confluence de bêta

La bêta-réduction est confluente (modulo alpha-conversion), comme démontré initialement par Church et Rosser (1936). Ces derniers ont utilisé la technique dite des résidus. Un autre technique plus simple a été utilisée par Tait et Martin-löf : la technique des réductions parallèles (Barendregt, 1984; Takahashi, 1995). On définit la relation $\rightrightarrows$ de la façon suivante :

i $t \rightrightarrows u$ si $t \equiv_\alpha u$;

ii $\lambda x.t \rightrightarrows \lambda x.u$ si $t \rightrightarrows u$;

iii $t\ u \rightrightarrows t'\ u'$ si $t \rightrightarrows t'$ et $u \rightrightarrows u'$;

iv $(\lambda x.t)\,u \rightrightarrows t'[u'/x]$ si $t \rightrightarrows t'$ et $u \rightrightarrows u'$ (et $[u'/x]$ évite les captures dans t').

Alors on peut montrer que $\to_\beta \subseteq \rightrightarrows \subseteq \to_\beta^*$, ce qui implique d'ailleurs que $\rightrightarrows^* = \to_\beta^*$. En outre, pour tout lambda-terme t, on définit le terme $(t)^*$ comme suit :

a $(x)^* = x$ pour toute variable x ;

b $(\lambda x.t)^* = \lambda x.(t)^*$;

c $(t\,u)^* = (t)^*\,(u)^*$ si t n'est pas une abstraction ;

d $((\lambda x.t)\,u)^* = (t)^*[(u)^*/x]$ (on peut supposer que $[(u)^*/x]$ évite les captures dans $(t)^*$).

Alors si $t \rightrightarrows u$, on peut démontrer que $u \rightrightarrows (t)^*$. Cela prouve que $\rightrightarrows$ est fortement confluente et donc confluente par la propriété 1.1.1. Puisque $\rightrightarrows^* = \to_\beta^*$, nous obtenons que $\to_\beta$ est confluente. Comme corollaire, nous obtenons la cohérence de $\equiv_{\alpha\beta}$. Il suffit en effet d'exhiber deux formes normales syntaxiquement distinctes : par exemple $\lambda xy.x$ et $\lambda xy.y$. Si ces deux termes étaient convertibles par $\equiv_{\alpha\beta}$, par confluence de $\to_\beta$, ils admettraient un réduit commun. Comme ce sont deux formes normales, ces deux termes ne peuvent qu'être égaux à ce réduit commun et doivent donc être égaux eux-mêmes (modulo alpha-conversion). Les termes $\lambda xy.x$ et $\lambda xy.y$ n'étant pas alpha-convertibles, nous obtenons que $\lambda xy.x \not\equiv_{\alpha\beta} \lambda xy.y$ et donc que $\equiv_{\alpha\beta}$ est consistant.

1.2.5 Sémantique du lambda-calcul : êta

Soit une relation $\sim$ sur les lambda-termes. Nous dirons que $\sim$ est faiblement extensionnelle si pour tous t et u tels que $t \sim u$ et pour toute variable x, alors $\lambda x.t \sim \lambda x.u$. Nous dirons que $\sim$ est (fortement) extensionnelle si pour tous t et u lambda-termes et pour toute variable x telle que $x \notin \mathcal{FV}(t) \cup \mathcal{FV}(u)$, si $tx \sim ux$ alors $t \sim u$. Puisque $\equiv_{\alpha\beta}$ est une relation close par contexte, elle est faiblement extensionnelle. Elle n'est par contre pas fortement extensionnelle : si x et y sont deux variables distinctes (X est un ensemble *infini* de variables), alors $yx \equiv_{\alpha\beta} (\lambda z.yz)x$ et par contre $y \not\equiv_{\alpha\beta} \lambda z.yz$ (par la confluence de $\to_\beta$ et puisque ces deux lambda-termes sont des formes normales pour $\to_\beta$). Lorsque l'on souhaite travailler dans un cadre fortement extensionnel, on utilise alors la réduction suivante.

Définition 1.2.7 (Êta-réduction). *La êta-réduction, que nous noterons $\to_\eta$, est la clôture par contexte de la relation $\mapsto_\eta$ définie par*

$$\lambda x.tx \mapsto_\eta t \quad \text{si } x \notin \mathcal{FV}(t)\ .$$

La relation $\leftarrow_\eta$ est appelée êta-expansion. La relation $\leftrightarrow_\eta$ est appelée êta-conversion.

Alors nous noterons $\to_{\beta\eta}$ la relation $\to_\beta \cup \to_\eta$. Nous noterons aussi $\equiv_{\alpha\beta\eta}$ la plus petite congruence contenant $\equiv_\alpha$, $\leftrightarrow_\beta^*$ et $\leftrightarrow_\eta^*$. Contrairement à $\equiv_{\alpha\beta}$, la relation $\equiv_{\alpha\beta\eta}$ est fortement extensionnelle. L'intérêt pour η provient des propriétés suivantes :

i (Curry *et al.*, 1972) tout lambda-terme t admet une forme normale pour $\rightarrow_{\beta\eta}$ si et seulement si il admet une forme normale pour $\rightarrow_{\beta}$;

ii (Böhm, 1968) si t et u sont deux formes normales pour $\rightarrow_{\beta\eta}$ tels que $t \not\equiv_{\alpha} u$, alors toute congruence $\sim$ contenant $\equiv_{\alpha\beta}$ telle que $t \sim u$ est inconsistante.

Alors si t et u sont deux termes qui normalisent au moins faiblement, la plus petite congruence contenant $\equiv_{\alpha\beta\eta} \cup (t, u)$ est soit inconsistante, soit $\equiv_{\alpha\beta\eta}$.

1.2.6 Expressivité du lambda-calcul

Malgré une syntaxe et une sémantique d'apparence simple, le lambda-calcul est un formalisme qui bénéficie d'une expressivité maximale pour l'informaticien. C'est-à-dire que le lambda-calcul permet de définir une notion de calculabilité qui, comme nous allons le voir, est équivalente aux notions définies par les machines de Turing ou encore aux fonctions récursives. C'est Church (1936) qui proposa la lambda-definissabilité comme une notion de calculabilité effective (*effective calculability*).

Définition 1.2.8 (Fonction lambda-définissable). *Une fonction $f : \mathbb{N}^n \rightarrow \mathbb{N}$ est lambda-définissable s'il existe un lambda-terme F tel que pour tout $(p_1, \ldots, p_n) \in \mathbb{N}^n$, si $f(p_1, \ldots, p_n) = k$ alors $F\,\overline{p_1} \ldots \overline{p_n} \equiv_{\alpha\beta\eta} \overline{k}$ où la notation $\overline{p}$ représente l'entier de Church associé à l'entier p (voir exemple 1.2.1).*

L'équivalence entre machines de Turing et lambda-définissabilité est déjà démontrée par Turing lui-même dans l'appendice de son article introduisant ces machines.

Propriété 1.2.3 (Lambda-définissabilité et Turing calculabilité (Turing, 1936)). *Une fonction est lambda-définissable si et seulement si elle est calculable par une machine de Turing.*

Je ne définirai pas ici ce qu'est une machine de Turing ni ce que signifie calculable par une telle machine. Le lecteur peut se référer au livre de Papadimitriou (1994). Enfin une dernière notion intéressante et toujours équivalente est celle de fonction récursive présentée par Kleene.

Définition 1.2.9 (Fonctions récursives). *Si i est un entier naturel, notons $\mathcal{A}_i$ l'ensemble des fonctions de $\mathbb{N}^i$ vers $\mathbb{N}$. Notons $\mathcal{A}$ l'union des $\mathcal{A}_i$. L'ensemble des fonctions récursives est le plus petit sous-ensemble de $\mathcal{A}$ tel que*

fonctions initiales *il contient la fonction $S : n \mapsto n + 1$, la fonction $Z : n \mapsto 0$ et pour tous entiers p et $i \leqslant p$, la fonction $U_i^p : (n_1, \ldots, n_p) \mapsto n_i$;*

composition *s'il contient une fonction $f \in \mathcal{A}_n$ et pour une séquence de taille n de fonctions $\psi_i \in \mathcal{A}_p$, il contient ψ_i pour tout $1 \leqslant i \leqslant n$, alors il contient la fonction définie par*

$$\begin{array}{rccc} \varphi : & \mathbb{N}^p & \rightarrow & \mathbb{N} \\ & (k_1 \ldots k_p) & \mapsto & f(\psi_1(k_1 \ldots k_p) \ldots \psi_n(k_1 \ldots k_p)) \end{array}$$

récursion primitive *s'il contient $\psi \in \mathcal{A}_{n+2}$ et $\varphi \in \mathcal{A}_n$, alors il contient la fonction définie par*

$$\begin{array}{rrcl} \chi : & \mathbb{N}^{n+1} & \to & \mathbb{N} \\ & (0, p_1 \dots p_n) & \mapsto & \varphi(p_1 \dots p_n) \\ & (k+1, p_1 \dots p_n) & \mapsto & \psi(\chi(k, p_1 \dots p_n), k, p_1 \dots p_n) \end{array}$$

minimalisation *s'il contient $\psi \in \mathcal{A}_{n+1}$ tel que pour tout $(p_1 \dots p_n) \in \mathbb{N}^n$, il existe un entier p_{n+1} tel que $\psi(p_1 \dots p_{n+1}) = 0$, alors il contient la fonction définie par*

$$\begin{array}{rrcl} \varphi : & \mathbb{N}^n & \to & \mathbb{N} \\ & (p_1 \dots p_n) & \mapsto & \textit{le plus petit } p_{n+1} \textit{ tel que } \psi(p_1 \dots p_{n+1}) = 0 \end{array}$$

Kleene (1936) a montré que les classes des fonctions recursives et des fonctions lambda-définissables concordaient.

Propriété 1.2.4 (Lambda-définissabilité et recursivité (Kleene, 1936)). *Toute fonction lambda-définissable est une fonction récursive et inversement.*

Tous ces résultats démontrent que le lambda-calcul est un outil adapté à la modélisation du calcul.

1.2.7 Stratégies d'évaluation pour le lambda-calcul

Comme nous l'avons déjà expliqué, le lambda-calcul est un outil performant pour définir la sémantique de langages de programmation. L'exécution des programmes par une machine est généralement définie comme une exécution déterministe. En particulier la sélection d'un redex puis sa réduction sont des processus déterministes. Le choix de ce redex n'est pas anodin. Considérons par exemple le lambda-terme $(\lambda x.\lambda y.y)\,\Omega$ (le lambda-terme Ω est défini dans l'exemple 1.2.2). Si l'on choisit de toujours sélectionner le redex de tête, alors on obtient la forme normale $\lambda y.y$. Par contre si l'on choisit de toujours sélectionner le redex intérieur (celui de Ω), on obtient la séquence de réduction infinie

$$(\lambda x.\lambda y.y)\,\Omega \to_\beta (\lambda x.\lambda y.y)\,\Omega \to_\beta \dots$$

La plupart des langages de programmation choisissent donc l'une des deux solutions suivantes : lorsqu'une fonction est appliquée à un argument, constituant ainsi un redex en tête, soit on réduit toujours l'argument avant de réduire le redex de tête, soit on réduit toujours directement le redex de tête. Ces deux *stratégies d'évaluation* s'appellent respectivement *appel par valeur* et *appel par nom*. L'ensemble des valeurs pour le lambda-calcul contient tous les lambda-termes qui sont une abstraction ou une variable. Nous utiliserons le symbole ν pour représenter les valeurs du lambda-calcul. À partir de maintenant nous utiliserons les symboles π, ρ, τ pour représenter des lambda-termes quelconques

Définition 1.2.10 (Appel par valeur). *La stratégie d'appel par valeur pour la bêta-réduction consiste à ne réduire que les bêta-redex qui ne se trouvent pas dans le corps d'une abstraction, qui ne se trouvent pas à droite d'une application dont la partie gauche n'est pas une valeur et qui sont de la forme* $(\lambda x.\pi)\,\nu$*. Nous noterons la bêta-réduction par appel par valeur* $\to_{\beta v}$*.*

Définition 1.2.11 (Appel par nom). *La stratégie d'appel par nom pour la bêta-réduction consiste à ne pas réduire les bêta-redex qui se trouvent dans le corps d'une abstraction ou à droite d'une application. Nous noterons la bêta-réduction par appel par nom* $\to_{\beta n}$*.*

Par exemple le terme $(\lambda x.\lambda y.y)\,\Omega$ se réduit uniquement en $\lambda y.y$ en appel par nom et en lui même en appel par valeur. Il est donc fortement normalisant pour $\to_{\beta n}$ et au contraire admet une séquence de réduction infinie pour $\to_{\beta v}$. Nous allons d'abord étudier la stratégie d'appel par valeur, puis notre attention se portera sur la stratégie d'appel par nom.

Appel par valeur

Un contexte en appel par valeur, ou CBV-contexte, est un élément du langage hors-contexte défini par la grammaire en forme de Backus-Naur suivante.

$$E, F \quad ::= \quad \Box \mid \nu\, E \mid E\, \pi$$

L'ensemble des contextes en appel par valeur est un sous-ensemble des contextes unaires du lambda-calcul (définition 1.1.12). Rappelons que les notations $E[F]$ et $E[\pi]$ représentent le remplacement dans E du symbole $\Box$ respectivement par F et π. Notons la propriété suivante.

Propriété 1.2.5. *Un lambda-terme* τ *se réduit en appel par valeur si et seulement si il est de la forme* $E[(\lambda x.\pi)\,\nu]$*. Lorsque ces conditions sont vérifiées, son unique redex en appel par valeur est cette occurrence de* $(\lambda x.\pi)\,\nu$ *et son unique réduit est* $E[\pi[\nu/x]]$*.*

Démonstration. L'équivalence entre « τ se réduit par $\to_{\beta v}$ » et « τ est de la forme $E[(\lambda x.\pi)\,\nu]$ » provient directement de la définition 1.2.10 et de la définition de CBV-contexte. Il nous reste à prouver l'unicité du redex en appel par valeur dans τ quand celui-ci se réduit par $\to_{\beta v}$. Prouvons le par induction sur τ.

- **Variable ou Abstraction**. τ ne se réduit pas par $\to_{\beta v}$.
- **Application dont la partie gauche est une variable**. Dans ce cas, τ ne se réduit pas en tête en appel par valeur, et la partie gauche de l'application ne se réduit pas en tête par appel par valeur. Donc si τ se réduit par $\to_{\beta v}$, la partie droite de l'application se réduit par $\to_{\beta v}$ et, par hypothèse d'induction, son redex en appel par valeur est unique. Le redex de τ en appel par valeur est donc unique.

- **Application dont la partie gauche est une abstraction et la partie droite est une valeur**. La partie gauche de l'application étant une abstraction, elle ne se réduit pas en appel par valeur. Comme la partie droite est une valeur, cette partie droite ne se réduit pas en appel par valeur et donc l'unique redex de τ en appel par valeur est en tête.
- **Application dont la partie gauche est une abstraction et la partie droite n'est pas une valeur**. La partie gauche de l'application étant une abstraction, elle ne se réduit pas en appel par valeur. Comme la partie droite de l'application n'est pas une valeur, τ ne se réduit pas en tête en appel par valeur. Finalement puisque τ se réduit par $\rightarrow_{\beta v}$, la partie droite de l'application se réduit par $\rightarrow_{\beta v}$ et, par hypothèse d'induction, son redex en appel par valeur est unique. Le redex de τ en appel par valeur est donc unique.
- **Application dont la partie gauche est une application**. $\tau = (\pi\,\rho)\,\rho'$. Alors par définition de l'appel par valeur et puisque $(\pi\,\rho)$ n'est pas une valeur, on ne peut réduire de redex de ρ'. Si τ se réduit par $\rightarrow_{\beta v}$, alors $(\pi\,\rho)$ se réduit par $\rightarrow_{\beta v}$ et, par hypothèse d'induction, son redex en appel par valeur est unique. Le redex de τ en appel par valeur est donc unique.

□

La propriété 1.2.5 se reformule ainsi : $\rightarrow_{\beta v}$ est définie alternativement par le schéma de règle de réécriture

$$E[(\lambda x.\pi)\,\nu] \quad \rightarrow_{\beta v} \quad \pi[\nu/x]\ .$$

La manipulation des CBV-contextes ainsi que la sélection des redex (ici en appel par valeur) peut être rendue explicite. Le résultat obtenu s'appelle une machine abstraite. Une machine abstraite pour la bêta-réduction en appel par valeur se définit de la façon suivante. Un CBV-état est un couple $\langle\pi|E\rangle$ contenant un lambda-terme π et un CBV-contexte E. Un CBV-état $\langle\pi|E\rangle$ représente le lambda-terme $E[\pi]$ où l'occurrence de π dans le contexte E est sélectionnée. Si un CBV-état donné représente toujours un lambda-terme unique, plusieurs CBV-états peuvent représenter le même lambda-terme. En revanche et comme le montre la propriété 1.2.5, il ne peut en exister qu'un seul où un redex en appel par valeur est sélectionné. Par exemple le lambda-terme $(x\,y)\,z$ est représenté par

$$\langle(x\,y)\,z|\Box\rangle\ , \quad \langle x\,y|\Box\,z\rangle\ , \quad \langle x|(\Box\,y)\,z)\rangle \quad \text{et} \quad \langle y|(x\,\Box)\,z)\rangle\ .$$

Les règles de réécriture suivantes représentent la manipulation des CBV-contextes pour la sélection puis la réduction du redex en appel par valeur.

$$\begin{array}{lcl}
\langle\pi\,\rho|E\rangle & \rightarrow & \langle\pi|E[\Box\,\rho]\rangle \\
\langle\nu|E[\Box\,\pi]\rangle & \rightarrow & \langle\pi|E[\nu\,\Box]\rangle \\
\langle\nu|E[(\lambda x.\pi)\,\Box]\rangle & \rightarrow & \langle\pi[\nu/x]|E\rangle
\end{array}$$

Ces trois règles constituent une *machine abstraite* pour le lambda-calcul en appel par valeur.

Appel par nom

Ce que nous venons d'écrire s'adapte à l'appel par nom de la façon suivante. L'ensemble des CBN-contextes est définie comme le langage hors-contexte associé à la grammaire en forme de Backus-Naur suivante.

$$G, H \quad ::= \quad \Box \mid G\,\pi$$

Tout comme dans le cas de l'appel par valeur, si π se réduit en appel par nom, il s'écrit de façon unique sous la forme $G[(\lambda x.\pi)\,\tau]$. La relation $\rightarrow_{\beta n}$ peut être définie de façon équivalente par le schéma de règle de réécriture

$$G[(\lambda x.\pi)\,\tau] \quad \rightarrow_{\beta n} G[\pi[\tau/x]]\ .$$

Un CBN-état est un couple $\langle \pi | G \rangle$ contenant un lambda-terme π et un CBN-contexte G. Alors la réduction de la machine abstraite pour la bêta-réduction en appel par nom est définie par les règles suivantes.

$$\begin{array}{lcl} \langle \pi\,\rho | G \rangle & \rightarrow & \langle \pi | G[\Box\,\rho] \rangle \\ \langle \lambda x.\pi | G[\Box\,\tau] \rangle & \rightarrow & \langle \pi[\nu/x] | G \rangle \end{array}$$

1.2.8 Opérateurs de contrôle

L'opérateur `call/cc` (*call with current continuation*) fut tout d'abord introduit dans le langage Scheme (Kelsey *et al.*, 1998). Dans un cadre de programmation fonctionnelle, il permet d'accéder à la *continuation courante*. Cet opérateur est maintenant présent dans des langages comme Scheme ou SML/NJ, et sous des formes faibles dans Java, OCaml, ou C++. Cette section présente des opérateurs pour le lambda-calcul introduits dans le but de modéliser des opérateurs de contrôle tels que `call/cc`. Néanmoins et avant de définir de tels opérateurs, il est nécessaire de définir ce qu'est une *continuation*. Cette notion varie lorsque l'on se place dans un cadre d'appel par valeur ou d'appel par nom. Nous nous placerons dans le cas de l'appel par valeur, mais la plupart des manipulations et résultats que nous allons obtenir s'écrivent aussi dans le cas de l'appel par nom. Dans le cadre du lambda-calcul en appel par valeur, une continuation est un CBV-contexte. Nous allons à présent définir des opérateurs de contrôle permettant d'effacer ou encore de stocker puis restaurer de telles continuations. La syntaxe du lambda-calcul est étendue à la grammaire suivante.

$$\pi, \rho, \tau \quad ::= \quad x \mid \lambda x.\pi \mid \pi\,\rho \mid \mathcal{C}(\pi) \mid \mathcal{A}(\pi)$$

Alors on rajoute à la bêta-réduction en appel par valeur les réductions suivantes.

$$\begin{array}{rcl} E[\mathcal{A}(\pi)] & \rightarrow & \pi \\ E[\mathcal{C}(\pi)] & \rightarrow & \pi\,(\lambda z.\mathcal{A}(E[z])) \end{array}$$

Le terme $\mathcal{A}(\pi)$ efface bel et bien le contexte courant et continue par l'évaluation de π, comme le montre la première des règles ci-dessus. Le terme $\mathcal{C}(\pi)$ permet quant

à lui de stocker le contexte actuel pour une réutilisation future. En effet considérons un terme $E[\mathcal{C}(\lambda x.\pi)]$. Il s'évalue par la deuxième règle ci-dessus puis par $\rightarrow_{\beta v}$ en $\pi[(\lambda z.\mathcal{A}(E[z]))/x]$. Le terme $\lambda z.\mathcal{A}(E[z])$ représente le contexte E : appliqué à une valeur ν, son évaluation effacera toujours le contexte courant pour considérer $E[\nu]$. L'évaluation de $E[\mathcal{C}(\lambda x.\pi)]$ consiste donc à *enregistrer* le contexte E sous la forme $\lambda z.\mathcal{A}(E[z])$ dans la variable x puis à continuer l'évaluation de π.

Pour manipuler les opérateurs de contrôle $\mathcal{C}$ et $\mathcal{A}$, les deux règles que nous rajoutons à la machine abstraite sont les suivantes.

$$\begin{array}{lcl} \langle \mathcal{A}(\pi)|E\rangle & \rightarrow & \langle \pi|\Box\rangle \\ \langle \mathcal{C}(\pi)|E\rangle & \rightarrow & \langle \pi|\Box\,(\lambda z.\mathcal{A}(E[z]))\rangle \end{array}$$

Remarquons que l'opérateur $\mathcal{A}$ peut être défini grâce à $\mathcal{C}$ par $\mathcal{A}(\pi) = \mathcal{C}(\lambda x.\pi)$ où x n'apparaît pas dans π. En outre l'opérateur $\mathcal{C}$ diffère de `call/cc` par le fait qu'il ne s'évalue pas dans la continuation dans laquel il se trouve. On peut utiliser un opérateur $\mathcal{K}$ plus proche de `call/cc` et dont la réduction serait

$$E[\mathcal{K}(\pi)] \rightarrow E[\pi\,(\lambda x.\mathcal{A}(E[z]))]\,.$$

En fait cet opérateur peut être défini par $\mathcal{K}(\pi) = \mathcal{C}(\lambda z.(z\,(\pi\,z)))$.

1.3 Calcul de réécriture

Le filtrage de motifs est l'un des mécanismes de base cruciaux de la réécriture. La capacité de discriminer de tels motifs a toujours été présente dans la modélisation des processus de traitement automatique de l'information. Le filtrage par motif est aussi largement utilisé par les langages de programmation fonctionnelle (comme ML, Haskell, Ocaml), logique (comme Prolog), à base de réécriture (comme Maude, Elan, Tom) ou encore par les assistants de preuve (comme Coq). Le calcul de réécriture, aussi appelé rho-calcul et introduit par Cirstea et Kirchner (2001); Cirstea (2000), propose de réunir le lambda-calcul et la réécriture dans un même formalisme. Tous les ingrédients de base de la réécriture, par exemple les notions de motif, de règle de réécriture, d'application d'une règle et de résultat, deviennent alors des objets explicites. Le filtrage de motifs peut alors être utilisé librement et les règles de réécriture deviennent des objets de première classe du calcul qui peuvent en particulier être modifiées par le calcul lui-même. Le rho-calcul a tout d'abord été introduit pour fournir une sémantique au langage ELAN (Borovanský *et al.*, 2001, 2004) basé sur la réécriture. Il peut être étendu de plusieurs façons intéressantes et différentes. Tout d'abord on peut choisir d'utiliser une version avec contraintes de filtrage explicites (Cirstea *et al.*, 2007). Une version impérative du calcul ainsi qu'un interpréteur certifié ont aussi été développés (Liquori et Serpette, 2008). Le rho-calcul peut aussi être étendu en généralisant les motifs en graphes (Baldan *et al.*, 2007). Notons aussi la comparaison entre rho-calcul et combinatory rewrite systems (Bertolissi *et al.*, 2006; Bertolissi et Kirchner, 2007).

Cette section est divisée en cinq sous-sections. Les deux premières introduisent respectivement la syntaxe et la sémantique du calcul de réécriture. La troisième étudie la confluence du calcul et mentionne des résultats la concernant. La quatrième définit un traîtement des échecs de filtrage dû à Benjamin Wack. Enfin la dernière sous-section établit l'expressivité du calcul obtenu en exhibant un plongement des systèmes de réécriture de termes dans le calcul du réécriture. Plusieurs de mes contributions personnelles sont présentées ici. Plus exactement, j'ai mis au point une preuve de la confluence de $\rightarrow^{v}_{\rho\delta\gamma}$ (propriété 1.3.2) puis, avec le concours de Benjamin Wack et Horatiu Cirstea, nous avons étendu l'encodage des systèmes de réécriture de termes convergents dans le calcul de réécriture présenté par Cirstea *et al.* (2003) en un encodage des systèmes de réécriture de termes sans restriction de convergence dans le calcul de réécriture. Ces résultats sont exposés dans un article (Cirstea *et al.*, 2006a) que j'ai présenté au workshop WRLA'06 en avril 2006.

1.3.1 Syntaxe du rho-calcul

L'idée à la base du calcul de réécriture est une analogie entre l'abstraction du lambda-calcul et les règles de réécriture. En effet une abstraction $\lambda x.t$ représente la fonction qui à tout lambda-terme x associe le terme t. Une règle de réécriture $l \rightarrow r$ représente un objet qui transforme toute instance de l en l'instance correspondante de r. Ces intuitions identifient alors les objets $x \rightarrow t$ et $\lambda x.t$. On pourrait d'ailleurs imaginer des systèmes de réécriture ne comprenant que des règles de la forme $x \rightarrow t$. Le rho-calcul est alors au lambda-calcul ce que la réécriture est à de tels systèmes. Pour l'obtenir, on remplace la lambda-abstraction $\lambda x.t$ du lambda-calcul par une rho-abstraction notée $u \rightharpoonup t$ (mais que l'on pourrait noter $\lambda u.t$) où u peut être n'importe quel objet du langage. Plus précisément, la syntaxe du rho-calcul est la suivante. Soit α une signature.

Définition 1.3.1 (Rho-termes). *L'ensemble des rho-termes est défini comme le langage associé à la grammaire hors-contexte décrite sous forme de Backus-Naur*

$$t, u, v \ ::= \ x \mid a \mid t \rightharpoonup u \mid t\ u \mid t \wr u$$

où x représente les éléments d'un ensemble infini dénombrable X de variables et a représente les éléments d'un ensemble infini dénombrable C de constantes. En général, on ne considère qu'un sous-ensemble de ces rho-termes que l'on choisit de la façon suivante : on définit tout d'abord un ensemble de motifs *$\mathcal{P}$ parmi les parties l'ensemble des rho-termes puis on considère l'ensemble de rho-termes associé à la grammaire*

$$t, u, v \ ::= \ x \mid a \mid p \rightharpoonup t \mid t\ u \mid t \wr u$$

où p représente les éléments de l'ensemble de motifs *$\mathcal{P}$.*

Les rho-termes de la forme $p \rightharpoonup t$ sont appelés des rho-abstractions. Ils représentent moralement des règles de réécriture $p \rightarrow t$. Les rho-termes de la forme $t\ u$

sont appelés des applications. Remarquons qu'en plus de l'extension de la lambda-abstraction en rho-abstraction, le langage des lambda-termes a été étendu grâce à une nouvelle construction syntaxique $t \wr u$ (pour tous rho-termes t et u) que nous appellerons *structure*.

La définition 1.3.1 est paramétrée par un ensemble de motifs (en plus des ensembles de variables et de constantes). Nous verrons que ce choix est crucial. Nous n'étudierons en détail que le cas où l'ensemble des motifs est le langage associé à la grammaire hors-contexte en forme de Backus-Naur suivante.

$$p \ ::= \ x \mid a \mid a\, p_1\, p_2 \ldots p_n$$

Les motifs de cet ensemble sont appelés des motifs algébriques ou encore *termes* algébriques.

1.3.2 Sémantique du rho-calcul

Tout comme la lambda-abstraction, la rho-abstraction est aussi un mécanisme de liaison de variables. On souhaite par exemple identifier les règles de réécriture $f(x,y) \to g(x,g(y))$ et $f(z,x') \to g(z,g(x'))$. Nous allons donc encore une fois utiliser une notion d'alpha-conversion. Contrairement au cas du lambda-calcul ou l'unique lieur λ ne lie toujours qu'une seule variable, ici le constructeur syntaxique $\rightharpoonup$ permet de lier un nombre arbitraire de variables. Pour être précis, dans une rho-abstraction $p \rightharpoonup t$, toutes les variables qui sont libres dans p sont liées dans $p \rightharpoonup t$. On peut donc définir, pour tout rho-terme t, l'ensemble des variables libres de t, dénoté $\mathcal{FV}(t)$, ainsi que l'ensemble des variables liées de t, dénoté $\mathcal{BV}(t)$, comme suit.

Définition 1.3.2 (Variables libres et liées d'un rho-terme).

$$\begin{array}{llll}
\mathcal{FV}(t) = \{x\} & \textit{et} & \mathcal{BV}(t) = \emptyset & \textit{si } t = x\ ; \\
\mathcal{FV}(t) = \emptyset & \textit{et} & \mathcal{BV}(t) = \emptyset & \textit{si } t = a\ ; \\
\mathcal{FV}(t) = \mathcal{FV}(u) \backslash \mathcal{FV}(p) & \textit{et} & \mathcal{BV}(t) = \mathcal{BV}(u) \cup \mathcal{FV}(p) & \textit{si } t = p \rightharpoonup u\ ; \\
\mathcal{FV}(t) = \mathcal{FV}(u) \cup \mathcal{FV}(v) & \textit{et} & \mathcal{BV}(t) = \mathcal{BV}(u) \cup \mathcal{BV}(v) & \textit{si } t = u\ v\ ; \\
\mathcal{FV}(t) = \mathcal{FV}(u) \cup \mathcal{FV}(v) & \textit{et} & \mathcal{BV}(t) = \mathcal{BV}(u) \cup \mathcal{BV}(v) & \textit{si } t = u \wr v\ .
\end{array}$$

Notons que les motifs algébriques ne contiennent pas de rho-abstraction. Par conséquent, puisque nous choisissons de ne considérer que des motifs algébriques, alors pour tout motif p, $\mathcal{BV}(p) = \emptyset$. Si l'on choisit de considérer des motifs contenant des lieurs, alors il faut reformuler en partie la définition 1.3.2 par

$$\mathcal{FV}(t) \ = \ \mathcal{FV}(u) \backslash \mathcal{FV}(p) \ \text{ et } \ \mathcal{BV}(t) \ = \ \mathcal{BV}(u) \cup \mathcal{FV}(p) \cup \mathcal{BV}(p) \ \text{ si } t = p \rightharpoonup u\,.$$

Remarquons aussi que lorsque les motifs contiennent des lieurs, il est impossible de définir les notions de position, contexte et substitution comme nous l'avons fait pour le lambda-calcul, c'est-à-dire en se reposant sur ces mêmes notions sur les algèbres de termes : en effet en section 1.2 nous avons défini l'ensemble des lieurs

du lambda-calcul comme un ensemble en bijection avec l'ensemble des variables. Il nous faudrait faire de même avec une bijection entre l'ensemble des lieurs du rho-calcul et l'ensemble des motifs. Hors si l'on veut autoriser les motifs à contenir des lieurs, cette définition devient circulaire. On doit donc définir ces trois notions de façon différente. Cela nous est épargné car nous avons choisi de ne considérer que des motifs algébriques et ceux-ci ne contiennent pas de lieur. Les notions de position, contexte et substitutions sont donc directement héritées des algèbres de termes, tout comme en section 1.2. La relation d'alpha-conversion est définie comme suit.

Définition 1.3.3 (Alpha-conversion sur les rho-termes). *La relation d'alpha-conversion sur les rho-termes, que nous noterons (aussi)* $\equiv_\alpha$*, est définie comme la plus petite congruence telle que*

$$p \rightharpoonup t \quad \equiv_\alpha \quad p[y/x] \rightharpoonup t[y/x]$$

pour tout motif p*, rho-terme* t *et variables* x *et* y *tels que* $x \notin \mathcal{BV}(t)$*,* $x \in \mathcal{FV}(p)$ *et* $y \notin \mathcal{FV}(t) \cup \mathcal{BV}(t)$ *.*

Tout comme dans le cadre du lambda-calcul et puisque nous sommes en présence de lieurs et d'alpha-conversion, nous choisissons de n'utiliser que des substitutions qui évitent les captures. La condition suffisante pour les éviter reste la même qu'en section 1.2 : nous n'appliquerons une substitution σ à un rho-terme t que si $\mathcal{BV}(t) \cap (\mathrm{dom}(\sigma) \cup \mathcal{FV}(\mathrm{codom}(\sigma))) = \emptyset$. Alors la rho-réduction est définie comme suit.

Définition 1.3.4 (Rho-réduction). *La rho-réduction en tête, notée* $\mapsto_\rho$*, est définie comme la relation* $\equiv_\alpha \ \circ \rho \circ \ \equiv_\alpha$ *où* ρ *est la relation définie par* $(p \rightharpoonup t)\ u \ \ \rho \ \ t\sigma$ *où* σ *est la solution du problème de filtrage syntaxique* $p \overset{?}{\preccurlyeq} u$ *et évite les captures dans* t*. La rho-réduction, notée* $\to_\rho$ *est alors définie comme* $\equiv_\alpha \ \circ \bar{\rho} \circ \ \equiv_\alpha$ *où* $\bar{\rho}$ *est la clôture par contexte de* $\mapsto_\rho$*. Finalement la rho-équivalence, notée* $\leftrightarrow_\rho^*$ *est définie comme la clôture réflexive, symétrique et transitive de* $\to_\rho$*.*

Remarquons tout d'abord que la substitution $[u/x]$ est toujours une solution du problème de filtrage $x \overset{?}{\preccurlyeq} u$. Le rho-terme $(x \rightharpoonup t)u$ se réduit donc toujours par $\to_\rho$ en $t[u/x]$. Cela suggère la traduction suivante du lambda-calcul vers le rho-calcul. Si t est un lambda-terme, alors t_ρ est défini comme suit : $[x]_\rho = x$ pour toute variable x ; $[\lambda x.t]_\rho = x \rightharpoonup [t]_\rho$ pour toute abstraction $\lambda x.t$; enfin $[t_1 t_2]_\rho = [t_1]_\rho\ [t_2]_\rho$ pour toute application $t_1 t_2$. Si t et u sont deux lambda-termes tels que $t \to_\beta u$, alors $[t]_\rho \to_\rho [u]_\rho$. On récupère ainsi dans le rho-calcul les examples 1.2.1 et 1.2.2 du lambda-calcul. En particulier la relation $\to_\rho$ n'est pas faiblement normalisante puisque $[\Omega]_\rho = (x \rightharpoonup xx)(x \rightharpoonup xx) \to_\rho (x \rightharpoonup xx)(x \rightharpoonup xx) = [\Omega]_\rho$.

Nous sommes restreints ici aux motifs algébriques. Dans ce cas, le résultat 1.1.5 s'applique. On peut toujours calculer l'*unique* substitution σ qui est solution d'un problème de filtrage $p \overset{?}{\preccurlyeq} u$. L'unicité et la décidabilité peuvent être perdue lorsqu'on considère un filtrage équationnel, ou encore lorsque les motifs contiennent des lieurs.

Dans ce cas, il faut résoudre des problèmes de filtrage équationnel pour une théorie équationnelle contenant $\equiv_\alpha$ et $\to_{\rho\delta}{}^*$ (Faure, 2007).

La construction syntaxique de structure $t \wr u$ est utilisée pour représenter des ensembles de résultats et des ensembles de règles de réécriture. Si t et u sont eux-mêmes des ensembles, alors il faut voir $t \wr u$ comme l'union de ces deux ensembles. En particulier lorsqu'on utilise un filtrage non unitaire, la règle de rho-réduction s'écrit

$$(p \rightharpoonup t)u \quad \mapsto_\rho \quad t\sigma_1 \wr t\sigma_2 \wr \cdots \wr t\sigma_n$$

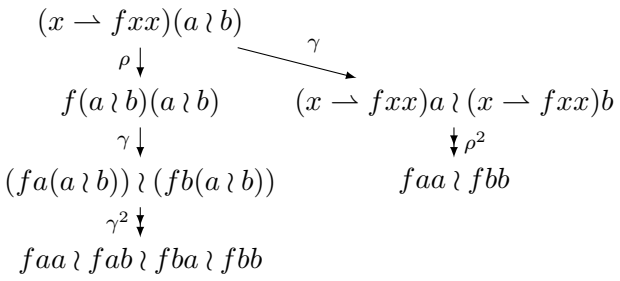
où $\sigma_1, \ldots, \sigma_n$ sont les solutions du problème de filtrage $p \overset{?}{\preccurlyeq} u$. Nous supposons d'ailleurs que le symbole $\wr$ est associatif et commutatif. Manipuler ces ensembles de résultats peut alors nécessiter deux réductions supplémentaires, décrites par les règles de réécriture suivantes.

$$\begin{aligned} \gamma : &\ \ t(u \wr v) \ \to\ tu \wr tv \\ \delta : &\ \ (t \wr u)v \ \to\ tv \wr uv \end{aligned}$$

Nous noterons $\to_{\rho\delta}$ et $\to_{\rho\delta\gamma}$ les extensions de $\to_\rho$ avec respectivement la règle δ et les règles δ et γ.

1.3.3 (Non) Confluence du rho-calcul

Notons que $\to_{\rho\delta\gamma}$ n'est pas confluent, comme le démontre le contre-exemple suivant proposé par Cirstea (2000).



Wack (2005) a montré que le contre-exemple de Klop que nous avons présenté en sous-section 1.1.3 peut aussi être transposé sous forme de rho-terme : soit Y le terme

$$(y \rightharpoonup x \rightharpoonup (x(yyx)))(y \rightharpoonup x \rightharpoonup (x(yyx))) \ .$$

Remarquons que ce rho-terme est aussi définissable dans le lambda-calcul et que sa propriété majeure est que pour tout terme t, on a $Yt \to_\rho^* t(Yt)$. Un tel terme est d'ailleurs appelé un combinateur de point fixe. On définit alors les termes C et A respectivement par $C = Y(y \rightharpoonup x \rightharpoonup ((dzz) \rightharpoonup e)(dx(yx)))$ et $A = YC$. Notons que $C \to_\rho^* (y \rightharpoonup x \rightharpoonup (dzz \rightharpoonup e)(dx(yx)))C \to_\rho x \rightharpoonup (dzz \rightharpoonup e)(dx(Cx))$. On a alors les rho-réductions suivantes.

$$\begin{array}{ccccc} A & \twoheadrightarrow & C\,A & \rightarrow & ((dzz) \rightharpoonup e)(dA(CA)) \\ & & \downarrow\!\!\downarrow & & \downarrow \\ & & C\,e & & ((dzz) \rightharpoonup e)(d(CA)(CA)) \\ & & & & \downarrow \\ & & & & e \end{array}$$

Cela montre que $\rightarrow_\rho$ n'est pas non plus confluent. La confluence du rho-calcul est étudiée de manière détaillée par Cirstea (2000). Un premier critère est celui des *rigid pattern*.

Définition 1.3.5 (Rigid Pattern Condition (RPC)). *On dit qu'un motif p est* rigide *si pour toute substitution σ et terme t tels que $p\sigma \rightarrow^*_{\rho\delta} t$, alors $t = p\sigma'$ avec $\sigma \rightarrow^*_{\rho\delta} \sigma'$.*

La notation $\sigma \rightarrow^*_{\rho\delta} \sigma'$ signifie ici que si $\sigma = [t_1/x_1 \dots t_n/x_n]$, alors il existe $t'_1 \dots t'_n$ tels que $t_i \rightarrow^*_{\rho\delta} t'_i$ pour tout i et $\sigma' = [t'_1/x_1 \dots t'_n/x_n]$. Notons que tout motif algébrique linéaire est rigide. On obtient alors le résultat général suivant.

Propriété 1.3.1 (Confluence par RPC (Barthe *et al.*, 2003)). *La relation $\rightarrow_{\rho\delta}$ est confluente sur l'ensemble des rho-termes dont les motifs sont rigides.*

Notons que cela ne s'applique pas aux filtrages non syntaxiques. Cela ne s'applique pas non plus à $\rightarrow_{\rho\delta\gamma}$. Une autre façon d'assurer la confluence du calcul est d'utiliser une stratégie d'appel par valeur, comme proposé par Cirstea (2000); Cirstea *et al.* (2006a). L'ensemble des valeurs pour $\rho\delta\gamma$ est défini comme le langage associé à la grammaire hors-contexte en forme de Backus-Naur suivante.

$$v^{\rho\delta\gamma} \quad ::= \quad a \mid a\,v_1^{\rho\delta\gamma} \dots v_n^{\rho\delta\gamma} \mid p \rightarrow t$$

où a, t et p représentent toujours respectivement les constantes, les rho-termes et les motifs. L'ensemble des valeurs pour $\rho\delta$ est défini comme suit.

$$v^{\rho\delta} \quad ::= \quad a \mid a\,v_1^{\rho\delta} \dots v_n^{\rho\delta} \mid p \rightarrow t \mid v_1^{\rho\delta} \wr v_2^{\rho\delta}$$

Notons $\rightarrow^v_{\rho\delta}$ la relation $\rightarrow_{\rho\delta}$ pour laquelle l'application des rho-réductions est restreinte aux redex de la forme $(p \rightharpoonup t)v^{\rho\delta}$. De même notons $\rightarrow^v_{\rho\delta\gamma}$ la relation $\rightarrow_{\rho\delta\gamma}$ pour laquelle l'application des rho-réductions est restreinte aux redex de la forme $(p \rightharpoonup t)v^{\rho\delta\gamma}$. Alors les confluences de chacune de ces deux réductions sont démontrées respectivement par Cirstea (2000) puis Cirstea *et al.* (2006a).

Propriété 1.3.2 (Confluence de l'appel par valeur (Cirstea *et al.*, 2006a)). *Les relations $\rightarrow^v_{\rho\delta}$ et $\rightarrow^v_{\rho\delta\gamma}$ sont confluentes sur les rho-termes dont les motifs sont linéaires.*

La preuve de la confluence de $\rightarrow^v_{\rho\delta\gamma}$ est l'une de mes contributions personnelles. Elle est détaillée dans la version longue (Cirstea *et al.*, 2006b) de l'article WRLA'06 (Cirstea *et al.*, 2006a).

1.3.4 Échecs de filtrage

Contrairement à la bêta-réduction pour laquelle tout terme $(\lambda x.t)u$ se réduit, pour la rho-réduction il existe des termes $(p \rightharpoonup t)u$ qui ne se réduisent pas, précisément quand le problème de filtrage $p \overset{?}{\preccurlyeq} u$ n'admet aucune solution. L'exemple le plus simple est le rho-terme $(a \rightharpoonup b)c$: comme a et c sont deux constantes distinctes, le problème de filtrage $a \overset{?}{\preccurlyeq} c$ n'admet aucune solution. Ce terme ne se réduit donc pas par $\to_\rho$. Considérons comme second exemple le terme $(fxx \rightharpoonup x)(fa((x \rightharpoonup x)a))$. Le filtrage $fxx \overset{?}{\preccurlyeq} (fa((x \rightharpoonup x)a))$ n'admet aucune solution, mais si l'on réduit le redex intérieur $(x \rightharpoonup x)a \to_\rho a$, alors on obtient le terme $(fxx \rightharpoonup x)(faa)$ et le problème de filtrage $fxx \overset{?}{\preccurlyeq} faa$. On peut donc réduire ce redex en a. Le rho-redex extérieur a été déclenché par la réduction du sous-terme $(x \rightharpoonup x)a$. Un rho-redex peut aussi être déclenché par la réduction d'un redex extérieur : considérons le terme $(x \rightharpoonup (a \rightharpoonup b)x)a$ (où a et b sont des constantes). Ici le rho-redex intérieur $(a \rightharpoonup b)x$ ne peut se réduire. Si l'on réduit le rho-redex extérieur, on obtient $(a \rightharpoonup b)a$ qui se réduit alors en b. La réduction du rho-redex extérieur, par l'instanciation de x, a permis que le rho-redex intérieur se réduise ensuite. Nous venons de mettre le doigt sur une différence importante entre le calcul de réécriture et les systèmes de réécriture : dans ces derniers, les erreurs de filtrage (« il n'existe aucune solution au problème de filtrage $p \overset{?}{\preccurlyeq} u$ » que nous noterons $p \overset{?}{\not\preccurlyeq} u$) sont éliminées et n'apparaissent jamais comme un *résultat* du calcul. Il est possible de tendre vers une telle gestion des erreurs dans le rho-calcul. On décide d'utiliser une nouvelle constante stk qui représentera les erreurs de filtrage. Il faut caractériser les rho-redex qui ne se réduiront jamais, malgré les réductions potentielles de redex intérieurs ou extérieurs : si pour aucune instanciation et réduction de l'argument, il n'existe de solution au filtrage, alors le ρ-redex ne se réduira pas. Autrement dit, si pour tous σ et u' tels que $u\sigma \to^*_{\rho(\delta)(\gamma)} u'$, $p \overset{?}{\not\preccurlyeq} u'$, alors $(p \rightharpoonup t)u$ ne se réduira pas. Une telle caractérisation pose problème car elle est probablement indécidable. En effet, u peut contenir un rho-terme ayant un nombre arbitraire (éventuellement infini) de réductions possibles, qu'il faut *a priori* toutes explorer pour décider si le problème de filtrage admettra une solution après réductions potentielles. Une solution proposée par Cirstea *et al.* (2003) et Wack (2005) est de se contenter d'une condition moins forte mais suffisante. La relation $\not\trianglelefteq$ entre motifs et rho-termes est définie comme suit.

$$
\begin{array}{lcll}
\mathsf{stk} & \not\trianglelefteq & g\,u_1 \dots u_n & \text{si } g \neq \mathsf{stk} \\
\mathsf{stk} & \not\trianglelefteq & p \rightharpoonup t & \\
f\,t_1 \dots t_m & \not\trianglelefteq & g\,u_1 \dots u_n & \text{si } f \neq \text{ ou } n \neq m \text{ ou } \exists i,\ t_i \not\trianglelefteq u_i \\
f\,t_1 \dots t_n & \not\trianglelefteq & \mathsf{stk} & \\
f\,t_1 \dots t_n & \not\trianglelefteq & p \rightharpoonup u & \\
f\,t_1 \dots t_n & \not\trianglelefteq & (p \rightharpoonup t)u & \text{si } p \not\trianglelefteq u \text{ ou } f\,t_1 \dots t_n \not\trianglelefteq t
\end{array}
$$

À partir de cette relation, on peut définir la réduction $\to_{\mathsf{stk}}$ qui gère la détection des échecs.

$$
\begin{array}{lcll}
(p \rightharpoonup t)u & \to_{\mathsf{stk}} & \mathsf{stk} & \text{si } p \not\trianglelefteq u \\
\mathsf{stk} \wr t & \to_{\mathsf{stk}} & t &
\end{array}
\qquad\qquad
\begin{array}{lcl}
t \wr \mathsf{stk} & \to_{\mathsf{stk}} & t \\
\mathsf{stk}\,t & \to_{\mathsf{stk}} & \mathsf{stk}
\end{array}
$$

Nous noterons respectivement $\to_{\rho}^{\mathsf{stk}}$, $\to_{\rho\delta}^{\mathsf{stk}}$ et $\to_{\rho\delta\gamma}^{\mathsf{stk}}$ les relations $\to_{\mathsf{stk}} \cup \to_{\rho}$, $\to_{\mathsf{stk}} \cup \to_{\rho\delta}$ et $\to_{\mathsf{stk}} \cup \to_{\rho\delta\gamma}$. La relation $\not\sqsubseteq$ est correcte dans le sens qu'elle implique l'échec du filtrage, comme démontré par Wack (2005) : si $p \not\sqsubseteq t$ et $t\sigma \to_{\rho\delta}^{\mathsf{stk}*} t'$, alors $p \overset{?}{\not\leftarrow} t'$.

1.3.5 Expressivité et encodage des systèmes de réécriture de termes

Le rho-calcul est un formalisme très expressif. Comme le démontre l'encodage $[\_]_{\rho}$ des lambda-termes vers les rho-termes, le rho-calcul a (au moins) le même pouvoir expressif que le lambda-calcul. Il est donc au moins Turing complet. Un autre résultat intéressant quant à l'expressivité du rho-calcul avec gestion des échecs de filtrage et que l'on peut y encoder tout système de réécriture de termes. Plus précisément, on peut encoder tout système de réécriture de termes convergent avec $\to_{\rho\delta}^{\mathsf{stk}}$, comme démontré par Cirstea *et al.* (2003). Cette encodage est étendu aux systèmes de réécriture de termes sans restriction de convergence avec $\to_{\rho\delta\gamma}^{\mathsf{stk}}$ par Cirstea *et al.* (2006a). J'ai personnellement contribué à cette extension. Il est néanmoins nécessaire d'utiliser l'appel par valeur et le résultat 1.3.2 (étendu avec la constante stk). Le rho-calcul sous-jacent est donc confluent. Nous noterons $\to_{\rho\delta\gamma}^{v\mathsf{stk}}$ la réduction $\to_{\rho\delta\gamma}^{\mathsf{stk}}$ restreinte à l'appel par valeur. Le reste de cette section présente cet encodage des systèmes de réécriture dans le rho-calcul et utilise $\to_{\rho\delta\gamma}^{v\mathsf{stk}}$. Pour être précis, à partir d'un système de réécriture de termes $\mathcal{R}$, nous allons construire deux rho-termes $\Omega_{\mathcal{R}}^{1}$ et $\Omega_{\mathcal{R}}$ tels que $\Omega_{\mathcal{R}}^{1}\, t$ représente (i.e. se réduit en) le réduit en un pas de t par $\mathcal{R}$ et $\Omega_{\mathcal{R}}\, t$ représente la forme normale de t par $\mathcal{R}$ (si elle existe).

Sélection des Règles. Comme nous souhaitons calculer des formes normales, il nous faut décider quand une réduction peut avoir lieu, c'est-à-dire quand une règle de $\mathcal{R}$ peut être utilisée, et agir en conséquence : si l'une des règles peut être appliquée à t, alors on réduit t ; sinon c'est une forme normale et t n'est pas modifié. La capacité à discriminer les règles que l'on peut appliquer à un certain argument est habituellement encodé dans le rho-calcul par le rho-terme $\mathsf{first} = x' \rightharpoonup y' \rightharpoonup x \rightharpoonup (\mathsf{stk} \rightharpoonup y'\, x \wr y \rightharpoonup y)\,(x'\, x)$. Lorsqu'on l'applique à trois rho-termes t, u et v, le terme first réduit tout d'abord $t\, v$. Si le résultat de cette dernière réduction n'est pas stk, alors first renvoie ce résultat. Sinon il réduit $u\, v$ et renvoie le résultat obtenu.

$$\mathsf{first}\, t\, u\, v \to_{\rho\delta\gamma}^{v\mathsf{stk}*} v^{\rho\delta\gamma} \text{ si } t\, v \to_{\rho\delta\gamma}^{v\mathsf{stk}*} v^{\rho\delta\gamma} \text{ ou bien si } t\, v \to_{\rho\delta\gamma}^{v\mathsf{stk}*} \mathsf{stk} \text{ et } u\, v \to_{\rho\delta\gamma}^{v\mathsf{stk}*} v^{\rho\delta\gamma} .$$

Intuitivement, si l'on remplace t par le rho-terme qui représente $\mathcal{R}$ et u par l'identité $x \rightharpoonup x$, alors $\mathsf{first}\, t\, u\, v$ appliquera $\mathcal{R}$ à v si une règle lui est applicable et le laissera inchangé sinon : le cas « $t\, v \to_{\rho\delta\gamma}^{v\mathsf{stk}*} v^{\rho\delta\gamma}$ » correspond à l'application d'une règle de $\mathcal{R}$ alors que le cas « $t\, v \to_{\rho\delta\gamma}^{v\mathsf{stk}*} \mathsf{stk}$ et $u\, v \to_{\rho\delta\gamma}^{v\mathsf{stk}*} v^{\rho\delta\gamma}$ » correspond au cas où aucune règle de $\mathcal{R}$ n'est applicable.

Propagation des Contextes. Les systèmes de réécriture permettent des transformations sur n'importe quel *sous-terme* de l'objet que l'on réécrit. Au contraire le

rho-calcul ne permet que d'appliquer une règle en tête de son argument. L'encodage des systèmes de réécriture doit donc explicitement propager l'application du système en profondeur dans le terme. Si la réécriture en tête échoue sur tous les sous-termes, il faut renvoyer stk de manière globale. La propagation de stk doit donc être faite pour tous les contextes. Pour cela et pour tout symbole f d'arité $n \geqslant 1$ de la signature α du système de réécriture $\mathcal{R}$, nous définissons un terme Γ_k^f comme suit.

$$\Gamma_k^f = \nu \rightharpoonup \overline{x}^n \rightharpoonup (nop\, z \rightharpoonup z)(\mathsf{first} \begin{pmatrix} \mathsf{stk} \rightharpoonup nop\, \mathsf{stk} \\ y \rightharpoonup nop(f\, x_1 \dots x_{k-1} y \dots x_n)) \end{pmatrix} (\nu\, x_k))$$

où nop est un symbole qui n'est pas dans α et pour tout $n \geqslant 1$, $\overline{x}^n \rightharpoonup t$ représente $x_1 \rightharpoonup x_2 \rightharpoonup \cdots \rightharpoonup x_n \rightharpoonup t$. Chaque Γ_k^f permet d'exprimer l'application d'un rho-terme au sous-terme t_k d'un autre rho-terme $f\, t_1 \dots t_n$. Leur comportement est résumé par la propriété suivante.

Propriété 1.3.3. *Soit f un symbole de α d'arité n. Soit $t_1 \dots t_n$ des termes algébriques et t un terme arbitraire. Soit $v_1^{\rho\delta\gamma} \dots v_p^{\rho\delta\gamma}$ des valeurs pour $\delta\gamma$. Alors*

$$\Gamma_k^f\, t\, t_1 \ \dots\ t_n \to_{\rho\delta\gamma}^{v\mathsf{stk}^*} \begin{pmatrix} f\, t_1 \dots t_{k-1} v_1^{\rho\delta\gamma} \dots t_n \\ \wr \cdots \wr \\ f\, t_1 \dots t_{k-1} v_p^{\rho\delta\gamma} \dots t_n \end{pmatrix}$$

si $t\, t_k \to_{\rho\delta\gamma}^{v\mathsf{stk}^*} v_1^{\rho\delta\gamma} \wr \cdots \wr v_p^{\rho\delta\gamma}$ et $\Gamma_k^f\, t\, t_1 \ \dots\ t_n \to_{\rho\delta\gamma}^{v\mathsf{stk}^*} \mathsf{stk}$ si $t\, t_k \to_{\rho\delta\gamma}^{v\mathsf{stk}^*} \mathsf{stk}$.

Remarquons que pour tous motifs p_1 et p_2, le terme $\mathsf{first}\ (p_1 \rightharpoonup \mathsf{stk})\ (p_2 \rightharpoonup t)\, u$ se réduira toujours vers le même terme que $(p_2 \rightharpoonup t)\, u$. En effet l'opérateur first ne vérifie pas si $p_1 \overset{?}{\preccurlyeq} u$ mais si $(p_1 \rightharpoonup \mathsf{stk})\, u$ se réduit en stk, ce qui est toujours vrai. Par conséquent le terme

$$\nu \rightharpoonup \overline{x}^n \rightharpoonup (\mathsf{first} \begin{pmatrix} \mathsf{stk} \rightharpoonup \mathsf{stk} \\ y \rightharpoonup f x_1 \dots x_{k-1} y \dots x_n \end{pmatrix} (\nu\, x_k))$$

n'a pas le même comportement que Γ_k^f. C'est l'usage de la constante nop dans ce dernier rho-terme qui nous permet de préciser qu'une réduction vers stk est équivalente, dans ce cas, à un échec du filtrage. Nous définissons alors le rho-terme Γ^f comme suit.

$$\Gamma^f = \Gamma_1^f \wr \cdots \wr \Gamma_n^f$$

Il représente intuitivement l'application d'un rho-terme à chaque sous-terme t_k d'une autre rho-terme $t = f\, t_1 \dots t_n$. La structure qui regroupe les différents résultats obtenus lorsqu'un terme u est appliqué à tous les sous-termes immédiats de t est obtenue en réduisant le rho-terme $\Gamma^f\, t\, u$:

$$\Gamma^f\, t\, u \to_{\rho\delta\gamma}^{v\mathsf{stk}^*} \Gamma_1^f\, t\, u \wr \cdots \wr \Gamma_n^f\, t\, u\ .$$

Réduction en un pas. Le rho-terme qui représente la réduction en un pas par le système de réécriture $\mathcal{R}$ est noté $\Omega^1_{\mathcal{R}}$ et défini par $\Omega^1_{\mathcal{R}} = \omega^1_{\mathcal{R}}\,\omega^1_{\mathcal{R}}$ où

$$\omega^1_{\mathcal{R}} = \pi \rightharpoonup \begin{pmatrix} \ldots \wr\, l_i \rightharpoonup r_i \wr \ldots \wr & \text{pour tout } l_i \to r_i \in \mathcal{R} \\ \ldots \wr\, f\, x_1 \ldots x_n \rightharpoonup \Gamma^f\,(\pi\,\pi)\, x_1 \ldots x_n \wr \ldots & \\ & \text{pour tout } f \text{ d'arité } n \geqslant 1 \end{pmatrix}.$$

Cette définition de $\omega^1_{\mathcal{R}}$ contient deux parties : la première encode la réécriture en tête par $\mathcal{R}$; la seconde utilise les termes Γ^f_k pour exprimer le fait que la réécriture peut aussi avoir lieu en profondeur. Le terme $\Omega^1_{\mathcal{R}}$ est un point fixe qui itère l'utilisation des Γ^k_f dans le but de descendre aussi profondément que possible dans le terme réécrit.

Propriété 1.3.4. *Soit t un terme algébrique. Si $\{u/t \to_{\mathcal{R}} u\} = \emptyset$ alors $\Omega^1_{\mathcal{R}}\, t \to^{v\mathsf{stk}^*}_{\rho\delta\gamma} \mathsf{stk}$. Sinon, $\Omega^1_{\mathcal{R}}\, t \to^{v\mathsf{stk}^*}_{\rho\delta\gamma} t_1 \wr \cdots \wr t_p$ où $\{u/t \to_{\mathcal{R}} u\} = \{t_1, \ldots t_p\}$. De plus, par confluence de $\to^{v\mathsf{stk}^*}_{\rho\delta\gamma}$ sur les rho-termes linéaires (propriété 1.3.2), si $\mathcal{R}$ est linéaire à gauche, les rho-termes $t_1 \wr \cdots \wr t_p$ et stk sont les uniques formes normales de $\Omega^1_{\mathcal{R}}\, M$.*

Réduction en Forme Normale. Nous définissons à présent le rho-terme qui représente la réduction en forme normale par le système de réécriture $\mathcal{R}$. Plus précisément, nous voulons définir un rho-terme $\Omega_{\mathcal{R}}$ tel que si nous l'appliquons à une deuxième terme t, $\Omega_{\mathcal{R}}$ applique tout d'abord $\Omega^1_{\mathcal{R}}$ à t puis si cela renvoie stk (c'est le cas quand t est une forme normale pour $\mathcal{R}$), alors $\Omega_{\mathcal{R}} t$ se réduit en t et sinon, il continue à applique $\Omega_{\mathcal{R}}$ au résultat de $\Omega^1_{\mathcal{R}} t$ Le rho-terme $\Omega_{\mathcal{R}}$ est définit par $\Omega_{\mathcal{R}} = \omega_{\mathcal{R}}\,\omega_{\mathcal{R}}$ où $\omega_{\mathcal{R}} = s \rightharpoonup x \rightharpoonup \mathsf{first}\,(\mathsf{stk} \rightharpoonup x)\,(z \rightharpoonup (s\,s)\,z)\,(\Omega^1_{\mathcal{R}}\, x)$. La relation $\Subset$ représente l'observabilité d'un *résultat* dans un rho-terme qui, à travers la structure $\_ \wr \_$, est un ensemble de termes.

Définition 1.3.6. *La relation $\in$ est définie inductivement comme la plus petite relation réflexive sur les rho-termes telle que pour tous termes t, u_1 et u_2 tels que $t \Subset u_1$, alors $t \Subset u_1 \wr u_2$ et $t \Subset u_2 \wr u_1$.*

Cette relation nous permet d'établir la correction et la complétude de l'encodage.

Propriété 1.3.5. *Si t et t' sont deux rho-termes algébriques tels que t' est une forme normale de t par $\mathcal{R}$, alors il existe u tel que $\Omega_{\mathcal{R}} t \to^{v\mathsf{stk}^*}_{\rho\delta\gamma} u$ et $t' \Subset u$. En outre, si $\mathcal{R}$ termine sur t, alors $\Omega_{\mathcal{R}}\, t \to^{v\mathsf{stk}^*}_{\rho\delta\gamma} t_1 \wr \cdots \wr t_p$ où l'ensemble de formes normales de t est $\{t_1, \ldots t_p\}$. Enfin puisque $\to^{v\mathsf{stk}^*}_{\rho\delta\gamma}$ est confluent sur les rho-termes linéaires, si $\mathcal{R}$ est linéaire, $t_1 \wr \cdots \wr t_p$ est l'unique forme normale de $\Omega_{\mathcal{R}}\, t$.*

Remarquons que si $\mathcal{R}$ ne termine pas sur t, alors la réduction du rho-terme $\Omega_{\mathcal{R}}\, t$ ne termine pas non plus car elle continuera sans cesse de générer de nouvelles formes normales. Ce qu'explique la propriété 1.3.5, c'est que toute forme normale sera éventuellement générée par la réduction (non terminante) de $\Omega_{\mathcal{R}}\, t$. En fait cette réduction est un parcours en largeur de l'arbre (infini) de réduction de t par $\mathcal{R}$.

Chapitre 2

Systèmes de logique formelle

Le besoin d'une formalisation des fondements des mathématiques et de la logique s'est d'abord traduit par la recherche d'un ensemble consistant d'axiomes assez puissants pour impliquer le raisonnement mathématique moderne. Cette recherche « culmina » lorsque Gödel (1931) présenta ses théorèmes d'incomplétude. Ces résultats modifièrent complètement la quête de fondements. Les ensembles d'axiomes qui étaient alors utilisés pour formaliser le raisonnement mathématique sont à présent appelés des systèmes déductifs à la Hilbert. Parmi ces systèmes on peut trouver par exemple le Begriffschrift de Frege (1879) ou encore les Principia Mathematica de Whitehead et Russell (1925). Ces systèmes sont construits autour d'un ensemble d'axiomes comme par exemple

$$
\begin{array}{ll}
A \Rightarrow B \Rightarrow (A \wedge B) & \forall x, 0 + x = 0 \\
B \Rightarrow (A \vee B) & \forall x\ y\ z, (x + y) + z = x + (y + z) \\
A \Rightarrow (A \vee B) & \dots \\
\dots &
\end{array}
$$

À partir d'un tel ensemble d'axiomes S, l'ensemble des formules prouvables est défini inductivement comme le plus petit ensemble $\bar{S}$ qui contient S et tel que

$$
\begin{array}{lllll}
\text{si} & \varphi \Rightarrow \psi \in \bar{S} \text{ et } \varphi \in \bar{S} & \text{alors} & \psi \in \bar{S} & \text{(Modus Ponens)}; \\
& \text{si } \varphi \in \bar{S} & \text{alors} & \forall x.\varphi \in \bar{S} & \text{(Généralisation universelle).}
\end{array}
$$

Ces règles sont habituellement écrites sous la forme de règles d'inférence.

$$
\frac{\phi \Rightarrow \psi \qquad \phi}{\psi}\ \text{(Modus Ponens)} \qquad \frac{\varphi}{\forall x.\varphi}\ \text{(Généralisation universelle)}
$$

La grand simplicité d'une telle définition est le seul avantage des systèmes à la Hilbert. En effet dès que ces systèmes sont confrontés à la réalité des mathématiques ou encore quand il s'agit d'étudier les propriétés de ces systèmes de preuve, le manque de structure des preuves qu'ils construisent devient un gros problème.

La préoccupation principale de Gentzen quand il introduisit la déduction naturelle et le calcul des séquents était de prouver la cohérence de l'arithmétique. Il proposa de découper la définition des systèmes de preuves en une partie contenant les axiomes logiques et une autre contenant les axiomes non-logiques. Cette approche se révéla fructueuse puisqu'il en résulte une méthode générique pour prouver la cohérence d'un système de déduction : l'Hauptsatz ou élimination des coupures. Dans ce chapitre, notre intérêt principal se porte sur la logique propositionnelle et le calcul des prédicats (aussi connu sous le nom de logique du premier ordre). Nous présentons des variantes des systèmes d'origine de Gentzen (1935) : la déduction naturelle et le calcul des séquents.

2.1 Formules et inférences

Soit $T_\alpha(X)$ une algèbre de termes que nous appellerons l'ensemble des termes du premier ordre. Le symbole t représentera des termes du premier ordre. Soit $\mathcal{A}$ une signature des prédicats, c'est-à-dire un ensemble de symboles, que nous appellerons des prédicats, tel que à chaque prédicat P et associé un entier naturel appelé l'arité de P et dénoté n_P. Alors l'ensemble des formules du premier ordre sur $T_\alpha(X)$ et $\mathcal{A}$ est défini comme le plus petit ensemble $\mathcal{F}_{\mathcal{A}T_\alpha(X)}$ tel que **1)** si P est un prédicat d'arité n_P et $(t_i)_{1\leqslant i\leqslant n_P}$ est une suite finie de termes du premier ordre de longueur n_P, alors $P(t_1,\ldots,t_{n_P}) \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$; **2)** $\bot \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$ et $\top \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$; **3)** si $\varphi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$, alors $\neg\varphi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$; **4)** si $\varphi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$ et $\psi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$, alors $\varphi\wedge\psi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$, $\varphi\vee\psi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$ et $\varphi\Rightarrow\psi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$; **5)** si $\varphi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$, $x \in X$, alors $\exists x.\varphi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$ et $\forall x.\varphi \in \mathcal{F}_{\mathcal{A}T_\alpha(X)}$. Les symboles φ et ψ dénoteront des formules. On peut définir alternativement l'ensemble des formules comme étant le langage hors-contexte associé à la grammaire en forme de Backus-Naur suivante.

$$\varphi,\psi \quad ::= \quad P(t_1,\ldots,t_{n_P}) \mid \bot \mid \top \mid \neg\varphi \mid \varphi\wedge\psi \mid \varphi\vee\psi \mid \varphi\Rightarrow\psi \mid \exists x.\varphi \mid \forall x.\varphi$$

les formules de la forme $P(t_1,\ldots,t_n)$ seront appelées des formules atomiques ou (par abus de langage) des prédicats atomiques. Nous appellerons les symboles $\bot$, $\top$, $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\exists$ et $\forall$ des connecteurs. Ils représentent respectivement le faux, le vrai, la négation, la conjonction, la disjonction, l'implication, la quantification existentielle et la quantification universelle. Les connecteurs $\forall$ est $\exists$ sont appelés des quantificateurs. Une formule propositionnelle est une formule qui ne contient pas de quantificateurs. La notion de position (définition 1.1.14) est étendue au formules du premier ordre de la façon suivante.

Définition 2.1.1 (Position dans une formule du premier ordre). *Une position est une suite finie d'entiers strictement positifs. Si φ est une formule, ν est une position, $\varphi_{|\nu}$*

est défini par

$$\begin{array}{ll}
\varphi_{|\nu} = \varphi & \text{si } \nu = \varepsilon \text{ ;} \\
\varphi_{|\nu} = \psi_{i|\nu'} & \text{si } \nu = i, \nu' \text{ et } \varphi = \psi_1 \Rightarrow \psi_2 \text{ ;} \\
\varphi_{|\nu} = \psi_{i|\nu'} & \text{si } \nu = i, \nu' \text{ et } \varphi = \psi_1 \wedge \psi_2 \text{ ;} \\
\varphi_{|\nu} = \psi_{i|\nu'} & \text{si } \nu = i, \nu' \text{ et } \varphi = \psi_1 \vee \psi_2 \text{ ;} \\
\varphi_{|\nu} = \psi_{|\nu'} & \text{si } \nu = 1, \nu' \text{ et } \varphi = \neg\psi \text{ ;} \\
\varphi_{|\nu} = \psi_{|\nu'} & \text{si } \nu = 1, \nu' \text{ et } \varphi = \forall x.\psi \text{ ;} \\
\varphi_{|\nu} = \psi_{|\nu'} & \text{si } \nu = 1, \nu' \text{ et } \varphi = \exists x.\psi \text{ ;} \\
\varphi_{|\nu} = t_{i|\nu'} & \text{si } \nu = i, \nu' \text{ et } \varphi = P(t_1, \ldots, t_n) \text{ .}
\end{array}$$

Pour une formule φ, la formule $\varphi_{|\nu}$ n'est pas définie pour toute position ν mais pour un ensemble fini de positions ν que nous appellerons les positions de φ. Nous dirons qu'une formule φ est une sous-formule d'une formule ψ s'il existe une position ν de φ telle que $\varphi_{|\nu} = \psi$. On peut alors définir les occurrences positives et négatives d'une formule comme suit.

Définition 2.1.2 (Occurrences positives et négatives). *Une formule φ admet une occurrence d'une formule ψ à la position ν quand $\varphi_{|\nu} = \psi$. La polarité* $\mathrm{pol}(\nu)_\varphi \in \mathbb{Z}/2\mathbb{Z}$ *de la position ν dans la formule φ est définie par*

$$\begin{array}{ll}
\mathrm{pol}(\nu)_\varphi = 1 & \text{si } \nu = \epsilon \text{ ;} \\
\mathrm{pol}(\nu)_\varphi = \mathrm{pol}(\nu')_{\psi_i} & \text{si } \nu = i, \nu' \text{ et } \varphi = \psi_1 \wedge \psi_2 \text{ ;} \\
\mathrm{pol}(\nu)_\varphi = \mathrm{pol}(\nu')_{\psi_i} & \text{si } \nu = i, \nu' \text{ et } \varphi = \psi_1 \vee \psi_2 \text{ ;} \\
\mathrm{pol}(\nu)_\varphi = \mathrm{pol}(\nu')_{\psi_i} + i & \text{si } \nu = i, \nu' \text{ et } \varphi = \psi_1 \Rightarrow \psi_2 \text{ ;} \\
\mathrm{pol}(\nu)_\varphi = \mathrm{pol}(\nu')_{\psi} + 1 & \text{si } \nu = 1, \nu' \text{ et } \varphi = \neg\psi \text{ ;} \\
\mathrm{pol}(\nu)_\varphi = \mathrm{pol}(\nu')_{\psi} & \text{si } \nu = 1, \nu' \text{ et } \varphi = \exists x.\psi \text{ ;} \\
\mathrm{pol}(\nu)_\varphi = \mathrm{pol}(\nu')_{\psi} & \text{si } \nu = 1, \nu' \text{ et } \varphi = \forall x.\psi \text{ .}
\end{array}$$

Lorsque $\mathrm{pol}(\nu)_\varphi = 0$*, nous parlerons d'occurrence négative (de $\varphi_{|\nu}$) dans φ. Nous parlerons d'occurrence positive sinon.*

La notion de contexte de la définition 1.1.12 peut aussi être transposée aux formules : On suppose donné un symbole $\Box$ qui n'est ni un symbole de fonction ou de variable de l'algèbre de termes du premier ordre $T_\alpha(X)$ ni un symbole de prédicat de $\mathcal{A}$. Un contexte de formule est une formule de $\mathcal{F}_{\mathcal{A}\cup\{\Box\}T_\alpha(X)}$ où l'arité de $\Box$ est 0. L'arité d'un contexte de formule est le nombre de symboles $\Box$ qu'il contient. Nous nous intéressons particulièrement aux contextes de formule unaires (c'est-à-dire d'arité 1) : ils permettent de définir la notion de relation close par contexte et de congruence, exactement comme dans la section 1.1. Une relation binaire $\rhd$ sur les formules du premier ordre est close par contexte si pour toutes formules φ et ψ telles que $\varphi \rhd \psi$ et pour tout contexte de formule C **unaire**, alors $C[\varphi] \rhd C[\psi]$. La clôture par contexte de $\rhd$ est définie comme la plus petite relation close par contexte contenant $\rhd$. Une congruence est une relation d'équivalence close par contexte.

Nous disposons déjà d'une notion de substitution sur les termes du premier ordre, puisque ceux-ci sont par définition membres d'une algèbre de termes $T_\alpha(X)$. L'application d'une substitution (du premier ordre) σ à une formule du premier ordre φ, dénotée $\varphi\sigma$, est alors définie de la façon suivante.

$$\begin{array}{ll}
\varphi\sigma = P(t_1\sigma, \ldots, t_n\sigma) & \text{si } \varphi = P(t_1, \ldots, t_n) \text{ ;} \\
\varphi\sigma = \top & \text{si } \varphi = \top \text{ ;} \\
\varphi\sigma = \bot & \text{si } \varphi = \bot \text{ ;} \\
\varphi\sigma = \neg(\psi\sigma) & \text{si } \varphi = \neg\psi \text{ ;} \\
\varphi\sigma = \psi\sigma \wedge \psi'\sigma & \text{si } \varphi = \psi \wedge \psi' \text{ ;} \\
\varphi\sigma = \psi\sigma \vee \psi'\sigma & \text{si } \varphi = \psi \vee \psi' \text{ ;} \\
\varphi\sigma = \psi\sigma \Rightarrow \psi'\sigma & \text{si } \varphi = \psi \Rightarrow \psi' \text{ ;} \\
\varphi\sigma = \forall x.(\psi\sigma) & \text{si } \varphi = \forall x.\psi \text{ ;} \\
\varphi\sigma = \exists x.(\psi\sigma) & \text{si } \varphi = \exists x.\psi \text{ .}
\end{array}$$

De manière équivalente et succinte, on peut résumer comme suit. L'extension d'une substitution aux formules atomiques est définie par

$$(P(t_1, \ldots, t_n))\sigma = P(t_1\sigma, \ldots, t_n\sigma) \text{ .}$$

L'extension d'une substitution aux formules est définie comme la clôture par contexte de la fonction de substitution sur les formules atomiques (en tant que relation binaire). Il reste alors à montrer que cette clôture définit bien une fonction (tâche que je délègue au lecteur qui s'ennuie).

Tout comme le symbole λ pour le lambda-calcul, les symboles $\exists$ et $\forall$ sont des lieurs. Tout comme la fonction carré peut s'écrire $x \mapsto x^2$ ou $y \mapsto y^2$, la formule « tout objet vérifie la propriété A » peut s'écrire indifféremment $\forall x.A(x)$ ou $\forall y.A(y)$ et la formule « il existe un objet vérifiant la propriété A » peut s'écrire indifféremment $\exists x.A(x)$ ou $\exists y.A(y)$. On peut alors définir les occurrences libres, liées et l'alpha-conversion sur les formules du premier ordre de la manière suivante : L'occurrence d'une variable du premier ordre x à une position ν dans une formule φ est liée s'il existe un préfixe ν' de ν tel que $\varphi_{|\nu'}$ est de la forme $\exists x.\psi$ ou bien de la forme $\forall x.\psi$. Sinon, l'occurrence est libre. L'ensemble des variables liées dans une formule φ, noté $\mathcal{BV}(\varphi)$, et l'ensemble des variables libres dans φ, noté $\mathcal{FV}(\varphi)$, sont définis par

$$\begin{array}{llll}
\mathcal{BV}(\varphi) = \emptyset & \text{et} \quad \mathcal{FV}(\varphi) = \bigcup_{1 \leqslant i \leqslant n} \mathcal{V}(t_i) & \text{si } \varphi = P(t_1, \ldots, t_n) \text{ ;} \\
\mathcal{BV}(\varphi) = \emptyset & \text{et} \quad \mathcal{FV}(\varphi) = \emptyset & \text{si } \varphi \in \{\bot, \top\} \text{ ;} \\
\mathcal{BV}(\varphi) = \mathcal{BV}(\psi) & \text{et} \quad \mathcal{FV}(\varphi) = \mathcal{FV}(\psi) & \text{si } \varphi = \neg\psi \text{ ;} \\
\mathcal{BV}(\varphi) = \mathcal{BV}(\psi) \cup \{x\} & \text{et} \quad \mathcal{FV}(\varphi) = \mathcal{FV}(\psi)/\{x\} & \text{si } \varphi \in \{\exists x.\psi, \forall x.\psi\} \text{ ;}
\end{array}$$

$$\left.\begin{array}{l}
\mathcal{BV}(\varphi) = \mathcal{BV}(\psi_1) \cup \mathcal{BV}(\psi_2) \\
\mathcal{FV}(\varphi) = \mathcal{FV}(\psi_1) \cup \mathcal{FV}(\psi_2)
\end{array}\right\} \quad \text{si } \varphi \in \{\psi_1 \Rightarrow \psi_2, \psi_1 \wedge \psi_2, \psi_1 \vee \psi_2\} \text{ ;}$$

L'alpha-conversion sur les formules (toujours notée $\equiv_\alpha$), est la plus petite congruence sur les formules telle que

$$\forall x.\varphi \;\equiv_\alpha\; \forall y.(\varphi[y/x]) \quad \text{et} \quad \exists x.\varphi \;\equiv_\alpha\; \exists y.(\varphi[y/x])$$

dès que $y \notin \mathcal{FV}(\varphi) \cup \mathcal{BV}(\varphi)$ et $x \notin \mathcal{BV}(\varphi)$. Tout comme dans le cadre du lambda-calcul, il peut y avoir capture lorsqu'on applique une substitution. $\forall x.P(z) \equiv_\alpha \forall y.P(z)$ mais

$$(\forall x.P(z))[x/z] = \forall x.P(x) \not\equiv_\alpha \forall y.P(x) = (\forall y.P(z))[x/z] \,.$$

$\forall x.P(x) \equiv_\alpha \forall y.P(y)$ mais

$$(\forall x.P(x))[z/x] = \forall x.P(z) \not\equiv_\alpha \forall y.P(y) = (\forall y.P(y))[z/x] \,.$$

Lorsque l'on souhaite appliquer une substitution σ à une formule φ et que σ n'évite pas les captures dans φ, c'est-à-dire que $\mathcal{BV}(\varphi) \cap (\mathrm{dom}(\sigma) \cup \mathcal{FV}(\mathrm{codom}(\sigma))) \neq \emptyset$, alors il est toujours possible de trouver une formule ψ alpha-équivalente à φ telle que σ évite les captures dans ψ. Nous travaillons donc modulo alpha-équivalence et nous supposerons toujours que les substitutions évitent les captures dans les formules sur lesquelles on les applique. La notion de sous-instance est définie comme suit.

Définition 2.1.3 (Sous-instance). *Nous noterons $\preccurlyeq$ la plus petite relation réflexive et transitive contenant la relation de sous-formule et telle que pour toute formule φ et pour tout terme t, $\varphi[t/x] \preccurlyeq \forall x.\varphi$ et $\varphi[t/x] \preccurlyeq \exists x.\varphi$. Nous dirons que φ est une sous-instance de ψ quand $\varphi \preccurlyeq \psi$.*

La différence majeure entre le mécanisme de liaison du lambda-calcul et le mécanisme de liaison des quantificateurs, c'est que les lieurs $\forall$ et $\exists$ opèrent sur la catégorie syntaxique des termes du premier ordre, objets qui ne contiennent pas de lieurs : lorsque nous écrirons $\forall x.A$, par exemple, x est une variable du premier-ordre qui représente « n'importe quel terme du premier ordre ». On ne pourra donc substituer x dans A que par des termes du permier ordre. Au contraire le lieur du lambda-calcul opère sur la catégorie syntaxique des lambda-termes qui contiennent, eux, des lieurs. L'exemple 1.2.2 (Oméga) ne pourra donc pas se traduire en utilisant les lieurs $\forall$ ou $\exists$.

Un contexte est un multi-ensemble de formules. Les symboles Γ et Δ représenteront des contextes. Un séquent est une paire de contextes $\Gamma \vdash \Delta$. Si Γ est le multi-ensemble $A_1, \ldots, A_n$ et Δ est le multi-ensemble $B_1, \ldots, B_m$, alors $\Gamma \vdash \Delta$ représente en fait la formule $(A_1 \wedge A_2 \wedge \cdots \wedge A_n) \Rightarrow (B_1 \vee B_2 \vee \cdots \vee B_m)$ libérée de la bureaucratie des parenthèses par l'associativité et la commutativité naturelle de la concaténation des multi-ensembles qui représentent elles-mêmes l'associativité et la commutativité des connecteurs $\vee$ et $\wedge$. Si ε est le multi-ensemble vide, alors $\varepsilon \vdash \Delta$ est aussi noté $\vdash \Delta$ et $\Gamma \vdash \varepsilon$ est aussi noté $\Gamma \vdash$. Les séquents $A_1, \ldots, A_n \vdash$ et $\vdash B_1, \ldots, B_m$ représentent en fait respectivement les formules $(A_1 \wedge \cdots \wedge A_n) \Rightarrow \bot$ et $\top \Rightarrow (B_1 \vee \cdots \vee B_m)$. Le symbole S représentera des séquents.

Définition 2.1.4 (Inférence). *Si n est un entier naturel non nul, une inférence d'arité n est un n-uplet contenant n séquents. Une inférence $(S_1, S_2, \ldots, S_n)$ avec $n > 1$ et une inférence (S_1) sont respectivement notées*

$$\frac{S_1 \qquad \ldots \qquad S_{n-1}}{S_n} \qquad \textit{et} \qquad \frac{}{S_1} \,.$$

Dans le cas $n > 1$, nous dirons que S_1,... S_{n-1} sont les prémisses et que S_n est la conclusion de l'inférence. Dans le cas $n = 1$, nous dirons que S_1 est la conclusion et que l'inférence n'admet pas de prémisse.

Définition 2.1.5 (Schéma d'inférence). *Un schéma ou règle d'inférence est un ensemble d'inférences généralement défini par un certain motif. La règle de généralisation universelle est par exemple définie par le motif*

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x.\varphi}\ x \notin \mathcal{FV}(\Gamma) \quad .$$

Plus formellement, cette règle dénote en fait l'ensemble d'inférences

$$\{(\Gamma \vdash \varphi, \Gamma \vdash \forall x.\varphi)/\Gamma \text{ contexte, } \varphi \text{ formule et } x \text{ variable telle que } x \notin \mathcal{FV}(\Gamma)\} \quad .$$

Si un inférence I est contenue dans un schéma C, nous dirons que I est une instance de C.

Un système de déduction est un ensemble d'inférences. Une dérivation ouverte dans un système $\mathcal{S}$ est un arbre fini dont chaque noeud est associé à un séquent et tel que si a est un noeud ayant $n > 0$ fils $(a_i)_{1 \leqslant i \leqslant n}$, a est associé au séquent S, et pour tout $1 \leqslant i \leqslant n$, a_i est associé au séquent S_i, alors $(S_1, \ldots, S_n, S)$ est une inférence de $\mathcal{S}$. Une feuille dans une dérivation ouverte est un noeud sans fils. Une feuille fermée dans une dérivation ouverte est une feuille associée à un séquent S tel que $\overline{S}$ appartient à $\mathcal{S}$. Au contraire, une feuille ouverte est une feuille qui n'est pas fermée. Une dérivation fermée dans un système de déduction $\mathcal{S}$ est une dérivation ouverte dont toutes les feuilles sont fermées. On parlera alors juste de dérivation. Les dérivations ouvertes ou fermées sont représentées habituellement de façon graphique similairement aux inférences. Si la racine d'une dérivation est associé à un séquent S, alors nous dirons que c'est une dérivation de S. Nous dirons qu'un séquent S est dérivable dans un système de déduction s'il existe dans ce système une dérivation de ce séquent.

Exemple 2.1.1 (Loi de Pierce). *Soit $\mathcal{MP}$ le système de déduction contenant les inférences*

$$\textsc{Modus Ponens}\ \frac{\vdash \varphi \Rightarrow \psi \qquad \vdash \varphi}{\vdash \psi}$$

$$\textsc{Ax1}\ \frac{}{\vdash \varphi \Rightarrow (\psi \Rightarrow \varphi)}$$

$$\textsc{Ax2}\ \frac{}{\vdash \varphi \Rightarrow (\psi \Rightarrow \varphi') \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow (\varphi \Rightarrow \varphi'))}$$

$$\textsc{Ax3}\ \frac{}{\vdash ((\varphi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \varphi}$$

pour toutes formules φ, ψ et φ'. Alors si φ et ψ sont deux formules, on peut prouver dans $\mathcal{MP}$ le séquent $\vdash ((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$. La preuve est détaillée par les figures 2.1 et 2.2.

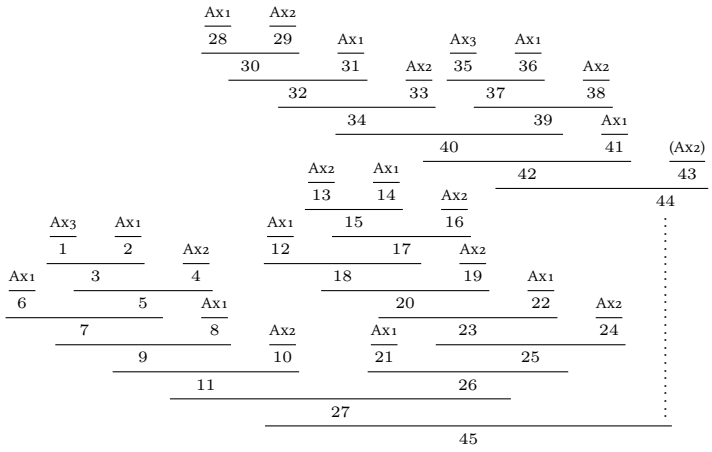


FIG. 2.1 – Preuve de la loi de Pierce dans le système $\mathcal{MP}$

Nous utiliserons aussi le mot *preuve* à la place du mot *dérivation*. Une inférence est donc un mécanisme qui permet de construire une preuve de sa conclusion à partir de preuves de ses premisses. Les inférences d'un système sont des mécanismes atomiques qui sont donnés par le système. On peut alors les composer pour obtenir des dérivations ouvertes ou fermées. Les séquents dérivables (nous dirons aussi prouvables) dans un système de déduction sont les séquents S tels qu'il existe une dérivation de S dans le système. Nous dirons qu'une inférence est admissible dans un système lorsqu'elle correspond à un mécanisme qui existe dans le système.

Définition 2.1.6 (Inférence et schéma admissible). *Soit un système déductif A et une inférence quelconque I. Nous dirons que I est admissible dans A lorsqu'il existe une dérivation ouverte dans A telle que toutes ses feuilles ouvertes sont associées à une prémisse de I et sa racine est associée à la conclusion de I. Un schéma d'inférence est admissible dans un système déductif quand toutes ses instances sont admissibles dans ce système.*

Le cas trivial apparaît lorsque l'inférence I est elle-même une inférence de A, car elle définit elle-même une dérivation ouverte de A qui démontre sa propre admissibilité dans A. Nous serons donc bien entendu intéressé par l'admissibilité d'inférences qui n'appartiennent pas au système. Par exemple l'inférence $(\vdash \varphi, \vdash \psi \Rightarrow \varphi)$ est admissible dans le système de l'exemple 2.1.1. Nous dirons qu'une inférence est inversible lorsque le mécanisme qu'elle représente l'est, c'est-à-dire

Définition 2.1.7 (Inférence et schéma inversible). *Une inférence I est inversible dans un système A lorsque s'il existe une dérivation de la conclusion de I dans A, alors il existe une dérivation des prémisses de A. Un schéma d'inférence est inversible dans un système déductif lorsque toutes ses instances le sont.*

Enfin nous utiliserons la définition suivante de cohérence.

1 $((\psi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \psi$
2 $(((\psi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \psi) \Rightarrow (\bot \Rightarrow (((\psi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \psi))$
3 $\vdash \bot \Rightarrow (((\psi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \psi)$
4 $(\bot \Rightarrow (((\psi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \psi)) \Rightarrow ((\bot \Rightarrow ((\psi \Rightarrow \bot) \Rightarrow \bot)) \Rightarrow (\bot \Rightarrow \psi))$
5 $(\bot \Rightarrow ((\psi \Rightarrow \bot) \Rightarrow \bot)) \Rightarrow (\bot \Rightarrow \psi)$
6 $\bot \Rightarrow ((\psi \Rightarrow \bot) \Rightarrow \bot)$
7 $\bot \Rightarrow \psi$
8 $(\bot \Rightarrow \psi) \Rightarrow (\varphi \Rightarrow (\bot \Rightarrow \psi))$
9 $\varphi \Rightarrow (\bot \Rightarrow \psi)$
10 $(\varphi \Rightarrow (\bot \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))$
11 $(\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)$
12 $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi))$
13 $((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))$
14 $(((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
$\Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))))$
15 $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
16 $(((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))))$
$\Rightarrow ((((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi)))$
$\Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))))$
17 $(((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi)))$
$\Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
18 $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))$
19 $(((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
$\Rightarrow ((((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
20 $(((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))$
21 $((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)))$
22 $((((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
$\Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))$
$\Rightarrow ((((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))))$
23 $((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))$
$\Rightarrow ((((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
24 $(((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))$
$\Rightarrow ((((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))))$
$\Rightarrow ((((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))))$
$\Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))))$
25 $(((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi))))$
$\Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)))$
26 $((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \psi)) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi))$
27 $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)$
28 $(\varphi \Rightarrow \bot) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot)) \Rightarrow (\varphi \Rightarrow \bot))$
29 $((\varphi \Rightarrow \bot) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot)) \Rightarrow (\varphi \Rightarrow \bot)))$
$\Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot))) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot)))$
30 $((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot))) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot))$
31 $(\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot))$
32 $(\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot)$
33 $((\varphi \Rightarrow \bot) \Rightarrow (\varphi \Rightarrow \bot)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \bot))$
34 $((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \bot)$
35 $((\varphi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \varphi$
36 $(((\varphi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \varphi))$
37 $((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \varphi)$
38 $(((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \varphi))$
$\Rightarrow ((((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \bot)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow \varphi))$
39 $(((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow \varphi))$
$\Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \bot)) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow \varphi)))$
40 $((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow \varphi$
41 $(((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow \varphi))$
42 $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow \varphi)$
43 $(((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow (((\varphi \Rightarrow \bot) \Rightarrow \varphi) \Rightarrow \varphi))$
$\Rightarrow ((((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi))$
44 $(((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow ((\varphi \Rightarrow \bot) \Rightarrow \varphi)) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi)$
45 $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$

FIG. 2.2 – Formules de la preuve en figure 2.1

Définition 2.1.8 (Cohérence). *Un système déductif est consistant s'il existe un séquent n'y admettant pas de dérivation.*

Généralement, le séquent $\vdash \top$ admet toujours une dérivation. La cohérence d'un système déductif correspond alors au fait que le système propose une distinction non triviale entre séquents dérivables et séquents non-dérivables. En outre, une dérivation de $\vdash \bot$ permet généralement de construire une dérivation de n'importe quel autre séquent. Démontrer la cohérence revient donc à prouver qu'il n'existe pas de dérivation de $\vdash \bot$. Notons enfin que cette notion de cohérence subsume en fait celle que nous avons utilisée au chapitre 1. En effet la cohérence d'une relation binaire $\sim$ sur un ensemble $\mathcal{A}$ est définie comme l'existence de deux éléments a et b de $\mathcal{A}$ tels que $a \not\sim b$. Il s'agit donc d'un séquent $\vdash a \sim b$ non dérivable.

2.2 Déduction naturelle

Nous présentons ici le premier système de déduction introduit par Gentzen (1935) : la déduction naturelle intuitionniste. Le même système fut indépendamment proposé par Jaśkowski (1934).

Définition 2.2.1 (Déduction naturelle intuitionniste NJ). *La déduction naturelle intuitionniste, notée NJ, est définie comme le système contenant les inférences de la figure 2.3 pour tout contexte Γ, formules φ, ψ et φ', et variable x et tout terme du premier ordre t. Nous avons groupé les inférences de la déduction naturelle en schémas d'inférence, comme par exemple le schéma* Axiom $\dfrac{}{\Gamma, \varphi \vdash \varphi}$ *qui représente en fait l'ensemble d'inférences*

$$\left\{ \frac{}{\Gamma, \varphi \vdash \varphi} \;/\; \Gamma \text{ un contexte et } \varphi \text{ une formule} \right\} .$$

Mis à part le schéma Axiom*, chaque schéma manipule un connecteur spécifique et peut être une introduction ou une manipulation. Inversement, chaque connecteur admet zéro, une ou deux schéma(s) d'introduction et d'élimination. Il est très important de remarquer que pour être valide (c'est-à-dire incluse dans le système déductif), une instance de $\forall$*Intro *et $\exists$*Elim *doit satisfaire la condition que nous avons écrite à côté du schéma correspondant. Nous appellerons ces conditions des conditions de bord.*

Ce système est exactement le système originel de Gentzen (1935). On peut par exemple y dériver le séquent $\vdash A \Rightarrow \neg(\neg A)$:

$$\dfrac{\dfrac{\dfrac{\text{Axiom}\ \dfrac{}{A, \neg A \vdash \neg A}}{\neg\text{Elim}} \qquad \text{Axiom}\ \dfrac{}{A, \neg A \vdash A}}{\neg\text{Intro}\ \dfrac{A, \neg A \vdash \bot}{A \vdash \neg(\neg A)}}}{\Rightarrow\text{Intro}\ \dfrac{}{\vdash A \Rightarrow \neg(\neg A)}}$$

$$\textsc{Axiom}\ \frac{}{\Gamma, \varphi \vdash \varphi}$$

$$\Rightarrow\textsc{Intro}\ \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \Rightarrow \psi} \qquad \Rightarrow\textsc{Elim}\ \frac{\Gamma \vdash \varphi \Rightarrow \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

$$\wedge\textsc{Intro}\ \frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} \qquad \wedge\textsc{Elim}_1\ \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} \qquad \wedge\textsc{Elim}_2\ \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}$$

$$\vee\textsc{Intro}_1\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} \qquad \vee\textsc{Intro}_2\ \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi}$$

$$\vee\textsc{Elim}\ \frac{\Gamma \vdash \varphi \vee \psi \qquad \Gamma, \varphi \vdash \varphi' \qquad \Gamma, \psi \vdash \varphi'}{\Gamma \vdash \varphi'}$$

$$\forall\textsc{Intro}\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x.\varphi}\ x \notin \mathcal{FV}(\Gamma) \qquad \forall\textsc{Elim}\ \frac{\Gamma \vdash \forall x.\varphi}{\Gamma \vdash \varphi[t/x]}$$

$$\exists\textsc{Intro}\ \frac{\Gamma \vdash \varphi[t/x]}{\Gamma \vdash \exists x.\varphi} \qquad \exists\textsc{Elim}\ \frac{\Gamma \vdash \exists x.\varphi \qquad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}\ x \notin \mathcal{FV}(\Gamma, \psi)$$

$$\neg\textsc{Intro}\ \frac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi} \qquad \neg\textsc{Elim}\ \frac{\Gamma \vdash \neg\varphi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \bot}$$

$$\bot\textsc{Elim}\ \frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi} \qquad \top\textsc{Intro}\ \frac{}{\Gamma \vdash \top}$$

FIG. 2.3 – Inférences en Déduction Naturelle Intuitionniste NJ

Par contre, nous verrons en section 2.4 qu'il est impossible d'y dériver les séquents $\vdash \neg(\neg A) \Rightarrow A$ ou $\vdash A \vee \neg A$. Cela mène à la définition de la déduction naturelle classique.

Définition 2.2.2 (Déduction naturelle classique NK). *La déduction naturelle classique, notée NK, est définie comme le système contenant toutes les inférences de NJ auxquelles sont rajoutées les inférences*

$$\neg\neg\textsc{Elim}\ \frac{\Gamma \vdash \neg\neg\varphi}{\Gamma \vdash \varphi}$$

pour toute formule φ et tout contexte Γ.

Remarquons que tous les séquents dérivables dans les systèmes NJ et NK sont de la forme $\Gamma \vdash \varphi$ où Γ est un contexte et φ est une formule. Nous appellerons prémisse principale la première prémisse (si elle existe, c'est-à-dire si l'inférence n'est pas une instance du schéma Axiom ou du schéma $\neg\neg$Elim) d'une inférence en déduction naturelle. Par définition, une inférence en déduction naturelle est soit une instance de Axiom ou de $\neg\neg$Elim, soit une introduction, soit une élimination. Dans le cas d'une introduction, si $\Gamma \vdash \varphi$ est la conclusion de l'inférence, nous dirons que φ est la formule introduite par l'inférence et que le connecteur de tête de φ est le connecteur introduit par l'inférence. Dans le cas d'une élimination, si $\Gamma \vdash \varphi$ est la prémisse principale de l'inférence, nous dirons que φ est la formule éliminée par l'inférence et que le connecteur de tête de φ est le connecteur éliminé par l'inférence. Si Q est le connecteur introduit par une inférence, alors celle-ci est une instance de QIntro ($Q\textsc{Intro}_i$ avec $i \in \{1,2\}$ si $Q = \vee$). Si Q est le connecteur éliminé par une inférence, alors celle-ci est une instance de QElim ($Q\textsc{Elim}_i$ avec $i \in \{1,2\}$ si $Q = \wedge$).

Une première propriété des systèmes NJ et NK est l'admissibilité des règles de contraction et d'affaiblissement (*weakening*) suivantes.

$$\textsc{Contraction}\ \frac{\Gamma, \varphi, \varphi \vdash \psi}{\Gamma, \varphi \vdash \psi} \qquad\qquad \textsc{Weakening}\ \frac{\Gamma \vdash \psi}{\Gamma, \varphi \vdash \psi}$$

L'admissibilité de l'affaiblissement peut aussi être formulé comme suit.

Propriété 2.2.1 (Admissibilité de l'affaiblissement). *S'il existe une preuve π dans NJ ou NK d'un séquent $\Gamma \vdash \varphi$, alors il existe une preuve que nous noterons $\Gamma' \cdot \pi$ de $\Gamma, \Gamma' \vdash \varphi$ dans le même système contenant exactement les inférences de π auxquelles est rajouté le contexte Γ'.*

Démonstration. La preuve de cette propriété se fait par induction sur la preuve. La structure de celle-ci n'est d'ailleurs aucunement modifiée : on ajoute seulement Γ' aux séquents des inférences de la preuve. □

Une autre propriété intéressante est la stabilité par substitution (tant que l'on évite les captures).

Propriété 2.2.2. *Si π est une preuve dans NJ ou NK d'un séquent $\Gamma \vdash \varphi$ et σ une substitution (du premier ordre), alors l'application de σ à toutes les formules des inférences de π constitue une preuve de $\Gamma\sigma \vdash \varphi\sigma$ que nous noterons $\pi\sigma$.*

Nous allons à présent nous intéresser à un ensemble précis de preuves en déduction naturelle (intuitionniste ou classique) que nous appellerons *preuves sans coupures*.

Définition 2.2.3 (Coupures en déduction naturelle). *Une coupure en déduction naturelle est une élimination dont la prémisse principale est une introduction.*

Par exemple, les inférences suivantes sont des coupures.

$$\wedge\textsc{Elim}2\ \dfrac{\wedge\textsc{Intro}\ \dfrac{\begin{array}{cc}\vdots & \vdots\\ \vdash \varphi & \vdash \psi\end{array}}{\vdash \varphi \wedge \psi}}{\vdash \psi} \qquad\qquad \Rightarrow\textsc{Elim}\ \dfrac{\Rightarrow\textsc{Intro}\ \dfrac{\begin{array}{c}\vdots\\ \varphi \vdash \psi\end{array}}{\vdash \varphi \Rightarrow \psi} \qquad \begin{array}{c}\vdots\\ \vdash \varphi\end{array}}{\vdash \psi}$$

Bien ententu, dans une coupure, la formule et donc le connecteur introduit par l'étape d'introduction sont les mêmes que ceux éliminés par l'étape d'élimination. Une propriété cruciale de la déduction naturelle (classique ou intuitionniste) et que l'on peut écrire une procédure, généralement sous la forme d'un système de réécriture, qui normalise des preuves quelconques en preuves sans coupure du même séquent.

Propriété 2.2.3 (Normalisation en déduction naturelle). *Toute preuve dans NJ ou NK peut être transformée en une preuve sans coupure du même séquent dans le même système.*

La preuve de cette propriété est détaillée par exemple dans l'ouvrage de Prawitz (1965). Elle repose majoritairement sur le processus permettant d'éliminer toute coupure sur une formule dont le connecteur de tête est une implication : ce processus est illustré par la figure 2.4. Comme le montrent les figures 2.5 et 2.6, une coupure sur une formule dont le connecteur n'est pas une implication peut être soit directement supprimée (par exemple s'il s'agit d'une coupure sur une quantification universelle), soit traitée comme une coupure sur une implication (par exemple s'il s'agit d'une coupure sur une quantification existentielle) : on l'élimine alors comme en figure 2.4.

L'intérêt de la propriété 2.2.3 provient du fait que les preuves sans coupures en déduction naturelle vérifient la propriété suivante.

Propriété 2.2.4. *Une preuve sans coupure d'un séquent $\vdash \varphi$ dans NJ se termine toujours par une introduction.*

Démonstration. Par induction sur la taille de la preuve. Celle-ci ne peut se terminer par une instance d'Axiome (il n'y a rien dans la partie gauche du séquent). Elle ne

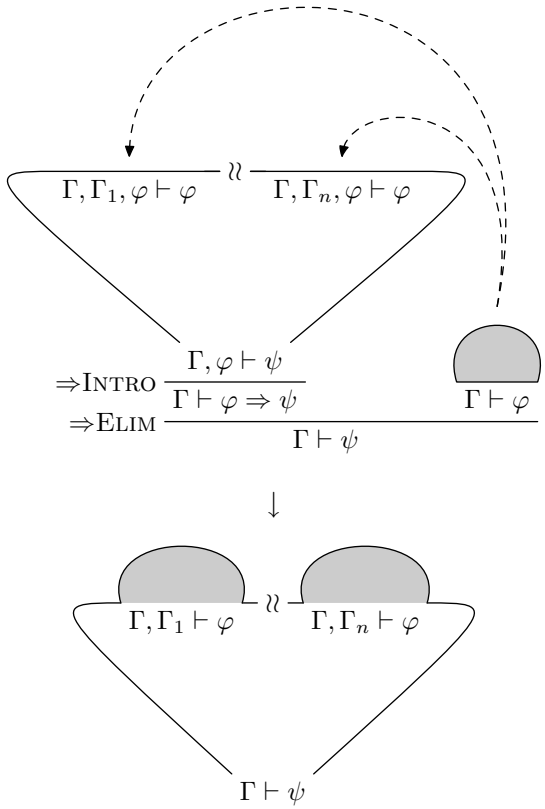


Fig. 2.4 – Normalisation en déduction naturelle, première partie

$$\wedge\textsc{Intro}\ \dfrac{\wedge\textsc{Elim1}\ \dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi} \quad \dfrac{(\pi_2)}{\Gamma \vdash \psi}}{\Gamma \vdash \varphi \wedge \psi}}{\Gamma \vdash \varphi} \quad \rightarrow \quad \dfrac{(\pi_1)}{\Gamma \vdash \varphi}$$

$$\wedge\textsc{Intro}\ \dfrac{\wedge\textsc{Elim2}\ \dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi} \quad \dfrac{(\pi_2)}{\Gamma \vdash \psi}}{\Gamma \vdash \varphi \wedge \psi}}{\Gamma \vdash \psi} \quad \rightarrow \quad \dfrac{(\pi_2)}{\Gamma \vdash \psi}$$

$$\forall\textsc{Intro}\ \dfrac{\forall\textsc{Elim}\ \dfrac{\dfrac{(\pi)}{\Gamma \vdash \varphi}\ x \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash \forall x.\varphi}}{\Gamma \vdash \varphi[t/x]} \quad \rightarrow \quad \dfrac{(\pi[t/x])}{\Gamma \vdash \varphi[t/x]}$$

$$\neg\textsc{Intro}\ \dfrac{\neg\textsc{Elim}\ \dfrac{\dfrac{(\pi_1)}{\Gamma, \varphi \vdash \bot}}{\Gamma \vdash \neg\varphi} \quad \dfrac{(\pi_2)}{\Gamma \vdash \varphi}}{\Gamma \vdash \bot} \quad \approx \quad \Rightarrow\textsc{Intro}\ \dfrac{\Rightarrow\textsc{Elim}\ \dfrac{\dfrac{(\pi_1)}{\Gamma, \varphi \vdash \bot} \quad \dfrac{(\pi_2)}{\Gamma \vdash \varphi}}{\Gamma \vdash \varphi \Rightarrow \bot}}{\Gamma \vdash \bot}$$

FIG. 2.5 – Normalisation en déduction naturelle, coupures sur $\wedge$, $\neg$ ou $\forall$

peut se terminer par une règle d'élimination car alors par hypothèse d'induction la preuve de sa prémisse principale étant sans coupure, elle termine par une introduction formant ainsi une coupure. Cette dernière règle ne peut donc être qu'une règle d'introduction. □

Cette propriété a une apparence très syntaxique. Combinée à la propriété 2.2.3, elle permet néanmoins de prouver la cohérence de la déduction naturelle intuitionniste NJ.

Propriété 2.2.5 (Cohérence de NJ). *Il n'existe pas dans NJ de preuve de* $\vdash \bot$.

Démonstration. S'il existait une preuve de $\vdash \bot$ dans NJ, il en existerait une sans coupure par la propriété 2.2.3, et la propriété 2.2.4 montre que celle-ci terminerait par une introduction. Une inspection des règles d'introduction démontre qu'il n'existe pas de preuve de $\vdash \bot$ terminant par une règle d'introduction. □

NJ propose donc une distinction non-triviale entre séquents dérivables et séquents non-dérivables. D'ailleurs s'il existait une dérivation de $\vdash \bot$, il existerait bien une dérivation de n'importe quel séquent par la règle $\bot$Elim puis affaiblissements successifs.

Les deux propriétés suivantes justifient le fait que NJ est un système *constructif* : lorsque l'on prouve que A ou B est vérifié, c'est que l'un des deux est vérifié ; lorsque l'on prouve qu'il existe un objet x tel que $A(x)$, c'est qu'on peut exhiber un tel objet.

$$\vee\textsc{Intro}1\;\vee\textsc{Elim}\;\dfrac{\dfrac{(\pi_1)}{\Gamma\vdash\varphi}}{\dfrac{\Gamma\vdash\varphi\vee\psi}{\Gamma\vdash\psi'}}\quad\dfrac{(\pi_2)}{\Gamma,\varphi\vdash\psi'}\quad\dfrac{(\pi_3)}{\Gamma,\psi\vdash\psi'}$$

$$\rightsquigarrow$$

$$\Rightarrow\textsc{Intro}\;\Rightarrow\textsc{Elim}\;\dfrac{\dfrac{(\pi_2)}{\Gamma,\varphi\vdash\psi'}}{\dfrac{\Gamma\vdash\varphi\Rightarrow\psi'}{\Gamma\vdash\psi'}}\quad\dfrac{(\pi_1)}{\Gamma\vdash\varphi}$$

$$\vee\textsc{Intro}2\;\vee\textsc{Elim}\;\dfrac{\dfrac{(\pi_1)}{\Gamma\vdash\psi}}{\dfrac{\Gamma\vdash\varphi\vee\psi}{\Gamma\vdash\psi'}}\quad\dfrac{(\pi_2)}{\Gamma,\varphi\vdash\psi'}\quad\dfrac{(\pi_3)}{\Gamma,\psi\vdash\psi'}$$

$$\rightsquigarrow$$

$$\Rightarrow\textsc{Intro}\;\Rightarrow\textsc{Elim}\;\dfrac{\dfrac{(\pi_3)}{\Gamma,\psi\vdash\psi'}}{\dfrac{\Gamma\vdash\psi\Rightarrow\psi'}{\Gamma\vdash\psi'}}\quad\dfrac{(\pi_1)}{\Gamma\vdash\psi}$$

$$\exists\textsc{Intro}\;\exists\textsc{Elim}\;\dfrac{\dfrac{(\pi_1)}{\Gamma\vdash\varphi[t/x]}}{\dfrac{\Gamma\vdash\exists x.\varphi}{\Gamma\vdash\psi}}\quad\dfrac{(\pi_2)}{\Gamma,\varphi\vdash\psi}\;x\notin\mathcal{FV}(\Gamma,\psi)$$

$$\rightsquigarrow$$

$$\Rightarrow\textsc{Intro}\;\Rightarrow\textsc{Elim}\;\dfrac{\dfrac{(\pi_2[t/x])}{\Gamma,\varphi[t/x]\vdash\psi}}{\dfrac{\Gamma\vdash\varphi[t/x]\Rightarrow\psi}{\Gamma\vdash\psi}}\quad\dfrac{(\pi_1)}{\Gamma\vdash\varphi[t/x]}$$

FIG. 2.6 – Normalisation en déduction naturelle, coupures sur $\vee$ et $\exists$

Propriété 2.2.6 (Propriété de la disjonction pour NJ). *Si π est une preuve sans coupure d'un séquent $\vdash \varphi \vee \psi$ dans NJ, alors la dernière étape de π est soit une instance de* $\wedge$Intro1*, soit une instance de* $\wedge$Intro2 *et donc contient soit une preuve de $\vdash \varphi$, soit une preuve de $\vdash \psi$.*

Démonstration. Par la propriété 2.2.4. □

Propriété 2.2.7 (Propriété du témoin pour NJ). *Si π est une preuve sans coupure d'un séquent $\vdash \exists x.\varphi$ dans NJ, alors la dernière étape de π est une instance de* $\exists$Intro *et donc π contient un terme t et une preuve de $\vdash \varphi[t/x]$.*

Démonstration. Par la propriété 2.2.4. □

Il est difficile (mais néanmoins possible) d'en dire plus sur la déduction naturelle sans introduire le calcul des séquents (ce que nous allons faire dans la section suivante). Les difficultés sont majoritairement posées par les règles $\vee$Elim et $\exists$Elim, comme analysé par Girard *et al.* (1989, chapitre 10).

2.3 Calcul des séquents

Nous présentons à présent le deuxième système introduit par Gentzen (1935). Le calcul des séquents classique est défini comme suit.

Définition 2.3.1 (Calcul des séquents classique LK). *Le calcul des séquents classique est le système contenant les inférences de la figure 2.7 pour tous contextes Γ et Δ, pour toutes formules φ, ψ et φ', pour toute variable x et pour tout terme du premier ordre t. Remarquons que ce système contient des règles structurelles* Axiom, Cut, ContrR *et* ContrL *ainsi que exactement une règle d'introduction pour chaque connecteur à gauche et à droite du symbole $\vdash$.*

Le calcul des séquents classique est organisé autour de la symétrie offerte par le symbole $\vdash$. Par exemple la règle $\wedge$R est la symétrique de la règle $\vee$L, la règle $\exists$L est la symétrique de la règle $\forall$R. Le calcul des séquents intuitionniste est défini comme suit.

Définition 2.3.2 (Calcul des séquents intuitionniste LJ). *Le calcul des séquents intuitionniste est le système contenant les inférences de la figure 2.8 pour tout contexte Γ, pour toutes formules φ, ψ et φ', pour toute variable x et pour tout terme du premier ordre t. Tout comme LK, ce système contient des règles structurelles* Axiom, Cut, ContrR *et* ContrL *ainsi que exactement une règle d'introduction pour chaque connecteur à gauche et à droite du symbole $\vdash$. Remarquons par ailleurs que comme dans NJ, les séquents dérivables dans LJ sont tous de la forme $\Gamma \vdash \varphi$ où Γ est un contexte et φ une formule.*

Mis à part les règles Axiom, Cut, ContrR et ContrL, les règles de LK et LJ sont toutes des introductions d'un connecteur à gauche ou à droite. Dans une telle règle,

$$\textsc{Axiom}\ \frac{}{\Gamma, \varphi \vdash \varphi, \Delta} \qquad \textsc{Cut}\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \varphi \vdash \Delta}{\Gamma \vdash \Delta}$$

$$\textsc{ContrL}\ \frac{\Gamma, \varphi, \varphi \vdash \Delta}{\Gamma, \varphi \vdash \Delta} \qquad \textsc{ContrR}\ \frac{\Gamma \vdash \varphi, \varphi, \Delta}{\Gamma \vdash \varphi, \Delta}$$

$$\Rightarrow\text{R}\ \frac{\Gamma, \varphi \vdash \psi, \Delta}{\Gamma \vdash \varphi \Rightarrow \psi, \Delta} \qquad \Rightarrow\text{L}\ \frac{\Gamma, \psi \vdash \Delta \qquad \Gamma \vdash \varphi, \Delta}{\Gamma, \varphi \Rightarrow \psi \vdash \Delta}$$

$$\wedge\text{R}\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \wedge \psi, \Delta} \qquad \wedge\text{L}\ \frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, \varphi \wedge \psi \vdash \Delta}$$

$$\neg\text{R}\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma \vdash \neg\varphi, \Delta} \qquad \neg\text{L}\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma, \neg\varphi \vdash \Delta}$$

$$\vee\text{R}\ \frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \varphi \vee \psi, \Delta} \qquad \vee\text{L}\ \frac{\Gamma, \varphi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \varphi \vee \psi \vdash \Delta}$$

$$\forall\text{R}\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \forall x.\varphi, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \qquad \forall\text{L}\ \frac{\Gamma, \varphi[t/x] \vdash \Delta}{\Gamma, \forall x.\varphi \vdash \Delta}$$

$$\top\text{R}\ \frac{}{\Gamma \vdash \top, \Delta} \qquad \top\text{L}\ \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta}$$

$$\exists\text{R}\ \frac{\Gamma \vdash \varphi[t/x], \Delta}{\Gamma \vdash \exists x.\varphi, \Delta} \qquad \exists\text{L}\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \exists x.\varphi \vdash \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta)$$

$$\bot\text{R}\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta} \qquad \bot\text{L}\ \frac{}{\Gamma, \bot \vdash \Delta}$$

FIG. 2.7 – Inférences en calcul des séquents classique LK

$$\textsc{Axiom}\ \frac{}{\Gamma, \varphi \vdash \varphi} \qquad \textsc{Cut}\ \frac{\Gamma \vdash \varphi \qquad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi} \qquad \textsc{ContrL}\ \frac{\Gamma, \varphi, \varphi \vdash \psi}{\Gamma, \varphi \vdash \psi}$$

$$\Rightarrow\text{R}\ \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \Rightarrow \psi} \qquad \Rightarrow\text{L}\ \frac{\Gamma, \psi \vdash \psi' \qquad \Gamma \vdash \varphi}{\Gamma, \varphi \Rightarrow \psi \vdash \psi'}$$

$$\wedge\text{R}\ \frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} \qquad \wedge\text{L}\ \frac{\Gamma, \varphi, \psi \vdash \psi'}{\Gamma, \varphi \wedge \psi \vdash \psi'}$$

$$\neg\text{R}\ \frac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi} \qquad \neg\text{L}\ \frac{\Gamma \vdash \varphi}{\Gamma, \neg\varphi \vdash \bot}$$

$$\vee\text{R1}\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} \qquad \vee\text{R2}\ \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} \qquad \vee\text{L}\ \frac{\Gamma, \varphi \vdash \psi' \qquad \Gamma, \psi \vdash \psi'}{\Gamma, \varphi \vee \psi \vdash \psi'}$$

$$\forall\text{R}\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x.\varphi}\ x \notin \mathcal{FV}(\Gamma) \qquad \forall\text{L}\ \frac{\Gamma, \varphi[t/x] \vdash \psi'}{\Gamma, \forall x.\varphi \vdash \psi'}$$

$$\exists\text{R}\ \frac{\Gamma \vdash \varphi[t/x]}{\Gamma \vdash \exists x.\varphi} \qquad \exists\text{L}\ \frac{\Gamma, \varphi \vdash \psi}{\Gamma, \exists x.\varphi \vdash \psi}\ x \notin \mathcal{FV}(\Gamma, \psi)$$

$$\top\text{R}\ \frac{}{\Gamma \vdash \top} \qquad \top\text{L}\ \frac{\Gamma \vdash \varphi}{\Gamma, \top \vdash \varphi} \qquad \bot\text{L}\ \frac{}{\Gamma, \bot \vdash \varphi}$$

FIG. 2.8 – Inférences en calcul des séquents intuitionniste LJ

la conclusion contient une formule qui est décomposée en sous-instances dans les prémisses. Par exemple $\wedge$R et $\wedge$L décomposent $\varphi \wedge \psi$ en φ et ψ ; $\exists$R décompose $\exists x.\varphi$ en $\varphi[t/x]$. La formule décomposée ainsi que son occurrence dans la conclusion de l'inférence sont respectivement appelées *formule principale* et *occurrence principale* de l'inférence. Les formules qui sont les résultats de cette décomposition ainsi que leurs occurrences dans les prémisses sont respectivement appelées les *formules secondaires* et les *occurrences secondaires* de l'inférence. Nous utiliserons la même terminologie pour les règles ContrR et ContrL. Dans ce cas les formules secondaires sont égales à la formule principale. Lorsque nous considérerons une instance de la règle Axiom, la formule (ou les formules) apparaissant des deux côtés du symbole $\vdash$ seront appelées *formules axiome*. Enfin lorsque nous considérerons une instance de la règle Cut, la formule apparaissant dans les deux prémisses et non dans la conclusion sera appelée *formule de coupure*.

Les inférences de LK et LJ respectent les polarités dans le sens suivant : si une prémisse d'une inférence de LK ou LJ admet une occurrence positive (respectivement négative) d'une formule, c'est forcément que sa conclusion admet aussi une occurrence positive (respectivement négative) de la même formule. D'ailleurs la seule façon de faire disparaître une formule en appliquant les règles de LK ou LJ du haut vers le bas, c'est d'utiliser une instance de la règle Cut. Inversement la seule façon de faire apparaître une formule φ, c'est de la décomposer en utilisant une introduction dont la formule principale est φ. On fait apparaître une occurrence positive dans le cas d'une introduction à droite et une occurrence négative dans le cas d'une introduction à gauche. Nous dirons donc que les règles correspondant à une occurrence positive d'un connecteur c sont les règles d'introduction à droite de c. Nous dirons que les règles correspondant à une occurrence négative d'un connecteur c sont les règles d'introduction à gauche de c.

Beaucoup de variations sont possibles autour de ces systèmes. En particulier et contrairement aux systèmes NJ et NK que nous avons présentés en section 2.2, les systèmes LJ et LK que nous venons de présenter ne sont pas les systèmes originaux de Gentzen (1935). Un large éventail de systèmes similaires est présenté par Kleene (1952a, chapitre 15). Une variation classique est l'utilisation de règles d'affaiblissement. La règle Axiom est alors remplacée par les trois règles

$$\text{Axiom}' \ \frac{}{\varphi \vdash \varphi} \qquad \text{WL} \ \frac{\Gamma \vdash \Delta}{\Gamma, \varphi \vdash \Delta} \qquad \text{WR} \ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \varphi, \Delta}$$

Les règles d'affaiblissement WR et WL étant admissibles dans les systèmes LK et LJ tels que nous les avons présentés, il est en fait possible de démontrer la propriété suivante.

Propriété 2.3.1 (Affaiblissement dans LK). *Si π est une preuve de $\Gamma \vdash \Delta$ dans LK et si Γ' et Δ' sont deux contextes, alors il existe une preuve notée $\Gamma' \cdot \pi \cdot \Delta'$ de $\Gamma, \Gamma' \vdash \Delta, \Delta'$ dans LK contenant exactement les inférences de π auxquelles sont rajoutés les contextes Γ' et Δ'.*

Démonstration. Un parcours inductif de la preuve π permet de construire $\Gamma' \cdot \pi \cdot \Delta'$. Les étapes $\forall$R et $\exists$L peuvent nécessiter l'utilisation d'alpha-conversion. □

Voici la propriété équivalente pour LJ.

Propriété 2.3.2 (Affaiblissement dans LJ). *Si π est une preuve de $\Gamma \vdash \varphi$ dans LJ et si Γ' est un contexte, alors il existe une preuve notée $\Gamma' \cdot \pi$ de $\Gamma, \Gamma' \vdash \varphi$ dans LJ contenant exactement les inférences de π auxquelles est rajouté le contexte Γ'.*

Une autre variation consiste à restreindre les formules axiomes des instances de Axiom aux formules atomiques. On obtient un système toujours équivalent. Par exemple toute instance de Axiom sur une disjonction $\varphi \vee \psi$ peut être décomposée de la manière suivante.

$$\dfrac{\dfrac{\dfrac{}{\varphi \vdash \varphi, \psi}\,\text{Axiom}}{\varphi \vdash \varphi \vee \psi}\,\vee\text{R} \qquad \dfrac{\dfrac{}{\psi \vdash \varphi, \psi}\,\text{Axiom}}{\psi \vdash \varphi \vee \psi}\,\vee\text{R}}{\varphi \vee \psi \vdash \varphi \vee \psi}\,\vee\text{L}$$

Cela mène à la propriété suivante.

Propriété 2.3.3 (Décomposition totale des axiomes). *S'il existe une preuve d'un séquent respectivement dans LK ou LJ, alors elle peut être complétée en une preuve du même séquent dans le même système telle que les formules axiomes des instances de* Axiom *qu'elle contient sont toutes des formules atomiques. Cette nouvelle preuve est un préfixe (en tant qu'arbre) de la preuve de départ.*

Démonstration. Il suffit d'écrire une preuve comme celle ci-dessus pour chaque connecteur et d'utiliser la propriété 2.3.1 ou 2.3.2. □

Tout comme les affaiblissements sont inclus dans la règle Axiom, il est possible de camoufler les contractions dans les règles d'introduction. Par exemple la règle $\exists$R peut être remplacée par

$$\exists\text{R+ContrR}\ \frac{\Gamma \vdash \varphi[t/x], \exists x.\varphi, \Delta}{\Gamma \vdash \exists x.\varphi, \Delta}$$

Si l'on applique cette transformation à toutes les règles d'introduction de LK, alors on peut supprimer les règles ContrR et ContrL. Le système obtenu est équivalent à notre système LK. La même manipulation peut être appliquée à LJ, mais uniquement du côté gauche. On modifie alors uniquement les règles d'introduction à gauche et on supprime la règle ContrL.

Enfin une dernière variation consiste à considérer les versions multiplicatives des inférences. Pour LK, les versions multiplicatives des règles Cut et $\vee$L sont par exemple

$$\text{Cut}\ \frac{\Gamma_1 \vdash \varphi, \Delta_1 \qquad \Gamma_2, \varphi \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \qquad \text{et} \qquad \vee\text{L}\ \frac{\Gamma_1, \varphi \vdash \Delta_1 \qquad \Gamma_2, \psi \vdash \Delta_2}{\Gamma_1, \Gamma_2, \varphi \vee \psi \vdash \Delta_1, \Delta_2}$$

Les versions de ces deux règles que nous avons utilisées pour définir notre système LK sont dites additives. La contraction et l'affaiblissement permettent de montrer que le système obtenu est équivalent à notre système LK. Ce n'est plus vrai si on les supprime. On obtient alors la logique linéaire (Girard, 1987) où chaque connecteur a une version additive et multiplicative qui ne sont plus équivalentes. Au sens de la logique linéaires, les règles $\vee$R et $\wedge$L que nous utilisons sont, par contre, additives. Puisque nous disposons de l'affaiblissement et de la contraction, nous pourrions les remplacer par leurs versions multiplicatives.

$$\vee\text{R}_1\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \varphi \vee \psi, \Delta} \qquad \vee\text{R}_2\ \frac{\Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \vee \psi, \Delta}$$

$$\wedge\text{L}_1\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \varphi \vee \psi \vdash \Delta} \qquad \wedge\text{L}_2\ \frac{\Gamma, \psi \vdash \Delta}{\Gamma, \varphi \vee \psi \vdash \Delta}$$

La notion de coupure est incarnée dans les systèmes LK et LJ par la règles d'inférence Cut. En particulier nous noterons LK^{cf} et LJ^{cf} les systèmes LK et LJ sans la règle de coupure Cut. L'intérêt principal du calcul des séquents réside dans le *Hauptsatz* démontré à l'origine par Gentzen.

Propriété 2.3.4 (Élimination des coupures (Hauptsatz)). *Toute preuve respectivement dans LK ou dans LJ peut être transformée en une preuve du même séquent dans* LK^{cf} *ou dans* LJ^{cf}.

En d'autres termes, la règle de coupure Cut est admissible dans LJ^{cf} et dans LK^{cf}. Il existe un grand nombre de preuves de cette proposition. On peut par exemple exhiber une transformation des preuves et prouver sa normalisation faible, comme le fait Gentzen (1935), voire sa normalisation forte, comme le fait Urban (2000, 2001). Usuellement on fait la différence entre deux types de coupures. Nous appellerons *coupures logiques* les instances de Cut dont chacune des deux prémisses est respectivement une introduction de la formule de coupure à gauche et à droite. Les coupures logiques sont généralement réduites de la façon décrite par les figures 2.9 et 2.10. Plus précisément elles sont réduites en nouvelles coupures dont les formules principales sont des sous-instances de la formule principale de départ. Remarquons que les coupures logiques sur les conjonctions, disjonctions et implications peuvent en fait être éliminées de deux façons différentes. Par exemple pour l'implication une coupure

$$\text{Cut}\ \frac{\Rightarrow\text{R}\ \dfrac{(\pi_1)}{\Gamma, \varphi \vdash \psi, \Delta}}{\Gamma \vdash \varphi \Rightarrow \psi, \Delta} \qquad \Rightarrow\text{L}\ \frac{(\pi_2) \qquad (\pi_3)}{\Gamma, \varphi \Rightarrow \psi \vdash \Delta}}{\Gamma \vdash \Delta}$$

$$\dfrac{\dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi, \Delta} \quad \dfrac{(\pi_2)}{\Gamma \vdash \psi, \Delta}}{\Gamma \vdash \varphi \wedge \psi, \Delta}\,\wedge\text{L} \qquad \dfrac{\dfrac{(\pi_3)}{\Gamma, \varphi, \psi \vdash \Delta}}{\Gamma, \varphi \wedge \psi \vdash \Delta}\,\wedge\text{R}}{\Gamma \vdash \Delta}\,\textsc{Cut}$$

$$\downarrow$$

$$\dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi, \Delta} \qquad \dfrac{\dfrac{(\varphi \cdot \pi_2)}{\Gamma, \varphi \vdash \psi, \Delta} \quad \dfrac{(\pi_3)}{\Gamma, \varphi, \psi \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}\,\textsc{Cut}}{\Gamma \vdash \Delta}\,\textsc{Cut}$$

$$\dfrac{\dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi, \psi, \Delta}}{\Gamma \vdash \varphi \vee \psi, \Delta}\,\vee\text{R} \qquad \dfrac{\dfrac{(\pi_2)}{\Gamma, \varphi \vdash \Delta} \quad \dfrac{(\pi_3)}{\Gamma, \psi \vdash \Delta}}{\Gamma, \varphi \vee \psi \vdash \Delta}\,\vee\text{L}}{\Gamma \vdash \Delta}\,\textsc{Cut}$$

$$\downarrow$$

$$\dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi, \psi, \Delta} \quad \dfrac{(\pi_3 \cdot \varphi)}{\Gamma, \psi \vdash \varphi, \Delta}}{\Gamma \vdash \varphi, \Delta}\,\textsc{Cut} \qquad \dfrac{(\pi_2)}{\Gamma, \varphi \vdash \Delta}\,\textsc{Cut}}{\Gamma \vdash \Delta}$$

$$\dfrac{\dfrac{\dfrac{(\pi_1)}{\Gamma, \varphi \vdash \psi, \Delta}}{\Gamma \vdash \varphi \Rightarrow \psi, \Delta}\,\Rightarrow\text{R} \qquad \dfrac{\dfrac{(\pi_2)}{\Gamma, \psi \vdash \Delta} \quad \dfrac{(\pi_3)}{\Gamma \vdash \varphi, \Delta}}{\Gamma, \varphi \Rightarrow \psi \vdash \Delta}\,\Rightarrow\text{L}}{\Gamma \vdash \Delta}\,\textsc{Cut}$$

$$\downarrow$$

$$\dfrac{\dfrac{\dfrac{(\pi_3 \cdot \psi)}{\Gamma \vdash \varphi, \psi, \Delta} \quad \dfrac{(\pi_1)}{\Gamma, \varphi \vdash \psi, \Delta}}{\Gamma \vdash \psi, \Delta}\,\textsc{Cut} \qquad \dfrac{(\pi_2)}{\Gamma, \psi \vdash \Delta}\,\textsc{Cut}}{\Gamma \vdash \Delta}$$

FIG. 2.9 – Coupures Logiques dans LK

$$\textsc{Cut}\ \dfrac{\neg\textsc{R}\ \dfrac{\overset{(\pi_1)}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \neg\varphi, \Delta} \qquad \neg\textsc{L}\ \dfrac{\overset{(\pi_2)}{\Gamma \vdash \varphi, \Delta}}{\Gamma, \neg\varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

$$\downarrow$$

$$\textsc{Cut}\ \dfrac{\overset{(\pi_2)}{\Gamma \vdash \varphi, \Delta} \qquad \overset{(\pi_1)}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

$$\textsc{Cut}\ \dfrac{\perp\textsc{R}\ \dfrac{\overset{(\pi)}{\Gamma \vdash \Delta}}{\Gamma \vdash \perp, \Delta} \qquad \perp\textsc{L}\ \dfrac{}{\Gamma, \perp \vdash \Delta}}{\Gamma \vdash \Delta} \quad \longrightarrow \quad \overset{(\pi)}{\Gamma \vdash \Delta}$$

$$\textsc{Cut}\ \dfrac{\top\textsc{R}\ \dfrac{}{\Gamma \vdash \top, \Delta} \qquad \top\textsc{L}\ \dfrac{\overset{(\pi)}{\Gamma \vdash \Delta}}{\Gamma, \top \vdash \Delta}}{\Gamma \vdash \Delta} \quad \longrightarrow \quad \overset{(\pi)}{\Gamma \vdash \Delta}$$

$$\textsc{Cut}\ \dfrac{\forall\textsc{R}\ \dfrac{\overset{(\pi_1)}{\Gamma \vdash \varphi, \Delta}}{\Gamma \vdash \forall x.\varphi, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \qquad \forall\textsc{L}\ \dfrac{\overset{(\pi_2)}{\Gamma, \varphi[t/x] \vdash \Delta}}{\Gamma, \forall x.\varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

$$\downarrow$$

$$\textsc{Cut}\ \dfrac{\overset{(\pi_1[t/x])}{\Gamma \vdash \varphi[t/x], \Delta} \qquad \overset{(\pi_2)}{\Gamma, \varphi[t/x] \vdash \Delta}}{\Gamma \vdash \Delta}$$

$$\textsc{Cut}\ \dfrac{\exists\textsc{R}\ \dfrac{\overset{(\pi_1)}{\Gamma \vdash \varphi[t/x], \Delta}}{\Gamma \vdash \exists x.\varphi, \Delta} \qquad \exists\textsc{L}\ \dfrac{\overset{(\pi_2)}{\Gamma, \varphi \vdash \Delta}}{\Gamma, \exists x.\varphi \vdash \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta)}{\Gamma \vdash \Delta}$$

$$\downarrow$$

$$\textsc{Cut}\ \dfrac{\overset{(\pi_1)}{\Gamma \vdash \varphi[t/x], \Delta} \qquad \overset{(\pi_2[t/x])}{\Gamma, \varphi[t/x] \vdash \Delta}}{\Gamma \vdash \Delta}$$

FIG. 2.10 – Coupures Logiques dans LK

peut être réduite en

$$\textsc{Cut}\ \dfrac{\dfrac{(\pi_3)}{\Gamma \vdash \varphi, \psi, \Delta} \qquad \dfrac{(\pi_1)}{\Gamma, \varphi \vdash \psi, \Delta}}{\textsc{Cut}\ \dfrac{\Gamma \vdash \psi, \Delta}{\Gamma \vdash \Delta} \qquad \dfrac{(\pi_2)}{\Gamma, \psi \vdash \Delta}}$$

ou en

$$\textsc{Cut}\ \dfrac{\dfrac{(\pi_3)}{\Gamma \vdash \varphi, \Delta} \qquad \textsc{Cut}\ \dfrac{\dfrac{(\pi_1)}{\Gamma, \varphi \vdash \psi, \Delta} \qquad \dfrac{(\pi_2)}{\Gamma, \varphi, \psi \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

Le point crucial est de savoir comment réduire les coupures qui ne sont pas logiques, et que nous appellerons d'ailleurs *coupures commutatives*. C'est d'ailleurs cela qui conditionne la normalisation forte ou parfois la confluence de la transformation d'élimination des coupures. Les coupures commutatives sur une instance de Axiom peuvent être aisément éliminées. Si la formule de coupure est la formule axiome, par exemple

$$\textsc{Cut}\ \dfrac{\textsc{Axiom}\ \dfrac{}{\Gamma, \varphi \vdash \varphi, \Delta} \qquad \dfrac{(\pi)}{\Gamma, \varphi, \varphi \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}\ , \quad \text{alors le réduit est} \quad \textsc{ContrL}\ \dfrac{\dfrac{(\pi)}{\Gamma, \varphi, \varphi \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}$$

Une coupure commutative sur une instance de ContrR ou ContrL doit, elle, être dupliquée. On peut par exemple décider naivement que

$$\textsc{Cut}\ \dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi, \Delta} \qquad \textsc{ContrL}\ \dfrac{\dfrac{(\pi_2)}{\Gamma, \varphi, \varphi \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

se réduit en

$$\textsc{Cut}\ \dfrac{\dfrac{(\pi_1)}{\Gamma \vdash \varphi, \Delta} \qquad \textsc{Cut}\ \dfrac{\dfrac{(\varphi \cdot \pi_1)}{\Gamma, \varphi \vdash \varphi, \Delta} \qquad \dfrac{(\pi_2)}{\Gamma, \varphi, \varphi \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

On peut d'ailleurs remplacer ici la preuve $\varphi \cdot \pi_1$ par une instance de Axiom. Une telle gestion des contractions s'avère rapidement pathologique. Une coupure sur une instance de Axiom se réduit en une contraction, et une coupure sur une contraction se réduit en deux coupures, dont l'une sur une instance de Axiom. On peut donc facilement aboutir à un contre-exemple à la normalisation forte (même en ne remplaçant pas $\varphi \cdot \pi_1$ par une instance de Axiom : en effet π_1 peut déjà être une instance de Axiom). Généralement, la manière dont les coupures commutatives sur les

contractions sont éliminées conditionne la normalisation forte de l'élimination des coupures. Enfin une coupure sur un affaiblissement, par exemple

$$\textsc{Cut}\ \dfrac{\textsc{WR}\ \dfrac{\overset{(\pi_1)}{\Gamma \vdash \Delta}}{\Gamma \vdash \varphi, \Delta} \qquad \overset{(\pi_2)}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

se réduit usuellement en

$$\overset{(\pi_1)}{\Gamma \vdash \Delta}$$

Dans ce cas la procédure d'élimination des coupures n'est pas confluente puisqu'une dérivation

$$\textsc{Cut}\ \dfrac{\textsc{WR}\ \dfrac{\overset{(\pi_1)}{\Gamma \vdash \Delta}}{\Gamma \vdash \varphi, \Delta} \qquad \textsc{WL}\ \dfrac{\overset{(\pi_2)}{\Gamma \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

se réduira en la dérivation π_1 ainsi qu'en la dérivation π_2, ces deux dérivation pouvant être totalement différentes. Même si les règles d'affaiblissement WR et WL ne sont pas explicites dans nos versions de LK et LJ, on pourra écrire le même contre-exemple sous la forme

$$\textsc{Cut}\ \dfrac{\overset{(\pi_1)}{\Gamma \vdash \varphi, \Delta} \qquad \overset{(\pi_2)}{\Gamma, \varphi \vdash \Delta}}{\Gamma \vdash \Delta}$$

où φ n'est la formule principale ou la formule axiome d'aucune inférence de π_1 et π_2. Dans ce cas, π_1 et π_2 s'écrivent respectivement $\pi_1' \cdot \varphi$ et $\varphi \cdot \pi_2'$ où π_1' et π_2' tous deux des preuves de $\Gamma \vdash \Delta$ et donc des réduits potentiels. Généralement, la manière dont les coupures commutatives sur les affaiblissements sont éliminée conditionne la confluence de l'élimination des coupures. Nous choisissons de ne pas résoudre ces problèmes pour le moment et admetterons donc la proposition 2.3.4. Urban (2000) propose une étude complète de ces questions dans.

La propriété 2.3.4 permet de déduire un certain nombre de corollaires sur les systèmes LK et LJ. Commençons par la propriété de sous-formule.

Propriété 2.3.5 (Propriété de la sous-formule). *Si π est une preuve sans coupure d'un séquent S dans LK ou LJ, alors les séquents des inférences de π ne contiennent que des sous-instances des formules de S.*

Démonstration. La seule règle de LK ou LJ dont les formules des prémisses ne sont pas toujours des sous-instances des formules de la conclusion est précisément Cut. □

C'est cette propriété qui explique l'utilisation du mot *analytique* pour désigner les preuves sans coupure : une preuve est analytique au sens de Kant (1781) si elle ne

fait qu'analyser le contenu du jugement qu'elle démontre sans tenter de l'étendre ou de le généraliser. Puisqu'une preuve sans coupure ne contient que des sous-instances du jugement qu'elle prouve, on dira qu'elle est analytique. Au contraire une preuve contenant une coupure peut utiliser la formule de coupure comme une généralisation du jugement $\Gamma \vdash \Delta$ qu'elle cherche à prouver. Un tel exemple serait une preuve

$$\pi \quad = \quad \textsc{Cut}\ \frac{\vdash \forall x.\varphi(x) \qquad \forall x.\varphi \vdash \varphi(t)}{\vdash \varphi(t)}$$

Bien évidemment, $\forall x.\varphi(x)$ est une généralisation de $\varphi(t)$. Supposons que la formule $\varphi(x)$ a le sens « si x est un entier, alors $2 \times x = x + x$ » et supposons que t représente le nombre 2. La preuve π prouve donc que $2 \times 2 = 2 + 2$. Plus précisément elle commence par démontrer que pour tout nombre entier x, $2 \times x = x + x$ et contient donc une récursion sur les nombres entiers. Au contraire une preuve analytique ne contiendra que le *calcul* qui explicite l'égalité $2 \times 2 = 2 + 2$. La récursion est bien évidemment, dans le cas d'une preuve de $2 \times 2 = 2 + 2$, un argument superflu et *synthétique* au sens de Kant, alors que le calcul est un argument *analytique*. Notons que dans le cas de la logique du premier ordre, l'analyse (c'est-à-dire le contenu) de certaines formules est infinie : c'est par exemple le cas des quantifications universelles $\forall x.\varphi$ dont l'analyse contient toutes les sous-instances $\varphi[t/x]$.

De manière plus pragmatique, la propriété a des conséquences intéressantes en termes de recherche de preuve. La méthode des tableaux (Hähnle, 2001) ainsi que la méthode inverse (Degtyarev et Voronkov, 2001) sont par exemple des méthodes de recherche de preuve qui s'appuient fortement sur cette propriété pour restreindre l'espace de recherche. Dans le cas propositionnel, l'ensemble des sous-instances des formules d'un séquent est toujours fini, et par finitude de l'espace de recherche, l'existence d'une dérivation sans coupure est décidable.

Une autre conséquence intéressante de la propriété 2.3.4 est la cohérence des système LK et LJ.

Propriété 2.3.6 (Cohérence de LK et LJ). *Il n'existe pas dans LK ou dans LJ de preuve de* $\vdash \bot$.

Démonstration. Si il existe une preuve de $\vdash \bot$ (dans LK ou LJ), par la propriété 2.3.4, il en existe une qui est sans coupure. Il nous suffit donc de montrer qu'il n'existe pas de preuve sans coupure de $\vdash \bot$ dans LK ou LJ. Le cas de LJ se règle en observant que la seule règle du système admettant une instance dont la conclusion est $\vdash \bot$ est la règle Cut. Une preuve sans coupure ne peut donc avoir comme conclusion $\vdash \bot$. Pour LK, nous prouvons qu'il n'existe pas de preuve sans coupure des séquents $\vdash \bot \ldots \bot$ (la séquence de $\bot$ peut être vide). Autrement dit notre but est de prouver que la conclusion de toute preuve sans coupure est un séquent qui n'est pas de la forme $\vdash \bot \ldots \bot$. Nous le prouvons par induction sur la preuve sans coupure. C'est clair si la preuve se termine par une règle qui n'est pas ContrR ou $\bot$R. Sinon c'est l'hypothèse d'induction appliquée à la prémisse de l'inférence en question qui démontre ce que nous cherchons. □

Enfin on peut aussi prouver la propriété du témoin ainsi que la propriété de la disjonction pour LJ. Toutes deux sont des corollaires de la propriété suivante.

Propriété 2.3.7. *Toute preuve sans coupure d'un séquent $\vdash \varphi$ dans LJ se termine par une introduction du connecteur de tête de φ.*

Démonstration. Les seules règles de LJ admettant comme instances des inférences dont la conclusion est $\vdash \varphi$ sont les règles d'introduction à droite. □

On en déduit alors

Propriété 2.3.8 (Propriété de la disjonction pour LJ). *Si π est une preuve sans coupure d'un séquent $\vdash \varphi \vee \psi$ dans LJ, alors la dernière étape de π est soit une instance de $\vee$R1, soit une instance de $\vee$R2 et donc contient soit une preuve de $\vdash \varphi$, soit une preuve de $\vdash \psi$.*

Démonstration. Par la propriété 2.3.7. □

Propriété 2.3.9 (Propriété du témoin pour LJ). *Si π est une preuve sans coupure d'un séquent $\vdash \exists x.\varphi$ dans LJ, alors la dernière étape de π est une instance de $\exists$R et donc π contient un terme t et une preuve de $\vdash \varphi[t/x]$.*

Démonstration. Par la propriété 2.3.7. □

En plus des coupures, une dérivation en calcul des séquents contient des étapes de décomposition (du bas vers le haut) des connecteurs de la dérivation. Le Hauptsatz montre que les coupures ne sont pas nécessaires. Il ne reste donc qu'a s'occuper des étapes de décomposition (et des contractions). Kleene (1952b) examine l'importance de l'ordre de ces décompositions. Le système déductif qu'il étudie est très proche de nos systèmes LK et LJ et ses résultats se transposent facilement à nos systèmes. Son premier lemme établit les permutations possibles entre inférences dans LK et LJ.

Propriété 2.3.10 (Lemme 7 et 8 de (Kleene, 1952b)). *Si dans une dérivation dans LK ou LJ, une inférence L_1 est placée juste au dessus d'une inférence L_2 et si la formule principale de L_1 n'est pas une formule secondaire de L_2, alors si $\frac{L_1}{L_2}$ n'est pas une instance de $\frac{\forall L \text{ ou } \exists R}{\forall R \text{ ou } \exists L}$ pour LK et*

$$\frac{\forall L}{\forall R} \quad \text{ou} \quad \frac{\forall L \text{ ou } \exists R}{\exists L} \quad \text{ou} \quad \frac{\Rightarrow L \text{ ou } \neg L}{\Rightarrow R \text{ ou } \neg R}$$

$$\text{ou} \quad \frac{\Rightarrow R \text{ ou } \Rightarrow L \text{ ou } \wedge R \text{ ou } \vee R \text{ ou } \neg R \text{ ou } \neg L \text{ ou } \forall R \text{ ou } \exists R}{\vee L}$$

pour LJ, on peut écrire la même dérivation où L_1 et L_2 ont été permutée. Pour LJ, on peut tout de même effectuer la permutation dans les cas

$$\frac{\Rightarrow R \text{ ou } \wedge R \text{ ou } \neg R \text{ ou } \forall R}{\vee L}$$

si toutes les inférences dont la conclusion est l'une des prémisses de L_2 sont des introductions de l'occurrence principale de L_1.

Kleene en déduit une généralisation sur les réarrangements possibles des inférences d'une dérivation.

Propriété 2.3.11 (Théorème 2 de (Kleene, 1952b)). *Soit $\Gamma \vdash \Delta$ et soit $(C_k)_{1 \leqslant k \leqslant n}$ une partition des positions de $\Gamma \vdash \Delta$ telle que*

- *si $(n, \varphi, \nu) \in C_i$ et $(n, \varphi, \nu') \in C_j$ et si ν' est un préfixe de ν, alors $i \leqslant j$;*
- *si a est une position de C_i correspondant à la règle d'inférence L_1 et b une position de C_j correspondant à la règle d'inférence L_2 et si $\frac{L_1}{L_2}$ est l'une des paires $\frac{\forall L \text{ ou } \exists R}{\forall R \text{ ou } \exists L}$ pour LK et*

$$\frac{\forall L}{\forall R} \quad ou \quad \frac{\forall L \text{ ou } \exists R}{\exists L} \quad ou \quad \frac{\Rightarrow L \text{ ou } \neg L}{\Rightarrow R \text{ ou } \neg R} \quad ou \quad \frac{\Rightarrow L \text{ ou } \vee R \text{ ou } \neg L \text{ ou } \exists R}{\vee L}$$

  pour LJ, alors $i \leqslant j$.

Alors si le séquent $\Gamma \vdash \Delta$ est dérivable, il en existe une dérivation telle que pour tout chemin de la racine à l'une des feuilles, la décomposition d'une occurrence correspondant à une position $a \in C_i$ apparaît toujours au dessus de la décomposition d'une occurrence correspondant à une position $b \in C_j$ dès que $i < j$.

De cette dernière propriété découle les résultats d'inversibilité suivants.

Propriété 2.3.12. *Dans LK, les règles $\Rightarrow R$, $\Rightarrow L$, $\wedge R$, $\wedge L$, $\vee R$, $\vee L$, $\neg R$, $\neg L$, $\bot R$, $\bot L$, $\top R$, $\top L$, $\exists L$, $\forall R$,* Cut, ContrL *et* ContrR *sont inversibles. Seule les règles $\exists R$ et $\forall L$ admettent des instances non inversibles.*

Les règles ∃R et ∀L admettent respectivement les instances

$$\frac{P(a) \vee P(b) \vdash P(a)}{P(a) \vee P(b) \vdash \exists x.P(x)} \qquad \text{et} \qquad \frac{P(a) \vdash P(a) \wedge P(b)}{\forall x.P(x) \vdash P(a) \wedge P(b)}$$

qui ne sont pas inversibles : il existe bien des dérivations de leurs prémisses respectives mais aucune de leurs conclusions. Le système LJ offre, lui, une souplesse réduite comme le démontre la propriété suivante, corollaire du théorème de Kleene.

Propriété 2.3.13. *Dans LJ, les règles $\Rightarrow R$, $\wedge R$, $\wedge L$, $\vee L$, $\neg R$, $\bot L$, $\top R$, $\exists L$, $\forall R$ et* ContrL *sont inversibles.*

Ces résultats ont une importance capitale lorsqu'on adopte l'approche qui consiste à appliquer les règles du système déductif du bas vers le haut en partant du jugement que l'on cherche à prouver. On espère ainsi obtenir des instances de la règle Axiom et construire une preuve complète du jugement. C'est par exemple l'approche utilisée par la méthode analytique des tableaux. Bien évidemment il est crucial de choisir la bonne inférence à appliquer au jugement à prouver. Si l'on se

rend compte plus tard qu'un mauvais choix a été fait, il faut alors faire du backtracking, retourner sur ses pas et choisir une inférence différente. Les inférences inversibles représentent alors les inférences dont on est sûr qu'elles représentent *un bon choix*. En effet puisqu'elles sont inversibles, si il existe une preuve de leur conclusion, il en existera une de leurs prémisses. Il sera donc inutile donc de remettre en cause un tel choix.

2.4 Logique intuitionniste, logique classique et modèles

Le calcul des séquents et la déduction naturelle sont deux formalismes équivalents en termes de prouvabilité. Nous verrons que tout séquent prouvable dans NJ est prouvable dans LJ et inversement. Le cas de NK et LK est un peu plus compliqué puisque tous les séquents prouvables dans NK n'ont qu'une seule formule à droite du symbole $\vdash$, ce qui n'est pas le cas du système LK. Nous appellerons de tels séquents des séquents unaires. Nous verrons que tout séquent unaire est prouvable dans LK si et seulement si il l'est dans NK. Nous dirons que les deux systèmes NK et LK sont des systèmes de preuve pour la logique *classique* alors que NJ et LJ sont des systèmes de preuve pour la logique *intuitionniste*. La logique classique n'est pas équivalente à la logique intuitionniste. Nous verrons que le séquent $\vdash \varphi \vee \neg\varphi$ n'est pas prouvable en logique intuitionniste alors qu'il l'est en logique classique.

2.4.1 Logique classique

L'équivalence entre LK et NK peut être démontrée grâce à des traductions des preuves d'un système vers l'autre et inversement. Par exemple une instance de $\neg$Intro dans NK

$$\neg\textsc{Intro}\ \dfrac{\overset{\pi}{\Gamma, \varphi \vdash \bot}}{\Gamma \vdash \neg\varphi} \qquad \text{peut être transformée dans LK en} \qquad \neg\textsc{R}\ \dfrac{\textsc{Cut}\ \dfrac{\overset{\widetilde{\pi}}{\Gamma, \varphi \vdash \bot} \qquad \bot\textsc{L}\ \dfrac{}{\Gamma, \varphi, \bot \vdash}}{\Gamma, \varphi \vdash}}{\Gamma \vdash \neg\varphi}\ .$$

De même une instance de $\vee$Intro1 dans NK

$$\vee\textsc{Intro1}\ \dfrac{\overset{\pi}{\Gamma \vdash \varphi}}{\Gamma \vdash \varphi \vee \psi} \qquad \text{peut être transformée dans LK en} \qquad \vee\textsc{R}\ \dfrac{\textsc{WR}\ \dfrac{\overset{\widetilde{\pi}}{\Gamma \vdash \varphi}}{\Gamma \vdash \varphi, \psi}}{\Gamma \vdash \varphi \vee \psi}$$

et similairement pour $\vee$Intro2. Bien entendu, WR n'est pas une règle de LK, mais puisqu'elle est admissible (propriété 2.3.1), nous sommes autorisés à l'utiliser. Les étapes d'élimination de NK nécessitent un peu plus d'attention. Une instance de la règle $\Rightarrow$Elim

$$\Rightarrow\textsc{Elim}\ \dfrac{\overset{\pi_1}{\Gamma \vdash \varphi} \qquad \overset{\pi_2}{\Gamma \vdash \varphi \Rightarrow \psi}}{\Gamma \vdash \psi}$$

peut être transformée dans LK en

$$\dfrac{\dfrac{\dfrac{\overset{\pi_2}{\Gamma \vdash \varphi \Rightarrow \psi}}{\Gamma \vdash \varphi \Rightarrow \psi, \psi}\,\text{WR} \qquad \dfrac{}{\Gamma, \psi \vdash \psi}\,\text{Axiom}}{\Gamma, \varphi \Rightarrow \psi \vdash \psi}\,\Rightarrow\text{L} \qquad \dfrac{\overset{\pi_1}{\Gamma \vdash \varphi}}{\Gamma \vdash \varphi, \psi}\,\text{WR}}{\Gamma \vdash \psi}\,\text{Cut}$$

Similairement pour les autres connecteurs, une élimination dans NK se traduit dans LK par une coupure couplée à une introduction à gauche. Néanmoins la traduction de LK vers NK est un peu plus compliquée. Nous allons donc prouver l'équivalence de ces deux systèmes en utilisant une troisième notion intermédiaire : celle de modèle. Rappelons que la définition de notre ensemble de formule $\mathcal{F}_{\mathcal{A}T_\alpha(X)}$ repose sur une algèbre de termes $T_\alpha(X)$ ainsi qu'un ensemble $\mathcal{A}$ de prédicats ayant tous une arité fixe. Nos noterons $\texttt{false} = 0$ et $\texttt{true} = 1$ les deux éléments de $\mathbb{B} = \mathbb{Z}/2\mathbb{Z}$. Nous noterons aussi

$$\begin{array}{rcl} \widetilde{\neg} : \mathbb{B} & \to & \mathbb{B} \\ x & \mapsto & x+1 \end{array} \qquad \begin{array}{rcl} \widetilde{\wedge} : \mathbb{B}^2 & \to & \mathbb{B} \\ x, y & \mapsto & \min(x,y) \end{array}$$

$$\begin{array}{rcl} \widetilde{\vee} : \mathbb{B}^2 & \to & \mathbb{B} \\ x, y & \mapsto & \max(x,y) \end{array} \qquad \begin{array}{rcl} \widetilde{\Rightarrow} : \mathbb{B}^2 & \to & \mathbb{B} \\ x, y & \mapsto & \max(x+1,y) \end{array}$$

Nous noterons d'ailleurs ces trois dernières fonctions en position infixe. Remarquons par exemple que pour tout $x, y \in \mathbb{B}$, $x \mathbin{\widetilde{\Rightarrow}} y = (\mathbin{\widetilde{\neg}} x) \mathbin{\widetilde{\vee}} y$ ou encore $x \mathbin{\widetilde{\vee}} (\mathbin{\widetilde{\neg}} x) = \texttt{true}$. Les tableaux suivants résument les valeurs de ces quatres fonctions.

x	$\widetilde{\neg}\, x$
false	true
true	false

x	y	$x \mathbin{\widetilde{\vee}} y$	$x \mathbin{\widetilde{\wedge}} y$	$x \mathbin{\widetilde{\Rightarrow}} y$
false	false	false	false	true
true	false	true	false	false
false	true	true	false	true
true	true	true	true	true

Définition 2.4.1 (Modèle Classique). *Un modèle classique $\mathcal{M}$ est un triplet contenant*

- *un domaine $\mathcal{D}$;*
- *une fonction d'interprétation des symboles de fonction $[\![\_]\!]$ associant à chaque symbole de fonction f d'arité n de la signature α une fonction $[\![f]\!] : \mathcal{D}^n \to \mathcal{D}$;*
- *une fonction d'interprétation des symboles de prédicat $(\!|\_|\!)$ associant à chaque symbole de prédicat P de $\mathcal{A}$ d'arité n une fonction $(\!|P|\!) : \mathcal{D}^n \to \mathbb{B}$.*

Définition 2.4.2 (Valuation). *Soit $\mathcal{M}$ un modèle ayant pour domaine un ensemble $\mathcal{D}$. Une valuation pour $\mathcal{M}$ est une fonction $\mu : X \to \mathcal{D}$.*

Notation 2.4.1. *Si μ est une valutaion pour un modèle $\mathcal{M}$ ayant comme domaine l'ensemble $\mathcal{D}$ et si $a \in \mathcal{D}$ et $x \in X$, alors $\mu[x \mapsto a]$ est la valuation (pour $\mathcal{M}$) $\mu' : X \to \mathcal{D}$ définie par $\mu'(y) = a$ si $y = x$; $\mu'(y) = \mu(y)$ sinon.*

$$
\begin{array}{lll}
(\!|\varphi|\!)^\mu &= \texttt{true} & \text{si } \varphi = \top \\
(\!|\varphi|\!)^\mu &= \texttt{false} & \text{si } \varphi = \bot \\
(\!|\varphi|\!)^\mu &= (\!|P|\!)([\![t_1]\!]^\mu \dots [\![t_n]\!]^\mu) & \text{si } \varphi = P(t_1 \dots t_n) \\
(\!|\varphi|\!)^\mu &= \widetilde{\neg}\,(\!|\psi|\!)^\mu & \text{si } \varphi = \neg\psi \\
(\!|\varphi|\!)^\mu &= (\!|\psi_1|\!)^\mu \,\widetilde{\wedge}\, (\!|\psi_2|\!)^\mu & \text{si } \varphi = \psi_1 \wedge \psi_2 \\
(\!|\varphi|\!)^\mu &= (\!|\psi_1|\!)^\mu \,\widetilde{\vee}\, (\!|\psi_2|\!)^\mu & \text{si } \varphi = \psi_1 \vee \psi_2 \\
(\!|\varphi|\!)^\mu &= (\!|\psi_1|\!)^\mu \,\widetilde{\Rightarrow}\, (\!|\psi_2|\!)^\mu & \text{si } \varphi = \psi_1 \Rightarrow \psi_2 \\
(\!|\varphi|\!)^\mu &= \min_{a\in\mathcal{D}}([\![\psi]\!]_{\mathcal{M}}^{\mu[x\mapsto a]}) & \text{si } \varphi = \forall x.\psi \\
(\!|\varphi|\!)^\mu &= \max_{a\in\mathcal{D}}([\![\psi]\!]_{\mathcal{M}}^{\mu[x\mapsto a]}) & \text{si } \varphi = \exists x.\psi
\end{array}
$$

FIG. 2.11 – Interprétation des Formules pour les Modèles Classiques

Soit un modèle $\mathcal{M} = (\mathcal{D}, [\![\_]\!], (\!|\_|\!))$ et μ une valuation pour $\mathcal{M}$, la fonction d'interprétation $[\![\_]\!]$ est étendue en une fonction $[\![\_]\!]^\mu$ sur les termes de $T_\alpha(X)$ inductivement comme suit : si t est une variable x, alors $[\![t]\!]^\mu = \mu(x)$; si $t = f(t_1 \dots t_n)$, alors $[\![t]\!]^\mu = [\![f]\!]([\![t_1]\!]^\mu \dots [\![t_n]\!]^\mu)$. L'interprétation $(\!|\varphi|\!)^\mu$ d'une formule φ dans le modèle $\mathcal{M}$ est alors définie inductivement en figure 2.11.

Notation 2.4.2. *Nous noterons $\mathcal{M} \models_{\mathcal{K}} \varphi$ le jugement « $(\!|\varphi|\!)^\mu = \texttt{true}$ ». Nous noterons $\models_{\mathcal{K}} \varphi$ le jugement « pour tout modèle $\mathcal{M}$, $\mathcal{M} \models_{\mathcal{K}} \varphi$ ». Nous noterons $\Gamma \models_{\mathcal{K}} \varphi$ le jugement « pour tout modèle $\mathcal{M}$, si $\mathcal{M} \models_{\mathcal{K}} \psi$ pour toute formule ψ de Γ, alors $\mathcal{M} \models_{\mathcal{K}} \varphi$ ». Enfin nous noterons $\Gamma \models_{\mathcal{K}} \Delta$ le jugement « pour tout modèle $\mathcal{M}$, si $\mathcal{M} \models_{\mathcal{K}} \psi$ pour toute formule ψ de Γ, alors il existe une formule φ de Δ telle que $\mathcal{M} \models_{\mathcal{K}} \varphi$ ».*

Définition 2.4.3 (Logique classique). *Une formule φ est classiquement valide si $\models_{\mathcal{K}} \varphi$. Un séquent $\Gamma \vdash \Delta$ est classiquement valide si $\Gamma \models_{\mathcal{K}} \Delta$.*

NK et LK sont corrects et complets pour la logique classique :

Correction tout séquent prouvable est classiquement valide ;

Complétude tout séquent classiquement valide est prouvable.

Nous allons à présent démontrer ces propriétés. Commençons par NK.

Propriété 2.4.1 (Correction de NK). *Tout séquent prouvable dans NK est classiquement valide.*

Démonstration. Soit donc un séquent prouvable dans NK. Une induction sur la preuve démontre qu'il est classiquement valide. Par exemple si la preuve se termine par

$$\Rightarrow\text{ELIM}\ \frac{\Gamma \vdash \varphi \Rightarrow \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

alors l'hypothèse d'induction montre que $\Gamma \models_{\mathcal{K}} \varphi \Rightarrow \psi$ et $\Gamma \models_{\mathcal{K}} \varphi$. Montrons que $\Gamma \models_{\mathcal{K}} \psi$. Soit donc un modèle $\mathcal{M}$ tel que $\mathcal{M} \models_{\mathcal{K}} \psi'$ pour toute formule ψ' de Γ. Alors $\mathcal{M} \models_{\mathcal{K}} \varphi \Rightarrow \psi$ et $\mathcal{M} \models_{\mathcal{K}} \varphi$. Alors

$$\begin{aligned} \texttt{true} &= (\!|\varphi \Rightarrow \psi|\!)^{\mu} = \max((\!|\varphi|\!)^{\mu} + 1, (\!|\psi|\!)^{\mu}) = \max(\texttt{true} + 1, (\!|\psi|\!)^{\mu}) \\ &= \max(\texttt{false}, (\!|\psi|\!)^{\mu}) = (\!|\psi|\!)^{\mu} \end{aligned}$$

ce qui démontre que $\mathcal{M} \models_{\mathcal{K}} \psi$. Finalement $\Gamma \models_{\mathcal{K}} \psi$. Les cas d'instances des autres règles d'inférences sont réglés de façon similaire. □

Propriété 2.4.2 (Complétude de NK). *Tout séquent $\Gamma \vdash \varphi$ classiquement valide est prouvable dans NK.*

Pour prouver cette propriété, nous allons nous baser sur les notions suivantes.

Définition 2.4.4 (Contexte consistant, théorie consistante, théorie complète). *Un contexte Γ est consistant si $\Gamma \vdash \bot$ n'est pas démontrable. Une théorie est un ensemble (potentiellement infini) de formules. Si $\mathcal{F}$ est une théorie, nous noterons $\mathcal{F} \vdash \varphi$ s'il existe un contexte Γ de formules de $\mathcal{F}$ tel que $\Gamma \vdash \varphi$ est prouvable (dans NK). Une théorie $\mathcal{F}$ est consistante si $\mathcal{F} \nvdash \bot$. Une théorie $\mathcal{F}$ est complète si elle est consistante et pour toute formule close φ, soit $\mathcal{F} \vdash \varphi$, soit $\mathcal{F} \vdash \neg\varphi$.*

Ces notions dépendent évidemment du système déductif dans lequel nous nous placerons. En particulier nous dirons qu'un contexte est consistant *dans un système déductif donné.*

Définition 2.4.5 (Contexte classiquement contradictoire). *Un contexte Γ est classiquement contradictoire s'il n'existe aucun modèle classique $\mathcal{M}$ tel que $\mathcal{M} \models_{\mathcal{K}} \varphi$ pour toute formule φ de Γ.*

Un corollaire de la propriété 2.4.1 est que tout contexte classiquement non contradictoire est consistant dans NK. Par contraposée, un contexte Γ non-consistant est tel que $\Gamma \vdash \bot$ admet une preuve et par la propriété 2.4.1, si $\mathcal{M} \models_{\mathcal{K}} \varphi$ pour toute formule φ de Γ, alors $\mathcal{M} \models_{\mathcal{K}} \bot$ ce qui est impossible. Γ est donc classiquement contradictoire. Inversement, la preuve de la propriété 2.4.2 va reposer sur le fait que tout contexte consistant dans NK n'est pas classiquement contradictoire.

Propriété 2.4.3. *Tout contexte consistant dans NK est classiquement non contradictoire.*

Démonstration. Supposons que Γ est consistant dans NK. On peut supposer sans perte de généralité que Γ est clos : sinon on peut rajouter à la signature α une constante pour chaque variable libre de Γ, substituer chaque variable libre de Γ par la constante qui lui est associée et ainsi obtenir Γ' consistant dans NK par la propriété 2.2.2. On suppose aussi donné un ensemble infini dénombrable $(c_i)_{i\in\mathbb{N}}$ de constantes de α n'apparaissant pas dans Γ'. La démonstration que nous allons exposer à présent repose sur le fait que les formules closes de $\mathcal{F}_{\mathcal{AT}_\alpha(X)}$ sont dénombrables. Il existe donc une suite infinie $(\psi_n)_{n\in\mathbb{N}}$ telle que pour toute formule close

φ, il existe n tel que $\varphi = \psi_n$. On définit alors la suite d'ensembles de formules closes $(K_i)_{i\in\mathbb{N}}$ récursivement comme suit :

- $K_0 = \Gamma$.
- Si K_n est complet pour NK, alors $K_{n+1} = K_n$.
- Sinon il existe un plus petit entier p tel que $K_n \nvdash \psi_p$ et $K_n \nvdash \neg\psi_p$ pour NK : si ψ_p n'est pas de la forme $\exists x.\varphi$, alors $K_{n+1} = K_n \cup \{\psi_p\}$; si ψ_p est de la forme $\exists x.\varphi$, alors $K_{n+1} = K_n \cup \{\psi_p, \varphi[c_i/x]\}$ où i est le plus petit entier tel que c_i n'apparaisse pas dans K_n.

L'ensemble $\mathfrak{K}$ est défini par $\mathfrak{K} = \cup_{i\in\mathbb{N}} K_i$. Alors $\mathfrak{K}$ est complet, consistant et $\Gamma \subseteq \mathfrak{K}$. On peut alors construire le modèle $\mathcal{M}$ suivant :

- son domaine est $T_\alpha(X)$;
- l'interprétation des variables est défini comme la fonction identité ;
- l'interprétation d'un symbole de fonction f d'arité n est la fonction

$$(t_1 \dots t_n) \mapsto f(t_1 \dots t_n) ;$$

- l'interprétation d'un prédicat P d'arité n est la fonction associant `true` à $(t_1 \dots t_n)$ si et seulement si $P(t_1 \dots t_n) \in \mathfrak{K}$.

Alors $\mathcal{M} \models_{\mathcal{K}} \varphi$ si et seulement si $\mathfrak{K} \vdash \varphi$ (par induction sur φ). En particulier si φ est une formule de Γ, alors $\varphi \in \mathfrak{K}$ donc $\mathfrak{K} \vdash \varphi$, d'où $\mathcal{M} \models_{\mathcal{K}} \varphi$. Finalement le modèle $\mathcal{M}$ montre que Γ est classiquement non contradictoire. □

Nous pouvons à présent démontrer la complétude de NK.

Démonstration de la propriété 2.4.2. Supposons que $\Gamma \models_{\mathcal{K}} \varphi$. Alors $\Gamma, \neg\varphi$ est classiquement contradictoire et donc par la propriété 2.4.3, $\Gamma, \neg\varphi$ est non-consistant dans NK : il existe une preuve de $\Gamma, \neg\varphi \vdash \bot$ dans NK, et donc par ¬Intro et ¬¬Elim, il existe une preuve de $\Gamma \vdash \varphi$. □

Notre système LK est lui aussi correct et complet par rapport à la logique classique.

Propriété 2.4.4 (Correction de LK). *Si $\Gamma \vdash \Delta$ est prouvable dans LK, alors $\Gamma \models_{\mathcal{K}} \Delta$.*

Démonstration. Par induction sur la preuve de $\Gamma \vdash \Delta$. □

Propriété 2.4.5 (Complétude de LK). *Si $\Gamma \models_{\mathcal{K}} \Delta$, alors $\Gamma \vdash \Delta$ est prouvable dans LK.*

Démonstration. Tout comme pour NK, tout contexte consistant dans LK est classiquement non contradictoire. Alors si $\Gamma \models_{\mathcal{K}} \Delta$, le contexte $\Gamma, \neg\Delta$ (si $\Delta = \varphi_1 \dots \varphi_n$, alors $\neg\Delta$ représente le contexte $\neg\varphi_1 \dots \neg\varphi_n$) est classiquement contradictoire et donc $\Gamma, \neg\Delta$ est non-consistant dans LK : il existe une preuve de $\Gamma, \neg\Delta \vdash \bot$ dans LK, et donc par inversibilité de ¬L (propriété 2.3.12), il existe une preuve de $\Gamma \vdash \Delta$. □

Finalement les propriétés 2.4.1, 2.4.2, 2.4.4 et 2.4.5 montrent l'équivalence entre NK et LK.

Propriété 2.4.6. *Tout séquent $\Gamma \vdash \varphi$ est prouvable dans NK si et seulement si il est prouvable dans LK. Tout séquent $\Gamma \vdash \varphi_1, \varphi_2 \dots \varphi_n$ est prouvable dans LK si et seulement si $\Gamma \vdash \varphi_1 \vee \varphi_2 \vee \dots \varphi_n$ est prouvable dans NK.*

Dualité de De Morgan

Remarquons que pour tous booléens $x, y \in \mathbb{B}$, les égalités

$$\widetilde{\neg}\,(x \,\widetilde{\vee}\, y) = \widetilde{\neg}\, x \,\widetilde{\wedge}\, \widetilde{\neg}\, y \quad \text{et} \quad \widetilde{\neg}\,(x \,\widetilde{\wedge}\, y) = \widetilde{\neg}\, x \,\widetilde{\vee}\, \widetilde{\neg}\, y$$

sont vérifiées. Cela implique que pour toutes formules φ et ψ, dans n'importe quel modèle classique, les formules $\neg(\varphi \wedge \psi)$ et $\neg\varphi \vee \neg\psi$ et respectivement $\neg(\varphi \vee \psi)$ et $\neg\varphi \wedge \neg\psi$ seront interprétées par la même valeur. Ces égalités traduisent une dualité entre les connecteurs $\wedge$ et $\vee$ que l'on appelle *dualité de De Morgan*. Elles se traduisent en déduction naturelle ou en calcul des séquents classiques par l'équivalence entre les formules $\neg(\varphi \wedge \psi)$ et $\neg\varphi \vee \neg\psi$ ainsi qu'entre les formules $\neg(\varphi \vee \psi)$ et $\neg\varphi \wedge \neg\psi$, et ce quelles que soient les formules φ et ψ. Notons aussi que cette dualité se traduit remarquablement bien en calcul des séquent par un symétrie flagrante entre les règles

$$\wedge\mathrm{R}\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \wedge \psi, \Delta} \quad \text{et} \quad \vee\mathrm{L}\ \frac{\Gamma, \varphi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \varphi \vee \psi \vdash \Delta}$$

ainsi qu'entre les règles

$$\vee\mathrm{R}\ \frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \varphi \vee \psi, \Delta} \quad \text{et} \quad \wedge\mathrm{L}\ \frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, \varphi \wedge \psi \vdash \Delta}\,.$$

Ces symétries se traduisent respectivement par le fait que réfuter une conjonction $\varphi \wedge \psi$ revient précisément à démontrer $\neg\varphi$ ou bien $\neg\psi$ et par le fait que réfuter une disjonction $\varphi \vee \psi$ revient exactement à démontrer $\neg\varphi$ et $\neg\psi$.

Une dualité similaire existe entre les quantificateurs $\exists$ et $\forall$. Tout d'abord pour toute formule φ et pour tout modèle classique, les formules $\neg\exists x.\varphi$ et $\forall x.\neg\varphi$ et respectivement $\neg\forall x.\varphi$ et $\exists x.\neg\varphi$ sont interprétées par la même valeur. En déduction naturelle et en calcul des séquents classiques, cela se traduit par l'équivalence des formules $\neg\exists x.\varphi$ et $\forall x.\neg\varphi$ ainsi qu'entre les formules $\neg\forall x.\varphi$ et $\exists x.\neg\varphi$. Enfin notons la symétrie qu'il existe entre les règles

$$\exists\mathrm{R}\ \frac{\Gamma \vdash \varphi[t/x], \Delta}{\Gamma \vdash \exists x.\varphi, \Delta} \quad \text{et} \quad \forall\mathrm{L}\ \frac{\Gamma, \varphi[t/x] \vdash \Delta}{\Gamma, \forall x.\varphi \vdash \Delta}$$

ainsi qu'entre les règles

$$\forall\mathrm{R}\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \forall x.\varphi, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \quad \text{et} \quad \exists\mathrm{L}\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \exists x.\varphi \vdash \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta)$$

$$\textsc{Axiom}\ \frac{}{\vdash \varphi, \neg\varphi, \Delta} \qquad \textsc{Contr}\ \frac{\vdash \Delta, \varphi, \varphi}{\vdash \Delta, \varphi} \qquad \textsc{Cut}\ \frac{\vdash \Delta, \varphi \qquad \vdash \Delta, \neg\varphi}{\vdash \Delta}$$

$$\wedge\ \frac{\vdash \Delta, \varphi \qquad \vdash \Delta, \psi}{\vdash \Delta, \varphi \wedge \psi} \qquad \vee\ \frac{\vdash \Delta, \varphi, \psi}{\vdash \Delta, \varphi \vee \psi} \qquad \Rightarrow\ \frac{\vdash \Delta, \neg\varphi, \psi}{\vdash \Delta, \varphi \Rightarrow \psi}$$

$$\bot\ \frac{\vdash \Delta}{\vdash \bot, \Delta} \qquad \top\ \frac{}{\vdash \Delta, \top} \qquad \exists\ \frac{\vdash \Delta, \varphi[t/x]}{\vdash \Delta, \exists x.\varphi} \qquad \forall\ \frac{\vdash \Delta, \varphi}{\vdash \Delta, \forall x.\varphi}\ x \notin \mathcal{FV}(\Delta)$$

FIG. 2.12 – Calcul des séquents à un seul côté LK1

Ces symétries se traduisent respectivement par le fait que réfuter une existence $\exists x.\varphi$ revient précisément à démontrer que pour tout x, $\neg\varphi$ est vérifiée et par le fait que réfuter une universalité $\forall x.\varphi$ revient exactement à démontrer l'existence d'un x vérifiant $\neg\varphi$. Nous verrons dans la sous-section suivante que ces dualités ne sont pas valables en logique intuitionniste.

Calcul des séquents à un seul côté

En logique classique, la dualité de De Morgan permet de définir un calcul des séquents correct et complet traitant de séquents à un seul côté : La négation en tant que connecteur syntaxique ne s'applique qu'aux propositions atomiques. Si φ contient un connecteur, sa négation est définie par la dualité de De Morgan comme suit.

$$\begin{array}{ll}
\neg\varphi = \bot & \text{si } \varphi = \top\ ; \\
\neg\varphi = \top & \text{si } \varphi = \bot\ ; \\
\neg\varphi = (\neg\psi_1) \wedge (\neg\psi_2) & \text{si } \varphi = \psi_1 \vee \psi_2\ ; \\
\neg\varphi = (\neg\psi_1) \vee (\neg\psi_2) & \text{si } \varphi = \psi_1 \wedge \psi_2\ ; \\
\neg\varphi = \psi_1 \wedge (\neg\psi_2) & \text{si } \varphi = \psi_1 \Rightarrow \psi_2\ ; \\
\neg\varphi = \exists x.\neg\psi & \text{si } \varphi = \forall x.\psi\ ; \\
\neg\varphi = \forall x.\neg\psi & \text{si } \varphi = \exists x.\psi\ .
\end{array}$$

Nous noterons LK1 le calcul des séquents à un seul côté contenant les inférences de la figure 2.12. LK1 est correct et complet par rapport à la logique classique : tout séquent $\vdash \Delta$ classiquement valide est dérivable dans LK1 et inversement.

2.4.2 Logique intuitionniste

Dans cette section, nous allons nous intéresser aux systèmes NJ et LJ. Ces deux systèmes définissent une notion de prouvabilité distincte de la celle de NK et LK : pour s'en rendre compte, il suffit de considérer une formule $P \vee \neg P$ où P est un prédicat.

Propriété 2.4.7. *Si P est un prédicat, il n'existe de dérivation de $\vdash P \vee \neg P$ ni dans NJ, ni dans LJ.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(\varphi \vee \neg\varphi), \varphi \vdash \neg(\varphi \vee \neg\varphi)}\,\text{Axiom}}{\vdash \neg\neg(\varphi \vee \neg\varphi)}\,\neg\text{Elim}}{\vdash \varphi \vee \neg\varphi}\,\neg\neg\text{Elim}}{\neg(\varphi \vee \neg\varphi) \vdash \bot}\,\neg\text{Intro}}{\neg(\varphi \vee \neg\varphi) \vdash \neg\varphi}\,\vee\text{Intro}_2}{\neg(\varphi \vee \neg\varphi) \vdash (\varphi \vee \neg\varphi)}\,\vee\text{Intro}_1 \qquad \dfrac{\dfrac{}{\neg(\varphi \vee \neg\varphi), \varphi \vdash \varphi}\,\text{Axiom}}{\neg(\varphi \vee \neg\varphi), \varphi \vdash \varphi \vee \neg\varphi}\,\vee\text{Intro}_1}{\neg(\varphi \vee \neg\varphi), \varphi \vdash \bot}\,\neg\text{Intro}$$

FIG. 2.13 – Tiers exclus dans NK

Démonstration. S'il existait une dérivation du séquent $\vdash P \vee \neg P$ dans NJ ou LJ, il en existerait une sans coupure par les propriétés 2.2.3 et 2.3.4. Celle-ci comporterait obligatoirement à sa racine une introduction (à droite) du $\vee$ par les propriétés 2.2.6 ou 2.3.8. Par conséquent il existerait une dérivation sans coupure de $\vdash P$ ou de $\vdash \neg P$. Or il ne peut exister de dérivation de $\vdash P$ sans coupure ni dans NJ ni dans LJ : aucune règle ne s'applique au séquent $\vdash P$. Il ne peut pas non plus exister de dérivation sans coupure de $\vdash \neg P$. En effet une simple induction sur les dérivations sans coupures de NJ ou LJ démontre que les séquents dérivables sans coupures ne sont pas de la forme $P, \ldots, P \vdash \bot$. Dans ce cas s'il existait une dérivation sans coupure de $\vdash \neg P$, elle comporterait à sa racine une introduction du $\neg$ et par conséquent inclurais une dérivation sans coupure de $P \vdash \bot$, ce qui est impossible. □

Au contraire, tout séquent de la forme $\vdash \varphi \vee \neg\varphi$ est dérivable dans NK et LK, comme le démontrent la dérivation écrite en figure 2.13 ainsi que la dérivation suivante.

$$\dfrac{\dfrac{\dfrac{}{\varphi \vdash \varphi}\,\text{Axiom}}{\vdash \varphi, \neg\varphi}\,\neg\text{R}}{\vdash \varphi \vee \neg\varphi}\,\vee\text{R}$$

L'objectif de cette section est de démontrer que les systèmes NJ et LJ définissent, à l'instar de NK et LK, la même notion de prouvabilité. Nous dirons alors que NJ et LJ sont des systèmes de preuve pour la logique intuitionniste. Tout comme pour la logique classique, nous allons utiliser une notion de modèle qui nous permettra de faire le lien entre déduction naturelle et calcul des séquents. Toutefois cette notion diffère de celle utilisée dans le cadre classique : remarquons en effet que toute formule $\varphi \vee \neg\varphi$ est valide dans tout modèle classique. Les modèles classiques ne sont donc pas en adéquation avec NJ et LJ. C'est bien naturel puisqu'ils sont en adéquation avec NK et LK qui diffèrent de NJ et LJ précisément pour la prouvabilité des formules de la forme $\varphi \vee \neg\varphi$. Nous allons donc utiliser une autre notion de modèle propre à la logique intuitionniste et que l'on doit à Kripke.

$$
\begin{array}{lll}
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \texttt{true} & \text{si } \varphi = \top \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \texttt{false} & \text{si } \varphi = \bot \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = (\!|P|\!)_{\alpha}([\![t_1]\!]^{\mu}, \ldots, [\![f_n]\!]^{\mu}) & \text{si } \varphi = P(t_1, \ldots, t_n) \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \max_{\alpha \leqslant \beta}(\langle\!|\psi|\!\rangle^{\mu}_{\alpha}) + 1 & \text{si } \varphi = \neg\psi \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \langle\!|\psi_1|\!\rangle^{\mu}_{\alpha} \mathbin{\widetilde{\wedge}} \langle\!|\psi_2|\!\rangle^{\mu}_{\alpha} & \text{si } \varphi = \psi_1 \wedge \psi_2 \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \langle\!|\psi_2|\!\rangle^{\mu}_{\alpha} \mathbin{\widetilde{\vee}} \langle\!|\psi_2|\!\rangle^{\mu}_{\alpha} & \text{si } \varphi = \psi_1 \vee \psi_2 \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \min_{\alpha \leqslant \beta}(\langle\!|\psi_1|\!\rangle^{\mu}_{\beta} \mathbin{\widetilde{\Rightarrow}} \langle\!|\psi_2|\!\rangle^{\mu}_{\beta}) & \text{si } \varphi = \psi_1 \Rightarrow \psi_2 \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \min_{\alpha \leqslant \beta,\ b \in \mathcal{D}_{\beta}} (\langle\!|\psi|\!\rangle^{\mu[x \mapsto b]}_{\beta}) & \text{si } \varphi = \forall x.\psi \\
\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} & = \max_{a \in \mathcal{D}_{\alpha}} (\langle\!|\psi|\!\rangle^{\mu[x \mapsto a]}_{\alpha}) & \text{si } \varphi = \exists x.\psi
\end{array}
$$

FIG. 2.14 – Interprétation dans un monde d'un modèle de Kripke

Définition 2.4.6 (Inclusion de modèles classiques). *Soit $\mathcal{M}_1 = (\mathcal{D}_1, [\![\_]\!]_1, (\!|\_|\!)_1)$ et $\mathcal{M}_2 = (\mathcal{D}_2, [\![\_]\!]_2, (\!|\_|\!)_2)$ deux modèles classiques. $\mathcal{M}_1$ est inclus dans $\mathcal{M}_2$, noté $\mathcal{M}_1 \Subset \mathcal{M}_2$ lorsque*

- *$\mathcal{D}_1 \subseteq \mathcal{D}_2$;*
- *pour tout symbole de fonction f d'arité n, $[\![f]\!]_1$ et $[\![f]\!]_2$ sont égales sur $\mathcal{D}_1^n$;*
- *pour tout symbole de prédicat P d'arité n, $[\![P]\!]_1$ est inférieure à $[\![P]\!]_2$ sur $\mathcal{D}_1^n$ (point à point et au sens $\texttt{false} < \texttt{true}$);*

Définition 2.4.7 (Modèle de Kripke). *Un modèle de Kripke est une paire $(K, \leqslant)$ où K est un ensemble de modèles classiques et $\leqslant$ est un préordre sur K inclus dans $\Subset$. Pour $\alpha \in K$, nous notons $\mathcal{D}_\alpha$, $[\![\_]\!]_\alpha$ et $(\!|\_|\!)_\alpha$ le domaine, la fonction d'interprétation des symboles de fonction et la fonction d'interprétation des symboles de prédicat de α. Si $\mu : X \to D_\alpha$ et t est un terme du premier ordre, alors le terme $[\![t]\!]^{\mu}_{\alpha}$ est défini comme en sous-section 2.4.1. Pour $\alpha \in K$ et $\mu : X \to D_\alpha$, l'interprétation d'une formule φ dans α du modèle de Kripke $(K, \leqslant)$, noté $\langle\!|\varphi|\!\rangle^{\mu}_{\alpha}$ est défini en figure 2.14. (Tout comme en sous-section 2.4.1, la notation $\mu[x \mapsto a]$ représente la fonction μ' définie par $\mu'(y) = a$ si $y = x$; $\mu'(y) = \mu(y)$ sinon.)*

Notation 2.4.3 (Réalisation d'une formule et d'un séquent). *Si $\mathcal{K} = (K, \leqslant)$ est un modèle de Kripke, une formule φ est réalisée dans le monde α, noté $\alpha \Vdash \varphi$, si pour toute valuation $\mu : X \to \mathcal{D}_\alpha$, $\langle\!|\varphi|\!\rangle^{\mu}_{\alpha} = \texttt{true}$. La formule est réalisée dans le modèle $\mathcal{K}$, noté $\mathcal{K} \Vdash \varphi$, si $\alpha \Vdash \varphi$ pour tout monde $\alpha \in K$. Un séquent unaire $\Gamma \vdash \varphi$ est réalisé, noté $\Gamma \Vdash \varphi$, lorsque pour tout modèle de Kripke $\mathcal{K} = (K, \leqslant)$, si $\mathcal{K} \Vdash \psi$ pour toute formule $\psi \in \Gamma$, alors $\mathcal{K} \Vdash \varphi$. Nous dirons aussi qu'un tel séquent est intuitionnistiquement valide.*

Exemple 2.4.1 (Tiers exclus). *Si P est un prédicat, le séquent $\vdash P \vee \neg P$ n'est pas intuitionnistiquement valide (autrement dit $\nVdash P \vee \neg P$). Pour le démontrer, il suffit*

de considérer un modèle de Kripke $\mathcal{K}$ contenant (au moins) deux modèles classiques α et β tels que $\alpha \leqslant \beta$, $(\!|P|\!)_\alpha = \texttt{false}$ et $(\!|P|\!)_\beta = \texttt{true}$. Alors $\beta \Vdash P$. Cela implique que $\alpha \nVdash \neg P$. En outre $\alpha \nVdash P$. Par conséquent $\alpha \nVdash P \vee \neg P$ et donc $\mathcal{K} \nVdash P \vee \neg P$. Finalement $\nVdash P \vee \neg P$.

Exemple 2.4.2 (Lois de De Morgan). *Les modèles de Kripke ne vérifient pas les lois de De Morgan. Si P et Q sont deux prédicats distincts, on peut par exemple construire un modèle de Kripke réalisant $\neg(P \wedge Q)$ sans réaliser $\neg P \vee \neg Q$. Considérons en effet un modèle de Kripke contenant trois modèles classiques α, β et γ tels que $\alpha \leqslant \beta$ et $\alpha \leqslant \gamma$ et tels que*

$$(\!|P|\!)_\beta = \texttt{true} \quad \textit{et} \quad (\!|P|\!)_\beta = \texttt{false}\,, \qquad (\!|P|\!)_\gamma = \texttt{false} \quad \textit{et} \quad (\!|P|\!)_\gamma = \texttt{true}$$
$$\textit{et} \quad (\!|P|\!)_\alpha = (\!|Q|\!)_\alpha = \texttt{false}\,.$$

Notons que $\alpha \Vdash \neg(P \wedge Q)$ et $\alpha \nVdash \neg P \wedge \neg Q$.

Nous allons à présent démontrer l'équivalence entre NJ et LJ en démontrant que la prouvabilité dans chacun de ces deux systèmes déductifs est équivalente à la notion de réalisation définie par les modèles de Kripke. Prouvons tout d'abord la correction de NJ et LJ par rapport à cette notion de réalisation.

Propriété 2.4.8 (Correction de NJ et LJ). *Si $\Gamma \vdash \varphi$ est un séquent dérivable dans NJ ou LJ, alors $\Gamma \Vdash \varphi$.*

Démonstration. Par inspection des inférences de NJ et LJ. □

La contraposée est plus difficile à démontrer. Introduisons d'abord un peu de terminologie. Rappelons qu'une *théorie* est un ensemble de formules et qu'une théorie $\mathcal{T}h$ est *consistante* respectivement dans NJ ou LJ s'il n'existe pas de sous-ensemble fini $\Gamma \subseteq \mathcal{T}h$ tel que $\Gamma \vdash \bot$ soit dérivable respectivement dans NJ ou LJ (définition 2.4.4).

Définition 2.4.8 (Théorie saturée). *Soit $\mathcal{C}$ un ensemble de constantes du premier ordre. Une théorie $\mathcal{T}h$ est $\mathcal{C}$-saturée (respectivement dans NJ ou LJ) si elle est consistante et que*

- *pour toutes formules φ et ψ, si $\mathcal{T}h \vdash \varphi \vee \psi$ est dérivable (respectivement dans NJ ou LJ), alors $\varphi \in \mathcal{T}h$ ou $\psi \in \mathcal{T}h$;*
- *pour toute formule φ, si $\mathcal{T}h \vdash \exists x.\varphi$ est dérivable (respectivement dans NJ ou LJ), alors il existe une constante $c \in \mathcal{C}$ telle que $\varphi[c/x] \in \mathcal{T}h$.*

Propriété 2.4.9 (Lemme de Henkin pour NJ). *Soit Γ un contexte et φ une formule tels que $\Gamma \vdash \varphi$ n'est pas démontrable dans NJ. Soit $\mathcal{C}$ un ensemble infini dénombrable de constantes n'apparaissant ni dans Γ ni dans φ. Alors il existe une théorie $\mathcal{T}h$ qui est $\mathcal{C}$-saturée dans NJ et telle que $\Gamma \subseteq \mathcal{T}h$ et $\mathcal{T}h \vdash \varphi$ n'est pas démontrable dans NJ.*

Démonstration. Les formules de la forme $\psi_1 \vee \psi_2$ ou $\exists x.\psi$ sont dénombrables. Nous supposons donc disposer d'une énumération de ces formules. Construisons à présent

par induction une suite de théories $(\mathcal{T}h_n)_{n\in\mathbb{N}}$. Simultanément à sa construction, nous montrons que cette suite vérifie que pour tout entier n, $\mathcal{T}h_n \vdash \varphi$ n'est pas démontrable dans NJ. Cette suite est définie comme suit. Tout d'abord $\mathcal{T}h_0 = \Gamma$. Effectivement, $\Gamma \vdash \varphi$ n'est pas dérivable dans NJ. Ensuite si nous avons déjà construit $\mathcal{T}h_n$, pour la construction de $\mathcal{T}h_{n+1}$, nous considérons la première formule ψ de notre énumération telle que $\mathcal{T}h_n \vdash \psi$ soit dérivable dans NJ.

cas 1 Si ψ est de la forme $\psi_1 \vee \psi_2$ et si $\mathcal{T}h_n, \psi_1 \vdash \varphi$ n'est pas dérivable dans NJ, on pose $\mathcal{T}h_{n+1} = \mathcal{T}h_n \cup \{\psi_1\}$. Par construction, $\mathcal{T}h_{n+1} \vdash \varphi$ n'est pas dérivable dans NJ.

cas 1 Si ψ est de la forme $\psi_1 \vee \psi_2$ et $\mathcal{T}h_n, \psi_1 \vdash \varphi$ est dérivable dans NJ, on pose $\mathcal{T}h_{n+1} = \mathcal{T}h_n \cup \{\psi_2\}$. Supposons par l'absurde que $\mathcal{T}h_{n+1} \vdash \varphi$ (c'est-à-dire $\mathcal{T}h_n, \psi_2 \vdash \varphi$) soit dérivable dans NJ. Alors en utilisant la dérivation π de $\mathcal{T}h_n \vdash \psi$ (c'est-à-dire de $\mathcal{T}h_n \vdash \psi_1 \vee \psi_2$), la dérivation π' de $\mathcal{T}h_n, \psi_1 \vdash \varphi$ et la dérivation π'' de $\mathcal{T}h_n, \psi_2 \vdash \varphi$, on peut construire la dérivation suivante dans NJ.

$$\vee\textsc{Elim}\ \frac{\begin{array}{ccc}\pi & \pi' & \pi'' \\ \mathcal{T}h_n \vdash \psi_1 \vee \psi_2 & \mathcal{T}h_n, \psi_1 \vdash \varphi & \mathcal{T}h_n, \psi_2 \vdash \varphi\end{array}}{\mathcal{T}h_n \vdash \varphi}$$

Puisqu'il n'existe pas de dérivation dans NJ de $\mathcal{T}h_n \vdash \varphi$, il n'en existe pas non plus de $\mathcal{T}h_{n+1} \vdash \varphi$.

cas 1 Si ψ est de la forme $\exists x.\psi'$, on choisit une constante $c \in \mathcal{C}$ qui ne figure pas dans $\mathcal{T}h_n \cup \{\varphi\}$ et $\mathcal{T}h_{n+1} = \mathcal{T}h_n \cup \{\psi'[c/x]\}$. Supposons par l'absurde qu'il existe une dérivation de $\mathcal{T}h_{n+1} \vdash \varphi$ (c'est-à-dire de $\mathcal{T}h_n, \psi'[c/x] \vdash \varphi$) dans NJ. Puisque c est une constante n'apparaissant ni dans $\mathcal{T}h_n$ ni dans φ cette dérivation peut être transformée en une dérivation π de $\mathcal{T}h_n, \psi' \vdash \varphi$. Alors en utilisant la dérivation π' de $\mathcal{T}h_n \vdash \psi$ (c'est-à-dire de $\mathcal{T}h_n \vdash \exists x.\psi'$), on peut construire la dérivation suivante dans NJ.

$$\exists\textsc{Elim}\ \frac{\begin{array}{cc}\pi' & \pi \\ \mathcal{T}h_n \vdash \exists x.\psi' & \mathcal{T}h_n, \psi' \vdash \varphi\end{array}}{\mathcal{T}h_n \vdash \varphi}$$

Puisqu'il n'existe pas de dérivation dans NJ de $\mathcal{T}h_n \vdash \varphi$, il n'en existe pas non plus de $\mathcal{T}h_{n+1} \vdash \varphi$.

Finalement nous posons $\mathcal{T}h = \bigcup_{n\in\mathbb{N}} \mathcal{T}h_n$. Alors $\mathcal{T}h \vdash \varphi$ n'est pas dérivable dans NJ, $\Gamma \subseteq \mathcal{T}h$ et $\mathcal{T}h$ est $\mathcal{C}$-saturé dans NJ (par construction). □

Nous pouvons à présent démontrer la complétude de NJ.

Propriété 2.4.10 (Complétude de NJ). *Si $\Gamma \Vdash \varphi$, alors il existe une dérivation de $\Gamma \vdash \varphi$ dans NJ.*

$$\textsc{Cut}\ \frac{\Gamma \vdash \psi_1 \vee \psi_2 \qquad \vee\textsc{L}\ \dfrac{\Gamma, \psi_1 \vee \psi_2 \vdash \Delta \qquad \Gamma, \psi_1 \vee \psi_2 \vdash \Delta}{\Gamma, \psi_1 \vee \psi_2 \vdash \Delta}}{\Gamma \vdash \varphi}$$

$$\textsc{Cut}\ \frac{\Gamma \vdash \exists x.\psi \qquad \exists\textsc{L}\ \dfrac{\Gamma, \psi \vdash \varphi}{\Gamma, \exists x.\psi \vdash \varphi}}{\Gamma \vdash \varphi}$$

FIG. 2.15 – Admissibilité de ∨ELIM et ∃ELIM dans LJ

Démonstration. Nous raisonnons par l'absurde. Supposons donc qu'il n'existe aucune dérivation de $\Gamma \vdash \varphi$ dans NJ. Montrons à présent que $\Gamma \nVdash \varphi$, c'est-à-dire qu'il existe un modèle réalisant toutes les formules de Γ sans réaliser φ. Pour cela, nous rajoutons au langage $\mathcal{F}_{\mathcal{A}T_\alpha(X)}$ des formules une suite $(\mathcal{C}_n)_{n\in\mathbb{N}}$ d'ensembles infinis dénombrables de constantes tels que les $\mathcal{C}_i$ sont disjoints deux à deux. Par la propriété 2.4.9, il existe une théorie $\mathcal{T}h_0$ qui est $\mathcal{C}_0$-saturée et telle que $\Gamma \subseteq \mathcal{T}h_0$ et $\mathcal{T}h_0 \vdash \varphi$ n'est pas dérivable dans NJ. Nous notons $\mathcal{C}_n^*$ l'ensemble de constantes $\bigcup_{1\leqslant i\leqslant n} \mathcal{C}_i$. On définit le modèle de Kripke $\mathcal{K} = (K, \leqslant)$ comme suit. Notons T l'ensemble $\{\mathcal{T}h / \mathcal{T}h_0 \subseteq \mathcal{T}h$ et il existe n tel que $\mathcal{T}h$ est $\mathcal{C}_n^*$-saturé$\}$. Si $\mathcal{T}h \in T$, alors le modèle $\mathcal{M}_{\mathcal{T}h} = (\mathcal{D}_{\mathcal{T}h}, [\![\_]\!]_{\mathcal{T}h}, (\!|\_|\!)_{\mathcal{T}h})$ est défini par

- $\mathcal{D}_{\mathcal{T}h}$ est l'ensemble des termes clos de $T_{\alpha\cup\mathcal{C}_n^*}(X)$ où n est le plus petit entier tel que $\mathcal{T}h$ est $\mathcal{C}_n^*$-saturé.
- Si f est un symbole de fonction d'arité n, $[\![f]\!]_{\mathcal{T}h}$ est la fonction

$$\begin{array}{rccc} [\![f]\!]_{\mathcal{T}h} : & \mathcal{D}_{\mathcal{T}h}^n & \to & \mathcal{D}_{\mathcal{T}h} \\ & (t_1, \ldots, t_n) & \mapsto & f(t_1, \ldots, t_n) \end{array}$$

- Si P est un symbole de prédicat d'arité n, $(\!|P|\!)_{\mathcal{T}h}$ est la fonction de $\mathcal{D}_{\mathcal{T}h}^n$ vers $\mathbb{B}$ telle que $(\!|P|\!)_{\mathcal{T}h}(t_1, \ldots, t_n) = \texttt{true}$ si et seulement si $P(t_1, \ldots, t_n) \in \mathcal{T}h$.

Alors $\mathcal{K}$ est défini par $K = \{\mathcal{M}_{\mathcal{T}h} / \mathcal{T}h \in T\}$ et $\mathcal{M}_{\mathcal{T}h_1} \leqslant \mathcal{M}_{\mathcal{T}h_2}$ si et seulement si $\mathcal{T}h_1 \subseteq \mathcal{T}h_2$. On peut alors vérifier que $\mathcal{K}$ est un modèle de Kripke réalisant toutes les formules de Γ sans réaliser la formule φ (en particulier $\mathcal{T}h_0 \nVdash \varphi$). □

Nous venons de traiter le cas de NJ. Le cas de LJ se règle similairement. La grande différence réside dans l'adaptation de la preuve du lemme de Henkin au calcul des séquents. En effet cette preuve utilise les schémas d'inférence ∨ELIM et ∃ELIM de NJ. Ces schémas sont néanmoins admissibles dans LJ comme le démontre les dérivations en figure 2.15. Nous en déduisons que LJ est lui aussi complet par rapport aux modèles de Kripke.

Propriété 2.4.11 (Complétude de LJ). *Si $\Gamma \Vdash \varphi$, alors $\Gamma \vdash \varphi$ est dérivable dans LJ.*

En corollaire, on obtient l'équivalence de NJ et LJ.

Propriété 2.4.12 (Équivalence de NJ et LJ). *Tout séquent est prouvable dans NJ si et seulement si il est prouvable dans LJ.*

Démonstration. Les propriétés 2.4.8, 2.4.10 et 2.4.11 montrent que pour tout séquent $\Gamma \vdash \varphi$,

$$\Gamma \vdash \varphi \text{ dérivable dans NJ} \quad \Leftrightarrow \quad \Gamma \Vdash \varphi \quad \Leftrightarrow \quad \Gamma \vdash \varphi \text{ dérivable dans LJ .} \quad \square$$

Nous déduisons aussi de l'exemple 2.4.1 que l'ensemble des séquents dérivables en logique intuitionniste est *strictement* inclus dans l'ensemble des séquents dérivables en logique classique.

Propriété 2.4.13.

$$\begin{array}{ccc} \Gamma \vdash \varphi \textit{ dérivable dans NJ} & \begin{array}{c}\Rightarrow\\ \not\Leftarrow\end{array} & \Gamma \vdash \varphi \textit{ dérivable dans NK} \\ \Updownarrow & & \Updownarrow \\ \Gamma \vdash \varphi \textit{ dérivable dans LJ} & \begin{array}{c}\Rightarrow\\ \not\Leftarrow\end{array} & \Gamma \vdash \varphi \textit{ dérivable dans LK} \end{array}$$

Démonstration. Les deux équivalences sont démontrées respectivement par les propriétés 2.4.12 et 2.4.6. L'implication de la logique intuitionniste vers la logique classique provient de l'inclusion (par définition) de NJ dans NK. Enfin la non-implication de la logique classique vers la logique intuitionniste provient de l'exemple 2.4.1 : si P est un prédicat, le séquent $\vdash P \vee \neg P$ est dérivable en logique classique, comme démontré par la dérivation en figure 2.13, mais n'est pas dérivable en logique intuitionniste, puisque $\nVdash P \vee \neg P$ (exemple 2.4.1) et par correction de NJ et LJ (propriété 2.4.8). $\square$

2.5 Recherche de preuves

Nous allons à présent nous intéresser à deux méthodes algorithmiques permettant d'infimer ou de confirmer la validité d'une formule. Comme la logique des prédicats n'est pas décidables, ces méthodes sont en fait des procédures de semi-décision. Nous allons tout d'abord présenter en sous-section 2.5.1 la méthode de résolution, puis en sous-section 2.5.2 la méthode analytique des tableaux. Nous les présentons comme des méthodes réfutationnelles qui permettent en fait de réfuter une formule, c'est-à-dire de montrer sa non-validité.

2.5.1 Résolution

La résolution (Robinson, 1965) est une méthode de démonstration automatique qui manipule **des littéraux**, c'est-à-dire des propositions atomiques ou des négations de propositions atomiques ; **des clauses**, c'est-à-dire des ensembles de littéraux représentant leur disjonction ; **des formes clausales**, c'est-à-dire des ensembles de clauses représentant leur conjonction. Nous utiliserons le symbole υ pour représenter des ensembles de formules (et en particulier des clauses) et le symboles Υ pour représenter des ensembles d'ensembles de formules (et en particulier des formes

$$
\begin{array}{lcl}
\Upsilon, \{\upsilon, \varphi \wedge \psi\} & \rightarrow & \Upsilon, \{\upsilon, \varphi\}, \{\upsilon, \psi\} \\
\Upsilon, \{\upsilon, \varphi \vee \psi\} & \rightarrow & \Upsilon, \{\upsilon, \varphi, \psi\} \\
\Upsilon, \{\upsilon, \varphi \Rightarrow \psi\} & \rightarrow & \Upsilon, \{\upsilon, \neg\varphi, \psi\} \\
\Upsilon, \{\upsilon, \top\} & \rightarrow & \Upsilon \\
\Upsilon, \{\upsilon, \bot\} & \rightarrow & \Upsilon, \{\upsilon\} \\
\Upsilon, \{\upsilon, \forall x.\varphi\} & \rightarrow & \Upsilon, \{\upsilon, \varphi\} \\
\Upsilon, \{\upsilon, \exists x.\varphi\} & \rightarrow & \Upsilon, \{\upsilon, \varphi[f(\bar{y})/x]\} \\
\multicolumn{3}{r}{\text{où } f \text{ est un symbole de fonction frais}} \\
\multicolumn{3}{r}{\text{et } \bar{y} = \mathcal{FV}(\exists x.\varphi)}
\end{array}
$$

$$
\begin{array}{lcl}
\Upsilon, \{\upsilon, \neg\neg\varphi\} & \rightarrow & \Upsilon, \{\upsilon\} \\
\Upsilon, \{\upsilon, \neg(\varphi \vee \psi)\} & \rightarrow & \Upsilon, \{\upsilon, \neg\varphi\}, \{\upsilon, \neg\psi\} \\
\Upsilon, \{\upsilon, \neg(\varphi \wedge \psi)\} & \rightarrow & \Upsilon, \{\upsilon, \neg\varphi, \neg\psi\} \\
\Upsilon, \{\upsilon, \neg(\varphi \Rightarrow \psi)\} & \rightarrow & \Upsilon, \{\upsilon, \varphi\}, \{\upsilon, \neg\psi\} \\
\Upsilon, \{\upsilon, \neg\top\} & \rightarrow & \Upsilon, \{\upsilon\} \\
\Upsilon, \{\upsilon, \neg\bot\} & \rightarrow & \Upsilon \\
\Upsilon, \{\upsilon, \neg(\exists x.\varphi)\} & \rightarrow & \Upsilon, \{\upsilon, \neg\varphi\} \\
\Upsilon, \{\upsilon, \neg(\forall x.\varphi)\} & \rightarrow & \Upsilon, \{\upsilon, \neg\varphi[f(\bar{y})/x]\} \\
\multicolumn{3}{r}{\text{où } f \text{ est un symbole de fonction frais}} \\
\multicolumn{3}{r}{\text{et } \bar{y} = \mathcal{FV}(\forall x.\varphi)}
\end{array}
$$

FIG. 2.16 – Mise en forme clausale

clausales). Si υ est un ensemble de formules et φ est une formule, nous noterons υ, φ l'ensemble $\upsilon \cup \{\varphi\}$. De même si Υ est un ensemble d'ensembles de formules et υ est un ensemble de formules, nous noterons Υ, υ l'ensemble $\Upsilon \cup \{\upsilon\}$. Toute forme clausale représente une formule. À titre d'exemple, si a, b et c sont des littéraux, la forme clausale $\{\{a, b\}, \{b, c\}\}$ représente en fait la formule $(a \vee b) \wedge (b \vee c)$. Inversement, toute formule propositionnelle peut être représentée par une forme clausale qui lui est équivalente en logique classique. Dans le cas de la logique des prédicats et pour traiter des quantificateurs, le procédé de *skolemisation* permet d'obtenir une forme clausale dont la satisfaisabilité est équivalente à la satisfaisabilité de la formule d'origine. Cette mise en forme clausale d'une formule est définie par le système de réécriture décrit par la figure 2.16. Ce système est convergent. Tout d'abord une analyse de ses paires critiques montre sa confluence. En outre, chaque étape de réécriture sur Υ fait décroître le multi-ensemble contenant pour chaque υ dans Υ la paire (a, b) où a est le nombre d'occurrences des symboles $\wedge, \vee, \Rightarrow, \forall, \exists$ et b est le nombre d'occurrences du symbole $\neg$ dans υ. Le système est donc aussi fortement normalisant. Si Υ est un ensemble d'ensembles de proprositions, nous noterons donc $\mathsf{clF}(\Upsilon)$ l'unique forme normale de Υ par ce système de réécriture. Cette forme nor-

male est bel est bien une forme clausale. La règle de résolution est la suivante.

$$\textsc{Resolution}\quad \frac{\{P_1, \ldots, P_n, Q_1, \ldots, Q_m\} \qquad \{\neg R_1, \ldots, \neg R_p, S_1, \ldots, S_q\}}{\{Q_1\sigma, \ldots, Q_m\sigma, S_1\sigma, \ldots, S_q\sigma\}}$$

où σ est l'unificateur plus général de $P_1, \ldots, P_n, R_1, \ldots, R_p$. Elle permet de combiner deux clauses (les deux prémisses) en une troisième (la conclusion). Ainsi quand une forme clausale contient deux clauses identifiables aux prémisses de cette règle, une nouvelle clause correspondant à la conclusion de la règle est rajoutée enrichissant ainsi la forme clausale de départ. La règle de résolution s'utilise comme une méthode réfutationnelle de la façon suivante. Le point de départ est une formule mise en forme clausale grâce au système de réécriture de la figure 2.16. On itère alors l'application de la règle de résolution afin de générer le maximum de nouvelles clauses, enrichissant ainsi la forme clausale sur laquelle on travaille. Si l'on obtient une forme clausale contenant la clause vide (c'est-à-dire un ensemble vide de formules), nous obtenons une réfutation de la formule de départ : celle-ci est insatisfaisable, c'est-à-dire que sa négation est classiquement valide. C'est la *correction* de cette méthode de résolution. Inversement si une formule est insatisfaisable, il est possible de générer la clause vide en partant de cette formule. C'est la *complétude réfutationnelle* de cette méthode de résolution. En particulier si l'on obtient une forme clausale ne contenant pas la clause vide, mais à laquelle la résolution ne peut plus rajouter de clauses, c'est que la formule de départ est satisfaisable, autrement dit sa négation est classiquement non-valide. Il peut aussi arriver que l'on ne puisse pas générer la clause vide ni une clause non-vide sur laquelle la résolution ne s'applique plus. Dans ce cas, bien que la formule soit satisfaisable, la méthode de résolution ne permet pas de le montrer puisqu'elle ne termine pas. Cette méthode permet donc de semi-decider l'insatisfaisabilité d'une formule.

2.5.2 Méthode analytique des tableaux

Nous allons à présent exposer la méthode analytique des tableaux (Smullyan, 1968). C'est aussi une méthode réfutationnelle cette fois basée sur le calcul des séquents plutôt que sur la mise en forme clausale et la seule règle de résolution. Pour réfuter une formule φ, l'idée est d'utiliser les règles d'introduction à gauche du calcul des séquents sur le séquent initial $\varphi \vdash$. On manipule par conséquent des séquents dont la partie droite est vide et que l'on appelle *branche*. La négation en tant que connecteur syntaxique ne s'applique que sur les propositions atomiques. Elle est définie pour les formules composées par la dualité de De Morgan. Une branche, à l'instar d'une clause, est un multi-ensemble de formules. Toutefois à l'inverse des clauses, une branche représente une conjonction. Un tableau, à l'instar d'une forme clausale, est un multi-ensemble de branches. Néanmoins, au lieu de représenter une conjonction de disjonctions, un tableau représente une disjonction de conjonctions. Par exemple si φ_1, φ_2 et φ_3 sont des formules, le tableau $\{\{\varphi_1, \varphi_2\}, \{\varphi_1, \varphi_3\}\}$ représente la formule $(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$. Nous utiliserons le symbole $\mathcal{T}$ pour

$$
\begin{array}{rcll}
\mathcal{B}, \varphi_1 \vee \varphi_2 \mid \mathcal{T} & \rightarrow & \mathcal{B}, \varphi_1 \mid \mathcal{B}, \varphi_2 \mid \mathcal{T} & \\
\mathcal{B}, \varphi_1 \wedge \varphi_2 \mid \mathcal{T} & \rightarrow & \mathcal{B}, \varphi_1, \varphi_2 \mid \mathcal{T} & \\
\mathcal{B}, \varphi_1 \Rightarrow \varphi_2 \mid \mathcal{T} & \rightarrow & \mathcal{B}, \neg\varphi_1 \mid \mathcal{B}, \varphi_2 \mid \mathcal{T} & \\
\mathcal{B}, \bot \mid \mathcal{T} & \rightarrow & \mathcal{T} & \\
\mathcal{B}, \top \mid \mathcal{T} & \rightarrow & \mathcal{B} \mid \mathcal{T} & \\
\mathcal{B}, \forall x.\varphi \mid \mathcal{T} & \rightarrow & \mathcal{B}, \forall x.\varphi, \varphi[t/x] \mid \mathcal{T} & \text{où } t \text{ est un terme clos} \\
\mathcal{B}, \exists x.\varphi \mid \mathcal{T} & \rightarrow & \mathcal{B}, \varphi[c/x] \mid \mathcal{T} & \text{où } c \text{ est une constante fraîche} \\
\mathcal{B}, P, \neg P \mid \mathcal{T} & \rightarrow & \mathcal{T} & (\textit{branch closure})
\end{array}
$$

FIG. 2.17 – Une méthode analytique des tableaux

représenter des tableaux et le symbole $\mathcal{B}$ pour représenter des branches. Si $\mathcal{T}$ est un tableau et $\mathcal{B}$ est une branche, nous noterons $\mathcal{T} \mid \mathcal{B}$ ou $\mathcal{B} \mid \mathcal{T}$ le tableau $\mathcal{T} \uplus \{\mathcal{B}\}$. Si $\mathcal{B}$ est une branche et φ est une formule, nous noterons $\mathcal{B}, \varphi$ la branche $\mathcal{B} \uplus \{\varphi\}$. La figure 2.17 décrit une méthode des tableaux traduisant directement les inférences de LK.

Tout comme la résolution, la méthode des tableaux est correcte ainsi que refutationnellement complète. *La correction* indique que si le tableau $\{\{\varphi\}\}$ se réécrit en $\emptyset$ (le tableau vide) par la relation de réécriture générée par les règles de la figure 2.17, alors la formule φ est insatisfaisable. *La complétude (réfutationnelle)* indique inversement que si une formule φ est insatisfaisable, alors le tableau $\{\{\varphi\}\}$ se réécrit en $\emptyset$ par cette même relation de réécriture.

Les règles traitant des quantificateurs peuvent être modifiées de manières à restreindre le non-déterminisme de leur utilisation. Une grande partie de ce non-déterminisme provient de l'étape d'instanciation contenue par la règle traitant du quantificateur universel : il s'agit de *choisir* un terme (du premier ordre) clos t. Il est possible de retarder ce choix en remplaçant ce terme par une variable en attente d'instanciation. On obtient la règle

$$\mathcal{B}, \forall x.\varphi \mid \mathcal{T} \quad \rightarrow \quad \mathcal{B}, \forall x.\varphi, \varphi[y/x] \mid \mathcal{T} \qquad \text{où } y \text{ est une variable fraîche}$$

L'instanciation est ainsi remise à plus tard. Notons que la décomposition des quantificateurs existentiels dépend néanmoins de la décomposition des quantificateurs universels : cette dépendance s'exprime par la condition *où c est une constante fraîche*. En effet une telle constante doit être fraîche par rapport à l'instanciation que l'on fera *plus tard* des variables introduites en décomposant des quantificateurs universels. Cette condition de fraîcheur est donc difficilement vérifiable. Une solution inspirée de la skolémisation consiste à modifier aussi la règle traitant des quantificateurs existentiels. Au lieu d'introduire une constante fraîche globalement, on introduit une fonction de Skolem fraîche dépendant des variables libres de la branche (c'est-à-dire des variables utilisées pour décomposer les quantificateurs universels et en attente d'instanciation). La fraîcheur est alors garantie quelque soit l'instanciation que l'on

fera des variables.

$$\mathcal{B}, \exists x.\varphi \mid \mathcal{T} \quad \rightarrow \quad \mathcal{B}, \varphi[f(y_1, \ldots, y_n)/x] \mid \mathcal{T}$$

où f est un symbole de fonction de Skolem frais pour tout le tableau et $y_1, \ldots y_n$ sont les variables libres dans la branche. Enfin il faut modifier la règle de *branch closure* pour qu'elle inclue le mécanisme d'instanciation que nous avons retardé. Cette règle devient

$$\mathcal{B}, Q, \neg P \mid \mathcal{T} \quad \rightarrow \quad \mathcal{T}$$

où P est Q sont deux propositions atomiques *unifiables*, c'est-à-dire qu'il existe une substitution σ telle que $P\sigma = Q\sigma$. Si l'on utilise à la fois le traitement modifié des quantificateurs avec instanciations retardées et skolemisation ainsi que la *branch closure* avec unification, on obtient une méthode des tableaux qui reste correcte et réfutationnellement complète. Il est possible de restreindre encore plus les règles traitant des quantificateurs : Hähnle et Schmitt (1994) proposent par exemple de restreindre les variables libres arguments des symboles de fonction des Skolem aux variables libres *de la formule que l'on décompose*, et non pas de toute la branche correspondant. Avec Beckert, ils proposent ensuite d'utiliser un seul symbole de Skolem frais pour toutes les décompositions d'une même formule $\exists x.\varphi$ dans la totalité du tableau (Beckert *et al.*, 1993).

Tout comme la résolution, la méthode des tableaux fournit une procédure de semi-décision pour l'insatisfaisabilité d'une formule. Si la méthode retourne le tableau vide, la formule est insatisfaisable. Si la méthode s'arrête sur un tableau non-vide, la formule est satisfaisable. Enfin si la méthode ne s'arrête pas, on ne peut conclure bien que la formule soit en fait satisfaisable.

Chapitre 3

Systèmes de typage, normalisation et calcul

La correspondance de Curry-De Bruijn-Howard est au cœur de notre travail. Elle relie calcul (chapitre 1) et logique (chapitre 2) et ce chapitre lui est dédié. Le point de départ est l'interprétation de Brouwer-Heyting-Kolmogorov proposées par Brouwer, Heyting et indépendamment par Kolmogorov. Cette interprétation propose non pas de voir les preuves comme des transcriptions syntaxiques mais comme des objets mathématiques, plus précisément les objets qui démontrent, réalisent ou témoignent de la vérité d'un jugement.

1. Une preuve de $\varphi \wedge \psi$ est un couple (π, ρ) tels que π est une preuve de φ et ρ est une preuve de ψ.
2. Une preuve de $\varphi \vee \psi$ est un couple (n, π) avec $n \in \mathbb{Z}/2\mathbb{Z}$ tel que si $n = 0$, π est une preuve de φ et si $n = 1$, π est une preuve de ψ.
3. Une preuve de $\exists x.\varphi(x)$ est un couple (t, π) tel que π est une preuve de $\varphi(t)$.
4. Une preuve de $\varphi \Rightarrow \psi$ est un calcul qui transforme toute preuve de φ en une preuve de ψ.
5. Une preuve de $\forall x.\varphi(x)$ est un calcul qui transforme tout élément t du domaine en une preuve de $\varphi(t)$.

Nous avons vu dans la section 1.2 que le lambda-calcul permet de définir efficacement la notion de fonction. Utiliser ce paradigme dans la partie 4 de l'interprétation de Brouwer-Heyting-Kolmogorov est le coeur de la correspondance de Curry-De Bruijn-Howard. L'histoire de cette correspondance remonte à des observations de Curry : les combinateurs de sa Logique Combinatoires correspondents aux schémas d'axiomes de la logique intuitionniste implicationnelle (Curry, 1934) mettant ainsi en correspondance son système et les systèmes à la Hilbert (Curry et Feys, 1958). Indépendamment le système Automath développé par De Bruijn dans les années 60 permettait de construire et de vérifier des preuves en utilisant les notations du lambda-calcul. Enfin en 1969, Howard proposa une interprétation de la déduction naturelle intuitionniste par une variante du lambda-calcul simplement typé. Nous

verrons que cette correspondance dépasse le simple cadre de la syntaxe. La bêta-réduction illustrée par la figure 1.2 est en correspondance avec l'élimination des coupures de la figure 2.4. On retrouve alors des résultats déjà énoncés, comme la propriété 2.2.4, mais on peut aussi prouver des résultats plus forts comme la normalisation *forte* de l'élimination des coupures en déduction naturelle.

Si la correspondance exprimée au départ par Howard concerne la déduction naturelle intuitionniste NJ, de nombreux travaux montrent qu'elle peut être étendu à la logique classiqe et au calcul des séquents (Griffin, 1990; Parigot, 1992; Curien et Herbelin, 2000; Urban, 2001; Urban et Bierman, 2001). L'intérêt que nous porterons au calcul des séquents classique LK aux chapitres 4, 6 et 7 nous amène naturellement à nous intéresser à ces approches.

3.1 Pour la déduction naturelle intuitionniste

Cette section présente la correspondance de Curry-De Bruijn-Howard entre le lambda-calcul simplement typé et la déduction naturelle intuitionniste LJ. Tout d'abord la sous-section 3.1.1 présente le lambda-calcul simplement typé et, puis la sous-section 3.1.2 présente la correspondance avec la déduction naturelle pour la logique intuitionniste minimale. Puis la sous-section 3.1.3 présente l'extension de cette correspondance au système NJ déjà présenté en section 2.2.

3.1.1 Lambda-calcul simplement typé

Un type est une catégorie d'informations qui ont une forme commune, comme par exemple la catégorie des nombres entiers ou des chaînes de caractères. L'intérêt des types est de définir des opérations qui leur sont spécifiques et qui n'ont de sens que lorsqu'elles sont appliquées à des objets du bon type. L'utilisation de cette même idée comme parade au contre-exemple de Russell mena ce dernier à la définition des bases de la théorie des types. Cette idée s'applique au cadre du lambda-calcul : certains lambda-termes représentent des objets d'un certain type et d'autres lambda-termes représenteront les opérations que l'on peut appliquer à ces objets. Soit donc un ensemble de types de base que nous appellerons des *types atomiques*. Alors l'ensemble des types simples est défini comme le langage hors-contexte associé à la grammaire hors-contexte sous forme de Backus-Naur suivante.

$$T, U \quad ::= \quad A \mid T \Rightarrow U$$

où le symbole A représente les types atomiques. Si T et U sont deux types simples, le type $T \Rightarrow U$ représente, lui, les opérations que l'on peut appliquer aux objets de type T pour obtenir comme résultat un objet de type U. Nous appellerons un tel type un *type fonctionnel*. Les objets de base du lambda-calcul étant les variables, on leur choisi un type de façon arbitraire en utilisant un *contexte* : un contexte de typage est un ensemble Γ de déclarations de typage de la forme « x est de type T ». Un tel jugement est d'ailleurs noté $x : T$. Nous n'étudierons que le cas où une variable ne

peut pas être déclarée comme ayant plusieurs types. Par conséquent, un contexte de typage n'est autorisé à contenir qu'une déclaration $x : T$ par symbole de variable x. Par abus de langage, nous identifierons $x : T$ et $\{x : T\}$. Nous écrirons aussi Γ, Γ' l'union disjointe des contextes Γ et Γ'. Enfin nous noterons $\Gamma \vdash t : T$ le jugement « le lambda-terme t est de type T dans le contexte Γ ». Alors la notion de typage, dans le cadre du lambda-calcul, se traduit par les principes suivants. Pour toute variable x, pour tous lambda-termes t et u, pour tout contexte Γ et pour tous types T et U,

Variable $\Gamma, x : T \vdash x : T$;

Abstraction si $\Gamma, x : T \vdash t : U$, alors $\Gamma \vdash \lambda x.t : T \Rightarrow U$;

Application si $\Gamma \vdash t : T$ et $\Gamma \vdash u : T \Rightarrow U$, alors $\Gamma \vdash t\,u : U$.

Ces trois principes peuvent être écrit sous la forme de schémas d'inférence

$$\textsc{Var}\ \frac{}{\Gamma, x : T \vdash x : T} \qquad \textsc{App}\ \frac{\Gamma, x : T \vdash t : U}{\Gamma \vdash \lambda x.t : T \Rightarrow U}$$

$$\textsc{Abs}\ \frac{\Gamma \vdash t : T \Rightarrow U \qquad \Gamma \vdash u : T}{\Gamma \vdash t\,u : U}$$

Alors un lambda-terme t est de type T dans le contexte Γ s'il existe une dérivation de $\Gamma \vdash t : T$. Nous dirons qu'un lambda-terme t est bien typé dans le contexte Γ s'il existe un type simple T tel que $\Gamma \vdash t : T$ est dérivable. Nous dirons qu'un lambda t est bien typé s'il existe un contexte Γ ainsi qu'un type T tels que $\Gamma \vdash t : T$ est dérivable.

Le typage des lambda-termes défini par les règles Var, Abs et App est dirigé par la syntaxe du lambda-terme t que l'on cherche à typer : si t est une variable, on ne pourra utiliser que la règle Var ; si t est une abstraction, on ne pourra utiliser que la règle Abs ; enfin si t est une application, on ne pourra utiliser que la règle App. Il existe donc une *bijection* entre les dérivations de typage et les jugements $\Gamma \vdash t : T$ dérivables. Ce typage des lambda-termes admet aussi les propriétés suivantes.

Propriété 3.1.1 (Substitution Lemma). *Si les jugements $\Gamma, x : T \vdash t : U$ et $\Gamma \vdash u : T$ sont dérivables, alors $\Gamma \vdash t[u/x] : U$ l'est aussi.*

Démonstration. Par induction sur le lambda-terme t. □

Propriété 3.1.2 (Subject Reduction). *Si t et u sont deux lambda-termes tels que $t \to_\beta u$ et si $\Gamma \vdash t : T$ est dérivable, alors $\Gamma \vdash u : T$ l'est aussi.*

Démonstration. Par induction sur t, le seul cas ou l'application triviale de l'hypothèse d'induction ne suffit pas étant le cas où $t \mapsto_\beta u$. Dans ce cas $t = (\lambda x.t_1)\,t_2$ et $u = t_1[t_2/x]$ et la dérivation de $\Gamma \vdash t : T$ est forcément de la forme

$$\textsc{App}\ \frac{\textsc{Var}\ \dfrac{\overset{(\pi_1)}{\Gamma, x : U \vdash t_1 : T}}{\Gamma \vdash \lambda x.t_1 : U \Rightarrow T} \qquad \overset{(\pi_2)}{\Gamma \vdash t_2 : U}}{\Gamma \vdash (\lambda x.t_1)\,t_2 : T}$$

Alors la propriété 3.1.1 appliquée aux dérivations π_1 et π_2 montre qu'il existe une dérivation de $\Gamma \vdash u : T$. □

Nous allons démontrer à présent que tout lambda-terme bien typé est fortement normalisant pour $\rightarrow_\beta$. La démonstration que nous allons développer ici se trouve dans l'ouvrage de Girard *et al.* (1989) et est basée sur une idée de Tait (1967). D'autres démonstrations sont possibles, comme par exemple celle exposée dans l'ouvrage de Terese (2003). Nous dirons à présent « fortement normalisant » au lieu de « fortement normalisant pour $\rightarrow_\beta$ » et « forme normale » au lieu de « forme normale pour $\rightarrow_\beta$ ». Remarquons tout d'abord que puisque $\rightarrow_\beta$ est à branchements finis, tout lambda-terme fortement normalisant admet un nombre fini de séquences de réduction (qui sont toutes finies) (propriété 1.1.4). Pour tout lambda-terme t fortement normalisant, la longueur maximale de ses séquences de réductions est notée $\nu(t)$. Remarquons que si t est fortement normalisant, alors tout sous-terme u de t l'est aussi et $\nu(u) \leqslant \nu(t)$. Notons aussi que si $t[u/x]$ est fortement normalisant, alors t l'est aussi et $\nu(t) \leqslant \nu(t[u/x])$. La preuve que nous présentons se base sur la définition de *candidats de réductibilité*.

Définition 3.1.1 (Termes réductibles). *Si T est un type simple, l'ensemble des termes réductibles de type T noté Red_T est défini par*

- *si T est atomique, Red_T est l'ensemble des termes fortement normalisants pour $\rightarrow_\beta$;*
- *si $T = U_1 \Rightarrow U_2$ alors Red_T est l'ensemble des lambda-termes t tels que pour tout $u \in \mathsf{Red}_{U_1}$, alors $t\,u \in \mathsf{Red}_{U_2}$.*

Nous dirons qu'un terme est *neutre* si c'est une variable ou une application. On démontre la propriété suivante.

Propriété 3.1.3. *Pour tout type T, Red_T est un candidat de réductibilité, c'est-à-dire un ensemble tel que pour tous lambda-termes t et u,*

(CR1) si $t \in \mathsf{Red}_T$, alors t est fortement normalisant ;

(CR2) si $t \in \mathsf{Red}_T$ et $t \rightarrow_\beta u$, alors $u \in \mathsf{Red}_T$;

(CR3) si t est neutre et tel que pour tout v tel que $t \rightarrow_\beta v$, $v \in \mathsf{Red}_T$, alors $t \in \mathsf{Red}_T$;

Remarquons que CR3 implique que si t est une forme normale neutre, alors $t \in \mathsf{Red}_T$.

Démonstration. La conjonction de CR1, CR2 et CR3 est prouvée par induction sur le type T.

Type Atomique Si T est un type atomique, alors par définition de Red_T et de la normalisation forte, CR1, CR2 et CR3 sont vérifiés.

Type Fonctionnel Si $T = U_1 \Rightarrow U_2$.

(CR1) Soit $t \in \mathsf{Red}_T$. Choisissons une variable x. Puisque x est neutre et par CR3 sur U_1, $x \in \mathsf{Red}_{U_1}$ et puisque $t \in \mathsf{Red}_T$, $t\,x \in \mathsf{Red}_{U_2}$. Par CR1 sur U_2, $t\,x \in \mathcal{SN}$ et donc $t \in \mathcal{SN}$.

(CR2) Soit $t \in \mathsf{Red}_T$ et u tel que $t \rightarrow_\beta u$. Soit $v \in \mathsf{Red}_{U_1}$. Par définition de

Red_T, $t\,v \in \mathsf{Red}_{U_2}$ et puisque $t\,v \to_\beta u\,v$, alors $u\,v \in \mathsf{Red}_{U_2}$ par CR2 sur U_2 (pour tout $v \in \mathsf{Red}_{U_1}$). D'où $u \in \mathsf{Red}_T$.

(CR3) Soit t neutre tel que si $t \to_\beta u$, alors $u \in \mathsf{Red}_T$. Soit $v \in \mathsf{Red}_{U_1}$. Par CR1 sur U_1, $v \in \mathcal{SN}$ et nous montrons alors par récurrence sur $\nu(v)$ que $t\,v \in \mathsf{Red}_{U_2}$. Puisque t est neutre, les réduits en un pas de $t\,v$ sont soit de la forme $t\,v'$ avec $v \to_\beta v'$, soit de la forme $t'\,v$ avec $t \to t'$. Dans le premier cas, $\nu(v') < \nu(v)$ et CR2 sur U_1 montre que $v' \in \mathsf{Red}_{U_1}$. D'où par hypothèse de récurrence $t\,v' \in \mathsf{Red}_{U_2}$. Dans le deuxième cas, puisque $t' \in \mathsf{Red}_T$ et par définition de Red_T, $t'\,v \in \mathsf{Red}_{U_2}$. Finalement tous les réduits du lambda-terme neutre $t\,v$ sont des éléments de Red_{U_2}. Par CR3 sur U_2, $t\,v \in \mathsf{Red}_{U_2}$ (pour tout $v \in \mathsf{Red}_{U_1}$) et donc nous en concluons que $t \in \mathsf{Red}_T$. □

On prouve la propriété suivante.

Propriété 3.1.4. *Supposons que pour tout $u \in \mathsf{Red}_U$, $v[u/x] \in \mathsf{Red}_T$. Alors $\lambda x.v \in \mathsf{Red}_{U\Rightarrow T}$.*

Démonstration. Il nous faut démontrer que $(\lambda x.v)\,u \in \mathsf{Red}_T$ pour tout $u \in \mathsf{Red}_U$. Nous le prouvons par une récurrence sur $\nu(v) + \nu(u)$. Le lambda-terme $(\lambda x.v)\,u$ se réduit en

- $v[u/x]$ qui par hypothèse est dans Red_T ;
- $(\lambda x.v')\,u$ avec $v \to_\beta v'$ et qui est donc par hypothèse de récurrence dans Red_T ;
- $(\lambda x.v)\,u'$ avec $u \to_\beta u'$ et qui est donc par hypothèse de récurrence dans Red_T .

Tous les réduits de $(\lambda x.v)\,u$ sont donc dans Red_T. Puisque ce terme est neutre et par CR3, il est lui-même dans Red_T. Enfin puisque $(\lambda x.v)\,u \in \mathsf{Red}_T$ pour tout $u \in \mathsf{Red}_U$, alors $\lambda x.v \in \mathsf{Red}_{U\Rightarrow T}$. □

Enfin on peut prouver la propriété suivante.

Propriété 3.1.5. *Tout lambda-terme t bien typé de type T appartient à Red_T.*

Démonstration. Par induction sur t en utilisant la propriété 3.1.4. □

Propriété 3.1.6 (Normalisation Forte). *Tout lambda-terme bien typé est fortement normalisant pour $\to_\beta$.*

Démonstration. Par la propriété 3.1.5 et CR1, tout lambda-terme bien typé est fortement normalisant. □

3.1.2 La correspondance

L'ensemble des types simples que nous avons défini en sous-section 3.1.1 est un sous-ensemble de l'ensemble des formules que nous avons défini au chapitre 2. Ce sont les formules construites à partir de propositions atomiques et de l'implication. Lorsqu'on efface les lambda-termes apparaissant dans un jugement de typage $S = x_1 :$

$\varphi_1 \dots x_n : \varphi_n \vdash t : \psi$, on obtient un séquent $\varphi_1 \dots \varphi_n \vdash \psi$. Nous noterons $\langle\!|\_|\!\rangle$ la fonction évidente d'effacement sur les contextes et jugements de typage étendue sur les inférences ainsi que sur les dérivations de typage. Cet effacement transforme les trois règles de typage VAR, ABS et APP respectivement en les règles AXIOM, ⇒INTRO et ⇒ELIM de NJ. Notons NM le système déductif contenant uniquement ces trois règles. NM est en correspondance avec le lambda-calcul simplement typé comme suit.

Propriété 3.1.7 (Correspondance de Curry-De Bruijn-Howard). *Toute instance des schémas* VAR, ABS *ou* APP *est respectivement transformée par* $\langle\!|\_|\!\rangle$ *en instance de* AXIOM, ⇒INTRO *ou* ⇒ELIM. *S'il existe une dérivation de typage* π *de* $\Gamma \vdash t : \varphi$, *alors* $\langle\!|\pi|\!\rangle$ *est une dérivation de* $\langle\!|\Gamma \vdash t : \varphi|\!\rangle$ *dans NM. Inversement s'il existe une dérivation* π *de* $\Gamma \vdash \varphi$ *dans NM, alors il existe un contexte de typage* Γ' *tel que* $\Gamma = \langle\!|\Gamma'|\!\rangle$, *un lambda-terme* t *ainsi qu'une dérivation de typage* π' *de* $\Gamma' \vdash t : \varphi$ *telle que* $\langle\!|\pi'|\!\rangle = \pi$.

Nous avons vu que le typage est dirigé par la syntaxe et que les règles VAR, ABS et APP correspondent aux trois constructions syntaxiques du lambda-calcul, c'est-à-dire la variable, l'abstraction et l'application. Alors les règles AXIOM, ⇒INTRO et ⇒ELIM correspondent au travers de la correspondance aux variables, abstraction et application du lambda-calcul. En particulier les bêta-redex bien typés sont exactement les coupures dans NM.

$$\begin{array}{l}\Rightarrow\textsc{Intro}\\ \Rightarrow\textsc{Elim}\end{array}\ \frac{\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \Rightarrow \psi} \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi} \qquad\qquad \begin{array}{l}\textsc{Abs}\\ \textsc{App}\end{array}\ \frac{\dfrac{\Gamma, x : \varphi \vdash t : \psi}{\Gamma \vdash \lambda x.t : \varphi \Rightarrow \psi} \qquad \Gamma \vdash u : \varphi}{\Gamma \vdash (\lambda x.t)\, u : \psi}$$

On peut alors transposer des propriétés du lambda-calcul simplement typés à la déduction naturelle.

La subject reduction (propriété 3.1.2) démontre que la transformation de la figure 2.4 transforme bien des dérivations de NM en dérivations de NM.

La normalisation forte (propriété 3.1.6) de la bêta-réduction dans le cadre bien typé (propriété 3.1.6) démontre que cette transformation est fortement normalisante sur les dérivations de NM. Nous n'avions pour l'instant démontré que sa normalisation *faible* (en section 2.2). Nous obtenons donc une preuve de la propriété 2.2.3 dans le cadre de NM.

Conversement des propriétés de la déduction naturelle se transposent au lambda-calcul simplement typé : par exemple la propriété 2.2.4 exprime le fait que si t est un lambda-terme en forme normale pour bêta et si t est de type φ dans le contexte vide, alors t est une abstraction.

3.1.3 Extension de la correspondance à la logique des prédicats

La correspondance que nous avons montrée en sous-section 3.1.2 se place dans le cadre de la logique minimale, c'est-à-dire pour des formules uniquement construites

à base d'implications. Elle peut néanmoins être étendue à tous les connecteurs de NJ. Tout comme pour le cas de l'implication, une règle d'introduction correspond à un constructeur et une règle d'élimination correspond à un destructeur. Par exemple pour l'implication, l'abstraction du lambda-calcul permet de *construire* une fonction, et l'application permet d'*utiliser* (ou détruire) une fonction. Pour ne plus confondre les termes et variables du premier ordre avec les lambda-termes et les variables du lambda-calcul, nous utiliserons les symboles x, y, z pour les variables du premier ordre ; t, m, n pour les termes du premier ordre ; x, y, z pour les variables du lambda-calcul et enfin π, ρ, τ pour les lambda-termes (étendus). L'ensemble des lambda-termes étendus est défini comme le langage hors-contexte associé à la grammaire en forme de Backus-Naur suivante.

$$
\begin{array}{rcll}
\pi, \rho, \tau & ::= & x & \text{variable} \\
& | & \lambda x.\pi \mid \pi\, \rho & \text{implication} \\
& | & \delta_{\bot}(\pi) & \text{faux} \\
& | & \lambda^{\neg} x.\pi \mid \delta_{\neg}(\pi, \rho) & \text{negation} \\
& | & (\pi, \rho) \mid \mathsf{fst}(\pi) \mid \mathsf{snd}(\pi) & \text{conjonction} \\
& | & i(\pi) \mid j(\pi) \mid \delta(\pi, x.\rho, y.\tau) & \text{disjonction} \\
& | & \lambda \mathsf{x}.\pi \mid \pi\, \mathsf{t} & \text{quantification universelle} \\
& | & \langle \mathsf{t}, \pi \rangle \mid \delta_{\exists}(\pi, x.\mathsf{x}.\rho) & \text{quantification existentielle}
\end{array}
$$

Les règles de typage sont décrites par la figure 3.1. Ce typage est dirigé par la syntaxe. On rajoute à l'alpha-conversion et à la bêta-réduction les règles de conversion qui correspondent aux réductions des figures 2.5 et 2.6.

$$
\begin{array}{rcl}
\delta_{\neg}(\lambda^{\neg} x.\pi, \rho) & \to & (\lambda x.\pi)\, \rho \\
\mathsf{fst}((\pi, \rho)) & \to & \pi \\
\mathsf{snd}((\pi, \rho)) & \to & \rho
\end{array}
\qquad
\begin{array}{rcl}
\delta(i(\pi), x.\rho, y.\tau) & \to & (\lambda x.\rho)\, \pi \\
\delta(j(\pi), x.\rho, y.\tau) & \to & (\lambda y.\tau)\, \pi \\
(\lambda \mathsf{x}.\pi)\, \mathsf{t} & \to & \pi[\mathsf{t}/\mathsf{x}] \\
\delta_{\exists}(\langle \mathsf{t}, \pi \rangle, x.\mathsf{x}.\rho) & \to & (\lambda x.\rho[\mathsf{t}/\mathsf{x}])\, \pi
\end{array}
$$

On peut aussi choisir d'utiliser des règles comme

$$\delta(i(\pi), x.\rho, y.\tau) \quad \to \quad \rho[\pi/x]$$

pour les connecteurs $\neg$, $\vee$ et $\exists$. Alors la propriété de subject reduction ainsi que la preuve de normalisation forte par candidats de réductibilité s'étendent à la syntaxe, au typage et à la réduction étendue. La correspondance de Curry-De Bruijn-Howard s'étend aussi entre ce lambda-calcul étendu et le système NJ. Nous obtenons donc par cette correspondance une preuve de la normalisation forte de la réduction définie par les figures 2.5, 2.6 et 2.4 puis une preuve de la propriété 2.2.3 (cette fois dans le cadre de NJ).

3.2 Pour la déduction naturelle classique

En 1990, Griffin propose un système de types pour Idealized Scheme. Son langage contient en particulier deux opérateurs de contrôle qui représentent l'instruction

$$\textsc{Var}\ \frac{}{\Gamma, x : \varphi \vdash x : \varphi} \qquad \delta_\bot()\ \frac{\Gamma \vdash \pi : \bot}{\Gamma \vdash \delta_\bot(\pi) : \varphi}$$

$$\textsc{Abs}\ \frac{\Gamma, x : \varphi \vdash \pi : \psi}{\Gamma \vdash \lambda x.\pi : \varphi \Rightarrow \psi} \qquad \textsc{App}\ \frac{\Gamma \vdash \pi : \varphi \Rightarrow \psi \qquad \Gamma \vdash \rho : \varphi}{\Gamma \vdash \pi\,\rho : \psi}$$

$$(,)\ \frac{\Gamma \vdash \pi : \varphi \qquad \Gamma \vdash \rho : \psi}{\Gamma \vdash (\pi, \rho) : \varphi \wedge \psi} \qquad \mathsf{fst}()\ \frac{\Gamma \vdash \pi : \varphi \wedge \psi}{\Gamma \vdash \mathsf{fst}(\pi) : \varphi} \qquad \mathsf{snd}()\ \frac{\Gamma \vdash \pi : \varphi \wedge \psi}{\Gamma \vdash \mathsf{snd}(\pi) : \psi}$$

$$i()\ \frac{\Gamma \vdash \pi : \varphi}{\Gamma \vdash i(\pi) : \varphi \vee \psi} \qquad j()\ \frac{\Gamma \vdash \pi : \psi}{\Gamma \vdash j(\pi) : \varphi \vee \psi}$$

$$\delta(,,)\ \frac{\Gamma \vdash \pi : \varphi \vee \psi \qquad \Gamma, x : \varphi \vdash \rho : \psi' \qquad \Gamma, y : \psi \vdash \tau : \psi'}{\Gamma \vdash \delta(\pi, x.\rho, y.\tau) : \psi'}$$

$$\lambda^\neg\ \frac{\Gamma, x : \varphi \vdash \pi : \bot}{\Gamma \vdash \lambda^\neg x.\pi : \neg\varphi} \qquad \delta_\neg(,)\ \frac{\Gamma \vdash \pi : \neg\varphi \qquad \Gamma \vdash \rho : \varphi}{\Gamma \vdash \delta_\neg(\pi, \rho) : \bot}$$

$$\textsc{Abs}'\ \frac{\Gamma \vdash \pi : \varphi}{\Gamma \vdash \lambda \mathsf{x}.\pi : \forall \mathsf{x}.\varphi}\ \mathsf{x} \notin \mathcal{FV}(\Gamma) \qquad \textsc{App}'\ \frac{\Gamma \vdash \pi : \forall \mathsf{x}.\varphi}{\Gamma \vdash \pi\,\mathsf{t} : \varphi[\mathsf{t}/\mathsf{x}]}$$

$$\langle,\rangle\ \frac{\Gamma \vdash \pi : \varphi[\mathsf{t}/\mathsf{x}]}{\Gamma \vdash \langle \mathsf{t}, \pi \rangle : \exists \mathsf{x}.\varphi} \qquad \delta_\exists(,)\ \frac{\Gamma \vdash \pi : \exists \mathsf{x}.\varphi \qquad \Gamma, x : \varphi \vdash \rho : \psi}{\Gamma \vdash \delta_\exists(\pi, x.\mathsf{x}.\rho) : \psi}\ \mathsf{x} \notin \mathcal{FV}(\Gamma, \psi)$$

FIG. 3.1 – Typage du lambda-calcul étendu

`call/cc` de Scheme : les opérateurs $\mathcal{A}$ et $\mathcal{C}$ de la sous-section 1.2.8. Griffin remarque que cet opérateur doit être typé par $\neg\neg\varphi \Rightarrow \varphi$: en supposant que $\lambda z.E[z]$ admet le type $\varphi \Rightarrow \psi$, alors la réduction

$$(\lambda z.\mathcal{A}(E[z]))\,\nu \;\to^{+}\; E[\nu] \;,$$

montre que $\lambda z.\mathcal{A}(E[z])$ admet ce même type $\varphi \Rightarrow \psi$. Analysons la règle

$$E[\mathcal{C}(\pi)] \;\to\; \pi\,(\lambda z.\mathcal{A}(E[z]))\;.$$

Le type de $\mathcal{C}(\pi)$ doit être φ et le type de $E[\mathcal{C}(\pi)]$ étant alors ψ, on en déduit que le type de π est $(\varphi \Rightarrow \psi) \Rightarrow \psi$. Alors le type de $\lambda x.\mathcal{C}(x)$ est $((\varphi \Rightarrow \psi) \Rightarrow \psi) \Rightarrow \varphi$. Le type de $\lambda x.\mathcal{K}(x)$ est, lui, la loi de Pierce $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$. Supposons à présent que π est un terme clos de type ψ, alors $\mathcal{A}(\pi) = \mathcal{C}(\lambda x.\pi)$, est de type φ.

Du point de vue logique, nous n'avons rien supposé sur φ (nous n'avons supposé que la prouvabilité de $\varphi \Rightarrow \psi$ et de ψ). Nous en avons néanmoins déduit la prouvabilité de φ. Il vaut mieux donc que ψ ne soit pas prouvable. Sans cela toute formule φ sera prouvable et le système logique sera inconsistant. En particulier on peut choisir pour ψ la formule $\bot$. On obtient un système consistant. Si l'on identifie les formules $(\varphi \Rightarrow \bot)$ et $\neg\varphi$ pour tout φ (ces formules sont toujours équiprouvables, que ce soit en logique classique ou intuitionniste), les règles de typage des opérateurs $\mathcal{C}$, $\mathcal{K}$ et $\mathcal{A}$ seront

$$\frac{\Gamma \vdash \pi : \neg\neg\varphi}{\Gamma \vdash \mathcal{C}(\pi) : \varphi}\;, \qquad \frac{\Gamma \vdash \pi : \neg\varphi \Rightarrow \varphi}{\Gamma \vdash \mathcal{K}(\pi) : \varphi} \quad \text{et} \quad \frac{\Gamma \vdash \pi : \bot}{\Gamma \vdash \mathcal{A}(\pi) : \varphi}\;.$$

La première règle s'associe bien évidemment à la règle $\neg\neg$Elim de NK. La troisième correspond quant à elle à la règle $\bot$Elim présente dans NJ et NK. Enfin la seconde correspond au raisonnement par l'absurde : la formule $\neg\varphi \Rightarrow \varphi$ prouve l'absurdité de $\neg\varphi$, et donc la vérité de φ. Par conséquent, rajouter ces opérateurs de contrôle ainsi que cette règle de typage permet d'obtenir un système de type et une correspondance de Curry-De Bruijn-Howard entre la logique classique et le lambda-calcul avec opérateurs de contrôle typé. Il est néanmoins ennuyeux de devoir utiliser une stratégie de réduction particulière pour obtenir une telle correspondance. Afin de corriger cela, Parigot (1992) proposa une extension du lambda-calcul en correspondance avec un système de déduction naturelle classique. Son système n'utilise pas de stratégie pour la bêta-réduction. Le système déductif que nous utiliserons et que nous appellerons NK_μ contient les règles d'inférence de la figure 3.2. Ce système est un système de déduction naturelle : il contient bien des règles d'introduction et d'élimination des connecteurs. Néanmoins et contrairement à NK, il permet de prouver des séquents de la forme $\Gamma \vdash \Delta$ où Δ est un contexte pouvant contenir un nombre arbitraire de formules. Notons que le système utilisé par Parigot (1992) est légèrement différent : les contractions sont implicites et les règles sont multiplicatives. La règle Contraction correspond à une contraction à droite du symbole $\vdash$.

$$\textsc{Axiom}\ \frac{}{\Gamma, \varphi \vdash \varphi, \Delta} \qquad \textsc{Contraction}\ \frac{\Gamma \vdash \varphi, \varphi, \Delta}{\Gamma \vdash \varphi, \Delta}$$

$$\Rightarrow\textsc{Intro}\ \frac{\Gamma, \varphi \vdash \psi, \Delta}{\Gamma \vdash \varphi \Rightarrow \psi, \Delta} \qquad \Rightarrow\textsc{Elim}\ \frac{\Gamma \vdash \varphi \Rightarrow \psi, \Delta \qquad \Gamma \vdash \varphi, \Delta}{\Gamma \vdash \psi, \Delta}$$

$$\neg\textsc{Intro}\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma \vdash \neg\varphi, \Delta} \qquad \neg\textsc{Elim}\ \frac{\Gamma \vdash \neg\varphi, \Delta \qquad \Gamma \vdash \varphi, \Delta}{\Gamma \vdash \Delta}$$

FIG. 3.2 – Système Déductif NK_μ

Néanmoins une contraction à gauche peut être simulée comme suit.

$$\dfrac{\dfrac{\dfrac{\Gamma, \varphi, \varphi \vdash \psi, \Delta}{\Gamma, \varphi \vdash \varphi \Rightarrow \psi, \Delta}\,\Rightarrow\textsc{Intro} \qquad \textsc{Axiom}\ \dfrac{}{\Gamma, \varphi \vdash \varphi, \Delta}}{\Gamma, \varphi \vdash \psi, \Delta}\,\Rightarrow\textsc{Elim}$$

C'est d'ailleurs aussi valable dans nos systèmes NJ et NK. Bien que NK_μ ne contienne pas la règle $\neg\neg$Elim de NK, celle-ci est admissible dans NK_μ. En fait les séquents prouvables dans NK_μ sont exactement les séquents $\Gamma \vdash \Delta$ classiquement valides et tels que les connecteurs de Γ et Δ ne sont que des implications et des négations. Par exemple on peut dériver le séquent $\vdash \neg\neg\varphi \Rightarrow \varphi$.

$$\dfrac{\dfrac{\dfrac{\textsc{Axiom}\ \dfrac{}{\neg\neg\varphi \vdash \neg\neg\varphi, \varphi} \qquad \neg\textsc{Intro}\ \dfrac{\textsc{Axiom}\ \dfrac{}{\neg\neg\varphi, \varphi \vdash \varphi}}{\neg\neg\varphi \vdash \neg\varphi, \varphi}}{\neg\neg\varphi \vdash \varphi}\,\neg\textsc{Elim}}{\vdash \neg\neg\varphi \Rightarrow \varphi}\,\Rightarrow\textsc{Intro}$$

On peut aussi y dériver la loi de Pierce.

$$\dfrac{\dfrac{\dfrac{\dfrac{\textsc{Axiom}\ \dfrac{}{(\psi \Rightarrow \varphi) \Rightarrow \psi \vdash (\psi \Rightarrow \varphi) \Rightarrow \psi, \psi} \qquad \Rightarrow\textsc{Intro}\ \dfrac{\textsc{Axiom}\ \dfrac{}{(\psi \Rightarrow \varphi) \Rightarrow \psi, \psi \vdash \varphi, \psi}}{(\psi \Rightarrow \varphi) \Rightarrow \psi \vdash \psi \Rightarrow \varphi, \psi}}{(\psi \Rightarrow \varphi) \Rightarrow \psi \vdash \psi, \psi}\,\Rightarrow\textsc{Elim}}{((\psi \Rightarrow \varphi) \Rightarrow \psi) \vdash \psi, \psi}\,\textsc{ContrR}}{((\psi \Rightarrow \varphi) \Rightarrow \psi) \vdash \psi}\,\Rightarrow\textsc{Intro}}{\vdash ((\psi \Rightarrow \varphi) \Rightarrow \psi) \Rightarrow \psi}$$

Nous allons à présent définir le $\lambda\mu$-calcul de Parigot. En plus de l'ensemble X de variables, on suppose donné un ensemble A de noms. Les symboles α, β, γ représenteront des noms. Les ensembles des $\lambda\mu$-termes anonymes et des $\lambda\mu$-termes nommés sont respectivement définis comme les langages hors-contexte associés aux grammaires en forme de Backus-Naur suivantes.

$$\begin{array}{rcl} \pi, \rho, \tau & ::= & x \mid \lambda x.\pi \mid \pi\,\rho \mid \mu\alpha.\eta \\ \eta, \vartheta, \kappa & ::= & [\alpha]\pi \end{array}$$

$$\textsc{Var}\ \frac{}{\Gamma, x : \varphi \vdash x : \varphi \mid \Delta} \qquad \textsc{Abs}\ \frac{\Gamma, x : \varphi \vdash \pi : \psi \mid \Delta}{\Gamma \vdash \lambda x.\pi : \varphi \Rightarrow \psi \mid \Delta}$$

$$\textsc{App}\ \frac{\Gamma \vdash \pi : \varphi \Rightarrow \psi \mid \Delta \qquad \Gamma \vdash \tau : \varphi \mid \Delta}{\Gamma \vdash \pi\,\tau : \psi \mid \Delta}$$

$$\textsc{Name}\ \frac{\Gamma \vdash \pi : \varphi \mid \Delta}{[\alpha]\pi \triangleright \Gamma \vdash \alpha : \varphi, \Delta} \qquad \textsc{Mu}\ \frac{\eta \triangleright \Gamma \vdash \alpha : \varphi, \Delta}{\Gamma \vdash \mu\alpha.\eta : \varphi \mid \Delta}$$

FIG. 3.3 – Typage du $\lambda\mu$-calcul

Les noms diffèrent des variables par le fait que l'on ne les remplacera jamais par des termes. Néanmoins on définit le remplacement $[\alpha/\beta]$ que l'on peut appliquer à tout terme nommé ou anonyme ne contenant pas α : il s'agit bien sûr de remplacer tout occurrence de β par α. Le symbole μ est un lieur pour les noms et on considère donc les termes anonymes et nommés modulo alpha-conversion sur les noms. En outre pour tout terme nommé η et pour tout terme anonyme τ, le terme $\eta[[\alpha](\_\,\pi)/[\alpha]\_]$ est défini comme étant obtenu à partir de η en remplaçant chaque sous-terme de la forme $[\alpha]\tau$ par $[\alpha](\tau\,\pi)$. Les règles de réduction du $\lambda\mu$-calcul sont alors les suivantes.

$$\begin{array}{lcl}(\lambda x.\pi)\rho & \to & \pi[\rho/x] \\ (\mu\alpha.\eta)\pi & \to & \mu\alpha.(\eta[[\alpha](\_\,\pi)/[\alpha]\_]) \\ [\alpha]\mu\beta.\eta & \to & \eta[\alpha/\beta]\end{array}$$

Nous noterons $\to_{\beta\mu}$ la réduction associée à ces règles de réduction. La première règle correspond à la bêta-réduction du lambda-calcul et représente une élimination des coupures logiques. La deuxième règle fait correspondre l'opérateur μ intuitivement à un `call/cc` puisque pour un nombre n quelconque d'arguments $\pi_1 \dots \pi_n$, la réduction de $(\mu\alpha.\eta)\pi_1 \dots \pi_n$ a pour effet d'ajouter les arguments $\pi_1 \dots \pi_n$ à tous les sous-termes de η nommés par α. Cet ensemble d'arguments est donc une *continuation* que l'on déplace jusqu'aux sous-termes nommés par α. Parigot a prouvé la confluence de $\to_{\beta\mu}$.

Propriété 3.2.1 (Parigot, 1992). *La relation $\to_{\beta\mu}$ est confluente.*

Pour définir le typage du $\lambda\mu$-calcul, nous allons utiliser deux différentes catégories de contextes de typage : des ensembles de déclarations de types $x : \varphi$ pour les variables et des ensembles de déclarations de types $\alpha : \varphi$ pour les noms. Les symboles Γ et Δ représenteront respectivement des contextes pour les variables et des contextes pour les noms. Tout comme dans la section 3.1, un contexte de typage n'est autorisé à contenir qu'une déclaration $x : \varphi$ ou $\alpha : \varphi$ pour une variable x donnée ou un nom α donné. Un jugement de typage s'écrit alors $\Gamma \vdash \pi : \varphi \mid \Delta$ ou $\eta \triangleright \Gamma \vdash \Delta$. Les règles de typage du $\lambda\mu$-calcul sont les règles de la figure 3.3. Le typage est encore une fois dirigé par la syntaxe. Parigot a montré que le typage est stable par réduction par $\to_{\beta\mu}$.

Propriété 3.2.2 (Subject reduction pour $\to_{\beta\mu}$ (Parigot, 1992)). *Si $\Gamma \vdash \pi : \varphi \mid \Delta$ est un jugement de typage valide et si $\pi \to_{\beta\mu} \tau$, alors $\Gamma \vdash \tau : \varphi \mid \Delta$ est un jugement de typage valide.*

Un jugement de typage $\eta \triangleright \Gamma \vdash \Delta$ correspond au séquent $\Gamma \vdash \Delta$ (où les variables et les noms ont été effacés). Un jugement de typage $\Gamma \vdash \pi : \varphi \mid \Delta$ correspond au séquent $\Gamma \vdash \varphi, \Delta$. Ce dernier jugement de typage ne contient pas seulement le $\lambda\mu$-terme anonyme π comme information supplémentaire par rapport au séquent sous-jacent : ce jugement contient aussi la zone située entre le symbole $\vdash$ et le symbole $\mid$ appelée *bénitier* qui souligne le fait que le type de π est φ (et non pas une autre formule de Γ ou Δ). Néanmoins ce bénitier peut être manipulé souplement : on peut sortir à tout moment une formule du bénitier à l'aide d'une instance de Name et l'on peut ensuite y introduire une formule arbitraire du contexte Δ à l'aide d'une instance de Mu. Ces deux règles Mu et Name permettent aussi de simuler la contraction à droite comme suit.

$$\textsc{Mu}\ \dfrac{\textsc{Name}\ \dfrac{\Gamma \vdash \pi : \varphi \mid \alpha : \varphi, \Delta}{[\alpha]\pi \triangleright \Gamma \vdash \alpha : \varphi, \Delta}}{\Gamma \vdash \mu\alpha.[\alpha]\pi : \varphi \mid \Delta}$$

Les règles Var, Abs et App représentent respectivement les règles Axiom, $\Rightarrow$Intro et $\Rightarrow$Elim de NK_μ. Les règles Mu et Name sont, elles, des règles bureaucratiques qui sont silencieuses dans NK_μ : les séquents correspondant respectivement à leur conclusion et à leur unique prémisse sont égaux. Nous obtenons donc une correspondance entre le typage du $\lambda\mu$-calcul et le fragment de NK_μ omettant les règles relatives à la négation. Pour obtenir une correspondance avec le système NK_μ complet, Parigot propose d'agir comme si un nom (frais) typé $\zeta : \bot$ apparaissait toujours du coté droit des jugements de typage sans pour autant qu'il soit écrit. On peut alors écrire les dérivations ouvertes de typage suivantes.

$$\dfrac{\eta \triangleright \Gamma \vdash \Delta}{\Gamma \vdash \mu\zeta.\eta : \bot \mid \Delta} \qquad \dfrac{\Gamma \vdash \pi : \bot \mid \Delta}{[\zeta]\pi \triangleright \Gamma \vdash \Delta}$$

Alors en identifiant les formules $(\varphi \Rightarrow \bot)$ et $\neg\varphi$ pour tout φ, les règles $\neg$Intro et $\neg$Elim de NK_μ sont représentée par

$$\textsc{Mu}\ \dfrac{\textsc{Abs}\ \dfrac{\eta \triangleright \Gamma, x : \varphi \vdash \Delta}{\Gamma, x : \varphi \vdash \mu\zeta.\eta : \bot \mid \Delta}}{\Gamma \vdash \lambda x.\mu\zeta.\eta : \neg\varphi \mid \Delta} \qquad \textsc{App}\ \dfrac{\textsc{Name}\ \dfrac{\Gamma \vdash \pi : \neg\varphi \mid \Delta \qquad \Gamma \vdash \rho : \varphi \mid \Delta}{\Gamma \vdash \pi\,\rho : \bot \mid \Delta}}{[\zeta](\pi\,\rho) \triangleright \Gamma \vdash \Delta}$$

Le typage de la figure 3.3 définit bien une correspondance entre le système déductif NK_μ et le $\lambda\mu$-calcul bien typé. La réduction $\to_{\beta\mu}$ correspond bien à l'élimination des coupures dans NK_μ. Griffin a proposé de typer les opérateurs $\mathcal{C}$ et $\mathcal{K}$ respectivement par les types $\neg\neg\varphi \Rightarrow \varphi$ et $((\psi \Rightarrow \varphi) \Rightarrow \psi) \Rightarrow \psi$. On peut écrire deux

opérateurs similaires dans le $\lambda\mu$-calcul. Le terme $\lambda y.\mu\alpha.[\beta](y\,\lambda x.\mu\zeta.[\alpha]x)$ admet le type $\neg\neg\varphi \Rightarrow \varphi$, comme le démontre la dérivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y : \neg\neg\varphi, x : \varphi \vdash x : \varphi \mid}\,\textsc{Var}}{[\alpha]x \triangleright y : \neg\neg\varphi, x : \varphi \vdash \alpha : \varphi}\,\textsc{Name}}{y : \neg\neg\varphi, x : \varphi \vdash \mu\zeta.[\alpha]x : \bot \mid \alpha : \varphi}\,\textsc{Mu}}{y : \neg\neg\varphi \vdash \lambda x.\mu\zeta.[\alpha]x : \neg\varphi \mid \alpha : \varphi}\,\textsc{Abs} \qquad \dfrac{}{y : \neg\neg\varphi \vdash y : \neg\neg\varphi \mid \alpha : \varphi}\,\textsc{Axiom}}{y : \neg\neg\varphi \vdash y\,\lambda x.\mu\zeta.[\alpha]x : \bot \mid \alpha : \varphi}\,\textsc{App}}{[\beta](y\,\lambda x.\mu\zeta.[\alpha]x) \triangleright y : \neg\neg\varphi \vdash \alpha : \varphi}\,\textsc{Name}}{y : \neg\neg\varphi \vdash \mu\alpha.[\beta](y\,\lambda x.\mu\zeta.[\alpha]x) : \varphi \mid}\,\textsc{Mu}}{\vdash \lambda y.\mu\alpha.[\beta](y\,\lambda x.\mu\zeta.[\alpha]x) : \neg\neg\varphi \Rightarrow \varphi \mid}\,\textsc{Abs}$$

Ce terme a d'ailleurs un comportement proche du terme $\mathcal{C}$:

$$\begin{array}{c}\Big(\ldots\Big(\Big(\overbrace{\lambda y.\mu\alpha.[\beta](y\,\lambda x.\mu\zeta.[\alpha]x)}^{\mathcal{C}}\;\overbrace{(u)}^{\pi}\Big)\;\overbrace{v_1\Big)\ldots v_n\Big)}^{E}\\ \downarrow\\ (\ldots(\mu\alpha.[\beta](u\,\lambda x.\mu\zeta.[\alpha]x)\,v_1)\ldots v_n)\\ \downarrow_*\\ \mu\alpha.[\beta]\Big(\underbrace{(u)}_{\pi}\;\Big(\lambda x.\mu\zeta.[\alpha](\ldots(x\;\underbrace{v_1)\ldots v_n)}_{E}\Big)\Big)\end{array}$$

Le terme $\lambda y.\mu\alpha.[\alpha](y\,\lambda x.\mu\delta.[\alpha]x)$ admet, lui, le type $((\psi \Rightarrow \varphi) \Rightarrow \varphi) \Rightarrow \varphi$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y : (\psi \Rightarrow \varphi) \Rightarrow \psi \vdash y : (\psi \Rightarrow \varphi) \Rightarrow \psi \mid \alpha : \psi}\,\textsc{Var}}{y : (\psi \Rightarrow \varphi) \Rightarrow \psi, x : \psi \vdash x : \psi \mid \delta : \varphi}\,\textsc{Var}}{[\alpha]x \triangleright y : (\psi \Rightarrow \varphi) \Rightarrow \psi, x : \psi \vdash \delta : \varphi, \alpha : \psi}\,\textsc{Name}}{y : (\psi \Rightarrow \varphi) \Rightarrow \psi, x : \psi \vdash \mu\delta.[\alpha]x : \varphi \mid \alpha : \psi}\,\textsc{Mu}}{y : (\psi \Rightarrow \varphi) \Rightarrow \psi \vdash \lambda x.\mu\delta.[\alpha]x : \psi \Rightarrow \varphi \mid \alpha : \psi}\,\textsc{Abs}}{y : ((\psi \Rightarrow \varphi) \Rightarrow \psi) \vdash y\,\lambda x.\mu\delta.[\alpha]x : \psi \mid \alpha : \psi}\,\textsc{App}}{[\alpha](y\,\lambda x.\mu\delta.[\alpha]x) \triangleright y : ((\psi \Rightarrow \varphi) \Rightarrow \psi) \vdash \alpha : \psi}\,\textsc{Name}}{y : ((\psi \Rightarrow \varphi) \Rightarrow \psi) \vdash \mu\alpha.[\alpha](y\,\lambda x.\mu\delta.[\alpha]x) : \psi \mid}\,\textsc{Mu}}{\vdash \lambda y.\mu\alpha.[\alpha](y\,\lambda x.\mu\delta.[\alpha]x) : ((\psi \Rightarrow \varphi) \Rightarrow \psi) \Rightarrow \psi \mid}\,\textsc{Abs}$$

Il se comporte similairement à l'opérateur $\mathcal{K}$.

$$\begin{array}{c}
\Big(\dots \Big(\Big(\overbrace{(\lambda y.\mu\alpha.[\alpha](y\ \lambda x.\mu\delta.[\alpha]x))}^{\mathcal{K}}\ \overbrace{(u)}^{\pi} \Big)\ v_1 \Big) \dots v_n \Big)^{E} \\
\downarrow \\
(\dots(\mu\alpha.[\alpha](u\ \lambda x.\mu\delta.[\alpha]x)\ v_1)\dots v_n) \\
\downarrow \\
\mu\alpha.[\alpha]\Big(\dots \Big(\Big(\underbrace{(u)}_{\pi}\ \Big(\lambda x.\mu\delta.[\alpha](\dots(x\ \underbrace{v_1)\dots v_n)}_{E}\Big)\Big)\ \underbrace{v_1\Big)\dots v_n\Big)}_{E}
\end{array}$$

3.3 Pour le calcul des séquents classique

Le calcul des séquents est un formalisme dont la symétrie apporte bien des bénéfices, notamment la propriété de la sous-formule ou encore le fait que les coupures soient clairement identifiées par la règle Cut. Il est donc naturel de vouloir transposer l'isomorphisme de Curry-De Bruijn-Howard aux calculs des séquents. C'est dans cet objectif que Curien et Herbelin (2000); Herbelin (1995) ont proposé le $\bar{\lambda}\mu\tilde{\mu}$-calcul, une symétrisation du $\lambda\mu$-calcul de Parigot. Le $\bar{\lambda}\mu\tilde{\mu}$-calcul est présenté en sous-section 3.3.1. Urban (2000) a proposé une deuxième approche déconnectée du lambda-calcul. Plutôt que de passer successivement du lambda-calcul au $\lambda\mu$ puis $\bar{\lambda}\mu\tilde{\mu}$ calcul, Urban étudie directement l'élimination des coupures en calcul des séquents classiques. Son calcul est présenté en sous-section 3.3.2.

3.3.1 Le $\bar{\lambda}\mu\tilde{\mu}$-calcul

L'intuition derrière le $\bar{\lambda}\mu\tilde{\mu}$-calcul est de souligner la dualité qui existe entre les *fonctions* et les *contextes*. En particulier un contexte peut signifier l'utilisation d'une fonction. Dans le lambda-calcul, l'abstraction étant le constructeur, son utilisation, c'est-à-dire l'application, est son destructeur. En déduction naturelle, l'abstraction correspond alors à une règle d'introduction et l'application correspond à une règle d'élimination. Nous avons vu en section 2.4 que les règles d'élimination peuvent se traduirent pas des règles d'introduction à gauche en calcul des séquents. Cette traduction s'apparente au passage des lambda-termes aux contextes : l'analyse de Griffin (1990) (voir section 3.2) montre qu'un solution acceptable est, si E est un contexte, de typer $\lambda z.E[z]$ par $(\varphi \Rightarrow \bot) = \neg\varphi$. Pour transposer la dualité de De Morgan (sous-section 2.4.1) aux lambda-termes et contextes, on peut choisir de typer le contexte E par φ. Le jugement « $E : \varphi$ » signifie alors « E est un contexte qui prend en *entrée* une valeur de type φ ». C'est bien le dual de « $\pi : \varphi$ », c'est-à-dire « π est un terme qui donne en *sortie* une valeur de type φ ». La règle $\Rightarrow$Elim correspond à l'application sur les lambda-termes. De manière duale, la règle $\Rightarrow$L correspond à

l'application dans les contextes.

$$\frac{E : \psi \vdash \qquad \vdash \pi : \varphi}{E[\square\, \pi] : \varphi \Rightarrow \psi \vdash}$$

Si E est un contexte en attente d'une valeur de type ψ et si π retourne une valeur de type φ, alors $E[\square\, \pi]$ est un contexte en attente d'une valeur de type $\varphi \Rightarrow \psi$: il applique cette dernière valeur (qui est une fonction puisque typée par une implication) à π puis utilise la valeur résultante dans le contexte E.

Définissons à présent le $\bar{\lambda}\mu\tilde{\mu}$-calcul. On se donne un ensemble X de variables et un ensemble A de *covariables*. Contrairement aux noms du $\lambda\mu$-calcul et similairement au variables, les covariables pourront effectivement être remplacées non pas par des termes mais par des contextes. Tout comme en section 3.2, les symboles x, y, z représenteront des variables et les symboles α, β, γ représenteront des covariables. La syntaxe du $\bar{\lambda}\mu\tilde{\mu}$-calcul contient donc trois catégories : les états ou *commandes*, les termes et les contextes. Le symbole c représentera un état, les symboles π, ρ, τ représenteront des termes et le symbole e représentera un contexte. Les ensembles des états, termes et contextes sont respectivement définis comme les langages hors-contexte associés aux grammaires en forme de Backus-Naur suivantes.

$$\begin{array}{rcl} c & ::= & \langle \pi | e \rangle \\ \pi & ::= & x \mid \mu\beta.c \mid \lambda x.\pi \\ e & ::= & \alpha \mid \tilde{\mu}x.c \mid \pi \cdot e \end{array}$$

Les constructeurs λ, μ et $\tilde{\mu}$ sont des lieurs. Nous parlerons de λ-abstraction, μ-abstraction ou $\tilde{\mu}$-abstraction. On considère donc les termes, contextes et commandes modulo alpha-consersion pour les variables et les covariables. Si x est une variable et π est un terme, alors la substitution $[\pi/x]$ est définie sur les commandes, termes et contextes comme suit.

$$\begin{array}{lcllcl} x[\pi/x] & = & \pi & \alpha[\pi/x] & = & \alpha \\ y[\pi/x] & = & y \text{ si } y \neq x & (\tilde{\mu}y.\rho)[\pi/x] & = & \tilde{\mu}y.(\rho[\pi/x]) \\ (\lambda y.\rho)[\pi/x] & = & \lambda y.(\rho[\pi/x]) & (\rho \cdot e)[\pi/x] & = & (\rho[\pi/x]) \cdot (e[\pi/x]) \\ (\mu\beta.c)[\pi/x] & = & \mu\beta.(c[\pi/x]) & \langle \rho | e \rangle[\pi/x] & = & \langle \rho[\pi/x] | e[\pi/x] \rangle \end{array}$$

Si α est une covariable et e est un contexte, alors la substitution $[e/\alpha]$ sur les commandes, termes et contextes est définie de façon similaire.

$$\begin{array}{lcllcl} x[e/\alpha] & = & x & \alpha[e/\alpha] & = & e \\ (\lambda x.\rho)[e/\alpha] & = & \lambda x.(\rho[e/\alpha]) & \beta[e/\alpha] & = & \beta \text{ si } \beta \neq \alpha \\ (\mu\beta.c)[e/\alpha] & = & \mu\beta.(c[e/\alpha]) & (\tilde{\mu}x.\rho)[e/\alpha] & = & \tilde{\mu}x.(\rho[e/\alpha]) \\ \langle \rho | e' \rangle[e/\alpha] & = & \langle \rho[e/\alpha] | e'[e/\alpha] \rangle & (\rho \cdot e')[e/\alpha] & = & (\rho[e/\alpha]) \cdot (e'[e/\alpha]) \end{array}$$

Les substitutions sur les variables ou les covariables ne sont appliquées que si elles évitent les captures. C'est pour cette raison que nous n'avons pas défini $(\lambda x.\rho)[\pi/x]$,

$$\textsc{CoVar}\ \frac{}{\Gamma \shortmid \alpha : \varphi \vdash \alpha : \varphi, \Delta} \qquad \textsc{Var}\ \frac{}{\Gamma, x : \varphi \vdash x : \varphi \shortmid \Delta}$$

$$\textsc{Cmd}\ \frac{\Gamma \vdash \pi : \varphi \shortmid \Delta \qquad \Gamma \shortmid e : \varphi \vdash \Delta}{\langle \pi | e \rangle \triangleright \Gamma \vdash \Delta}$$

$$\textsc{App}\ \frac{\Gamma \vdash \pi : \varphi \shortmid \Delta \qquad \Gamma \shortmid e : \psi \vdash \Delta}{\Gamma \shortmid \pi \cdot e \vdash \Delta} \qquad \textsc{Abs}\ \frac{\Gamma, x : \varphi \vdash \pi : \psi \shortmid \Delta}{\Gamma \vdash \lambda x.\pi : \varphi \Rightarrow \psi \shortmid \Delta}$$

$$\textsc{Mu}\ \frac{c \triangleright \Gamma \vdash \beta : \varphi, \Delta}{\Gamma \vdash \mu\beta.c : \varphi \shortmid \Delta} \qquad \textsc{Mu}\tilde{}\ \frac{c \triangleright \Gamma, x : \varphi \vdash \Delta}{\Gamma \shortmid \tilde{\mu}x.c : \varphi \vdash \Delta}$$

FIG. 3.4 – Typage du $\bar{\lambda}\mu\tilde{\mu}$-calcul

$(\tilde{\mu}x.\rho)[\pi/x]$ ou $(\mu\alpha.e)[e'/\alpha]$: ces cas impliquent une capture. Appliquer ces substitutions sous-entend alors une étape d'alpha-conversion. Les règles de réduction du $\bar{\lambda}\mu\tilde{\mu}$-calcul sont les suivantes.

$$\begin{array}{lcl} \langle \lambda x.\pi | \rho \cdot e \rangle & \to & \langle \rho | \tilde{\mu}x.\langle \pi | e \rangle \rangle \\ \langle \mu\alpha.c | e \rangle & \to & c[e/\alpha] \\ \langle \pi | \tilde{\mu}x.c \rangle & \to & c[\pi/x] \end{array}$$

La première des ces trois règles sera appelée $\bar{\beta}$, la deuxième sera appelée μ et la troisième sera appelée $\tilde{\mu}$. Nous noterons $\to_{\bar{\beta}\mu\tilde{\mu}}$ la réduction associée à ces trois règles. Pour définir le typage du $\bar{\lambda}\mu\tilde{\mu}$-calcul, nous allons utiliser deux catégories de contextes de typage : des ensembles de déclarations de types $x : \varphi$ pour les variables et des ensembles de déclarations de types $\alpha : \varphi$ pour les covariables. Les symboles Γ et Δ représenteront respectivement des contextes pour les variables et des contextes pour les covariables. Tout comme dans la section 3.1, un contexte de typage n'est autorisé à contenir qu'une déclaration $x : \varphi$ ou $\alpha : \varphi$ pour une variable x donnée ou une covariable α donnée. Un jugement de typage s'écrit alors soit $\Gamma \vdash \pi : \varphi \shortmid \Delta$, soit $\Gamma \shortmid e : \varphi \vdash \Delta$, soit $c \triangleright \Gamma \vdash \Delta$. Nous dirons respectivement que le focus est à droite, à gauche ou au centre. Les règles de typage du $\bar{\lambda}\mu\tilde{\mu}$-calcul sont celles de la figure 3.4. Tout d'abord notons que trois versions typées de l'affaiblissement respectivement à droite et à gauche sont admissibles.

$$\textsc{WL1}\ \frac{c \triangleright \Gamma \vdash \Delta}{c \triangleright \Gamma, x : \varphi \vdash \Delta} \qquad \textsc{WR1}\ \frac{c \triangleright \Gamma \vdash \Delta}{c \triangleright \Gamma \vdash \alpha : \varphi, \Delta}$$

$$\textsc{WL2}\ \frac{\Gamma \vdash \pi : \psi \shortmid \Delta}{\Gamma, x : \varphi \vdash \pi : \psi \shortmid \Delta} \qquad \textsc{WR2}\ \frac{\Gamma \vdash \pi : \psi \shortmid \Delta}{\Gamma \vdash \pi : \psi \shortmid \alpha : \varphi, \Delta}$$

$$\textsc{WL3}\ \frac{\Gamma \shortmid e : \psi \vdash \Delta}{\Gamma, x : \varphi \shortmid e : \psi \vdash \Delta} \qquad \textsc{WR3}\ \frac{\Gamma \shortmid e : \psi \vdash \Delta}{\Gamma \shortmid e : \psi \vdash \alpha : \varphi, \Delta}$$

Une instance de WL1, WL2 ou WL3 est toujours admissible si x n'apparaît pas dans la dérivation de la prémisse. Une instance de WR1, WR2 ou WR3 l'est toujours si α

n'apparaît pas dans la dérivation de la prémisse. La contraction est aussi admissible sous plusieurs formes. Les dérivations

$$\textsc{Cmd}\ \frac{\Gamma \vdash \pi : \varphi \shortmid \alpha : \varphi, \Delta \qquad \textsc{CoVar}\ \dfrac{}{\Gamma \shortmid \alpha : \varphi \vdash \alpha : \varphi, \Delta}}{\langle \pi | \alpha \rangle \triangleright \Gamma \vdash \alpha : \varphi, \Delta}$$

et

$$\begin{array}{l}\textsc{Var}\ \dfrac{}{\Gamma, x : \varphi \vdash x : \varphi \shortmid \Delta} \qquad \Gamma \shortmid e : \varphi \vdash \Delta \\ \textsc{Cmd}\ \dfrac{}{\langle x | e \rangle \triangleright \Gamma, x : \varphi \vdash \Delta}\end{array}$$

permettent par exemple de fusionner un terme $\pi : \varphi$ ou un contexte $e : \varphi$ respectivement avec une variable $x : \varphi$ ou une covariable $\alpha : \varphi$ du même séquent. Notons qu'à partir du typage d'un terme ou d'un contexte, on obtient le typage d'une commande. Ce passage correspond à la règle Name du $\lambda\mu$-calcul passant d'un terme anonyme à un terme nommé. Tout comme la règle Mu du $\lambda\mu$-calcul, les règles Mu et Mũ correspondent aux passages inverses. Elles permettent de passer tout jugement de typage typant une commande à un jugement de typage typant un terme ou un contexte. On peut donc passer souplement d'un focus quelconque (gauche, droite ou centre) à un autre focus. Des jugements de typage de la forme $c \triangleright \Gamma \vdash \Delta$, $\Gamma \vdash \pi : \varphi \shortmid \Delta$ et $\Gamma \shortmid e : \varphi \vdash \Delta$ représentent respectivement les séquents $\Gamma \vdash \Delta$, $\Gamma \vdash \varphi, \Delta$ et $\Gamma, \varphi \vdash \Delta$. Les règles Cmd, Var et CoVar correspondent aux règles Cut et Axiom de LK. Les règles App et Abs correspondent aux règles ⇒L et ⇒R. Nous avons vu que les contractions et les affaiblissements sont admissibles. Les preuves dans le fragment de LK dont les règles d'introduction sont restreintes aux règles relatives à l'implication sont donc en correspondance aux dérivations de typage pour le $\bar{\lambda}\mu\tilde{\mu}$-calcul.

La réduction $\to_{\bar{\beta}\mu\tilde{\mu}}$ n'est pas confluente. En effet pour toutes commandes c_1 et c_2 quelconques, la commande $\langle \mu\alpha.c_1 | \tilde{\mu}x.c_2 \rangle$ se réduit par μ en $c_1[\tilde{\mu}x.e/\alpha]$ et par $\tilde{\mu}$ en $c_2[\mu\alpha.\pi/x]$. En particulier ces deux commandes sont respectivement c_1 et c_2 si α n'apparaît pas libre dans c_1 et si x n'apparaît pas libre dans c_2. On en déduit que (la plus petite congruence contenant) $\to_{\bar{\beta}\mu\tilde{\mu}}$ n'est pas consistante sur les commandes : toute commande c_1 est convertible à toute autre commande c_2, il suffit d'exhiber une variable x et une covariable α fraîches. Il s'ensuit que

$$c_1 \quad {}_{\mu}\!\leftarrow \quad \langle \mu\alpha.c_1 | \tilde{\mu}x.c_2 \rangle \quad \to_{\tilde{\mu}} \quad c_2\ .$$

Notons d'ailleurs que seuls μ et $\tilde{\mu}$ suffisent pour avoir cette incohérence. Nous noterons $\to^{n}_{\bar{\beta}\mu\tilde{\mu}}$ la réduction qui restreint l'application de μ aux commandes $\langle \mu\alpha.c | e \rangle$ telles que e n'est pas une $\tilde{\mu}$-abstraction et qui ne réduit que la commande de tête. Nous noterons $\to^{v}_{\bar{\beta}\mu\tilde{\mu}}$ la réduction qui restreint l'application de $\tilde{\mu}$ aux commandes $\langle \pi | \tilde{\mu}x.c \rangle$ telles que π n'est pas une μ-abstraction et qui ne réduit que la commande de tête.

Si l'on omet les constructions μ et $\tilde{\mu}$, la syntaxe des termes ressemble étrangement à celle des valeurs du lambda-calcul en appel par valeur (section 1.2.7) et la

syntaxe des contextes ressemble étrangement à celle des CBN-contextes du lambda-calcul en appel par nom, leurs grammaires respectives étant

$$\nu \ ::= \ x \mid \lambda x.\pi \qquad \text{et} \qquad G \ ::= \ \Box \mid G\,\pi \ .$$

Le $\bar{\lambda}\mu\tilde{\mu}$-calcul contient effectivement les deux, puisque l'on peut traduire le lambda-calcul dans le $\bar{\lambda}\mu\tilde{\mu}$-calcul de sorte que la réduction $\to^{v}_{\bar{\beta}\mu\tilde{\mu}}$ corresponde à la bêta-réduction en appel par valeur et la réduction $\to^{v}_{\bar{\beta}\mu\tilde{\mu}}$ corresponde à la bêta-réduction en appel par nom. Le plongement présenté ici est une restriction du plongement du $\lambda\mu$-calcul en appel par valeur dans le $\bar{\lambda}\mu\tilde{\mu}$-calcul que l'on peut trouver par exemple dans les travaux de Curien et Herbelin (2000); Herbelin (2005). Si π est un lambda-terme, alors on définit $\langle\!\langle \pi \rangle\!\rangle$ par

$$\begin{array}{lll} \langle\!\langle x \rangle\!\rangle & = x & \langle\!\langle \pi\,\rho \rangle\!\rangle = \mu\alpha.\langle\!\langle \pi\,\rho \rangle\!\rangle_\alpha \quad \text{où } \alpha \text{ est une covariable fraîche} \\ \langle\!\langle \lambda x.\pi \rangle\!\rangle & = \lambda x.\langle\!\langle \pi \rangle\!\rangle & \\ \langle\!\langle \pi\,\rho \rangle\!\rangle_e & = \langle\!\langle \pi \rangle\!\rangle_{\langle\!\langle \rho \rangle\!\rangle \cdot e} & \langle\!\langle \nu \rangle\!\rangle_e \ = \langle \langle\!\langle \nu \rangle\!\rangle | e \rangle \end{array}$$

(rappelons que le symbole ν représente les valeurs, c'est-à-dire les variables ou les abstraction). Une induction sur les lambda-termes montre que tout lambda-terme π s'écrit de manière unique $G[\nu]$ où ν est une valeur et G est un CBN-contexte (sous-section 1.2.7). La commande $\langle\!\langle \pi \rangle\!\rangle_\alpha$ est alors tout à fait similaire à l'état de machine abstraite $\langle \nu | G \rangle$. si $\pi = (\dots(\nu\,\tau_1)\dots t_n)$, alors $\langle \nu | G \rangle = \langle \nu | \Box\,\tau_1 \dots \tau_n \rangle$ et $\langle\!\langle \pi \rangle\!\rangle_\alpha = \langle \langle\!\langle \nu \rangle\!\rangle | \tau_1 \dots \tau_n \cdot \alpha \rangle$. Alors les réductions $\to^{n}_{\bar{\beta}\mu\tilde{\mu}}$ et $\to^{v}_{\bar{\beta}\mu\tilde{\mu}}$ correspondent respectivement à la bêta-réduction en appel par nom et en appel par valeur. En effet si l'on considère la traduction d'un bêta-redex $\langle\!\langle (\lambda x.\pi)\,\rho \rangle\!\rangle_\alpha = \langle \lambda x.\langle\!\langle \pi \rangle\!\rangle | \langle\!\langle \rho \rangle\!\rangle \cdot \alpha \rangle$, il se réduit par la règle $\bar{\beta}$ en $\langle \langle\!\langle \rho \rangle\!\rangle | \tilde{\mu}x.\langle \langle\!\langle \pi \rangle\!\rangle | \alpha \rangle \rangle$. Alors en appel par nom, la priorité est donnée à la règle $\tilde{\mu}$ et l'on obtient directement $\langle \langle\!\langle \pi[\rho/x] \rangle\!\rangle | \alpha \rangle$. Par contre en appel par valeur, la règle $\tilde{\mu}$ n'est déclenchée que si $\langle\!\langle \rho \rangle\!\rangle$ n'est pas une μ-abstraction, c'est-à-dire si ρ est une valeur. La bêta-réduction n'est donc totalement effectuée que si ρ est une valeur. Dans le cas ou ρ est une application, une réduction par μ donne $\langle\!\langle \rho \rangle\!\rangle_{\tilde{\mu}x.\langle \langle\!\langle \pi \rangle\!\rangle | \alpha \rangle}$ qui représente la réduction de ρ dans le contexte $(\lambda x.\pi)\,\Box$. Afin d'illustrer ce propos, les réductions par $\to^{n}_{\bar{\beta}\mu\tilde{\mu}}$ et $\to^{v}_{\bar{\beta}\mu\tilde{\mu}}$ à partir de la commande $\langle\!\langle (\lambda x.\lambda y.y)\,((\lambda z.z)\,x') \rangle\!\rangle$ sont décrite par la figure 3.5.

3.3.2 Le langage d'urban

Urban et Bierman (2001) ont proposé un autre langage de termes de preuve pour le calcul des séquents classique. Leur approche est différente de celle de Curien et Herbelin : au lieu de passer par étapes du lambda-calcul pour la déduction naturelle intuitionniste au $\lambda\mu$-calcul pour la déduction naturelle classique et finalement au $\bar{\lambda}\mu\tilde{\mu}$-calcul pour le calcul des séquents classique, le point de départ de leur travail est le calcul des séquents classique pour lequel des constructeurs sont proposés, représentant exactement chaque règle d'inférence de LK. Un grande partie de cette étude consiste alors à comparer les différentes procédures d'élimination des coupures possibles, comme par exemple la procédure originale de Gentzen (1935). Notamment

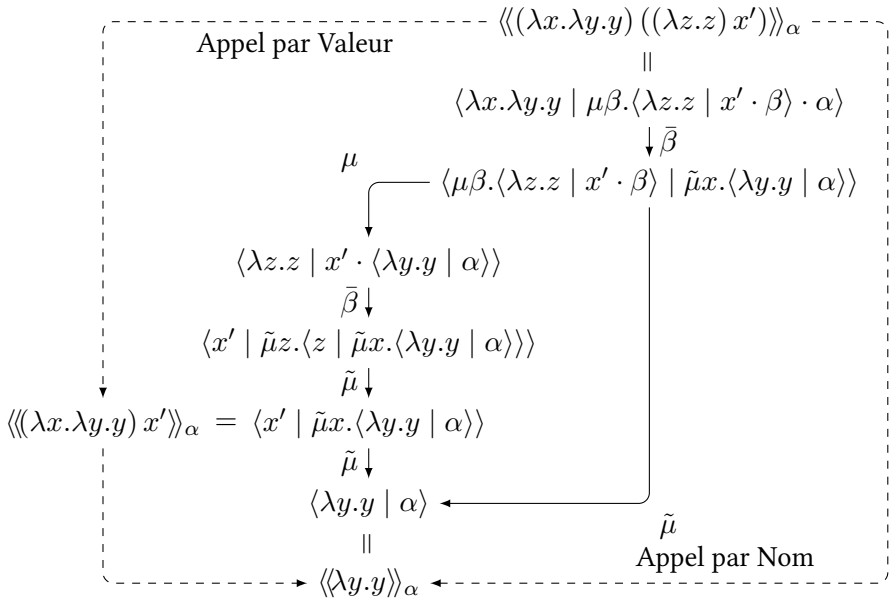


FIG. 3.5 – Appel par Nom et Appel par Valeur en $\bar{\lambda}\mu\tilde{\mu}$-calcul

deux procédures d'élimination des coupures sont proposées et leurs normalisations fortes sont démontrées parmi d'autres propriétés appropriées : la première procédure est exposée par Urban et Bierman (2001) et la seconde (qualifiée de *Gentzen-like*), par Urban (2001). Lengrand (2003) nomme $\lambda\xi$ le fragment implicatif du langage de termes d'Urban et le relie avec le $\bar{\lambda}\mu\tilde{\mu}$-calcul en plongeant l'un dans l'autre et inversement. En particulier Lengrand simule le $\bar{\lambda}\mu\tilde{\mu}$-calcul dans $\lambda\xi$, ce qui prouve que la normalisation forte du deuxième (démontrée par Urban (2000)) induit la normalisation forte du premier. Ce même fragment implicatif du langage d'Urban est renommé $\mathcal{X}$ et considéré dans le cadre non typé par Bakel *et al.* (2005). Son pouvoir expressif y est démontré puisque des interprétations du lambda-calcul, du $\lambda\mu$-calcul et du lambda-calcul à substitutions explicites $\lambda\mathbf{x}$ de Bloo et Rose (1995) y sont décrites. L'un des inconvénients du $\bar{\lambda}\mu\tilde{\mu}$-calcul comme outil décrivant les preuves de LK réside dans la bureaucratie introduite par le focus dans les dérivations de typage. Par exemple la dérivation de LK

$$\vee\mathrm{L}\,\dfrac{\Rightarrow\mathrm{R}\,\dfrac{\varphi_3 \vdash \varphi_1, \varphi_4 \qquad \varphi_2, \varphi_3 \vdash \varphi_4}{\varphi_1 \Rightarrow \varphi_2, \varphi_3 \vdash \varphi_4}}{\varphi_1 \Rightarrow \varphi_2 \vdash \varphi_3 \Rightarrow \varphi_4}$$

s'exprime dans le $\bar{\lambda}\mu\tilde{\mu}$-calcul par la dérivation de typage

$$\textsc{App}\ \frac{\textsc{Cmd}\ \dfrac{\textsc{Mu}\ \dfrac{\textsc{Abs}\ \dfrac{\textsc{Var}\ \dfrac{}{x : \varphi_1 \Rightarrow \varphi_2, y : \varphi_3 \vdash x : \varphi_1 \Rightarrow \varphi_2 \vdash \alpha : \varphi_4}}{\langle x | \pi \cdot e \rangle \triangleright x : \varphi_1 \Rightarrow \varphi_2, y : \varphi_3 \vdash \alpha : \varphi_4}}{x : \varphi_1 \Rightarrow \varphi_2, y : \varphi_3 \vdash \mu\alpha.\langle x | \pi \cdot e \rangle : \varphi_4 \vdash}}{x : \varphi_1 \Rightarrow \varphi_2 \vdash \lambda y.\mu\alpha.\langle x | \pi \cdot e \rangle : \varphi_3 \Rightarrow \varphi_4 \vdash} \qquad \frac{y : \varphi_3 \vdash e : \varphi_2 \vdash \alpha : \varphi_4 \qquad y : \varphi_3 \vdash \pi : \varphi_1 \vdash \alpha : \varphi_4}{y : \varphi_3 \vdash e \cdot \pi : \varphi_1 \Rightarrow \varphi_2 \vdash \alpha : \varphi_4}$$

dont les inférences Mu, Cmd et Var ne correspondent à aucune inférence de LK. Du point de vue logique, ce ne sont que des étapes de bureaucratie du typage du $\bar{\lambda}\mu\tilde{\mu}$-calcul.

Nous supposons donnés deux ensembles infinis dénombrables X et A de noms et de conoms. Contrairement aux variables et covariables du $\bar{\lambda}\mu\tilde{\mu}$-calcul et similairement aux noms du $\lambda\mu$-calcul, nous n'attribuerons pas de valeurs à ces noms et conoms. Plus précisément nous ne définirons pas pas de mécanisme de substitution qui les remplacerait par un quelconque objet, mis à part, respectivement, un autre nom ou un autre conom. Nous utiliserons les symboles x, y, z pour représenter des noms et les symboles α, β, γ pour représenter des conoms. Tout comme au chapitre 2, nous supposons aussi donnée une algèbre de termes $T_\alpha(X)$ que nous appellerons des termes du premier ordre. Les symboles x, y, z représenteront des variables du premier ordre et le symbole t représentera des termes du premier ordre. Enfin nous supposons donné une signature des prédicats $\mathcal{A}$ nous permettant ainsi de manipuler les formules de l'ensemble $\mathcal{F}_{\mathcal{A}T_\alpha(X)}$. Les symboles φ, ψ représenteront des formules. Un contexte de typage à gauche est un ensemble Γ de couples $x : \varphi$ tel qu'il n'existe qu'un seul φ tel que $x : \varphi \in \Gamma$. Un contexte de typage à droite est un ensemble Δ de couples $\alpha : \varphi$ tel qu'il n'existe qu'un seul φ tel que $\alpha : \varphi \in \Gamma$. L'ensemble des termes de preuve est le langage hors-contexte associé à la grammaire en forme de Backus-Naur suivante.

$$\begin{aligned} M, N \quad ::= \quad & \mathsf{Ax}(x, a) \mid \mathsf{Cut}(\widehat{a}M, \widehat{x}N) \mid \mathsf{False}_L(x) \mid \mathsf{True}_R(a) \\ \mid\ & \mathsf{Not}_R(\widehat{x}M, a) \mid \mathsf{Not}_L(\widehat{a}M, x) \mid \mathsf{Imp}_R(\widehat{x}\widehat{a}M, b) \mid \mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, y) \\ \mid\ & \mathsf{And}_R(\widehat{a}M, \widehat{b}N, c) \mid \mathsf{And}_L(\widehat{x}\widehat{y}M, z) \mid \mathsf{Or}_R(\widehat{a}\widehat{b}M, c) \mid \mathsf{Or}_L(\widehat{x}M, \widehat{y}N, z) \\ \mid\ & \mathsf{Exists}_R(\widehat{a}M, \mathsf{t}, b) \mid \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, y) \mid \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M, b) \mid \mathsf{Forall}_L(\widehat{x}M, \mathsf{t}, y) \end{aligned}$$

Les symboles M, N seront utilisés pour représenter des termes de preuve. Par souci de simplicité nous dirons « terme » pour les termes de preuve et « terme du premier ordre » pour les termes de $T_\alpha(X)$. La notion de position se définit pour les termes de la façon attendue.

Définition 3.3.1 (Position et Occurrence dans un Terme). *Une position est une suite finie d'entiers naturels non nuls, potentiellement vide (la suite vide étant notée ε).*

Pour tout terme M et toute position ν, le terme $M_{|\nu}$ est defini par

$$
\begin{aligned}
M_{|\nu} &= M \qquad \textit{si } \nu = \varepsilon \ ; \\
M_{|\nu} &= M_{i|\nu'} \quad \textit{si } \nu = i, \nu' \textit{ avec } i \in \{1,2\} \\
&\qquad\qquad \textit{et } M \in \{\mathsf{Cut}(\widehat{a}M_1, \widehat{x}M_2), \mathsf{Imp}_L(\widehat{x}M_1, \widehat{a}M_2, y), \\
&\qquad\qquad\qquad \mathsf{And}_R(\widehat{a}M_1, \widehat{b}M_2, c), \mathsf{Or}_L(\widehat{x}M_1, \widehat{y}M_2, z)\} \ ; \\
M_{|\nu} &= N_{|\nu'} \quad \textit{si } \nu = 1, \nu' \textit{ avec } M \in \{\mathsf{Not}_R(\widehat{x}N, a), \mathsf{Not}_L(\widehat{a}N, x), \\
&\qquad\qquad\qquad \mathsf{Imp}_R(\widehat{x}\widehat{a}N, b), \mathsf{And}_L(\widehat{x}\widehat{y}N, z), \mathsf{Or}_R(\widehat{a}\widehat{b}N, c), \\
&\mathsf{Exists}_R(\widehat{a}N, \mathsf{t}, b), \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}N, y), \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}N, b), \mathsf{Forall}_L(\widehat{x}N, \mathsf{t}, y)\} \ ;
\end{aligned}
$$

Remarquons que $M_{|\nu}$ n'est pas défini pour tout terme M et toute position ν. Les positions d'un terme M sont les positions ν pour lesquelles $M_{|\nu}$ est effectivement défini. Encore une fois la notion de position définit une notion d'occurrence : lorsque $N = M_{|\nu}$, nous dirons que M admet une occurrence de N à la position ν. Enfin si x est un nom, nous dirons que M admet une occurrence de x à la position ν si $M_{|\nu}$ est de la forme $\mathsf{Ax}(x,a)$, $\mathsf{False}_L(x)$, $\mathsf{Not}_L(\widehat{a}M, x)$, $\mathsf{Imp}_L(\widehat{y}M, \widehat{a}N, x)$, $\mathsf{And}_L(\widehat{y}\widehat{z}M, x)$, $\mathsf{Or}_L(\widehat{y}M, \widehat{z}N, x)$, $\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}M, x)$ ou $\mathsf{Forall}_L(\widehat{y}M, \mathsf{t}, x)$. Si a est un conom, nous dirons que M admet une occurrence de a à la position ν si $M_{|\nu}$ est de la forme $\mathsf{Ax}(x,a)$, $\mathsf{True}_R(a)$, $\mathsf{Not}_R(\widehat{x}M, a)$, $\mathsf{Imp}_R(\widehat{x}\widehat{b}M, a)$, $\mathsf{And}_R(\widehat{b}M, \widehat{c}N, a)$, $\mathsf{Or}_R(\widehat{b}\widehat{c}M, a)$, $\mathsf{Exists}_R(\widehat{b}M, \mathsf{t}, a)$, $\mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}M, a)$.

Comme le symbole λ du lambda-calcul, l'accent circonflexe est un lieur. Il permet de lier noms, conoms et variables du premier ordre. Nous dirons qu'une occurrence de a en position ν dans M est liée si

$$
\begin{aligned}
M_{|\nu_1} &= \mathsf{Cut}(\widehat{a}M_1, \widehat{x}M_2) && \text{et } \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{Not}_L(\widehat{a}M_1, x) && \text{et } \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{Imp}_R(\widehat{x}\widehat{a}M_1, b) && \text{et } \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{Imp}_L(\widehat{x}M_1, \widehat{a}M_2, y) && \text{et } \nu = \nu_1, 2, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{And}_R(\widehat{a}M_1, \widehat{b}M_2, c) && \text{et } \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{And}_R(\widehat{b}M_1, \widehat{a}M_2, c) && \text{et } \nu = \nu_1, 2, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{Or}_R(\widehat{a}\widehat{b}M_1, c) && \text{et } \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{Or}_R(\widehat{b}\widehat{a}M_1, c) && \text{et } \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{Exists}_R(\widehat{a}M_1, \mathsf{t}, b) && \text{et } \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} &= \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M_1, b) && \text{et } \nu = \nu_1, 1, \nu_2 \ ;
\end{aligned}
$$

et qu'une occurrence de x en position ν dans M est liée si

$$\begin{array}{lll}
M_{|\nu_1} = \mathsf{Cut}(\widehat{a}M_1, \widehat{x}M_2) & \text{et} & \nu = \nu_1, 2, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{Not}_R(\widehat{x}M_1, a) & \text{et} & \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{Imp}_R(\widehat{x}\widehat{a}M_1, b) & \text{et} & \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{Imp}_L(\widehat{x}M_1, \widehat{a}M_2, y) & \text{et} & \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{And}_L(\widehat{x}\widehat{y}M_1, z) & \text{et} & \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{And}_L(\widehat{y}\widehat{x}M_1, z) & \text{et} & \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{Or}_L(\widehat{x}M, \widehat{y}M_1, z) & \text{et} & \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{Or}_L(\widehat{y}M, \widehat{x}M_1, z) & \text{et} & \nu = \nu_1, 2, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M_1, y) & \text{et} & \nu = \nu_1, 1, \nu_2 \ ; \\
M_{|\nu_1} = \mathsf{Forall}_L(\widehat{x}M_1, \mathsf{t}, y) & \text{et} & \nu = \nu_1, 1, \nu_2 \ .
\end{array}$$

Nous dirons qu'une occurrence d'un nom ou d'un conom est libre si ce n'est pas une occurrence liée. L'ensemble des symboles liés d'un terme M, que nous noterons $\mathcal{BS}(M)$ et l'ensemble des symboles libres d'un terme M, que nous noterons $\mathcal{FS}(M)$, sont définis inductivement par la figure 3.6. Nous considérons les termes modulo alpha-conversion sur les noms et conoms et variables du premier ordre. Par exemple

$$\mathsf{Cut}(\widehat{a}M, \widehat{x}N) \quad \equiv_\alpha \quad \mathsf{Cut}(\widehat{b}M[b/a], \widehat{x}N)$$

dès que $b \notin \mathcal{FS}(M) \cup \mathcal{BS}(M)$ et $a \notin \mathcal{BS}(M)$ et où $M[b/a]$ représente le terme M où toute occurrence (obligatoirement libre) du symbole a est remplacée par le symbole b. Ce remplacement ainsi que le remplacement similaire pour les noms est formellement défini par la figure 3.8. Alors l'alpha-conversion, notée $\equiv_\alpha$, est la plus petite congruence contenant la relation de la figure 3.9. Les règles de typage sont décrites par la figure 3.10. Tout comme dans l'article de Urban et Bierman (2001), les occurrences du symbole « , » dans les règles de typage ont des significations différentes si elles apparaissent dans une prémisse ou dans une conclusion. Dans le premier cas, elles dénotent une union disjointe. Dans le deuxième cas, elles dénotent une union. Par exemple l'inférence

$$\frac{M\triangleright \vdash a : \varphi, c : \varphi \wedge \psi \qquad N\triangleright \vdash b : \psi, c : \varphi \wedge \psi}{\mathsf{And}_R(\widehat{a}M, \widehat{b}N, c)\triangleright \vdash c : \varphi \wedge \psi}$$

est bien une instance de la règle de typage $\wedge$R. Logiquement elle contient une instance des règles ContrR puis $\wedge$R de LK.

$$\textsc{ContrR}\ \frac{\wedge\text{R}\ \dfrac{\vdash \varphi, \varphi \wedge \psi \qquad \vdash \psi, \varphi \wedge \psi}{\vdash \varphi \wedge \psi, \varphi \wedge \psi}}{\vdash \varphi \wedge \psi}$$

Cette étape de contraction *doit* être effectuée dès que le conom c apparaît libre dans M ou N. En effet, le typage vérifie l'invariant suivant : si $M \triangleright \Gamma \vdash \Delta$ est dérivable, alors tout nom libre de M apparaît dans Γ est tout conom libre de M apparaît dans

$$
\begin{array}{lcl}
\mathcal{BS}(\mathsf{Ax}(x,a)) & = & \emptyset \\
\mathcal{BS}(\mathsf{Cut}(\widehat{a}M,\widehat{x}N)) & = & \mathcal{BS}(M)\cup\mathcal{BS}(N) \\
 & & \qquad\cup\{a,x\} \\
\mathcal{BS}(\mathsf{False}_L(x)) & = & \emptyset \\
\mathcal{BS}(\mathsf{True}_R(a)) & = & \emptyset \\
\mathcal{BS}(\mathsf{Not}_R(\widehat{x}M,a)) & = & \mathcal{BS}(M)\cup\{x\} \\
\mathcal{BS}(\mathsf{Not}_L(\widehat{a}M,x)) & = & \mathcal{BS}(M)\cup\{a\} \\
\mathcal{BS}(\mathsf{Imp}_R(\widehat{x}\widehat{a}M,b)) & = & \mathcal{BS}(M)\cup\{a,x\} \\
\mathcal{BS}(\mathsf{Imp}_L(\widehat{x}M,\widehat{a}N,y)) & = & \mathcal{BS}(M) \\
 & & \cup\mathcal{BS}(N)\cup\{a,x\} \\
\mathcal{BS}(\mathsf{And}_R(\widehat{a}M,\widehat{b}N,c)) & = & \mathcal{BS}(M) \\
 & & \cup\mathcal{BS}(N)\cup\{a,b\} \\
\mathcal{BS}(\mathsf{And}_L(\widehat{x}\widehat{y}M,z)) & = & \mathcal{BS}(M)\cup\{x,y\} \\
\mathcal{BS}(\mathsf{Or}_R(\widehat{a}\widehat{b}M,c)) & = & \mathcal{BS}(M)\cup\{a,b\} \\
\mathcal{BS}(\mathsf{Or}_L(\widehat{x}M,\widehat{y}N,z)) & = & \mathcal{BS}(M) \\
 & & \cup\mathcal{BS}(N)\cup\{x,y\} \\
\mathcal{BS}(\mathsf{Exists}_R(\widehat{a}M,\mathsf{t},b)) & = & \mathcal{BS}(M)\cup\{a\} \\
\mathcal{BS}(\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M,y)) & = & \mathcal{BS}(M)\cup\{x,\mathsf{x}\} \\
\mathcal{BS}(\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M,b)) & = & \mathcal{BS}(M)\cup\{a,\mathsf{x}\} \\
\mathcal{BS}(\mathsf{Forall}_L(\widehat{x}M,\mathsf{t},y)) & = & \mathcal{BS}(M)\cup\{x\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{FS}(\mathsf{Ax}(x,a)) & = & \{x,a\} \\
\mathcal{FS}(\mathsf{Cut}(\widehat{a}M,\widehat{x}N)) & = & (\mathcal{FS}(M)\backslash\{a\}) \\
 & & \qquad\cup(\mathcal{FS}(N)\backslash\{x\}) \\
\mathcal{FS}(\mathsf{False}_L(x)) & = & \{x\} \\
\mathcal{FS}(\mathsf{True}_R(a)) & = & \{a\} \\
\mathcal{FS}(\mathsf{Not}_R(\widehat{x}M,a)) & = & (\mathcal{FS}(M)\backslash\{x\})\cup\{a\} \\
\mathcal{FS}(\mathsf{Not}_L(\widehat{a}M,x)) & = & (\mathcal{FS}(M)\backslash\{a\})\cup\{x\} \\
\mathcal{FS}(\mathsf{Imp}_R(\widehat{x}\widehat{a}M,b)) & = & (\mathcal{FS}(M)\backslash\{x,a\})\cup\{b\} \\
\mathcal{FS}(\mathsf{Imp}_L(\widehat{x}M,\widehat{a}N,y)) & = & (\mathcal{FS}(M)\backslash\{x\}) \\
 & & \qquad\cup(\mathcal{FS}(N)\backslash\{a\})\cup\{y\} \\
\mathcal{FS}(\mathsf{And}_R(\widehat{a}M,\widehat{b}N,c)) & = & (\mathcal{FS}(M)\backslash\{x\}) \\
 & & \qquad\cup(\mathcal{FS}(N)\backslash\{b\})\cup\{c\} \\
\mathcal{FS}(\mathsf{And}_L(\widehat{x}\widehat{y}M,z)) & = & (\mathcal{FS}(M)\backslash\{x,y\})\cup\{z\} \\
\mathcal{FS}(\mathsf{Or}_R(\widehat{a}\widehat{b}M,c)) & = & (\mathcal{FS}(M)\backslash\{a,b\})\cup\{c\} \\
\mathcal{FS}(\mathsf{Or}_L(\widehat{x}M,\widehat{y}N,z)) & = & (\mathcal{FS}(M)\backslash\{x\}) \\
 & & \qquad\cup(\mathcal{FS}(N)\backslash\{y\})\cup\{a\} \\
\mathcal{FS}(\mathsf{Exists}_R(\widehat{a}M,\mathsf{t},b)) & = & (\mathcal{FS}(M)\backslash\{a\})\cup\{b\}\cup\mathcal{V}(\mathsf{t}) \\
\mathcal{FS}(\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M,y)) & = & (\mathcal{FS}(M)\backslash\{x,\mathsf{x}\})\cup\{y\} \\
\mathcal{FS}(\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M,b)) & = & (\mathcal{FS}(M)\backslash\{a,\mathsf{x}\})\cup\{b\} \\
\mathcal{FS}(\mathsf{Forall}_L(\widehat{x}M,\mathsf{t},y)) & = & (\mathcal{FS}(M)\backslash\{x\})\cup\{y\}\cup\mathcal{V}(\mathsf{t})
\end{array}
$$

FIG. 3.6 – Symboles liés et libres d'un terme

$$
\begin{array}{lcl}
\mathsf{Ax}(x,a)[b/a] & = & \mathsf{Ax}(x,b) \\
\mathsf{Cut}(\widehat{c}M,\widehat{x}N)[b/a] & = & \mathsf{Cut}(\widehat{c}M[b/a],\widehat{x}N[b/a]) \\
\mathsf{False}_L(x)[b/a] & = & \mathsf{False}_L(x) \\
\mathsf{True}_R(c)[b/a] & = & \mathsf{True}_R(c[b/a]) \\
\mathsf{Not}_R(\widehat{x}M,c)[b/a] & = & \mathsf{Not}_R(\widehat{x}M[b/a],c[b/a]) \\
\mathsf{Not}_L(\widehat{c}M,x)[b/a] & = & \mathsf{Not}_L(\widehat{c}M[b/a],x) \\
\mathsf{Imp}_R(\widehat{x}\widehat{d}M,c)[b/a] & = & \mathsf{Imp}_R(\widehat{x}\widehat{d}M[b/a],c[b/a]) \\
\mathsf{Imp}_L(\widehat{x}M,\widehat{c}N,y)[b/a] & = & \mathsf{Imp}_L(\widehat{x}M,\widehat{c}N[b/a],y) \\
\mathsf{And}_R(\widehat{d}M,\widehat{e}N,c)[b/a] & = & \mathsf{And}_R(\widehat{d}M[b/a],\widehat{d}N[b/a],c[b/a]) \\
\mathsf{And}_L(\widehat{x}\widehat{y}M,z)[b/a] & = & \mathsf{And}_L(\widehat{x}\widehat{y}M[b/a],z) \\
\mathsf{Or}_R(\widehat{d}\widehat{e}M,c)[b/a] & = & \mathsf{Or}_R(\widehat{d}\widehat{e}M[b/a],c[b/a]) \\
\mathsf{Or}_L(\widehat{x}M,\widehat{y}N,z)[b/a] & = & \mathsf{Or}_L(\widehat{x}M[b/a],\widehat{y}N[b/a],z) \\
\mathsf{Exists}_R(\widehat{d}M,\mathsf{t},c)[b/a] & = & \mathsf{Exists}_R(\widehat{d}M[b/a],\mathsf{t},c[b/a]) \\
\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M,y)[b/a] & = & \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M[b/a],y) \\
\mathsf{Forall}_R(\widehat{d}\widehat{\mathsf{x}}M,c)[b/a] & = & \mathsf{Forall}_R(\widehat{d}\widehat{\mathsf{x}}M[b/a],c[b/a]) \\
\mathsf{Forall}_L(\widehat{x}M,\mathsf{t},y)[b/a] & = & \mathsf{Forall}_L(\widehat{x}M[b/a],\mathsf{t},y) \\
\text{où} \quad c[b/a] & = & \begin{cases} b & \text{si } c=a \\ c & \text{sinon.} \end{cases}
\end{array}
$$

FIG. 3.7 – Remplacement pour les conoms

$$
\begin{array}{lcl}
\mathsf{Ax}(z,a)[y/x] & = & \mathsf{Ax}(z[y/x],a) \\
\mathsf{Cut}(\widehat{a}M,\widehat{z}N)[y/x] & = & \mathsf{Cut}(\widehat{a}M[y/x],\widehat{z}N[y/x]) \\
\mathsf{False}_L(z)[y/x] & = & \mathsf{False}_L(z[y/x]) \\
\mathsf{True}_R(a)[y/x] & = & \mathsf{True}_R(a) \\
\mathsf{Not}_R(\widehat{z}M,a)[y/x] & = & \mathsf{Not}_R(\widehat{z}M[y/x],a) \\
\mathsf{Not}_L(\widehat{a}M,z)[y/x] & = & \mathsf{Not}_L(\widehat{a}M[y/x],z[y/x]) \\
\mathsf{Imp}_R(\widehat{z}\widehat{a}M,b)[y/x] & = & \mathsf{Imp}_R(\widehat{z}\widehat{a}M[y/x],b) \\
\mathsf{Imp}_L(\widehat{z_1}M,\widehat{a}N,z_2)[y/x] & = & \mathsf{Imp}_L(\widehat{z_1}M[y/x],\widehat{a}N[y/x],z_2[y/x]) \\
\mathsf{And}_R(\widehat{a}M,\widehat{b}N,c)[y/x] & = & \mathsf{And}_R(\widehat{a}M[y/x],\widehat{b}N[y/x],c) \\
\mathsf{And}_L(\widehat{z_1}\widehat{z_2}M,z_3)[y/x] & = & \mathsf{And}_L(\widehat{z_1}\widehat{z_2}M[y/x],z_3[y/x]) \\
\mathsf{Or}_R(\widehat{a}\widehat{b}M,c)[y/x] & = & \mathsf{Or}_R(\widehat{a}\widehat{b}M[y/x],c) \\
\mathsf{Or}_L(\widehat{z_1}M,\widehat{z_2}N,z_3)[y/x] & = & \mathsf{Or}_L(\widehat{z_1}M[y/x],\widehat{z_2}N[y/x],z_3[y/x]) \\
\mathsf{Exists}_R(\widehat{a}M,\mathsf{t},b)[y/x] & = & \mathsf{Exists}_R(\widehat{a}M[y/x],\mathsf{t},b) \\
\mathsf{Exists}_L(\widehat{z_1}\widehat{\mathsf{x}}M,z_2)[y/x] & = & \mathsf{Exists}_L(\widehat{z_1}\widehat{\mathsf{x}}M[y/x],z_2[y/x]) \\
\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M,b)[y/x] & = & \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M[y/x],b) \\
\mathsf{Forall}_L(\widehat{z_1}M,\mathsf{t},z_2)[y/x] & = & \mathsf{Forall}_L(\widehat{z_1}M[y/x],\mathsf{t},z_2[y/x]) \\
\text{où} \quad z[y/x] & = & \begin{cases} y & \text{si } z=x \\ z & \text{sinon.} \end{cases}
\end{array}
$$

FIG. 3.8 – Remplacement pour les noms

$$
\begin{aligned}
\mathsf{Cut}(\widehat{a}M, \widehat{x}N) &\equiv_\alpha \mathsf{Cut}(\widehat{a}M, \widehat{y}N[y/x]) \\
&\quad \text{si } y \notin \mathcal{BS}(N) \cup \mathcal{FS}(N) \text{ et } x \notin \mathcal{BS}(N) \\
\mathsf{Cut}(\widehat{a}M, \widehat{x}N) &\equiv_\alpha \mathsf{Cut}(\widehat{b}M[b/a], \widehat{x}N) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{Not}_R(\widehat{x}M, a) &\equiv_\alpha \mathsf{Not}_R(\widehat{y}M[y/x], a) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M) \\
\mathsf{Not}_L(\widehat{a}M, x) &\equiv_\alpha \mathsf{Not}_L(\widehat{b}M[b/a], x) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{Imp}_R(\widehat{x}\widehat{a}M, b) &\equiv_\alpha \mathsf{Imp}_R(\widehat{y}\widehat{a}M[y/x], b) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M) \\
\mathsf{Imp}_R(\widehat{x}\widehat{a}M, c) &\equiv_\alpha \mathsf{Imp}_R(\widehat{x}\widehat{b}M[b/a], c) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, z) &\equiv_\alpha \mathsf{Imp}_L(\widehat{y}M[y/x], \widehat{a}N, z) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M) \\
\mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, y) &\equiv_\alpha \mathsf{Imp}_L(\widehat{x}M, \widehat{b}N[b/a], y) \\
&\quad \text{si } b \notin \mathcal{BS}(N) \cup \mathcal{FS}(N) \text{ et } a \notin \mathcal{BS}(N) \\
\mathsf{And}_R(\widehat{a}M, \widehat{c}N, d) &\equiv_\alpha \mathsf{And}_R(\widehat{b}M[b/a], \widehat{c}N, d) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{And}_R(\widehat{c}M, \widehat{a}N, d) &\equiv_\alpha \mathsf{And}_R(\widehat{c}M, \widehat{b}N[b/a], d) \\
&\quad \text{si } b \notin \mathcal{BS}(N) \cup \mathcal{FS}(N) \text{ et } a \notin \mathcal{BS}(N) \\
\mathsf{And}_L(\widehat{x}\widehat{z_1}M, z_2) &\equiv_\alpha \mathsf{And}_L(\widehat{y}\widehat{z_1}M[y/x], z_2) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M) \\
\mathsf{And}_L(\widehat{z_1}\widehat{x}M, z_2) &\equiv_\alpha \mathsf{And}_L(\widehat{z_1}\widehat{y}M[y/x], z_2) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M) \\
\mathsf{Or}_R(\widehat{a}\widehat{c}M, d) &\equiv_\alpha \mathsf{Or}_R(\widehat{b}\widehat{c}M[b/a], d) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{Or}_R(\widehat{c}\widehat{a}M, d) &\equiv_\alpha \mathsf{Or}_R(\widehat{c}\widehat{b}M[b/a], d) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{Or}_L(\widehat{x}M, \widehat{z_1}N, z_2) &\equiv_\alpha \mathsf{Or}_L(\widehat{y}M[y/x], \widehat{z_1}N, z_2) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M) \\
\mathsf{Or}_L(\widehat{z_1}M, \widehat{x}N, z_2) &\equiv_\alpha \mathsf{Or}_L(\widehat{z_1}M, \widehat{y}N[y/x], z_2) \\
&\quad \text{si } y \notin \mathcal{BS}(N) \cup \mathcal{FS}(N) \text{ et } x \notin \mathcal{BS}(N) \\
\mathsf{Exists}_R(\widehat{a}M, \mathsf{t}, c) &\equiv_\alpha \mathsf{Exists}_R(\widehat{b}M[b/a], \mathsf{t}, c) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, z) &\equiv_\alpha \mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}M[y/x], z) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M) \\
\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M, c) &\equiv_\alpha \mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}M[b/a], c) \\
&\quad \text{si } b \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } a \notin \mathcal{BS}(M) \\
\mathsf{Forall}_L(\widehat{x}M, \mathsf{t}, z) &\equiv_\alpha \mathsf{Forall}_L(\widehat{y}M[y/x], \mathsf{t}, z) \\
&\quad \text{si } y \notin \mathcal{BS}(M) \cup \mathcal{FS}(M) \text{ et } x \notin \mathcal{BS}(M)
\end{aligned}
$$

FIG. 3.9 – Alpha-conversion

$$\textsc{Axiom}\ \frac{}{\mathsf{Ax}(x,a) \triangleright \Gamma, x : \varphi \vdash a : \varphi, \Delta}$$

$$\textsc{Cut}\ \frac{M \triangleright \Gamma \vdash a : \varphi, \Delta \qquad N \triangleright \Gamma, x : \varphi \vdash \Delta}{\mathsf{Cut}(\widehat{a}M, \widehat{x}N) \triangleright \Gamma \vdash \Delta}$$

$$\neg\mathrm{R}\ \frac{M \triangleright \Gamma, x : A \vdash \Delta}{\mathsf{Not}_R(\widehat{x}M, a) \triangleright \Gamma \vdash a : \neg A, \Delta}$$

$$\neg\mathrm{L}\ \frac{M \triangleright \Gamma \vdash a : A, \Delta}{\mathsf{Not}_L(\widehat{a}M, x) \triangleright \Gamma, x : \neg A \vdash \Delta}$$

$$\bot\mathrm{L}\ \frac{}{\mathsf{False}_L(x) \triangleright \Gamma, x : \bot \vdash \Delta}$$

$$\top\mathrm{R}\ \frac{}{\mathsf{True}_R(a) \triangleright \Gamma \vdash a : \top, \Delta}$$

$$\wedge\mathrm{R}\ \frac{M \triangleright \Gamma \vdash a : \varphi_1, \Delta \qquad N \triangleright \Gamma \vdash b : \varphi_2, \Delta}{\mathsf{And}_R(\widehat{a}M, \widehat{b}N, c) \triangleright \Gamma \vdash c : \varphi_1 \wedge \varphi_2, \Delta}$$

$$\wedge\mathrm{L}\ \frac{M \triangleright \Gamma, x : \varphi_1, y : \varphi_2 \vdash \Delta}{\mathsf{And}_L(\widehat{x}\widehat{y}M, z) \triangleright \Gamma, z : \varphi_1 \wedge \varphi_2 \vdash \Delta}$$

$$\vee\mathrm{R}\ \frac{M \triangleright \Gamma \vdash a : \varphi_1, b : \varphi_2, \Delta}{\mathsf{Or}_R(\widehat{a}\widehat{b}M, c) \triangleright \Gamma \vdash c : \varphi_1 \vee \varphi_2, \Delta}$$

$$\vee\mathrm{L}\ \frac{M \triangleright \Gamma, x : \varphi_1 \vdash \Delta \qquad N \triangleright \Gamma, y : \varphi_2 \vdash \Delta}{\mathsf{Or}_L(\widehat{x}M, \widehat{y}N, z) \triangleright \Gamma, z : \varphi_1 \vee \varphi_2 \vdash \Delta}$$

$$\Rightarrow\mathrm{R}\ \frac{M \triangleright \Gamma, x : \varphi_1 \vdash a : \varphi_2, \Delta}{\mathsf{Imp}_R(\widehat{x}\widehat{a}M, b) \triangleright \Gamma \vdash b : \varphi_1 \Rightarrow \varphi_2, \Delta}$$

$$\Rightarrow\mathrm{L}\ \frac{M \triangleright \Gamma, x : \varphi_2 \vdash \Delta \qquad N \triangleright \Gamma \vdash a : \varphi_1, \Delta}{\mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, y) \triangleright \Gamma, y : \varphi_1 \Rightarrow \varphi_2 \vdash \Delta}$$

$$\exists\mathrm{L}\ \frac{M \triangleright \Gamma, x : \varphi \vdash \Delta}{\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, y) \triangleright \Gamma, y : \exists \mathsf{x}.\varphi \vdash \Delta}\ \mathsf{x} \notin \mathcal{FS}(\Gamma, \Delta)$$

$$\exists\mathrm{R}\ \frac{M \triangleright \Gamma \vdash a : \varphi[\mathsf{t}/\mathsf{x}], \Delta}{\mathsf{Exists}_R(\widehat{a}M, t, b) \triangleright \Gamma \vdash b : \exists \mathsf{x}.\varphi, \Delta}$$

$$\forall\mathrm{R}\ \frac{M \triangleright \Gamma \vdash a : \varphi, \Delta}{\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M, b) \triangleright \Gamma \vdash b : \forall \mathsf{x}.\varphi, \Delta}\ \mathsf{x} \notin \mathcal{FS}(\Gamma, \Delta)$$

$$\forall\mathrm{L}\ \frac{M \triangleright \Gamma, x : \varphi[\mathsf{t}/\mathsf{x}] \vdash \Delta}{\mathsf{Forall}_L(\widehat{x}M, t, y) \triangleright \Gamma, y : \forall \mathsf{x}.\varphi \vdash \Delta}$$

FIG. 3.10 – Typage pour le calcul d'Urban

Δ. Si c apparaît libre dans M ou N, c'est bien qu'il faut conserver $c : \varphi \wedge \psi$ dans le contexte de typage pour typer M ou N. Au contraire si c n'apparaît pas libre dans M ou N, c'est que l'étape de contraction est superflue. Nous dirons alors que $\mathsf{And}_R(\widehat{a}M, \widehat{b}N, c)$ introduit fraîchement le conom c.

Définition 3.3.2 (Introduction Fraîche). *Si l'unique occurrence d'un nom x (respectivement d'un conom a) dans un terme M est à la position ε, nous dirons que M introduit fraîchement x (respectivement a).*

Notons qu'encore une fois le typage est dirigé par la syntaxe. La correspondance entre une dérivation de typage dans le calcul d'Urban et LK est immédiat. Mis à part les contractions que nous venons de traîter, une règle de LK correspond à la règle de typage du même nom. La procédure d'élimination des coupures exposée par Urban et Bierman (2001) est décrite par la figure 3.11. Les coupures sont représentées par le constructeur $\mathsf{Cut}(\_, \_)$ et par la règle de typage Cut. Elles sont classées en deux catégories. Tout d'abord les coupures logiques sont de la forme $\mathsf{Cut}(\widehat{a}M, \widehat{x}N)$ ou M et N introduisent fraîchement respectivement a et x. Les autres coupures sont appelées des coupures commutatives et déclenchent un mécanisme de substitution (bien que le terme « substitution » ne soit pas totalement approprié) qui déplace la coupure en profondeur jusqu'àux positions introduisant le nom ou conom dont il est question. Ce mécanisme est décrit par les figures 3.12 et 3.13.

Cette procédure d'élimination des coupures vérifie les propriétés de subject reduction et de normalisation forte, comme démontré par Urban et Bierman (2001); Urban (2000). La preuve de normalisation forte est hardue mais néanmoins formalisée en Isabelle/HOL (Urban et Zhu, 2008).

Propriété 3.3.1 (Subject Reduction). *S'il existe une dérivation de typage de $M \triangleright \Gamma \vdash \Delta$ et $M \to^* N$, alors il existe une dérivation de typage de $N \triangleright \Gamma \vdash \Delta$.*

Propriété 3.3.2 (Normalisation Forte). *La procédure d'élimination des coupures du calcul d'Urban est fortement normalisante.*

Par contre et à l'instar du $\bar{\lambda}\mu\tilde{\mu}$-calcul, cette élimination des coupures n'est ni confluente ni consistante puisque, pour tous termes M et N, il existe un nom x et un conom a frais (c'est-à-dire tels que $a \notin \mathcal{FS}(M)$ et $x \notin \mathcal{FS}(N)$) et

$$M \quad \leftarrow \quad \mathsf{Cut}(\widehat{a}M, \widehat{x}N) \quad \rightarrow \quad N$$

Coupures logiques :

$$
\begin{array}{rcll}
\mathsf{Cut}(\widehat{a}M, \widehat{x}\mathsf{Ax}(x,b)) & \to & M[a \mapsto b] & \text{si } M \text{ introduit fraîchement } a \\
\mathsf{Cut}(\widehat{a}\mathsf{Ax}(y,a), \widehat{x}M) & \to & M[x \mapsto y] & \text{si } M \text{introduit fraîchement } x \\
\mathsf{Cut}(\widehat{a}\mathsf{True}_R(a), \widehat{x}M) & \to & M & \text{si } M \text{introduit fraîchement } x \\
\mathsf{Cut}(\widehat{a}M, \widehat{x}\mathsf{False}_L(x)) & \to & M & \text{si } M \text{introduit fraîchement } a
\end{array}
$$

$$
\mathsf{Cut}(\widehat{a}\mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, a), \widehat{x}\mathsf{And}_L(\widehat{y}\widehat{z}N, x)) \to \begin{cases} \mathsf{Cut}(\widehat{b}M_1, \widehat{y}\mathsf{Cut}(\widehat{c}M_2, \widehat{z}N)) \\ \mathsf{Cut}(\widehat{c}M_2, \widehat{z}\mathsf{Cut}(\widehat{b}M_1, \widehat{y}N)) \end{cases}
$$

si $\mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, a)$ introduit fraîchement a
et $\mathsf{And}_L(\widehat{y}\widehat{z}N, x)$ introduit fraîchement x

$$
\mathsf{Cut}(\widehat{a}\mathsf{Or}_R(\widehat{b}\widehat{c}M, a), \widehat{x}\mathsf{Or}_L(\widehat{y}N_1, \widehat{z}N_2, x)) \to \begin{cases} \mathsf{Cut}(\widehat{b}\mathsf{Cut}(\widehat{c}M, \widehat{z}N_2), \widehat{y}N_1) \\ \mathsf{Cut}(\widehat{c}\mathsf{Cut}(\widehat{b}M, \widehat{y}N_1), \widehat{z}N_2) \end{cases}
$$

si $\mathsf{Or}_R(\widehat{b}\widehat{c}M, a)$ introduit fraîchement a
et $\mathsf{Or}_L(\widehat{y}N_1, \widehat{z}N_2, x)$ introduit fraîchement x

$$
\mathsf{Cut}(\widehat{a}\mathsf{Imp}_R(\widehat{x}\widehat{b}M, a), \widehat{y}\mathsf{Imp}_L(\widehat{z}N_1, \widehat{c}N_2, y)) \to \begin{cases} \mathsf{Cut}(\widehat{b}\mathsf{Cut}(\widehat{c}N_2, \widehat{x}M), \widehat{z}N_1) \\ \mathsf{Cut}(\widehat{c}N_2, \widehat{x}\mathsf{Cut}(\widehat{b}M, \widehat{z}N_1)) \end{cases}
$$

si $\mathsf{Imp}_R(\widehat{x}\widehat{b}M, a)$ introduit fraîchement a
et $\mathsf{Imp}_L(\widehat{z}N_1, \widehat{c}N_2, y)$ introduit fraîchement y

$$
\mathsf{Cut}(\widehat{a}\mathsf{Exists}_R(\widehat{b}M, t, a), \widehat{x}\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}N, x)) \to \mathsf{Cut}(\widehat{b}M, \widehat{y}N[\mathsf{x} := t])
$$

si $\mathsf{Exists}_R(\widehat{b}M, t, a)$ introduit fraîchement a
et $\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}N, x)$ introduit fraîchement x

$$
\mathsf{Cut}(\widehat{a}\mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}M, a), \widehat{x}\mathsf{Forall}_L(\widehat{y}N, t, x)) \to \mathsf{Cut}(\widehat{b}M[\mathsf{x} := t], \widehat{y}N)
$$

si $\mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}M, a)$ introduit fraîchement a
et $\mathsf{Forall}_L(\widehat{y}N, t, x)$ introduit fraîchement x

Coupures commutatives :

$$
\mathsf{Cut}(\widehat{a}M, \widehat{x}N) \to \begin{cases} M[a := \widehat{x}N] & \text{si } M \text{ n'introduit pas fraîchement } a \\ N[x := \widehat{a}M] & \text{si } M \text{ n'introduit pas fraîchement } x \end{cases}
$$

FIG. 3.11 – Élimination des coupures de Urban et Bierman (2001)

Sur les termes admettant une occurrence de c à la position ε.

$$
\begin{array}{lcl}
\mathsf{Ax}(x,c)[c := \widehat{y}M] & = & M[x/y] \\
\mathsf{True}_R(c)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{c}\mathsf{True}_R(c), \widehat{y}M) \\
\mathsf{Not}_R(\widehat{x}N, c)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{c}\mathsf{Not}_R(\widehat{x}N[c := \widehat{y}M], c), \widehat{y}M) \\
\mathsf{Imp}_R(\widehat{x}\widehat{a}N, c)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{c}\mathsf{Imp}_R(\widehat{x}\widehat{a}N[c := \widehat{y}M], c), \widehat{y}M) \\
\mathsf{Or}_R(\widehat{a}\widehat{b}N, c)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{c}\mathsf{Or}_R(\widehat{a}\widehat{b}N[c := \widehat{y}M], c), \widehat{y}M) \\
\mathsf{And}_R(\widehat{a}N_1, \widehat{b}N_2, c)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{c}\mathsf{And}_R(\widehat{a}N_1[c := \widehat{y}M], \\
 & & \qquad \widehat{b}N_2[c := \widehat{y}M], c), \widehat{y}M) \\
\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}N, c)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{c}\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}N[c := \widehat{y}M], c), \widehat{y}M) \\
\mathsf{Exists}_R(\widehat{a}N, t, c)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{c}\mathsf{Exists}_R(\widehat{a}M[c := \widehat{y}M], t, c), \widehat{y}M)
\end{array}
$$

Sur les termes n'admettant pas une occurrence de c à la position ε.

$$
\begin{array}{lcl}
\mathsf{Ax}(x,a)[c := \widehat{y}M] & = & \mathsf{Ax}(x,a) \\
\mathsf{Cut}(\widehat{a}N_1, \widehat{x}N_2)[c := \widehat{y}M] & = & \mathsf{Cut}(\widehat{a}N_1[c := \widehat{y}M], \widehat{x}N_2[c := \widehat{y}M]) \\
\mathsf{False}_L(x)[c := \widehat{y}M] & = & \mathsf{False}_L(x) \\
\mathsf{True}_R(a)[c := \widehat{y}M] & = & \mathsf{True}_R(a) \\
\mathsf{Not}_R(\widehat{x}N, a)[c := \widehat{y}M] & = & \mathsf{Not}_R(\widehat{x}N[c := \widehat{y}M], a) \\
\mathsf{Not}_L(\widehat{a}N, x)[c := \widehat{y}M] & = & \mathsf{Not}_L(\widehat{a}N[c := \widehat{y}M], x) \\
\mathsf{Imp}_R(\widehat{x}\widehat{a}N, b)[c := \widehat{y}M] & = & \mathsf{Imp}_R(\widehat{x}\widehat{a}N[c := \widehat{y}M], b) \\
\mathsf{Imp}_L(\widehat{x}N_1, \widehat{a}N_2), y)[c := \widehat{y}M] & = & \mathsf{Imp}_L(\widehat{x}N_1[c := \widehat{y}M], \widehat{a}N_2[c := \widehat{y}M], y) \\
\mathsf{And}_R(\widehat{a}N_1, \widehat{b}N_2, d)[c := \widehat{y}M] & = & \mathsf{And}_R(\widehat{a}N_1[c := \widehat{y}M], \widehat{b}N_2[c := \widehat{y}M], d) \\
\mathsf{And}_L(\widehat{x}\widehat{z}N, z')[c := \widehat{y}M] & = & \mathsf{And}_L(\widehat{x}\widehat{z}N[c := \widehat{y}M], z') \\
\mathsf{Or}_R(\widehat{a}\widehat{b}N, d)[c := \widehat{y}M] & = & \mathsf{Or}_R(\widehat{a}\widehat{b}N[c := \widehat{y}M], d) \\
\mathsf{Or}_L(\widehat{x}N_1, \widehat{z}N_2, z')[c := \widehat{y}M] & = & \mathsf{Or}_L(\widehat{x}N_1[c := \widehat{y}M], \widehat{z}N_2[c := \widehat{y}M], z') \\
\mathsf{Exists}_R(\widehat{a}N, \mathsf{t}, b)[c := \widehat{y}M] & = & \mathsf{Exists}_R(\widehat{a}N[c := \widehat{y}M], \mathsf{t}, b) \\
\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}N, y)[c := \widehat{y}M] & = & \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}N[c := \widehat{y}M], y) \\
\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}N, b)[c := \widehat{y}M] & = & \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}N[c := \widehat{y}M], b) \\
\mathsf{Forall}_L(\widehat{x}N, \mathsf{t}, y)[c := \widehat{y}M] & = & \mathsf{Forall}_L(\widehat{x}N[c := \widehat{y}M], \mathsf{t}, y)
\end{array}
$$

FIG. 3.12 – Définition de $[c := \widehat{y}M]$

Sur les termes admettant une occurrence de z à la position ε.

$$
\begin{array}{lcl}
\mathsf{Ax}(z,a)[z := \widehat{b}M] & = & M[a/c] \\
\mathsf{False}_L(z)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{b}M, \widehat{z}\mathsf{False}_L(z)) \\
\mathsf{Not}_L(\widehat{a}N, z)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{b}M, \widehat{z}\mathsf{Not}_L(\widehat{a}N[z := \widehat{b}M], z)) \\
\mathsf{Imp}_L(\widehat{x}N_1, \widehat{a}N_2, z)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{b}M, \widehat{z}\mathsf{Imp}_L(\widehat{x}N_1[z := \widehat{b}M], \\
 & & \qquad \widehat{a}N_2[z := \widehat{b}M], z)) \\
\mathsf{Or}_L(\widehat{x}N_1, \widehat{y}N_2, z)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{b}M, \widehat{z}\mathsf{Or}_L(\widehat{x}N_1[z := \widehat{b}M], \\
 & & \qquad \widehat{y}N_2[z := \widehat{b}M], z)) \\
\mathsf{And}_L(\widehat{x}\widehat{y}N, z)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{b}M, \widehat{z}\mathsf{And}_L(\widehat{x}\widehat{y}N[z := \widehat{b}M], z)) \\
\mathsf{Forall}_L(\widehat{x}N, \mathsf{t}, z)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{b}M, \widehat{z}\mathsf{Forall}_L(\widehat{x}N[z := \widehat{b}M], \mathsf{t}, z)) \\
\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}N, z)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{b}M, \widehat{z}\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}N[z := \widehat{b}M], z))
\end{array}
$$

Sur les termes n'admettant pas une occurrence de c à la position ε.

$$
\begin{array}{lcl}
\mathsf{Ax}(x,a)[z := \widehat{b}M] & = & \mathsf{Ax}(x,a) \\
\mathsf{Cut}(\widehat{a}N_1, \widehat{x}N_2)[z := \widehat{b}M] & = & \mathsf{Cut}(\widehat{a}N_1[z := \widehat{b}M], \widehat{x}N_2[z := \widehat{b}M]) \\
\mathsf{False}_L(x)[z := \widehat{b}M] & = & \mathsf{False}_L(x) \\
\mathsf{True}_R(a)[z := \widehat{b}M] & = & \mathsf{True}_R(a) \\
\mathsf{Not}_R(\widehat{x}N, a)[z := \widehat{b}M] & = & \mathsf{Not}_R(\widehat{x}N[z := \widehat{b}M], a) \\
\mathsf{Not}_L(\widehat{a}N, x)[z := \widehat{b}M] & = & \mathsf{Not}_L(\widehat{a}N[z := \widehat{b}M], x) \\
\mathsf{Imp}_R(\widehat{x}\widehat{a}N, c)[z := \widehat{b}M] & = & \mathsf{Imp}_R(\widehat{x}\widehat{a}N[z := \widehat{b}M], c) \\
\mathsf{Imp}_L(\widehat{x}N_1, \widehat{a}N_2, y)[z := \widehat{b}M] & = & \mathsf{Imp}_L(\widehat{x}N_1[z := \widehat{b}M], \widehat{a}N_2[z := \widehat{b}M], y) \\
\mathsf{And}_R(\widehat{a}N_1, \widehat{c}N_2, d)[z := \widehat{b}M] & = & \mathsf{And}_R(\widehat{a}N_1[z := \widehat{b}M], \widehat{b}N_2[z := \widehat{b}M], c) \\
\mathsf{And}_L(\widehat{x}\widehat{y}N, z')[z := \widehat{b}M] & = & \mathsf{And}_L(\widehat{x}\widehat{y}N[z := \widehat{b}M], z) \\
\mathsf{Or}_R(\widehat{a}\widehat{c}N, d)[z := \widehat{b}M] & = & \mathsf{Or}_R(\widehat{a}\widehat{b}N[z := \widehat{b}M], c) \\
\mathsf{Or}_L(\widehat{x}N_1, \widehat{y}N_2, z')[z := \widehat{b}M] & = & \mathsf{Or}_L(\widehat{x}N_1[z := \widehat{b}M], \widehat{y}N_2[z := \widehat{b}M], z) \\
\mathsf{Exists}_R(\widehat{a}N, \mathsf{t}, c)[z := \widehat{b}M] & = & \mathsf{Exists}_R(\widehat{a}N[z := \widehat{b}M], \mathsf{t}, c) \\
\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}N, y)[z := \widehat{b}M] & = & \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}N[z := \widehat{b}M], y) \\
\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}N, c)[z := \widehat{b}M] & = & \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}N[z := \widehat{b}M], c) \\
\mathsf{Forall}_L(\widehat{x}N, \mathsf{t}, y)[z := \widehat{b}M] & = & \mathsf{Forall}_L(\widehat{x}N[z := \widehat{b}M], \mathsf{t}, y)
\end{array}
$$

FIG. 3.13 – Définition de $[z := \widehat{b}M]$

Chapitre 4

Coupures canoniques

Le calcul des séquents ne permet d'appliquer qu'une seule règle à la fois : une dérivation en calcul des séquents est un arbre où chaque noeud correspond à une unique inférence. Cet arbre définit un ordre sur les noeuds et donc sur ces applications d'inférences. Considérons par exemple les dérivations

$$\dfrac{\dfrac{\dfrac{}{A \vdash A, B, C, D}\,\text{Axiom}}{A \vdash A \vee B, C, D}\,\vee\text{R}}{A \vdash A \vee B, C \vee D}\,\vee\text{R} \quad \text{et} \quad \dfrac{\dfrac{\dfrac{}{A \vdash A, B, C, D}\,\text{Axiom}}{A \vdash A, B, C \vee D}\,\vee\text{R}}{A \vdash A \vee B, C \vee D}\,\vee\text{R}\ .$$

L'intuition dicte que les deux inférences de la première dérivation (respectivement de la deuxième) sont totalement indépendantes. Plus précisément, les résultats de Kleene (c'est-à-dire la propriété 2.3.12) démontrent qu'en termes de prouvabilité, l'ordre attribué à ces deux inférences n'est pas pertinent : s'il existe une dérivation de $A \vdash A \vee B, C \vee D$ (c'est bien le cas), il en existera une quelque soit l'ordre choisi entre les décompositions des conjonctions. Effectivement si l'on choisit de décomposer d'abord la conjonction de gauche, on obtient la première dérivation ci-dessus. Si l'on choisit de décomposer d'abord la conjonction de droite, on obtient la deuxième dérivation. Autrement dit, bien que ces deux dérivations soient *syntaxiquement* différentes, elles sont égales modulo *permutations des inférences*. C'est ce que Girard (1989) appelle la *bureaucratie de la syntaxe*. Les résultats de Kleene suggèrent donc d'étudier plus profondément l'identité des preuves modulo permutations. Comme nous l'avons déjà vu en section 2.3, certaines instances des introductions de quantificateurs ne sont pas permutables. Par exemple les inférences de la dérivation

$$\dfrac{\dfrac{\dfrac{}{A(x) \vdash A(x)}\,\text{Axiom}}{A(x) \vdash \exists y.A(y)}\,\exists\text{R}}{\exists x.A(x) \vdash \exists y.A(y)}\,\exists\text{L}$$

ne peuvent être réarrangées de manière à faire apparaître l'instance de $\exists$L au dessus de l'instance de $\exists$R.

Notre intérêt pour ces permutations d'inférences est aussi motivé par l'étude de l'élimination des coupures. Jusqu'à présent, toutes les procédures d'élimination des coupures que nous avons considérées (par exemple en section 2.3 ou 3.3) sont non-confluentes et même non-consistantes. Plus précisément la congruence qu'elles génèrent identifie toutes les dérivations d'un même séquent : considérons dans LK deux dérivations π_1 et π_2 d'un même séquent $\Gamma \vdash \Delta$ syntaxiquement distinctes. Alors nous avons vu que les règles réduisant la coupure

$$\dfrac{\dfrac{\overset{(\pi_1)}{\Gamma \vdash \Delta}}{\Gamma \vdash \varphi, \Delta}\,\text{WR} \qquad \dfrac{\overset{(\pi_2)}{\Gamma \vdash \Delta}}{\Gamma, \varphi \vdash \Delta}\,\text{WL}}{\Gamma \vdash \Delta}\,\text{Cut}$$

induit la paire critique (π_1, π_2). Par conséquent π_1 et π_2 sont identifiés par toute congruence contenant l'élimination des coupures. Pour caractériser l'élimination des coupures comme un calcul, il est naturel de rechercher une procédure confluente et surtout consistante. Ce qui est gênant dans la coupure ci-dessus, c'est que son élimination consiste à *choisir* entre π_1 et π_2. La dérivation choisie devient le réduit, l'autre est considérée comme du *code mort* puisqu'elle est simplement supprimée. Cette suppression est ennuyeuse : π_1 et π_2 sont toutes deux des réponses *valides* à la question $\Gamma \vdash \Delta$. Ces réponses ne sont en aucun cas du code mort. En particulier on ne peut pas les supprimer toutes les deux. La coupure ci-dessus représente moralement une troisième réponse composée de π_1 et π_2. L'élimination des coupures se comporte alors comme un choix non-déterministe entre π_1 et π_2. Pour que l'élimination des coupures ne contienne que du calcul (et soit confluente), on peut la modifier de façon à ce qu'elle n'opère plus aucun choix. La coupure ci-dessus se réduirait alors vers une troisième réponse représentant l'union des réponses π_1 et π_2. Le choix on-déterministe serait en quelque sorte laissé à l'utilisateur.

Si le contre-exemple que nous venons de considérer dans le précédent paragraphe est bien *la* cause de non-cohérence de nos procédures d'élimination des coupures, il convient de remarquer que ce n'est pas la seule cause de non-confluence puisqu'on peut trouver de nombreuses autres paires critiques qui ne sont pas convergentes. Celles-ci sont généralement des coupures commutatives. Considérons par exemple la coupure

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A, A \vee B}\,\text{Axiom}}{A \wedge B \vdash A, A \vee B}\,\wedge\text{L} \qquad \dfrac{\dfrac{}{A \wedge B, A \vdash A, B}\,\text{Axiom}}{A \wedge B, A \vdash A \vee B}\,\vee\text{R}}{A \wedge B \vdash A \vee B}\,\text{Cut}\ .$$

La plupart des procédures d'élimination des coupures proposent deux manières de l'éliminer : soit la coupure est remontée dans la dérivation de sa prémisse de gauche, soit elle est remontée dans la dérivation de sa prémisse de droite. Dans le premier

cas, on obtient les dérivations

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A, A \vee B}\,\textsc{Axiom}}{A \wedge B \vdash A \vee B}\,\wedge\text{L}}{A \wedge B \vdash A \vee B}\,\textsc{Cut} \qquad \dfrac{\dfrac{\dfrac{}{A, B, A \vdash A, B}\,\textsc{Axiom}}{A, B, A \vdash A \vee B}\,\vee\text{R}}{A, B \vdash A \vee B}\,\wedge\text{L} \qquad \text{puis} \qquad \dfrac{\dfrac{\dfrac{}{A, B \vdash A, B}\,\textsc{Axiom}}{A, B \vdash A \vee B}\,\vee\text{R}}{A \wedge B \vdash A \vee B}\,\wedge\text{L}\,.$$

Dans le deuxième cas, on obtient les dérivations

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash A, A, B}\,\textsc{Axiom}}{A \wedge B \vdash A, A, B}\,\wedge\text{L} \qquad \dfrac{}{A \wedge B, A \vdash A, B}\,\textsc{Axiom}}{A \wedge B \vdash A, B}\,\textsc{Cut}}{A \wedge B \vdash A \vee B}\,\vee\text{R} \qquad \text{puis} \qquad \dfrac{\dfrac{\dfrac{}{A, B \vdash A, B}\,\textsc{Axiom}}{A \wedge B \vdash A, B}\,\wedge\text{L}}{A \wedge B \vdash A \vee B}\,\vee\text{R}\,.$$

Notons que les deux formes normales obtenues diffèrent syntaxiquement mais sont égales modulo permutations des inférences $\wedge$L et $\vee$R. Des paires critiques similaires peuveut aussi être trouvées pour les autres connecteurs. Ces paires critiques qui convergent seulement modulo permutations nous amènent à étudier plus en détail la permutabilité des inférences en calcul des séquents. Plus précisément, notre but est de définir une représentation des classes d'équivalence modulo permutations puis d'écrire une élimination des coupures sur ces représentants qui soit à la fois confluente et fortement normalisante.

L'approche que nous allons développer au cours de ce chapitre est celle des *réseaux de preuve*. Si leur introduction initiale est d'habitude attribuée à Girard (1987) dans le cadre de la logique linéaire, l'idée originale se trouve aussi dans les *Expansion Trees* de Miller (1987) introduits à la même période pour une version de la logique classique basée sur la théorie des types simples de Church. Notons aussi les travaux de Robinson (2003) et ceux de Lamarche et Straßburger (2005) dans le cadre de la logique propositionnelle classique. Ces derniers proposent de représenter les classes d'équivalence modulo permutation par des réseaux de preuve puis obtiennent une élimination des coupures confluente et fortement normalisante pour la logique propositionnelle. Nous allons à présent reformuler cette présentation dans les sections 4.1, 4.2 et 4.3. La première présente le paradigme des réseaux de preuve pour la logique propositionnelle classique. La section 4.2 fait le lien avec le calcul des séquents par le biais des processus de *séquentialisation* et de *parallélisation*. La section 4.3 étudie l'élimination des coupures sur les réseaux de preuve. Ces trois sections ne contiennent donc pas de grande nouveauté et reformulent des idées déjà présentes dans les travaux de Miller, Robinson ou Lamarche et Straßburger. Néanmoins, la section 4.4 contient des travaux originaux : elle propose une représentation dans un espace à trois dimensions des réseaux de preuve pour la logique classique propositionnelle. Cette représentation est basée sur la dualité de De Morgan entre conjonctions et disjonctions et est présentée dans un article non publié (Houtmann, 2009).

4.1 Réseaux de preuves pour la logique classique

Cette section a pour but l'introduction des réseaux de preuve pour la logique propositionnelle classique. Cette présentation est une reformulation des travaux de Lamarche et Straßburger (2005). Comme eux et par souci de simplicité, nous ne considérerons que des formules construites à partir de propositions atomiques (P), de négations de propositions atomiques ($\bar{P}$) et des connecteurs $\top$, $\bot$, $\wedge$ et $\vee$. L'ensemble des formules est donc le langage hors-contexte associé à la grammaire sous forme de Backus-Naur suivante.

$$\varphi, \psi \quad ::= \quad \underbrace{P \mid \bar{P} \mid \top \mid \bot}_{\text{atomes}} \mid \varphi \wedge \psi \mid \varphi \vee \psi \; .$$

La négation n'est pas un connecteur à part entière, mais une opération sur les formules définie inductivement à partir des lois de De Morgan comme suit.

$$\neg P = \bar{P} \; , \qquad \neg \bar{P} = P \; , \qquad \neg \bot = \top \; , \qquad \neg \top = \bot \; ,$$
$$\neg(\varphi \wedge \psi) = (\neg\varphi) \vee (\neg\psi) \; , \qquad \neg(\varphi \vee \psi) = (\neg\varphi) \wedge (\neg\psi) \; .$$

Cette négation nous permet par exemple d'écrire les dérivations en calcul des séquents en utilisant des séquents à un seul côté en utilisant par exemple la restriction de LK1 (figure 2.12) aux connecteurs $\wedge$, $\vee$, $\bot$ et $\top$. Un tel système logique est équivalent au calcul des séquents usuel LK, du moins pour les séquents que nous considérons ici.

À présent et lorsque nous considérerons un séquent L, nous nous intéresserons aux occurrences des sous-formules de L. Nous supposons donné un moyen d'identifier de telles occurrences (positions, contextes, zippers…) sans pour autant entrer dans les détails d'une telle présentation. Les symboles x, y, z représenterons de telles occurrences. Si dans un séquent L, l'occurrence x correspond à une sous-formule φ, nous la noterons aussi $x : \varphi$ et la formule φ sera aussi notée $L_{|x}$. Nous pouvons à présent définir ce qu'est un réseau de preuve.

Définition 4.1.1 (Réseau de preuve). *Un réseau de preuve est une paire $(L, \sim)$ où L est un séquent et $\sim$ est une relation symétrique sur les occurrences dans L telle que*

$$x \sim y \quad \textit{implique} \quad L_{|x} = \neg L_{|y} \;\; \textit{ou} \;\; L_{|x} = L_{|y} = \top \; .$$

La figure 4.1 représente un réseau de preuve pour le séquent

$$\vdash \top \vee (P \wedge \top), (P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot) \; .$$

Les réseaux de preuve représentent en fait des dérivations potentiellement ouvertes dans le calcul des séquents classique propositionnel sans coupure. Dans un réseau de preuve $(L, \sim)$, un lien $x \sim y$ avec $\varphi = L_{|x} = \neg L_{|y}$ représente en fait l'utilisation de la règle AXIOM et un lien $x \sim y$ avec $L_{|x} = L_{|y} = \top$ représente une introduction du connecteur $\top$.

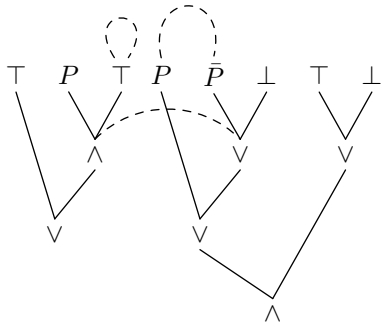


FIG. 4.1 – Un réseau pour $\vdash \top \vee (P \wedge \top), (P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot)$.

Il est important de savoir caractériser quels réseaux de preuve représentent effectivement des dérivations fermées. Pour cela, nous nous reposons sur la notion de résolution conjonctive introduite par Lamarche et Straßburger (2005). Soit L un séquent. Soit O l'ensemble des occurrences de L. Les *résolutions conjonctives* pour L sont les sous-ensembles C de O suivants.

- Si L consiste seulement en un atome φ dont l'occurrence est notée x, alors sa seule résolution conjonctive est $C = O = \{x : \varphi\}$.
- Si L est une formule $\varphi \vee \psi$ dont l'occurrence est notée x, alors ses résolutions conjonctives sont $C_1 \cup C_2 \cup \{x : \varphi \vee \psi\}$ où C_1 et C_2 sont respectivement des résolutions conjonctives pour φ et pour ψ.
- Si L est une formule $\varphi \wedge \psi$ dont l'occurrence est notée x, alors ses résolutions conjonctives sont $C \cup \{x : \varphi \wedge \psi\}$ où C est une résolution conjonctive pour φ ou pour ψ.
- Si L est un séquent composé L', φ, alors ses résolutions conjonctives sont $C_1 \cup C_2$ où C_1 et C_2 sont respectivement des résolutions conjonctives pour L' et pour φ.

Grâce à l'associativité et la commutativité de l'union ensembliste, cette définition des résolutions conjonctives est bien formée. Nous pouvons à présent nous pencher sur la notion de réseau CR-valide qui, nous le verrons par la suite, caractérise les réseaux correspondant à des dérivations fermées du calcul des séquents.

Définition 4.1.2 (Réseau de preuve CR-valide). *Un réseau de preuve $(L, \sim)$ est CR-valide si pour toute résolution conjonctive C de L, la restriction de $\sim$ à C (c'est-à-dire l'ensemble $\sim \cap (C \times C)$) n'est pas vide.*

Considérons par exemple le réseau de preuve décrit en figure 4.1. Le séquent correspondant à ce réseau de preuve admet quatre résolutions conjonctives qui sont décrites par la figure 4.2. Cette figure représente aussi la restriction de la relation associée à chaque résolution conjonctive. Remarquons que la restriction de la relation du réseau associée à la troisième résolution conjonctive est vide. Le réseau de preuve de la figure 4.1 n'est donc pas une réseau CR-valide. Toutefois cela peut être facilement corrigé en reliant par exemple l'occurrence de $\top$ de gauche à elle-même.

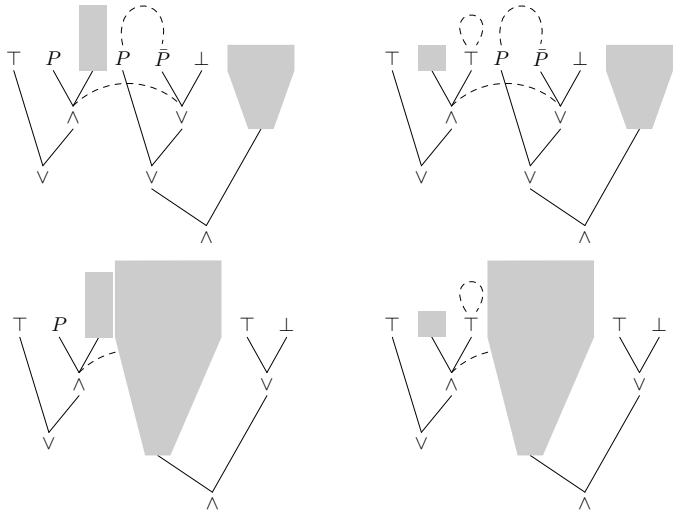


FIG. 4.2 – Résolutions conjonctives du réseau de preuve de la figure 4.1

Pour représenter les réseaux de preuve avec coupures, Lamarche et Straßburger (2005) introduisent un nouveau connecteur. Ce connecteur est en fait une conjonction et repose sur l'équivalence dans LK de la règle de coupure avec

$$\frac{\vdash \Gamma, \varphi \wedge (\neg\varphi), \Delta}{\vdash \Gamma, \Delta} \quad .$$

Les *formules de coupure* sont donc des conjonctions spéciales notées $\varphi \sqcap \psi$ au lieu de $\varphi \wedge \psi$. Elles ne peuvent pas apparaître à l'intérieur d'une autre formule : dans un séquent $\vdash \varphi_1, \ldots, \varphi_n$, seules les φ_i peuvent être des formules de coupure. En outre si $(L, \sim)$ est un réseau de preuve avec $x \sim y$, alors en aucun cas $L_{|x}$ ou $L_{|y}$ ne peuvent être des formules de coupure. En ce qui concerne les résolutions conjonctives, les formules de coupure sont traitées comme des conjonctions usuelles. Si L est un séquent ne contenant aucune formule de coupure et L' est un séquent ne contenant que des formules de coupure, alors nous dirons qu'un réseau de preuve $(L \cup L', \sim)$ est un *réseau pour L avec les coupures L'*. Si L' est vide, alors le réseau est *sans coupure*.

4.2 Parallèlisation et séquentialisation

Dans cet section, nous allons comparer les réseaux de preuve et les dérivations en calcul des séquents. Plus précisément nous allons retranscrire le résultat de Lamarche et Straßburger (2005) qui explique qu'il existe un réseau de preuve CR-valide pour un séquent L (avec coupures L') si et seulement si il existe un dérivation de L en calcul des séquents. Tout comme en section 2.4, nous pourrions passer par une

notion de modèle (ici pour la logique classique). Bien qu'intrinsèquement intéressante, cette approche ne sera pas développée ici : nous allons directement traduire les dérivations du calcul des séquents en réseaux de preuve et inversement. Nous appellerons ces deux traductions respectivemement *parallélisation* et *séquentialisation*. La première est tout à fait simple. Dans le cas propositionnel, il s'agit de paralléliser au maximum la décomposition des connecteurs. La séquentialisation n'est pas beaucoup plus compliquée lorsque l'on connaît les résultats d'inversibilité dans le calcul des séquents : il s'agit de séquentialiser la décomposition des connecteurs et l'ordre choisi n'a aucune importance, du moins dans le cadre propositionnel. Notons que lorsque nous parlons de dérivation dans le calcul des séquents, nous nous référons à LK1.

4.2.1 Parallèlisation : des dérivations vers les réseaux

Commençons donc par la première traduction.

Propriété 4.2.1. *S'il existe une dérivation sans coupure de L, alors il existe un réseau de preuve sans coupure CR-valide d'un séquent L.*

Démonstration. Par induction sur la dérivation de L.

- Si cette dérivation consiste en une instance de la règle Axiom, alors il existe une occurrence d'une formule φ et de sa négation $\neg\varphi$. Soit $\sim$ la relation symétrique reliant l'occurrence de φ à l'occurrence de $\neg\varphi$. Alors $(L, \sim)$ est un réseau CR-valide pour L.
- Si cette dérivation consiste en une instance de la règle $\top$ ou la règle Axiom, alors il existe une occurrence de $\top$ dans L. Soit $\sim$ la relation symétrique reliant l'occurrence de $\top$ à elle-même. Alors $(L, \sim)$ est un réseau CR-valide pour L.
- Si cette dérivation contient à sa racine une instance de la règle $\wedge$ de la forme

$$\wedge \frac{\vdash \varphi, \Delta \qquad \vdash \psi, \Delta}{\vdash \varphi \wedge \psi, \Delta}$$

  alors par hypothèse d'induction, il existe deux réseaux CR-valides $(\vdash \varphi, \Delta \ , \ \sim_1)$ et $(\vdash \psi, \Delta \ , \ \sim_2)$. Alors $(\vdash \varphi \wedge \psi, \Delta \ , \ \sim_1 \cup \sim_2)$ est un réseau CR-valide pour $\vdash \varphi \wedge \psi, \Delta$ (notons que les relations $\sim_1$ et $\sim_2$ doivent être respectivement mises à jour pour traiter des occurrences du séquent $\vdash \varphi \wedge \psi, \Delta$ au lieu des occurrences des séquents $\vdash \varphi, \Delta$ et $\vdash \psi, \Delta$).
- Si cette dérivation contient à sa racine une instance de la règle $\vee$ de la forme

$$\vee \frac{\vdash \varphi, \psi, \Delta}{\vdash \varphi \vee \psi, \Delta}$$

  alors par hypothèse d'induction, il existe un réseau CR-valide $(\vdash \varphi, \psi, \Delta \ , \ \sim)$. Alors $(\vdash \varphi \vee \psi, \Delta \ , \ \sim)$ est un réseau CR-valide pour $\vdash \varphi \vee \psi, \Delta$ (notons que la relation $\sim$ doit être mise à jour pour traiter

des occurrences du séquent $\vdash \varphi \vee \psi, \Delta$ au lieu des occurrences du séquent $\vdash \varphi, \psi, \Delta$). □

4.2.2 Séquentialisation : des réseaux vers les dérivations

A présent exposons la deuxième traduction.

Propriété 4.2.2. *S'il existe un réseau de preuve sans coupure CR-valide d'un séquent L, alors il existe une dérivation sans coupure de L.*

Démonstration. Soit donc un réseau de preuve $(L, \sim)$ sans coupure. Nous raisonnons par induction sur le séquent L.

- Si L est un séquent atomique, c'est-à-dire qui ne contient que des atomes, alors puisque $(L, \sim)$ est un réseau valide et puisque L est atomique, il existe deux occurrences x et y de L telles que $x \sim y$. Si ces occurrences correspondent à une formule φ et à sa négation, L peut être dérivé grâce à la règle Axiom. Si ces occurrences correspondent à $\top$, L peut être dérivé grâce à la règle $\top$. Dans les deux cas nous obtenons une dérivation sans coupure de L.
- Si L est un séquent de la forme $\vdash \varphi \wedge \psi, \Delta$, la première chose à faire et de remplacer dans le réseau tout lien éventuel de l'occurrence de $\varphi \wedge \psi$ vers une occurrence de $\neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi$ par deux liens reliant l'occurrence de φ à l'occurrence de $\neg\varphi$ et l'occurrence de ψ à l'occurrence de $\neg\psi$. Le réseau obtenu $(L, \sim)$ reste CR-valide. Soient $\sim_1$ et $\sim_2$ les restrictions de $\sim$ aux séquents $\vdash \varphi, \Delta$ et $\vdash \psi, \Delta$ (mises à jour pour bien traiter des occurrences de ces séquents et non des occurrences de $\vdash \varphi \wedge \psi, \Delta$). Alors $(\vdash \varphi, \Delta \ , \ \sim_1)$ et $(\vdash \psi, \Delta \ , \ \sim_2)$ sont deux réseaux CR-valides pour $\vdash \varphi, \Delta$ et $\vdash \psi, \Delta$. Par hypothèse d'induction, on obtient des dérivations pour chacun de ces séquents et la règle $\wedge$ nous permet alors de construire une dérivation de L.
- Si L est un séquent de la forme $\vdash \varphi \vee \psi, \Delta$, la première chose à faire et de remplacer dans le réseau tout lien éventuel de l'occurrence de $\varphi \vee \psi$ vers une occurrence de $\neg(\varphi \vee \psi) = \neg\varphi \wedge \neg\psi$ par deux liens reliant l'occurrence de φ à l'occurrence de $\neg\varphi$ et l'occurrence de ψ à l'occurrence de $\neg\psi$. Le réseau obtenu $(L, \sim)$ reste CR-valide. Alors $(\vdash \varphi, \psi, \Delta \ , \ \sim)$ est un réseau CR-valide pour $\vdash \varphi, \psi, \Delta$ ($\sim$ doit être mise à jour pour bien traiter des occurrences de ce séquent et non des occurrences de $\vdash \varphi \vee \psi, \Delta$). Par hypothèse d'induction, on obtient une dérivation pour le séquent $\vdash \varphi, \psi, \Delta$ et la règle $\vee$ nous permet alors de construire une dérivation de L. □

4.2.3 Traduction des coupures

Bien que nous n'ayons prouvé les propriétés 4.2.1 et 4.2.2 que dans le cadre des réseaux et dérivations sans coupure, ces résultats se transfèrent facilement aux réseaux et dérivations avec coupures : encore une fois une coupure $\varphi \sqcap \neg\varphi$ correspond directement à une conjonction $\varphi \wedge \neg\varphi$.

Si L et L' sont deux séquents, nous dirons qu'une dérivation de L en calcul des séquents est une dérivation de L avec coupures L' si toute instance de la règle Cut que contient cette dérivation a pour formule de coupure une formule de L'.

Propriété 4.2.3. *S'il existe un réseau de preuve CR-valide pour un séquent L avec coupures L', alors il existe une dérivation de L.*

Propriété 4.2.4. *S'il existe une dérivation de L avec coupures L', alors il existe un réseau de preuve CR-valide pour le séquent L avec coupures L'.*

Pour la séquentialisation, un réseau $(\vdash \varphi \sqcap \neg\varphi, \Delta\ ,\ \sim)$ se transforme en réseaux $(\vdash \varphi, \Delta\ ,\ \sim_1)$ et $(\vdash \neg\varphi, \Delta\ ,\ \sim_2)$ exactement comme si la coupure était en fait une conjonction. L'hypothèse d'induction nous fournit alors des dérivations de $\vdash \varphi, \Delta$ et $\vdash \neg\varphi, \Delta$ que l'on peut transformer en dérivations de $\vdash \Delta$ en appliquant une coupure sur la formule φ.

Pour la parallélisation, si la dérivation est de la forme

$$\textsc{Cut}\ \frac{\vdash \varphi, \Delta \qquad \vdash \neg\varphi, \Delta}{\vdash \Delta}$$

alors il suffit d'appliquer l'hypothèse d'induction sur les dérivations des prémisses de cette inférence pour obtenir des réseaux CR-valides R_1 et R_2 pour ces prémisses. Superposer ces réseaux nous donne alors un réseau CR-valide pour $\vdash \varphi \sqcap \neg\varphi, \Delta$.

Puisque nous venons de démontrer l'équivalence entre réseaux de preuve CR-valides sans coupure et dérivations en calcul des séquents sans coupure (propriétés 4.2.2 et 4.2.1), puis l'équivalence entre réseaux de preuve CR-valides et dérivations en calcul des séquents (propriétés 4.2.3 et 4.2.4), nous pouvons transposer directement l'élimination des coupures du calcul des séquents (propriété 2.3.4) vers les réseaux de preuves.

Propriété 4.2.5. *S'il existe un réseau de preuve pour un séquent L avec coupures L', alors il en existe un pour L sans coupure.*

La section 4.3 sera dédiée à la preuve directe de cette propriété grâce à une procédure d'élimination des coupures pour les réseaux.

4.2.4 Typage réversible

La séquentialisation que nous venons d'exposer contient deux mécanismes importants. Tout d'abord, il s'agit de transformer un lien entre une formule $\varphi_1 \wedge \varphi_2$ et sa négation $\psi_1 \vee \psi_2$ en deux liens respectivement entre φ_1 et ψ_1 ainsi qu'entre ψ_1 et ψ_2. Puis on applique les transformations suivantes.

1. Lorsque le réseau obtenu (CR-valide) est de la forme $(\vdash \varphi \wedge \psi, \Delta\ ,\ \sim)$, on construit deux réseaux CR-valides pour les séquents $\vdash \varphi, \Delta$ et $\vdash \psi, \Delta$.

$$\textsc{Axiom}\ \frac{}{x \sim y \ \vdash x : \varphi, y : \neg\varphi, \Delta} \qquad \top\ \frac{}{x \sim y \ \vdash x : \top, y : \top, \Delta}$$

$$\wedge\ \frac{R_1 \ \vdash \varphi, \Delta \qquad R_1 \ \vdash \psi, \Delta}{R_1 \bowtie R_2 \ \vdash \varphi \wedge \psi, \Delta} \qquad \vee\ \frac{R \ \vdash \varphi, \psi, \Delta}{R \ \vdash \varphi \vee \psi, \Delta}$$

$$\textsc{Union}\ \frac{R_1 \ \vdash \Delta \qquad R_2 \ \vdash \Delta}{R_1 \cup R_2 \ \vdash \Delta} \qquad \textsc{Cut}\ \frac{R_1 \ \vdash \varphi, \Delta \qquad R_2 \ \vdash \neg\varphi, \Delta}{R_1 \bowtie R_2 \ \vdash \varphi \sqcap \neg\varphi, \Delta}$$

FIG. 4.3 – Typage Commutatif

2. Lorsque le réseau obtenu est de la forme $(\vdash \varphi \vee \psi, \Delta \ , \ \sim)$, on construit un réseau CR-valide pour le séquent $\vdash \varphi, \psi, \Delta$

Ces transformations sont appliquées récursivement jusqu'à obtenir des réseaux CR-valides pour des séquents atomiques. Les règles Axiom et $\top$ permettent alors de dériver ces séquents atomiques. Les règles $\wedge$ et $\vee$ permettent alors de reconstruire une dérivation pour le séquent d'origine selon l'ordre d'application des transformations 1 et 2.

La parallélisation est tout à fait similaire et inclut les transformations inverses.

a Lorsque l'on est en présence d'une dérivation de la forme

$$\wedge\ \frac{\vdash \varphi, \Delta \qquad \vdash \psi, \Delta}{\vdash \varphi \wedge \psi, \Delta}$$

et que la récursion nous a fourni des réseaux pour les deux prémisses de cette inférence, alors il nous suffit de construire la superposition de ces deux réseaux pour obtenir un réseau CR-valide pour $\vdash \varphi \wedge \psi, \Delta$.

b Lorsque l'on est en présence d'une dérivation de la forme

$$\vee\ \frac{\vdash \varphi, \psi, \Delta}{\vdash \varphi \vee \psi, \Delta}$$

et que la récursion nous a fourni un réseau pour la prémisse de cette inférence, alors ce réseau est (presque) directement un réseau pour $\vdash \varphi \vee \psi, \Delta$.

L'ordre d'application des transformations 1 et 2 sont importantes et modifient la dérivation finale. Par contre, l'ordre d'application des transformations a et b n'a aucune importance : on obtiendra de toute façon le même réseau. Notons en outre que si R_1 et R_2 sont deux réseaux CR-valides pour un même séquent, alors $R_1 \cup R_2$ est aussi un réseau CR-valide pour ce séquent. Par conséquent si l'on note $\bowtie$ l'opération de superposition relative à la transformation a, on peut écrire un système de type pour les réseaux comme le décrit la figure 4.3. Remarquons que l'associativité de $\bowtie$ incarne la permutabilité des instances des règles $\wedge$ et Cut entre elles. La permutabilité des instances de la règle $\vee$ entre elles ainsi qu'avec les instances des règles $\wedge$ et Cut est incarnée par le fait que la règle $\vee$ ne modifie pas le réseau de preuve.

Nous nous devons d'éclaircir un point concernant les règles $\wedge$, Cut et Union. Considérons par exemple la règle $\wedge$. Du point de vue *bottom-up*, si l'on part d'un réseau de preuve R CR-valide pour un séquent $\vdash \varphi \wedge \psi, \Delta$, il peut exister plusieurs paires de réseaux (R_1, R_2) telles que $R_1 \bowtie R_2 = R$. Toutes ces paires ne constituent d'ailleurs pas forcément des réseaux CR-valides pour $\vdash \varphi, \Delta$ et $\vdash \psi, \Delta$. Néanmoins il est garanti qu'il *existe* une paire constituant des réseaux CR-valides pour ces deux séquents : pour les règles $\wedge$ et Cut, c'est justement la propriété 4.2.2.

La correspondance entre mécanisme de séquentialisation est aussi valable pour les autres règles de typages. Ainsi la séquentialisation s'exprime comme l'application *bottom-up* du système de typage de la figure 4.3. De manière duale, le mécanisme de parallélisation s'exprime comme l'application *top-down* du système de typage de la figure 4.3.

4.3 Coupures commutatives

Comme nous l'avons expliqué dans l'introduction du présent chapitre, notre intêrêt pour les permutations d'inférences (que nous venons de personnaliser par le typage réversible de la précédente section) provient en grande partie du comportement de l'élimination des coupures. En effet notre objectif est de trouver des représentants des classes d'équivalence de dérivations en calcul des séquents modulo permutations puis d'étudier l'élimination des coupures sur ces représentants. Plus précisément nous souhaitons obtenir une élimination des coupures fortement normalisante et confluente. L'approche de Lamarche et Straßburger que nous venons de développer est, dans le cadre de la logique propositionnelle, concluante : ils proposent une élimination des coupures fortement normalisante et confluente sans utiliser aucune stratégie restrictive. Notons tout d'abord que la règle Union du système de type de la figure 4.3 permet de corriger la première cause de non-confluence : une dérivation

$$\text{Cut}\ \dfrac{\text{WR}\ \dfrac{R_1 \vdash \Delta}{R_1 \vdash \varphi, \Delta} \qquad \text{WL}\ \dfrac{R_2 \vdash \Delta}{R_2 \vdash \neg\varphi, \Delta}}{R_1 \bowtie R_2 \vdash \Delta}$$

peut alors se réduire vers

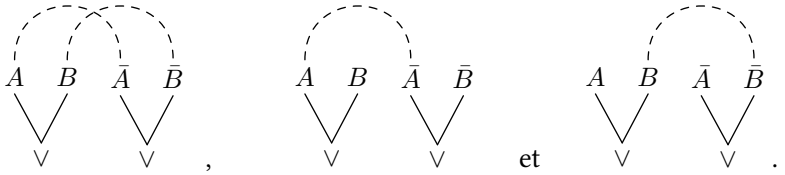
$$\text{Union}\ \dfrac{R_1 \vdash \Delta \qquad R_2 \vdash \Delta}{R_1 \cup R_2 \vdash \Delta}$$

Quant à la deuxième cause de non-confluence que nous avons décrite au début du présent chapitre et relative à l'ordre d'application des décompositions indépendantes de connecteurs, il est clair que l'élimination de la *bureaucratie de la syntaxe* opérée par l'approche des réseaux de preuve la supprime pûrement et simplement : par

exemple les preuves

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A, B}\,\text{Axiom}}{A, B \vdash A \vee B}\,\vee\text{R}}{A \wedge B \vdash A \vee B}\,\wedge\text{L} \qquad \text{et} \qquad \dfrac{\dfrac{\dfrac{}{A, B \vdash A, B}\,\text{Axiom}}{A \wedge B \vdash A, B}\,\wedge\text{L}}{A \wedge B \vdash A \vee B}\,\vee\text{R}$$

sont représentées par les mêmes réseaux de preuve, en l'occurrence



4.4 Représentation dans l'espace

Les développements détaillés par cette section constituent l'une de mes contributions originales. Je les expose par ailleurs dans un rapport de recherche (Houtmann, 2009). Dans cette section, nous proposons une représentation des réseaux de preuve dans un espace à trois dimensions. Celle-ci repose sur une représentation des séquents dans un espace à deux dimensions. Plus précisément, un séquent est représenté par un graphe orienté acyclique planaire ayant un seul noeud initial ainsi qu'un seul noeud terminal. Pour cela, nous définissons d'abord deux opérations auxiliaires sur de tels graphes. Si A et B sont de tels graphes, le graphe $A|B$ est le graphe obtenu en reliant simplement le noeud initial de A au noeud terminal de B ; le graphe $\frac{A}{B}$ est le graphe obtenu en rajoutant un noeud initial frais relié aux noeuds initiaux de A et B ainsi qu'un noeud terminal frais auquel les noeuds terminaux de A et B sont reliés. Si A et B sont tous les deux des graphes orientés acycliques planaires, alors $A|B$ et $\frac{A}{B}$ le sont aussi.

Considérons à présent la liste des atomes apparaissant dans un séquent. Une telle liste est un objet à une dimension qui ne traduit pas la structure logique du séquent. Cette structure est héritée des connecteurs et se traduit par les résolutions conjonctives selectionnant des sous-listes de la liste initiale d'atomes. Pour prouver un séquent, une liaison doit être trouvée pour chaque sous-liste (définition 4.1.2). Nous proposons à présent de remplacer la liste unidimensionnel d'atomes par un objet bidimensionnel qui représente le séquent en entier et contient en particulier la structure logique correspondant aux résolutions conjonctives : la représentation d'une formule φ, noté $\langle\varphi\rangle$, est le graphe orienté acyclique planaire ayant un seul noeud initial et un seul noeud terminal défini inductivement comme suit.

- Si φ est un atome, alors $\langle\varphi\rangle$ est un noeud unique étiqueté par l'atome φ.
- Si φ est une disjonction $\psi_1 \vee \psi_2$, alors $\langle\varphi\rangle$ est le graphe $\langle\psi_1\rangle|\langle\psi_2\rangle$.
- Si φ est une conjonction $\psi_1 \wedge \psi_2$, alors $\langle\varphi\rangle$ est le graphe $\frac{\langle\psi_1\rangle}{\langle\psi_2\rangle}$.

Le graphe $\langle\vdash \varphi_1, \ldots, \varphi_n\rangle$ associé à un séquent $\vdash \varphi_1, \ldots, \varphi_n$ est le graphe

$$\langle\varphi_1\rangle|\ldots|\langle\varphi_n\rangle\ .$$

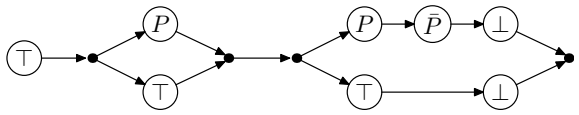


FIG. 4.4 – $\langle \vdash \top \vee (P \wedge \top), (P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot) \rangle$

Par exemple le graphe

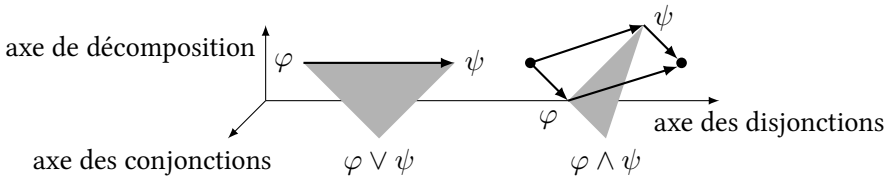
$$\langle \vdash \top \vee (P \wedge \top), (P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot) \rangle$$

est représenté en figure 4.4. Si L est un séquent, chaque chemin depuis le noeud initial jusqu'au noeud terminal dans $\langle L \rangle$ correspond à un séquent $\vdash a_1, \ldots, a_n$ où les a_i sont les étiquettes des noeuds du chemin. Les séquents ainsi obtenus correspondent exactement aux séquents que l'on peut obtenir en décomposant tous les connecteurs de L en utilisant des règles du calcul des séquents. Ils correspondent aussi aux résolutions conjonctives de L. En particulier pour le graphe représenté par la figure 4.4, les chemins du noeud initial au noeud terminal correspondent bien aux résolutions conjonctives de la figure 4.2 : dans ce cas, les séquents sont

$$\vdash \top, P, P, \bar{P}, \bot\,, \quad \vdash \top, \top, P, \bar{P}, \bot\,, \quad \vdash \top, P, \top, \bot\,, \quad \text{and} \quad \vdash \top, \top, \top, \bot\,.$$

Pour chaque séquent L, le graphe $\langle L \rangle$ est censé être tracé sur le plan. Tout comme dans un séquent $\vdash \varphi_1, \ldots, \varphi_n$, l'axe horizontale représente les disjonctions. L'axe verticale représente les conjonctions. Un graphe $\langle \varphi \vee \psi \rangle = \langle \varphi \rangle | \langle \psi \rangle$ est tracé comme une séquence horizontale $\langle \varphi \rangle \to \langle \psi \rangle$ et un graphe $\langle \varphi \wedge \psi \rangle = \frac{\langle \varphi \rangle}{\langle \psi \rangle}$ est tracé comme un *fork* vertical $\cdot \begin{smallmatrix} \nearrow \langle \varphi \rangle \searrow \\ \searrow \langle \psi \rangle \nearrow \end{smallmatrix} \cdot$. Le tracé obtenu se comporte particulièrement bien par rapport à la négation qui, si l'on échange l'axe des disjonctions et l'axe des conjonctions (en effectuant par exemple une symétrie d'axe $x = y$), ne modifie pas la position des noeuds : le graphe associé à la négation du sequent $\vdash \top \vee (P \wedge \top), (P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot)$ est représenté en figure 4.5, au dessus d'une représentation en pointillé du graphe de la figure 4.4.

Nous sommes maintenant capables de représenter des séquents dans un espace à deux dimensions. Cette représentation, tout comme la représentation usuelle des réseaux de preuve, consiste à décomposer recursivement les connecteurs du séquent. Nous proposons de traduire cette itération récursive sur une troisième axe appelée l'*axe de décomposition* que nous ajoutons à l'*axe des conjonctions* et à l'*axe des disjonctions*. Les disjonctions et les conjonctions sont alors représentées de la manière suivante.



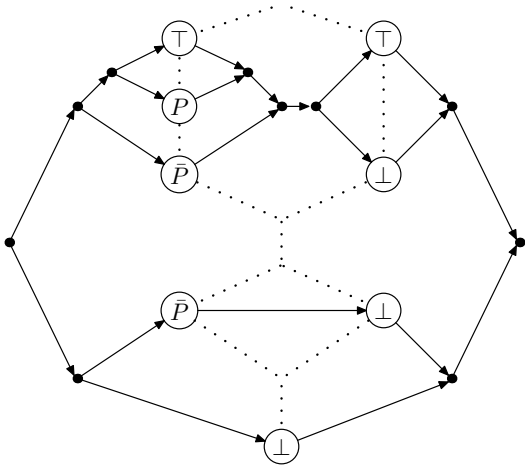


FIG. 4.5 – Negation du graphe de la figure 4.4

La représentation d'une formule consiste alors à représenter récursivement chacun de ses connecteurs. On obtient ainsi un arbre en trois dimensions. La représentation d'un séquent complet consiste alors à représenter chacune de ses formules côte à côte le long de l'axe des disjonctions. Chaque étape du processus de représentation correspond à un graphe partiellement calculé. Chaque décomposition d'un connecteur correspond à un calcul de plus vers le graphe final correspondant au séquent initial. Les feuilles de la forêt tridimensionnelle obtenue finalement correspondent au graphe associé au séquent initial. Par exemple, l'arbre en trois dimensions associé au séquent $\vdash \top \vee (P \wedge \top), (P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot)$ est tracé en figure 4.6. La figure contient aussi le tracé de plusieurs graphes correspondants à des décompositions partielles des connecteurs du séquent. Plus précisément de bas en haut sont tracés les graphes correspondant à

$$[\top \vee (P \wedge \top)] \mid [(P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot)]\,, \quad [\top] \mid [(P \wedge \top)] \mid \frac{[P \vee (\bar{P} \vee \bot)]}{[\top \vee \bot]}\,,$$
$$[\top] \mid \frac{[P]}{[\top]} \mid \frac{[P] \mid [\bar{P} \vee \bot]}{[\top] \mid [\bot]} \quad \text{et} \quad [\top] \mid \frac{[P]}{[\top]} \mid \frac{[P] \mid [\bar{P}] \mid [\bot]}{[\top] \mid [\bot]}\,.$$

Remarquons que nous aurions pu aussi représenter des graphes obliques tels que celui correspondant à la décomposition $[\top] \mid \frac{[P]}{[\top]} \mid \frac{[P \vee (\bar{P} \vee \bot)]}{[\top \vee \bot]}$.

Nous n'avons pas encore expliqué comment tracer la relation $\sim$ d'un réseau de preuve $(L, \sim)$. Si l et m sont deux positions de L telles que $l \sim m$, alors les positions l et m apparaissent comme des sommets de la forêt tridimensionnelle correspondant à L et le lien $l \sim m$ est représenté par une courbe reliant ces deux sommets. Par conséquent la relation $\sim$ est représentée par un ensemble de courbes reliant les sommets de la forêt tridimensionnelle Pour chaque graphe, une résolution conjonctive correspond à un chemin du noeud initial au noeud terminal. Similairement dans la forêt, elle correspond à une surface (un ensemble de polygones) entre la courbe

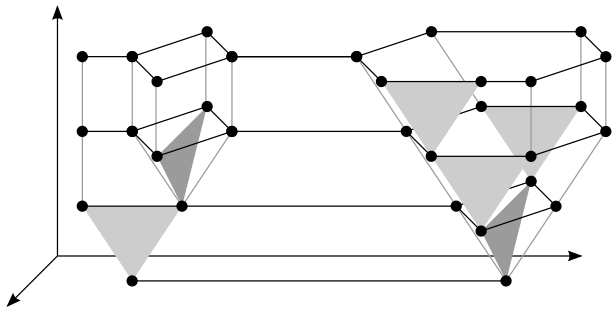
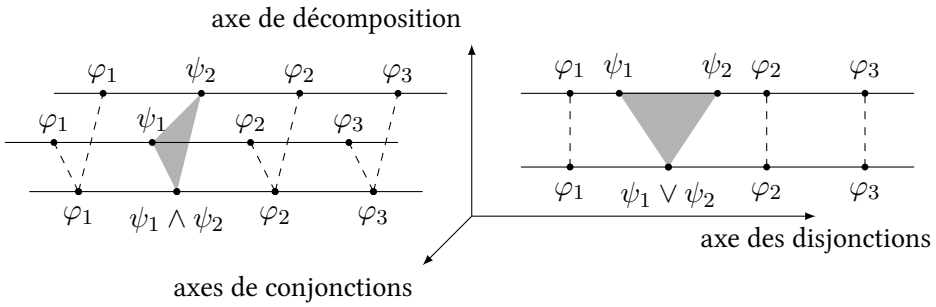


FIG. 4.6 – Un réseau de preuve tridimensionnel

composée des arêtes reliant les noeuds initaux et la courbe composée des arêtes reliant les noeuds terminaux. Si chaque surface reliant ces deux courbes contient une courbe de la relation $\sim$, alors la forêt représente un réseau de preuve CR-valide.

Nous avons vu qu'en termes de graphes à deux dimensions, la negation est un échange de l'axe des disjonctions avec l'axe des conjonctions. Cela reste valide pour les réseaux de preuve en trois dimensions : les sommets correspondant aux formules et les triangles correspondant aux décompositions des connecteurs ne sont pas autrement modifiés. Par contre, les graphes doivent être commutés comme nous l'avons fait en figure 4.5.

Si L est un séquent, chaque chemin du noeud initial de $\langle L \rangle$ jusqu'au noeud terminal correspond à un séquent atomique. La liste de ces chemins (ou de façon équivalente de ces séquents) est une version expansée du graphe $\langle L \rangle$. La taille d'une telle liste de chemins peut d'ailleurs être exponentielle en la taille du graphe initial (par exemple pour le séquent $\vdash \varphi_1 \wedge \psi_1, \varphi_2 \wedge \psi_2, \ldots, \varphi_n \wedge \psi_n$). Cette expansion en liste de séquents (objet bidimensionnel) correspond à l'expansion qui permet de transformer des réseaux de preuve en dérivation du calcul des séquents. Chaque graphe bidimensionnel est expansé en une liste de séquents bidimensionnelle et chaque réseau de preuve tridimensionnel est expansé en une dérivation du calcul des séquents tridimensionnelle. Cette expansion est bel et bien la *séquentialisation* étudiée en section 4.2. La différence entre réseaux de preuve et calcul des séquents réside dans la décomposition respectivement parallèle et séquentielle des connecteurs. En calcul des séquents, on ne peut décomposer qu'un seul connecteur à la fois. Les contextes (c'est-à-dire les autres formules du séquent) sont juste copiées dans les conclusions et prémisses de l'inférence. La séquentialisation correspond donc à transformer une seule étape contenant plusieurs décompositions simultanées en une séquence d'étapes contenant chacune une seule décomposition (en l'occurrence de $\vee$ ou de $\wedge$) et plusieurs duplications. Les représentations en trois dimensions de l'introduction du connecteur $\wedge$ et du connecteur $\vee$ sont les suivantes.



Dans de telles inférences tridimensionnelles, un séquent apparaît comme un segment parallèle à l'axe des disjonctions. Si l'on lit ces inférences du bas vers le haut, à gauche la décomposition du segment $\ldots \varphi_1, \psi_1 \wedge \psi_2, \varphi_2, \varphi_3 \ldots$ crée deux nouveaux segments $\ldots \varphi_1, \psi_1, \varphi_2, \varphi_3 \ldots$ et $\ldots \varphi_1, \psi_2, \varphi_2, \varphi_3 \ldots$; à droite la décomposition du segment $\ldots \varphi_1, \psi_1 \vee \psi_2, \varphi_2, \varphi_3 \ldots$ crée un nouveau segment $\ldots \varphi_1, \psi_1, \psi_2, \varphi_2, \varphi_3 \ldots$. La duplication des contextes est représentée par des lignes pointillées. L'utilisation de ces inférences tridimensionnelles permet de construire des dérivations du calcul des séquents en trois dimensions. Par exemple la dérivation tridimensionnelle correspondant à

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top, \top, P, \bar{P}, \bot}{\vdash \top, \top, P, (\bar{P} \vee \bot)}}{\vdash \top, \top, (P \vee (\bar{P} \vee \bot))} \quad \dfrac{\vdash \top, \top, \top, \bot}{\vdash \top, \top, (\top \vee \bot)}}{\vdash \top, \top, (P \vee (\bar{P} \vee \bot)) \wedge (\top \vee \bot)}}{\vdots}}{\vdots}}{\vdots}}{\vdots}}{\vdots}}{\vdots}}{\vdots}}{\vdots}}{\vdots}}{\vdots}$$

est représentée en figure 4.7. Remarquons qu'il existe une bijection entre les sommets de la représentation tridimensionnelle et les occurrences de formules de la dérivation initiale ainsi qu'entre les segments parallèles à l'axe des disjonctions et les occurrences de séquents. La même dérivation tridimensionnelle peut être obtenue en appliquant un processus de séquentialisation sur le réseau tridimensionnel de la figure 4.6. La séquentialisation consiste à choisir un ordre pour la décomposition des connecteurs du séquent initial. Par exemple si un séquent contient à la fois les formules $\varphi_1 \wedge \varphi_2$ et $\psi_1 \vee \psi_2$, on peut choisir de décomposer d'abord la conjonction ou bien la disjonction. Cela mène alors respectivement aux dérivations en calcul des séquent

$$\dfrac{\dfrac{\vdash \varphi_1, \psi_1, \psi_2 \qquad \vdash \varphi_2, \psi_1, \psi_2}{\vdash \varphi_1 \wedge \varphi_2, \psi_1, \psi_2}}{\vdash \varphi_1 \wedge \varphi_2, \psi_1 \vee \psi_2} \qquad \text{et} \qquad \dfrac{\dfrac{\vdash \varphi_1, \psi_1, \psi_2}{\vdash \varphi_1, \psi_1 \vee \psi_2} \qquad \dfrac{\vdash \varphi_2, \psi_1, \psi_2}{\vdash \varphi_2, \psi_1 \vee \psi_2}}{\vdash \varphi_1 \wedge \varphi_2, \psi_1 \vee \psi_2} \; .$$

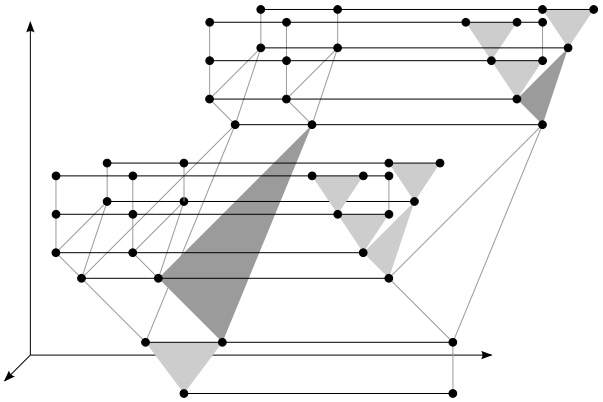


FIG. 4.7 – Une dérivation tridimensionnelle

Par conséquent la séquentialisation d'un réseau tridimensionnel n'est pas unique (à l'instar de la séquentialisation d'un réseau usuel) : la paire critique contenant ces deux dérivations est par exemple traduite en une paire critique pour la séquentialisation en trois dimensions représentée en figure 4.8.

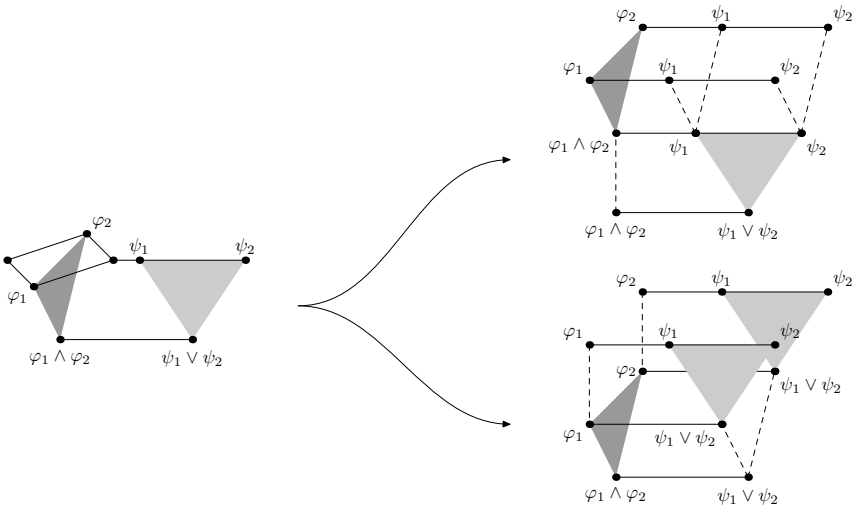


FIG. 4.8 – Étapes de séquentialisation

Deuxième partie

De la théorie à la pratique, pratique des théories

Chapitre 5

Déduction modulo

Toute déduction repose sur une certaine quantité d'axiomes supposés évidents, non démontrables et universels. Lorsqu'on choisit de s'appuyer sur un plus grand nombre d'axiomes, on accroît naturellement les conclusions que l'on peut atteindre. Le pouvoir déductif est donc augmenté. Inversement sans aucun axiome, c'est-à-dire de la déduction pure, ne dérivent que des tautologies d'un intérêt limité. Le point crucial de la définition d'un système déductif est donc la description de l'utilisation des axiomes qu'on lui donne. Plus cette utilisation sera intuitive, plus l'utilisation du système déductif sera simple. Considérons à titre d'exemple la preuve de la loi de Pierce dans le système $\mathcal{MP}$ décrite par la figure 2.1. Le calcul des séquents LK permet de prouver cette même formule de façon bien plus concise.

$$\dfrac{\dfrac{\dfrac{}{\varphi \vdash \varphi}\,\textsc{Axiom}}{\vdash \varphi \Rightarrow \psi, \varphi}\,\Rightarrow\!\text{R}}{\vdash ((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi}\,\Rightarrow\!\text{R}$$

Considérons à présent un deuxième exemple. Munissons nous des axiomes

$$\forall x.\forall y.\quad x = y \Rightarrow S(x) = S(y)\ , \qquad \forall x.\forall y.\quad x + S(y) = S(x + y)\ ,$$
$$\forall x.\quad x + Z = x \quad \text{et} \quad \forall x.\forall y.\forall z.\quad x = y \Rightarrow y = z \Rightarrow x = z\ .$$

Notons $\mathcal{Th}$ le multi-ensemble contenant une fois chacune de ces formules. Nous utiliserons les symboles 0, 1, 2, 3, 4… pour représenter les termes Z, $S(Z)$, $S(S(Z))$, $S(S(S(Z)))$, $S(S(S(S(Z))))$… Une preuve dans NJ du séquent $\mathcal{Th} \vdash 2+2 = 4$ est décrite en figure 5.1. Les abbréviations $\mathrm{Tr}(t, u, v)$, $\mathrm{Zero}(t)$, $\mathrm{Rec}(t, u)$, et $\mathrm{Succ}(t, u)$ y représentent respectivement les dérivations

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{Th} \vdash \forall x.\forall y.\forall z.x = y \Rightarrow y = z \Rightarrow x = z}\,\textsc{Axiom}}{\mathcal{Th} \vdash \forall y.\forall z.t = y \Rightarrow y = z \Rightarrow t = z}\,\forall\textsc{Elim}}{\mathcal{Th} \vdash \forall z.t = u \Rightarrow u = z \Rightarrow t = z}\,\forall\textsc{Elim}}{\mathcal{Th} \vdash t = u \Rightarrow u = v \Rightarrow t = v}\,\forall\textsc{Elim}\quad ,$$

$$\dfrac{\dfrac{\text{Succ}(2+0,2)}{\mathcal{T}h \vdash 2+0=2 \Rightarrow S(2+0)=3} \qquad \dfrac{\text{Zero}(2)}{\mathcal{T}h \vdash 2+0=2}}{\mathcal{T}h \vdash S(2+0)=3}$$

$$\dfrac{\dfrac{\dfrac{\text{Tr}(2+1,S(2+0),3)}{\mathcal{T}h \vdash 2+1=S(2+0) \Rightarrow S(2+0)=3 \Rightarrow 2+1=3} \quad \dfrac{\text{Rec}(2,0)}{\mathcal{T}h \vdash 2+1=S(2+0)}}{\mathcal{T}h \vdash S(2+0)=3 \Rightarrow 2+1=3} \qquad \vdots}{\mathcal{T}h \vdash 2+1=3}$$

$$\dfrac{\dfrac{\text{Succ}(2+1,3)}{\mathcal{T}h \vdash 2+1=3 \Rightarrow S(2+1)=4} \qquad \vdots}{\mathcal{T}h \vdash S(2+1)=4}$$

$$\dfrac{\dfrac{\dfrac{\text{Tr}(2+2,S(2+1),4)}{\mathcal{T}h \vdash 2+2=S(2+1) \Rightarrow S(2+1)=4 \Rightarrow 2+2=4} \quad \dfrac{\text{Rec}(2,1)}{\mathcal{T}h \vdash 2+2=S(2+1)}}{\mathcal{T}h \vdash S(2+1)=4 \Rightarrow 2+2=4} \qquad \vdots}{\mathcal{T}h \vdash 2+2=4}$$

FIG. 5.1 – Preuve de $\mathcal{T}h \vdash 2+2$ dans NJ

$$\dfrac{\text{Axiom}\ \dfrac{}{\mathcal{T}h \vdash \forall x.x+0=x}}{\mathcal{T}h \vdash t+0=t}\ \forall\text{Elim} \quad ,$$

$$\dfrac{\dfrac{\text{Axiom}\ \dfrac{}{\mathcal{T}h \vdash \forall x.\forall y.x=y \Rightarrow S(x)=S(y)}}{\mathcal{T}h \vdash \forall y.t=y \Rightarrow S(t)=S(y)}\ \forall\text{Elim}}{\mathcal{T}h \vdash t=u \Rightarrow S(t)=S(u)}\ \forall\text{Elim}$$

et

$$\dfrac{\dfrac{\text{Axiom}\ \dfrac{}{\mathcal{T}h \vdash \forall x.\forall y.x+S(y)=S(x+y)}}{\mathcal{T}h \vdash \forall y.t+S(y)=S(t+y)}\ \forall\text{Elim}}{\mathcal{T}h \vdash t+S(u)=S(t+u)}\ \forall\text{Elim} \quad .$$

Observons que nous venons de dépenser une quantité considérable d'énergie pour finalement écrire

$$2+2=S(2+1)=S(S(2+0))=S(S(2))=4\ .$$

Cet argument est purement calculatoire. On peut par exemple le représenter par la réduction $2 + 2 \to^* 4$ dans le système de réduction contenant les règles

$$\begin{cases} x + Z & \to x\,; \\ x + S(y) & \to S(x + y)\,. \end{cases}$$

Ce système de réécriture étant convergent (voir exemple 1.1.3), on peut décider $t \leftrightarrow^* u$ pour tous termes t et u. Par conséquent, non seulement la dérivation en figure 5.1 est longue, mais elle est inutile car elle certifie une propriété vérifiable automatiquement. Poincaré qualifie une telle preuve de « stérile » en soulignant la différence entre vérification (c'est-à-dire le calcul) et démonstration (c'est-à-dire la déduction).

> *La vérification diffère précisément de la véritable démonstration, parce qu'elle est purement analytique et parce qu'elle est stérile. Elle est stérile parce que la conclusion n'est que la traduction des prémisses dans un autre langage. La démonstration véritable est féconde au contraire parce que la conclusion y est, en un sens, plus générale que les prémisses.*
>
> Henri Poincaré, *La Science et l'Hypothèse* (1902).

Si l'on accepte ce principe, on se doit d'omettre les calculs lors de l'écriture des démonstrations car ils ne consistent qu'en de futiles *réécritures de prémisses dans un autre langage*. La déduction modulo propose d'intégrer ce principe dans les systèmes permettant d'écrire des démonstrations. La déduction est incarnée par la déduction naturelle ou le calcul des séquents ; le calcul est incarné par la réécriture. Le reste de ce chapitre consiste en une introduction de la déduction modulo et des systèmes qui en découle ainsi que d'une étude de leurs propriétés.

5.1 Principes de la déduction modulo

La déduction modulo est une approche qui propose de supprimer les arguments calculatoires des preuves. En effet il n'est pas souhaitable que de tels arguments apparaissent explicitement dans une preuve car un calcul peut toujours être automatisé. C'est même une propriété intrinsèque du calcul. Si certains arguments calculatoires d'une preuve sont omis, il peuvent toujours être retrouvés automatiquement et la preuve reste donc un certificat moralement valide. L'approche « modulo » consiste donc à représenter le calcul par une congruence sur les formules. Le mécanisme de déduction opère alors *modulo* cette congruence. Plus précisément, cette congruence est définie comme la congruence $(\equiv) = (\leftrightarrow^*)$ générée par une relation de réécriture $\to$. La déduction opère alors sur les classes d'équivalence associées à cette congruence. Dans la pratique et lorsqu'on utilise des systèmes déductifs comme la déduction naturelle ou le calcul des séquents, cela revient à rajouter les règles

$$\textsc{ConvR}_{\equiv}\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \psi, \Delta}\ \varphi \equiv \psi \qquad \text{et} \qquad \textsc{ConvL}_{\equiv}\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \psi \vdash \Delta}\ \varphi \equiv \psi\,.$$

Par exemple si $\equiv$ est la congruence générée par les règles de réécriture

$$\begin{cases} x + Z & \to x \ ; \\ x + S(y) & \to S(x + y) \ ; \\ x = x & \to \top \ ; \end{cases}$$

alors la dérivation de la figure 5.1 devient

$$\dfrac{\dfrac{}{\vdash \top}\ \top\text{R ou } \top\text{Elim}}{\vdash 2 + 2 = 4}\ \text{ConvR}_{\equiv} \ .$$

Dans le système de réécriture ci-dessus, une différence de taille sépare la dernière règle de réécriture des deux autres. En effet les deux premières règles agissent sur les termes du premier ordre alors que la dernière agit sur les formules. Cette différence est cruciale. Nous verrons qu'il est par exemple impossible de rendre un système déductif comme NJ ou LK inconsistant en utilisant seulement des règles de réécriture sur les termes. Par contre c'est très facile avec des règles de réécriture sur les formules, comme la règle $\top \to \bot$. Cette règle égalise le vrai et le faux. Plus généralement, il est peu souhaitable de vouloir égaliser des formules dont les connecteurs de tête sont différents. En effet le *sens* des connecteurs est en quelque sorte donné par leur comportement dans les règles d'inférences. Par exemple le sens du connecteur $\wedge$ est donné par les règles $\wedge$L et $\wedge$R dans LK. Si l'on décide d'égaliser une certaine conjonction avec, disons, une disjonction, le sens donné à la conjonction est irrémédiablement modifié. D'un point de vue plus pragmatique, cela introduit un chevauchement et une paire critique entre le traîtement respectifs par les règles d'inférences de la conjonction et de la disjonction. Nous ne nous intéresserons donc qu'aux congruences vérifiant la propriété de protection des connecteurs définie comme suit.

Définition 5.1.1 (Protection des connecteurs). *Nous dirons qu'une congruence $\equiv$ vérifie la propriété de protection des connecteurs lorsque*

- *si $\psi \equiv \varphi$ avec $\psi \in \{\top, \bot\}$, alors φ est une proposition atomique ou bien $\varphi = \psi$;*
- *si $\neg\psi \equiv \varphi$, alors φ est une proposition atomique ou bien $\varphi = \neg\varphi'$ avec $\psi \equiv \varphi'$;*
- *si $\psi_1 \otimes \psi_2 \equiv \varphi$ avec $\otimes \in \{\Rightarrow, \wedge, \vee\}$, alors φ est une proposition atomique ou bien $\varphi = \varphi_1 \otimes \varphi_2$ avec $\psi_1 \equiv \varphi_1$ et $\psi_2 \equiv \varphi_2$;*
- *si $Qx.\psi \equiv \varphi$ avec $Q \in \{\exists, \forall\}$, alors φ est une proposition atomique ou bien $\varphi = Qx.\varphi'$ avec $\psi \equiv \varphi'$.*

Pour résumer, si $\equiv$ satisfait la propriété de protection des connecteurs, c'est que lorsque $\varphi \equiv \psi$, soit φ ou ψ est une proposition atomique, soit ces deux formules ont le même connecteur de tête. En particulier les congruences que nous utiliserons en général sont celles engendrées par des règles de termes et des règles de prédicats.

Définition 5.1.2 (Règles de termes et de prédicats). *Une règle de termes est simplement une règle de réécriture sur les termes du premier ordre. Une règle de prédicats est une règle de réécriture sur les formules dont la partie gauche est une proposition atomique.*

Nous nous restreindrons d'ailleurs aux systèmes de réécriture confluents. Dans ce cas, une congruence engendrée par de telles règles vérifie bien la propriété de protection des connecteurs.

Propriété 5.1.1. *Toute congruence engendrée par un ensemble de règles de termes et de prédicats correspondant à une relation de réécriture convergente vérifie la propriété de protection des connecteurs.*

Démonstration. Soit donc un ensemble de règles de termes et de prédicats correspondant à une relation de réécriture convergente et engendrant une congruence $\equiv$. Supposons par exemple que $\varphi_1 \wedge \varphi_2 \equiv \psi$. Remarquons tout d'abord que tous les réduits de $\varphi_1 \wedge \varphi_2$ seront des conjonctions de la forme $\varphi'_1 \wedge \varphi'_2$ où φ'_i est un réduit de φ_i pour $i \in \{1, 2\}$. Remarquons en outre que seuls les conjonctions et les propositions atomiques peuvent avoir pour réduits des conjonctions. Puisque $\varphi_1 \wedge \varphi_2 \equiv \psi$ et puisque la relation de réécriture est convergente, ces deux termes admettent un réduit commun ce qui démontre que ψ est soit une conjonction, soit une proposition atomique. En outre si ψ est une conjonction $\psi_1 \wedge \psi_2$, ses réduits sont de la forme $\psi'_1 \wedge \psi'_2$ où ψ'_i est un réduit de ψ_i pour $i \in \{1, 2\}$. Le réduit commun de ψ et $\varphi_1 \wedge \varphi_2$ est donc de la forme $\varphi'_1 \wedge \varphi'_2$ avec φ'_i réduit commun de φ_i et ψ_i pour $i \in \{1, 2\}$. Cela démontre alors que $\varphi_i \equiv \psi_i$ pour $i \in \{1, 2\}$. □

Les systèmes de déduction modulo sont alors définis comme suit.

Définition 5.1.3 (Déduction naturelle intuitionniste modulo). *Si $\equiv$ est une congruence sur les formules, alors la déduction naturelle intuitionniste modulo $\equiv$, notée $N\mathcal{J}_{\equiv}$, est définie comme l'ensemble contenant les inférences de la figure 5.2.*

Définition 5.1.4 (Déduction naturelle classique modulo). *Si $\equiv$ est une congruence sur les formules, alors la déduction naturelle classique modulo $\equiv$, notée $NK_{\equiv}$, est définie comme l'ensemble contenant les inférences de $N\mathcal{J}_{\equiv}$ ainsi que les inférences*

$$\textsc{Excluded Middle}\ \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi}\ \psi \equiv \neg\neg\varphi\ .$$

Définition 5.1.5 (Calcul des séquents intuitionniste modulo). *Si $\equiv$ est une congruence sur les formules, alors le calcul des séquents intuitionniste modulo $\equiv$, noté $L\mathcal{J}_{\equiv}$, est défini comme l'ensemble contenant les inférences de la figure 5.3.*

Définition 5.1.6 (Calcul des séquents classique modulo). *Si $\equiv$ est une congruence sur les formules, alors le calcul des séquents classique modulo $\equiv$, noté $LK_{\equiv}$, est défini comme l'ensemble contenant les inférences de la figure 5.4.*

$$\textsc{Axiom}\ \frac{}{\Gamma, \varphi \vdash \psi}\ \psi \equiv \varphi$$

$$\Rightarrow\textsc{Intro}\ \frac{\Gamma, \psi_1 \vdash \psi_2}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \Rightarrow \psi_2 \qquad \Rightarrow\textsc{Elim}\ \frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi_1}{\Gamma \vdash \psi_2}\ \varphi \equiv \psi_1 \Rightarrow \psi_2$$

$$\wedge\textsc{Intro}\ \frac{\Gamma \vdash \psi_1 \qquad \Gamma \vdash \psi_2}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \wedge \psi_2 \qquad \wedge\textsc{Elim}_1\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \psi_1}\ \varphi \equiv \psi_1 \wedge \psi_2$$

$$\vee\textsc{Intro}_1\ \frac{\Gamma \vdash \psi_1}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \vee \psi_2 \qquad \wedge\textsc{Elim}_2\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \psi_2}\ \varphi \equiv \psi_1 \wedge \psi_2$$

$$\vee\textsc{Intro}_2\ \frac{\Gamma \vdash \psi_2}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \vee \psi_2$$

$$\vee\textsc{Elim}\ \frac{\Gamma \vdash \varphi \qquad \Gamma, \psi_1 \vdash \varphi' \qquad \Gamma, \psi_2 \vdash \varphi'}{\Gamma \vdash \varphi'}\ \varphi \equiv \psi_1 \vee \psi_2$$

$$\forall\textsc{Intro}\ \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi}\ \begin{cases} x \notin \mathcal{FV}(\Gamma) \\ \varphi \equiv \forall x.\psi \end{cases} \qquad \forall\textsc{Elim}\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \psi[t/x]}\ \varphi \equiv \forall x.\psi$$

$$\exists\textsc{Intro}\ \frac{\Gamma \vdash \psi[t/x]}{\Gamma \vdash \varphi}\ \varphi \equiv \exists x.\psi \qquad \exists\textsc{Elim}\ \frac{\Gamma \vdash \varphi \quad \Gamma, \psi \vdash \varphi'}{\Gamma \vdash \varphi'}\ \begin{cases} x \notin \mathcal{FV}(\Gamma, \varphi') \\ \varphi \equiv \exists x.\psi \end{cases}$$

$$\neg\textsc{Intro}\ \frac{\Gamma, \psi \vdash \bot}{\Gamma \vdash \varphi}\ \varphi \equiv \neg\psi \qquad \neg\textsc{Elim}\ \frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \bot}\ \varphi \equiv \neg\psi$$

$$\top\textsc{Intro}\ \frac{}{\Gamma \vdash \varphi}\ \varphi \equiv \top \qquad \bot\textsc{Elim}\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \psi}\ \varphi \equiv \bot$$

FIG. 5.2 – Système de déduction naturelle modulo $\mathrm{NJ}_{\equiv}$

$$\textsc{Axiom}\ \frac{}{\Gamma, \varphi \vdash \psi}\ \psi \equiv \varphi \qquad \textsc{Cut}\ \frac{\Gamma \vdash \varphi \qquad \Gamma, \psi \vdash \psi'}{\Gamma \vdash \psi'}\ \psi \equiv \varphi$$

$$\textsc{ContrL}\ \frac{\Gamma, \varphi, \psi \vdash \psi'}{\Gamma, \varphi \vdash \psi'}\ \psi \equiv \varphi$$

$$\Rightarrow\text{R}\ \frac{\Gamma, \psi_1 \vdash \psi_2}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \Rightarrow \psi_2 \qquad \Rightarrow\text{L}\ \frac{\Gamma, \psi_2 \vdash \psi' \qquad \Gamma \vdash \psi_1}{\Gamma, \varphi \vdash \psi'}\ \varphi \equiv \psi_1 \Rightarrow \psi_2$$

$$\wedge\text{R}\ \frac{\Gamma \vdash \psi_1 \qquad \Gamma \vdash \psi_2}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \wedge \psi_2 \qquad \wedge\text{L}\ \frac{\Gamma, \psi_1, \psi_2 \vdash \psi'}{\Gamma, \varphi \vdash \psi'}\ \varphi \equiv \psi_1 \wedge \psi_2$$

$$\vee\text{R1}\ \frac{\Gamma \vdash \psi_1}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \vee \psi_2 \qquad \vee\text{R2}\ \frac{\Gamma \vdash \psi_2}{\Gamma \vdash \varphi}\ \varphi \equiv \psi_1 \vee \psi_2$$

$$\vee\text{L}\ \frac{\Gamma, \psi_1 \vdash \psi' \qquad \Gamma, \psi_2 \vdash \psi'}{\Gamma, \varphi \vdash \psi'}\ \varphi \equiv \psi_1 \vee \psi_2$$

$$\forall\text{R}\ \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi}\ \begin{cases} x \notin \mathcal{FV}(\Gamma) \\ \varphi \equiv \forall x.\psi \end{cases} \qquad \forall\text{L}\ \frac{\Gamma, \psi[t/x] \vdash \psi'}{\Gamma, \varphi \vdash \psi'}\ \varphi \equiv \forall x.\psi$$

$$\exists\text{R}\ \frac{\Gamma \vdash \psi[t/x]}{\Gamma \vdash \varphi}\ \varphi \equiv \exists x.\psi \qquad \exists\text{L}\ \frac{\Gamma, \psi \vdash \psi'}{\Gamma, \varphi \vdash \psi'}\ \begin{cases} x \notin \mathcal{FV}(\Gamma, \psi') \\ \varphi \equiv \exists x.\psi \end{cases}$$

$$\neg\text{R}\ \frac{\Gamma, \psi \vdash \bot}{\Gamma \vdash \varphi}\ \varphi \equiv \neg\psi \qquad \neg\text{L}\ \frac{\Gamma \vdash \psi}{\Gamma, \varphi \vdash \bot}\ \varphi \equiv \neg\psi$$

$$\top\text{R}\ \frac{}{\Gamma \vdash \varphi}\ \varphi \equiv \top \qquad \top\text{L}\ \frac{\Gamma \vdash \psi}{\Gamma, \varphi \vdash \psi}\ \varphi \equiv \top \qquad \bot\text{L}\ \frac{}{\Gamma, \varphi \vdash \psi}\ \varphi \equiv \bot$$

FIG. 5.3 – Système calcul des séquents intuitionniste modulo $\text{LJ}_{\equiv}$

$$\textsc{Axiom}\ \frac{}{\Gamma, \varphi \vdash \psi, \Delta}\ \varphi \equiv \psi \qquad \textsc{Cut}\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma \vdash \Delta}\ \varphi \equiv \psi$$

$$\textsc{ContrL}\ \frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \psi \qquad \textsc{ContrR}\ \frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \psi$$

$$\neg\mathrm{R}\ \frac{\Gamma, \psi \vdash \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \neg\psi \qquad \neg\mathrm{L}\ \frac{\Gamma \vdash \psi, \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \neg\psi$$

$$\Rightarrow\mathrm{R}\ \frac{\Gamma, \psi_1 \vdash \psi_2, \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \psi_1 \Rightarrow \psi_2 \qquad \Rightarrow\mathrm{L}\ \frac{\Gamma, \psi_2 \vdash \Delta \quad \Gamma \vdash \psi_1, \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \psi_1 \Rightarrow \psi_2$$

$$\wedge\mathrm{R}\ \frac{\Gamma \vdash \psi_1, \Delta \quad \Gamma \vdash \psi_2, \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \psi_1 \wedge \psi_2 \qquad \wedge\mathrm{L}\ \frac{\Gamma, \psi_1, \psi_2 \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \psi_1 \wedge \psi_2$$

$$\vee\mathrm{R}\ \frac{\Gamma \vdash \psi_1, \psi_2, \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \psi_1 \vee \psi_2 \qquad \vee\mathrm{L}\ \frac{\Gamma, \psi_1 \vdash \Delta \quad \Gamma, \psi_2 \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \psi_1 \vee \psi_2$$

$$\forall\mathrm{R}\ \frac{\Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi, \Delta}\ \begin{cases} x \notin \mathcal{FV}(\Gamma, \Delta) \\ \varphi \equiv \forall x.\psi \end{cases} \qquad \forall\mathrm{L}\ \frac{\Gamma, \psi[t/x] \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \forall x.\psi$$

$$\exists\mathrm{R}\ \frac{\Gamma \vdash \psi[t/x], \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \exists x.\psi \qquad \exists\mathrm{L}\ \frac{\Gamma, \psi \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\ \begin{cases} x \notin \mathcal{FV}(\Gamma, \Delta) \\ \varphi \equiv \exists x.\varphi \end{cases}$$

$$\top\mathrm{R}\ \frac{}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \top \qquad \top\mathrm{L}\ \frac{\Gamma \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \top$$

$$\bot\mathrm{R}\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv \bot \qquad \bot\mathrm{L}\ \frac{}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv \bot$$

FIG. 5.4 – Système de calcul des séquents classique modulo $\mathrm{LK}_{\equiv}$

Bien que les règles $\text{ConvR}_{\equiv}$ et $\text{ConvL}_{\equiv}$ ne soient pas explicitement présentes dans les systèmes $\text{NJ}_{\equiv}$, $\text{NK}_{\equiv}$, $\text{LJ}_{\equiv}$ ou $\text{LK}_{\equiv}$, elles y sont admissibles. En fait tout séquent prouvable respectivement dans $\text{NJ}_{\equiv}$, $\text{NK}_{\equiv}$, $\text{LJ}_{\equiv}$ ou $\text{LK}_{\equiv}$ l'est aussi dans $\text{NJ}+\text{ConvL}_{\equiv}+\text{ConvR}_{\equiv}$, $\text{NK}+\text{ConvL}_{\equiv}+\text{ConvR}_{\equiv}$, $\text{LJ}+\text{ConvL}_{\equiv}+\text{ConvR}_{\equiv}$ ou $\text{LK}+\text{ConvL}_{\equiv}+\text{ConvR}_{\equiv}$ et inversement. Par exemple la dérivation de $\vdash 2+2=4$ que nous avons écrite plus haut à l'aide des règles d'inférence $\top\text{R}$ ou $\top\text{Elim}$ et $\text{ConvR}_{\equiv}$ peut s'écrire directement

$$\top\text{R ou }\top\text{Elim}\ \frac{}{\vdash 2+2=4}\ 2+2=4 \equiv \top\ .$$

La déduction modulo est donc un paradigme qui permet de raisonner modulo une congruence définie sur les formules. Cette congruence se doit de représenter un calcul. Plus précisément on se doit de pouvoir décider automatiquement si deux formules sont congrues. Savoir le décider est équivalent à savoir vérifier qu'une dérivation est effectivement valide en déduction modulo. En effet et pour une congruence donnée, si un algorithme permet de répondre, pour toutes formules φ et ψ, à la question « Est-ce que $\varphi \equiv \psi$? », alors il existe un algorithme qui permet de répondre, pour toute dérivation π, à la question « Est-ce que π est valide en déduction modulo ? ». Inversement et toujours pour une congruence donnée, s'il existe un algorithme qui permet de répondre aux questions du deuxième groupe, il en existe un qui permet de répondre aux questions du premier groupe : en effet si φ et ψ sont deux formules, la dérivation

$$\text{Axiom}\ \frac{}{\varphi \vdash \psi}$$

sera valide en déduction modulo si et seulement si $\varphi \equiv \psi$. Répondre à la question « Est-ce que cette dérivation est valide en déduction modulo ? » revient donc à répondre à la question « Est-ce que $\varphi \equiv \psi$? » et inversement.

5.2 Correction et complétude, théories et égalité

À première vue, la déduction modulo est donc une approche qui permet d'étendre un système déductif $\mathcal{A}$ en un système $\mathcal{A}_{\equiv}$ par le biais d'une congruence $\equiv$ engendrée généralement par un ensemble $\mathcal{R}$ de règles de termes et de prédicats. Le système étendu peut être défini comme $\mathcal{A}+\text{ConvR}_{\equiv}+\text{ConvL}_{\equiv}$ ou de manière équivalente en utilisant des définitions similaires aux définitions 5.1.3, 5.1.4, 5.1.5 et 5.1.6. Une analyse un peu plus détaillée de ce paradigme montre que ce n'est en fait pas une *extension* du système de déduction, mais plutôt une *migration* du pouvoir expressif : la congruence $\equiv$ utilisée par la déduction modulo représente certains axiomes que l'on préfère en fait écrire sous forme de règles de réécriture. Notons qu'il peut exister plusieurs façons de représenter une théorie à l'aide de règles de réécriture et inversement plusieurs façons de représenter un système de réécriture à l'aide d'axiomes. Si $\mathcal{R}$ est un ensemble de règles de termes et de prédicats, Dowek *et al.* (2003) proposent de qualifier les théories permettant de représenter $\mathcal{R}$ comme *compatibles avec* $\mathcal{R}$.

Définition 5.2.1 (Théories compatibles). *Une théorie $\mathcal{T}h$ est compatible avec un ensemble de règles de termes et de prédicats $\mathcal{R}$ dans un système déductif $\mathcal{A}$ si, $\equiv$ étant la congruence engendrée par $\mathcal{R}$,*

Correction *pour toutes formules φ et ψ telles que $\varphi \equiv \psi$, le séquent $\mathcal{T}h \vdash \varphi \Leftrightarrow \psi$ est démontrable dans $\mathcal{A}$;*

Complétude *pour toute formule φ de $\mathcal{T}h$, le séquent $\vdash \varphi$ est démontrable dans $\mathcal{A}$+ConvR$_{\equiv}$+ConvL$_{\equiv}$.*

Par abus de langage, la correction et la complétude du paradigme de la déduction modulo *pour un système déductif $\mathcal{A}$ donné* consiste en l'existence, pour tout système de termes et de prédicats $\mathcal{R}$, d'une théorie $\mathcal{T}h$ compatible avec $\mathcal{R}$ dans $\mathcal{A}$. Comme le remarque Burel (2009), la compatibilité d'une théorie avec un ensemble de règles dépend du système déductif que l'on considère.

Après avoir introduit la déduction modulo, Gilles Dowek, Thérèse Hardin et Claude Kirchner ont ensuite démontré qu'il existait toujours pour tout système $\mathcal{R}$ de règles de termes et de prédicats une théorie compatible avec $\mathcal{R}$. Si $\equiv$ est une congruence (par exemple la congruence engendrée par $\mathcal{R}$), la théorie $\mathcal{T}h_{\equiv}$ qu'elle représente est en fait l'ensemble de formules $\{\forall \bar{x}.(\varphi \Leftrightarrow \psi)/\varphi \equiv \psi\}$. La propriété suivante le démontre formellement dans le cas de la déduction naturelle et du calcul des séquents.

Propriété 5.2.1 (Correction et complétude de la déduction modulo (Dowek *et al.*, 2003)). *Il existe une dérivation de $\Gamma \vdash \Delta$ respectivement dans NJ$_{\equiv}$, NK$_{\equiv}$, LJ$_{\equiv}$ ou LK$_{\equiv}$ si et seulement si il existe une dérivation de $\mathcal{T}h_{\equiv}, \Gamma \vdash \Delta$ dans NJ, NK, LJ ou LK.*

Bien entendu et puisque $\mathcal{T}h_{\equiv}$ est un ensemble infini de formules, lorsque nous écrivons « il existe une dérivation de $\mathcal{T}h_{\equiv}, \Gamma \vdash \Delta$ », il faut lire « il existe un sous-ensemble fini $A \subset \mathcal{T}h_{\equiv}$ tel qu'il existe une dérivation de $A, \Gamma \vdash \Delta$ ».

La théorie $\mathcal{T}h_{\equiv}$ peut être définie à partir de moins d'axiomes, en particulier puisque $\equiv$ est définie à partir de règles de termes et de prédicats. Le cas des règles de prédicats est assez simple : on associe à chaque règle $P \longrightarrow \varphi$ l'axiome $\forall \bar{x}.(P \Leftrightarrow \varphi)$ où $\bar{x}$ représente l'ensemble des variables libres de P (rappelons que $\mathcal{FV}(\varphi) \subseteq \mathcal{FV}(P)$ par la définition 1.1.24 de règle de réécriture). Le traitement des règles de termes est un peu plus problématique. L'intuition est de représenter chaque règle $t \longrightarrow u$ par l'axiome $\forall \bar{x}.(t = u)$ où $\bar{x}$ représente l'ensemble des variables de t (encore une fois $\mathcal{V}(u) \subseteq \mathcal{V}(t)$) et $=$ représente *un symbole d'égalité*. Reste à définir ce symbole d'égalité. C'est là que réside la difficulté. Une solution est d'utiliser l'axiome de Leibniz

$$\forall x.\forall y.(x = y \Leftrightarrow (\forall P.P(x) \Rightarrow P(y))) .$$

Le fait que le symbole $=$ définisse une congruence sur les termes est alors dérivable dans le système déductif. La solution de Leibniz nous est néanmoins proscrite car elle utilise une *quantification d'ordre supérieur*, c'est-à-dire une quantification sur

les formules « $\forall P$ ». On peut remplacer cet axiome par les axiomes

$$\begin{array}{ll}
\text{reflexivité} & \forall x.\ \ x = x \\
\text{transitivité} & \forall x.\forall y.\forall z.\ \ x = y \Rightarrow y = z \Rightarrow x = z \\
\text{symétrie} & \forall x.\forall y.\ \ x = y \Rightarrow y = x
\end{array}$$

congruence pour tout symbole de fonction f d'arité n

$$\forall x_1 \dots x_n.\forall y_1 \dots y_n.\ \ x_1 = y_1 \Rightarrow \dots \Rightarrow x_n = y_n \Rightarrow f(x_1 \dots x_n) = f(y_1 \dots y_n)$$

congruence pour tout symbole de prédicat P d'arité n

$$\forall x_1 \dots x_n.\forall y_1 \dots y_n.\ \ x_1 = y_1 \Rightarrow \dots \Rightarrow x_n = y_n \Rightarrow P(x_1 \dots x_n) \Leftrightarrow P(y_1 \dots y_n)$$

Remarquons que nous avons là un nombre d'axiome proportionnel au nombre de symboles de fonction de l'algèbre de termes et au nombre de symboles de prédicats sous-jacents. En particulier le nombre d'axiomes est infini si le nombre de symboles de fonction ou le nombre de symboles de prédicats est infini. En plus de tous ces axiomes, la théorie associée à un ensemble de règles de termes et de prédicats contient donc

$$\begin{array}{lll}
 & \forall \bar{x}.(t = u) & \text{pour toute règle de termes } t \to u \\
\text{et} & \forall \bar{x}.(P \Leftrightarrow \varphi) & \text{pour toute règle de prédicats } P \to \varphi\ .
\end{array}$$

Si $\mathcal{R}$ est l'ensemble de règles de termes et de prédicats, nous noterons $\mathcal{T}h_{\mathcal{R}}$ cette théorie. Lorsque $\mathcal{R}$ ne contient que des règles de prédicats, nous enleverons les axiomes concernant l'égalité de $\mathcal{T}h_{\mathcal{R}}$.

Propriété 5.2.2 (Correction et complétude de la déduction modulo, version 2). *Soit $\equiv$ la congruence engendrée par un ensemble $\mathcal{R}$ de règles de termes et de prédicats. Il existe une dérivation de $\Gamma \vdash \Delta$ respectivement dans $N\mathcal{J}_{\equiv}$, $NK_{\equiv}$, $L\mathcal{J}_{\equiv}$ ou $LK_{\equiv}$ si et seulement si il existe une dérivation de $\mathcal{T}h_{\mathcal{R}}, \Gamma \vdash \Delta$ dans $N\mathcal{J}$, NK, $L\mathcal{J}$ ou LK.*

5.3 Élimination des coupures en déduction modulo

Nous avons vu au chapitre 2 que, dans les systèmes de déduction naturelle ou de calcul des séquents, les coupures peuvent toujours être éliminées : nous avons en effet écrit des procédures terminantes qui éliminent les coupures. Par abus de langage et par parenté avec la propriété de *normalisation forte* des systèmes de réécriture, nous appellerons cette propriété des systèmes déductifs la *normalisation*. Une propriété un peu plus faible mais néanmoins intéressante est l'admissibilité des coupures, vérifiée lorsque, s'il existe une dérivation d'un certain séquent, alors il en existe toujours une sans coupure. La normalisation implique l'admissibilité des coupures, puisqu'elle induit un processus permettant de construire une dérivation sans coupure. L'admissibilité des coupures implique à son tour la cohérence (propriété 2.2.5 pour NJ, propriété 2.3.6 pour LJ et LK). Ces deux implications sont aussi

vérifiées dans le cadre de la déduction modulo, du moins pour les congruences satisfaisant la propriété de protection des connecteurs. En revanche, si la normalisation, l'admissibilité des coupures et la cohérence sont vérifiées dans le cadre des systèmes de déduction naturelle et de calcul des séquents, ce n'est pas forcément le cas en déduction modulo. Tout d'abord la cohérence d'un système de déduction modulo est équivalent à la cohérence des théories qui lui sont compatibles. Il est par exemple très simple de noter que si l'on utilise la règle de prédicats $P \longrightarrow \neg P$, le système $\mathrm{LK}_{\equiv}$ obtenu est inconsistant : à partir de la preuve du tiers exclu $P \vee \neg P$ (dans $\mathrm{LK} \subseteq \mathrm{LK}_{\equiv}$), on obtient facilement une preuve de P ainsi qu'une preuve de $\neg P$, puis finalement une preuve de $\bot$. Il est donc important de savoir quelles propriétés des systèmes de règles de termes et de prédicats permettent d'assurer la normalisation, l'admissibilité des coupures et la cohérence du système. Il peut s'avérer difficile d'avoir des critères syntaxiques à la fois corrects et complets. Par conséquent on s'attache à rechercher les critères corrects les plus larges qu'on puisse trouver.

La sous-section 5.3.1 contient des contre-exemples à la cohérence, à l'admissibilité des coupures ainsi qu'à la normalisation de systèmes de déduction modulo. La sous-section 5.3.2 décrit des méthodes permettant de démontrer ces propriétés lorsque certains critères sont vérifiés par le système de règles de termes et de prédicats considéré. Ces deux sous-sections regroupent des résultats obtenus par Dowek et Werner (2003); Hermant (2005).

5.3.1 Contre-exemples

Le premier exemple que je vais présenter est présenté par Burel (2009). Ce contre-exemple est une adaptation d'un contre-exemple de Crabbé démontrant la non-normalisation de la déduction naturelle lorsqu'on lui rajoute des règles d'inférences correspondant au schéma de compréhension restreinte. Afin d'introduire la définition de l'ensemble $\{x \in A / \varphi(x)\}$, on peut utiliser les règles d'inférence

$$\in\textsc{Intro}\ \frac{\Gamma \vdash t \in A \qquad \Gamma \vdash \varphi(t)}{\Gamma \vdash t \in \{x \in A / \varphi(x)\}}$$

$$\in\textsc{Elim1}\ \frac{\Gamma \vdash t \in \{t \in A / \varphi(t)\}}{\Gamma \vdash \varphi(t)} \qquad \in\textsc{Elim2}\ \frac{\Gamma \vdash t \in \{t \in A / \varphi(t)\}}{\Gamma \vdash t \in A}$$

Notons r_a le terme $\{x \in a / x \notin x\}$. En utilisant ces règles, la dérivation π suivante démontre $r_a \in a \vdash r_a \notin r_a$.

$$\dfrac{\dfrac{\dfrac{}{r_a \in r_a, r_a \in a \vdash r_a \in r_a}\,\textsc{Axiom}}{r_a \in r_a, r_a \in a \vdash r_a \notin r_a}\,\in\textsc{Elim1} \qquad \dfrac{}{r_a \in r_a, r_a \in a \vdash r_a \in r_a}\,\textsc{Axiom}}{\dfrac{r_a \in r_a, r_a \in a \vdash \bot}{r_a \in a \vdash r_a \notin r_a}\,\neg\textsc{Intro}}\,\neg\textsc{Elim}$$

Alors la dérivation π' suivante démontre $\vdash r_a \notin a$.

$$\neg\textsc{Elim}\ \dfrac{\overset{\pi}{r_a \in a \vdash r_a \notin r_a} \qquad \in\textsc{Intro}\ \dfrac{\textsc{Axiom}\ \dfrac{}{r_a \in a \vdash r_a \in a} \qquad \overset{\pi}{r_a \in a \vdash r_a \notin r_a}}{r_a \in a \vdash r_a \in r_a}}{\neg\textsc{Intro}\ \dfrac{r_a \in a \vdash \bot}{\vdash r_a \notin a}}$$

Notons que l'ultime étape de π est une instance de $\neg$Intro. Cette dérivation étant directement utilisée par une instance de $\neg$Elim dans π', nous avons ici une coupure dont la réduction renvoie la dérivation π'' suivante.

$$\neg\textsc{Elim}\ \dfrac{\in\textsc{Elim1}\ \dfrac{\in\textsc{Intro}\ \dfrac{\textsc{Axiom}\ \dfrac{}{r_a \in a \vdash r_a \in a} \qquad \overset{\pi}{r_a \in a \vdash r_a \notin r_a}}{r_a \in a \vdash r_a \in r_a} \qquad \vdots}{r_a \in a \vdash r_a \notin r_a}}{\neg\textsc{Intro}\ \dfrac{r_a \in a \vdash \bot}{\vdash r_a \notin a}} \qquad \in\textsc{Intro}\ \dfrac{\textsc{Axiom}\ \dfrac{}{r_a \in a \vdash r_a \in a} \qquad \overset{\pi}{r_a \in a \vdash r_a \notin r_a}}{r_a \in a \vdash r_a \in r_a}$$

Dans cette dernière preuve π'' se trouve une instance de la coupure

$$\in\textsc{Elim1}\ \dfrac{\in\textsc{Intro}\ \dfrac{\overset{\pi_1}{\Gamma \vdash t \in A} \qquad \overset{\pi_2}{\Gamma \vdash \varphi(t)}}{\Gamma \vdash t \in \{x \in A / \varphi(t)\}}}{\Gamma \vdash \varphi(t)}$$

qui se réduit naturellement en π_2 et l'on obtient alors une réduction de π'' vers π'. Pour résumer, π' se réduit en un pas en π'' par la réduction d'une coupure $\neg$Intro/Elim, et π'' se réduit en π' par la réduction d'une coupure $\in$Intro/Elim.

Mis à part les règles de NJ, ce contre-exemple ne se sert que des instances de $\in$Elim1 et $\in$Intro pour r_a, c'est-à-dire

$$\in\textsc{Intro}\ \dfrac{\Gamma \vdash r_a \in a \qquad \Gamma \vdash r_a \notin r_a}{\Gamma \vdash r_a \in r_a} \qquad\qquad \in\textsc{Elim1}\ \dfrac{\Gamma \vdash r_a \in r_a}{\Gamma \vdash r_a \notin r_a}$$

Pour obtenir ces règles en déduction modulo, on peut utiliser tout simplement la règle de prédicats $r_a \in r_a \to r_a \in a \wedge r_a \notin r_a$. En fait et comme présenté par Dowek et Werner (2003), on peut même remplacer $r_a \in r_a$ et $r_a \in a$ par des prédicats d'arité 0 que nous noterons respectivement A et B. La règle de réécriture

devient $A \longrightarrow B \wedge \neg A$ et π' devient

$$\neg\textsc{Elim}\ \dfrac{\dfrac{\pi}{B \vdash \neg A} \quad \in\textsc{Intro}\ \dfrac{\textsc{Axiom}\ \dfrac{}{B \vdash B}}{B \vdash A}}{\neg\textsc{Intro}\ \dfrac{B \vdash \bot}{\vdash \neg B}} \qquad \overbrace{\neg\textsc{Elim}\ \dfrac{\in\textsc{Elim1}\ \dfrac{\textsc{Axiom}\ \dfrac{}{A, B \vdash A}}{A, B \vdash \neg A} \quad \textsc{Axiom}\ \dfrac{}{A, B \vdash A}}{B \vdash \neg A}}^{\pi}$$

et en deux étapes de réduction, se réduit en lui-même. Cette même dérivation ainsi que la réduction qui lui est associée peut d'ailleurs être traduite en calcul des séquents (classique ou intuitionniste). On perd alors, en plus de la normalisation, la propriété d'admissibilité des coupures.

Notons que l'exemple de Crabbé provient de la même intuition mathématique que le paradoxe de Russell. Des contre-exemples peuvent être imaginés à partir de l'autre côté du miroir : le côté calcul. Considérons par exemple le terme $\Omega = (\lambda x.x\,x)\,(\lambda x.x\,x)$ de l'exemple 1.2.2. Il n'est pas typable dans le lambda-calcul simplement typé car x ne peut avoir en même temps un type A et le type $A \Rightarrow A$. Lorsque l'on travaille en déduction modulo, on peut choisir de considérer la règle de prédicats $A \longrightarrow A \Rightarrow A$ qui identifie ces deux types. Ainsi Ω prend le type A.

$$\Rightarrow\textsc{Elim}\ \dfrac{\overbrace{\Rightarrow\textsc{Intro}\ \dfrac{\Rightarrow\textsc{Elim}\ \dfrac{\textsc{Axiom}\ \dfrac{}{A \vdash A \Rightarrow A}\ {\scriptstyle A \equiv A \Rightarrow A} \qquad \textsc{Axiom}\ \dfrac{}{A \vdash A}}{A \vdash A}}{\vdash A \Rightarrow A}}^{\omega} \qquad \omega\ {\scriptstyle (A \equiv A \Rightarrow A)}\ \vdash A}{\vdash A}$$

Cette dérivation contient une seule coupure et se réduit en elle-même. C'est donc un deuxième contre-exemple à la normalisation en déduction modulo.

Le problème qui se pose alors naturellement consiste à caractériser les systèmes de règles de termes et de prédicats qui induisent la normalisation et l'admissibilité des coupures en déduction modulo. Les deux contre-exemples que nous venons de décrire, c'est-à-dire la règle $A \longrightarrow B \wedge \neg A$ puis la règle $A \longrightarrow A \Rightarrow A$, contiennent des règles de prédicats. Nous avons déjà expliqué que celles-ci pouvaient interférer beaucoup plus fortement avec le système d'inférences. Il est donc légitime de se concentrer tout d'abord sur les systèmes de réécriture ne contenant que des règles de termes : pour ceux-ci, la normalisation est toujours vérifiée, comme l'ont démontré Dowek et Werner (2003). Bien évidemment, des règles de prédicats, même satisfaisant la propriété de protection des connecteurs 5.1.1, peuvent menacer la normalisation du système de déduction modulo : c'est le cas des deux contre-exemples

que nous venons de décrire. Remarquons que ces deux contre-exemples définissent des systèmes de réécriture qui ne terminent pas. On peut donc penser que la terminaison ou encore la convergence du système de réécriture permet d'assurer la normalisation du système de déduction modulo associé. Ce n'est malheureusement pas le cas : Dowek et Werner (2003) décrivent un système de règles de termes et de prédicats convergent et induisant un système de déduction modulo ne vérifiant pas la propriété de normalisation. Ce système est le suivant.

$$\begin{aligned}\mathcal{R}_1 &: R \in R \to \forall y.(y \simeq R \Rightarrow (R \in y \Rightarrow \bot))\\ \mathcal{R}_2 &: y \simeq z \to \forall y.(x \in y \Rightarrow z \in y)\end{aligned}$$

Notons $\equiv_{\mathcal{R}}$ la congruence engendrée par $\mathcal{R}_1$ et $\mathcal{R}_2$. Pour cette congruence, des preuves de $\vdash \bot$ en déduction naturelle modulo puis en calcul des séquents modulo sont décrites respectivement en figures 5.5 et 5.6. Remarquons aussi que la formule $\bot$ ne se réécrit en aucune autre formule par $\mathcal{R}_1$ ou par $\mathcal{R}_2$, et qu'aucune formule ne se réécrit en $\bot$ par ces mêmes règles. Par conséquent, si $\varphi \equiv_{\mathcal{R}} \bot$, alors $\varphi = \bot$. La même analyse que celle de la démonstration de la propriété 2.2.5 démontre alors qu'il n'existe pas de démonstration sans coupure de $\vdash \bot$ dans $\mathrm{NJ}_{\equiv_{\mathcal{R}}}$. De même, une analyse similaire à celle de la démonstration de la propriété 2.3.6 démontre alors qu'il n'existe pas de démonstration sans coupure de $\vdash \bot$ dans $\mathrm{LK}_{\equiv_{\mathcal{R}}}$. Les preuves des figures 5.5 et 5.6 ne normalisent donc pas.

Pour résumer, nous venons de décrire plusieurs contre-exemples à la normalisation. Notons que seul le dernier traduit une théorie inconsistante. Les autres, bien que ne satisfaisant pas la propriété de normalisation, induisent des systèmes de déduction modulo consistants. On peut donc casser la normalisation sans pour autant casser la cohérence. Plus précisément, Hermant (2005) a démontré que l'on pouvait perdre l'admissibilité de la coupure en restant consistant, et que l'on pouvait perdre la normalisation en conservant l'admissibilité des coupures. Il produit d'ailleurs dans les deux cas des contre-exemples convergents. Ces contre-exemples sont résumés en figure 5.7.

5.3.2 Preuves d'élimination des coupures

Nous allons à présent décrire trois méthodes efficaces permettant d'obtenir la normalisation ou l'admissibilité des coupures en déduction modulo. La première, proposée par Gilles Dowek et Benjamin Werner, se base sur les candidats de réducibilité. Cela amène les amène à proposer les notions de prémodèle et de superconsistance (Dowek et Werner, 2003; Dowek, 2006; Dowek et Hermant, 2007). La deuxième approche, proposée par Olivier Hermant, se base sur la notion de modèles et sur des preuves de complétude forte de la déduction modulo (Hermant, 2005; Bonichon et Hermant, 2006a,b). Enfin la dernière, proposée par Guillaume Burel et Claude Kirchner, définit une méthode de complétion d'un système de règle de termes et de prédicats, l'objectif étant d'obtenir finalement un système de réécriture correspondant à un système de déduction modulo vérifiant l'admissibilité des coupures (Burel, 2009; Burel et Kirchner, 2007, 2009).

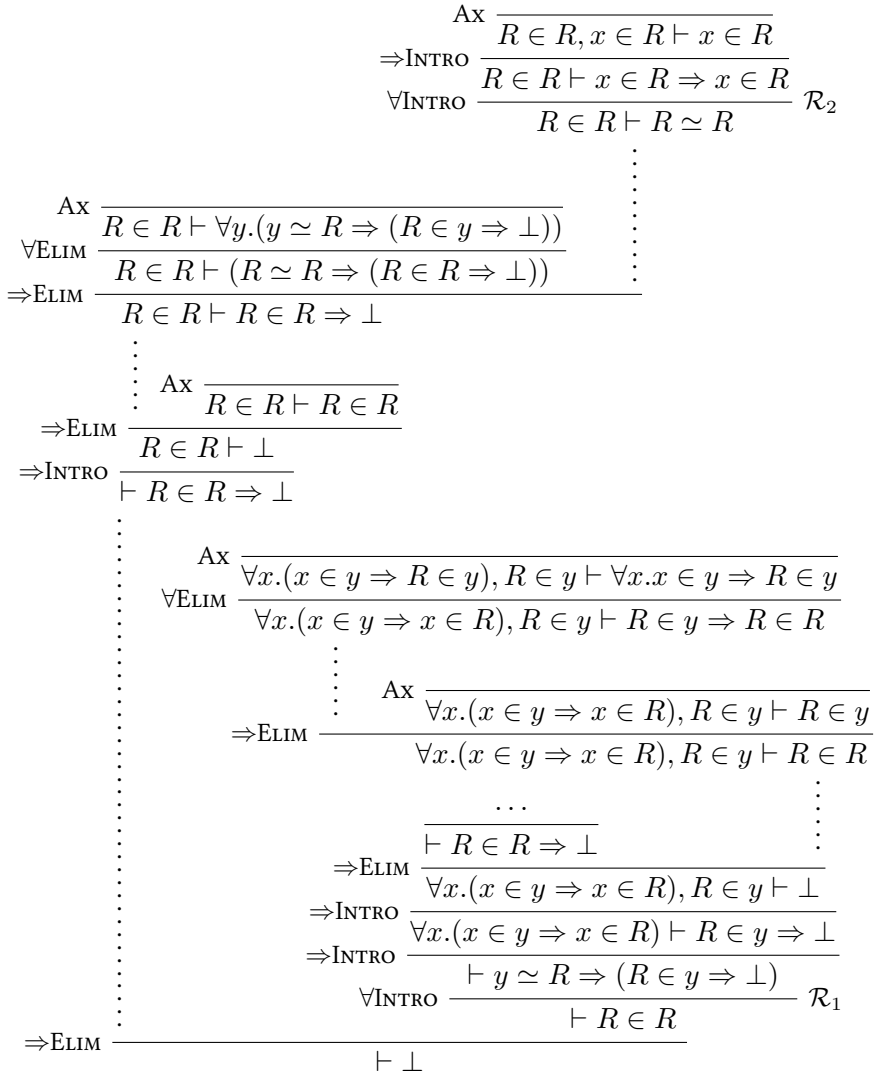


FIG. 5.5 – Incohérence de $\mathrm{NJ}_{\equiv_{\mathcal{R}}}$

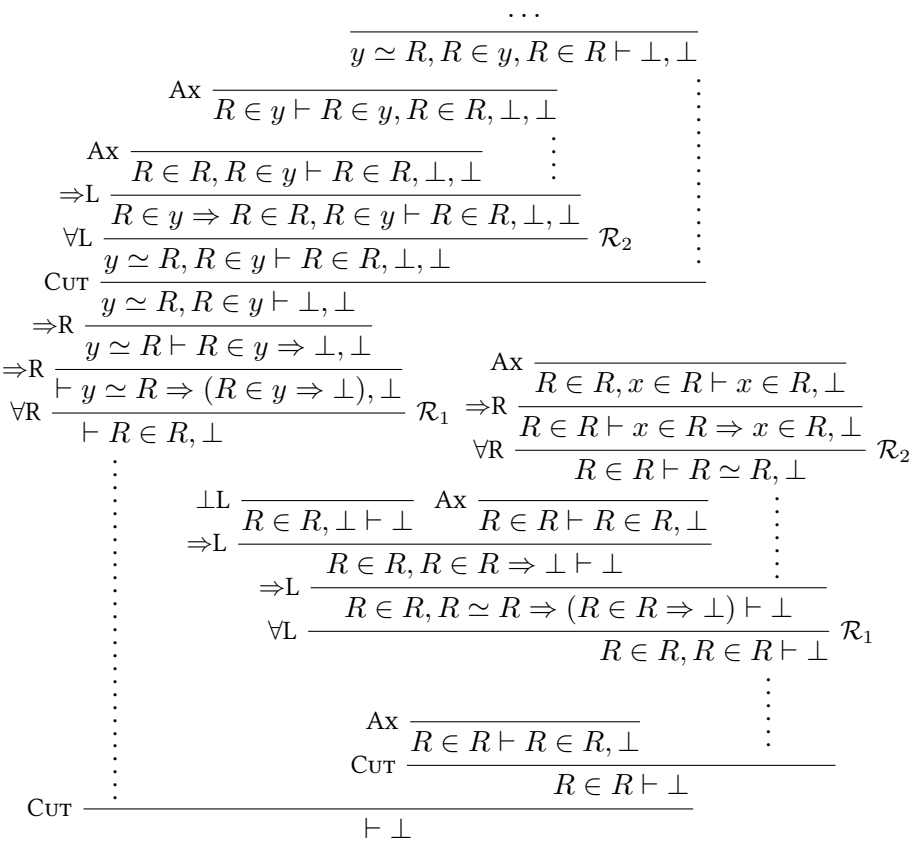


FIG. 5.6 – Incohérence de $\mathrm{LK}_{\equiv_{\mathcal{R}}}$

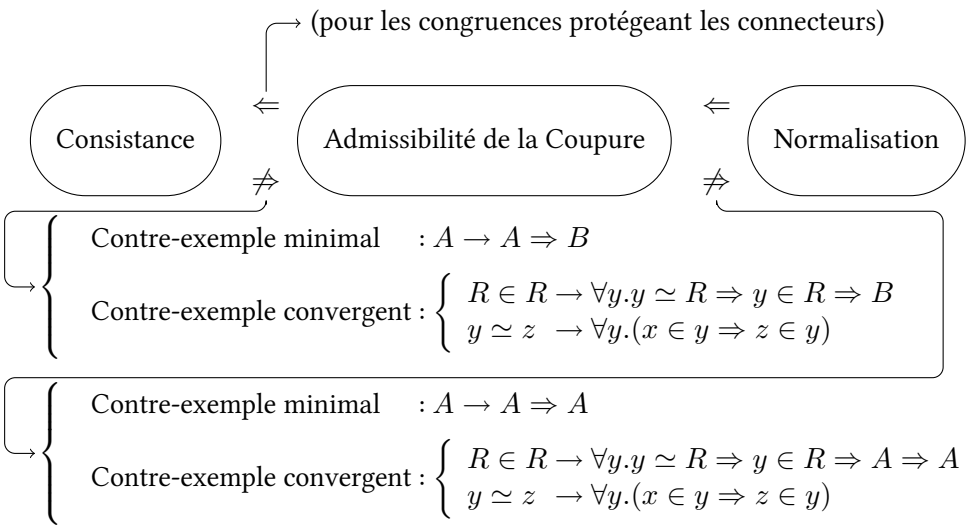


FIG. 5.7 – Contre-exemples de Hermant (2005) à la normalisation en déduction modulo

Prémodèles et pré-algèbres de Heyting

Afin d'étudier la normalisation de $\mathrm{NJ}_{\equiv}$, Gilles Dowek et Benjamin Werner proposent de représenter les dérivations sous forme de lambda-termes étendus comme nous l'avons fait en sous-section 3.1.3, mais cette fois pour la déduction naturelle modulo. Rappelons que l'ensemble des lambda-termes étendus est défini comme le langage hors-contexte associé à la grammaire en forme de Backus-Naur suivante.

$$\begin{aligned}\pi, \rho, \tau \quad ::= \quad & x \mid \lambda x.\pi \mid \pi\,\rho \mid \delta_{\bot}(\pi) \mid \lambda^{\neg} x.\pi \mid \delta_{\neg}(\pi, \rho) \\ & \mid (\pi, \rho) \mid \mathsf{fst}(\pi) \mid \mathsf{snd}(\pi) \mid i(\pi) \mid j(\pi) \mid \delta(\pi, x.\rho, y.\tau) \\ & \mid \lambda \mathsf{x}.\pi \mid \pi\,\mathsf{t} \mid \langle \mathsf{t}, \pi \rangle \mid \delta_{\exists}(\pi, x.\mathsf{x}.\rho)\end{aligned}$$

Les règles de typage sont décrites par la figure 5.8. Ce typage est dirigé par la syntaxe. Nous dirons qu'un lambda-terme est une introduction si la règle qui le type correspond à une règle d'introduction de $\mathrm{NJ}_{\equiv}$; nous dirons que c'est une élimination si la règle qui le type correspond à une règle d'élimination de $\mathrm{NJ}_{\equiv}$. Enfin nous dirons qu'un lambda-terme étendu est neutre si c'est une élimination ou une variable. On rajoute à l'alpha-conversion et à la bêta-réduction les règles de conversion suivantes (modifiées par rapport aux règles de la sous-section 3.1.3).

$$\begin{array}{rcl@{\qquad}rcl} (\lambda x.\pi)\,\rho & \to & \pi[\rho/x] & \delta(i(\pi), x.\rho, y.\tau) & \to & \rho[\pi/x] \\ \delta_{\neg}(\lambda^{\neg} x.\pi, \rho) & \to & \pi[\rho/x] & \delta(j(\pi), x.\rho, y.\tau) & \to & \tau[\pi/y] \\ \mathsf{fst}((\pi, \rho)) & \to & \pi & (\lambda \mathsf{x}.\pi)\,\mathsf{t} & \to & \pi[\mathsf{t}/\mathsf{x}] \\ \mathsf{snd}((\pi, \rho)) & \to & \rho & \delta_{\exists}(\langle \mathsf{t}, \pi \rangle, x.\mathsf{x}.\rho) & \to & (\rho[\mathsf{t}/\mathsf{x}])[\pi/x] \end{array}$$

L'objectif est donc de démontrer la normalisation forte de cette réduction sur tous les lambda-termes étendus bien typés. Pour cela, un simple raisonnement récursif

$$\textsc{Var}\ \frac{}{\Gamma, x : \varphi \vdash x : \psi}\ \varphi \equiv \psi \qquad \delta_\bot()\ \frac{\Gamma \vdash \pi : \varphi}{\Gamma \vdash \delta_\bot(\pi) : \psi}\ \varphi \equiv \bot$$

$$\textsc{Abs}\ \frac{\Gamma, x : \psi_1 \vdash \pi : \psi_2}{\Gamma \vdash \lambda x.\pi : \varphi}\ \varphi \equiv \psi_1 \Rightarrow \psi_2$$

$$\textsc{App}\ \frac{\Gamma \vdash \pi : \psi_1 \Rightarrow \psi_2 \qquad \Gamma \vdash \rho : \psi_1}{\Gamma \vdash \pi\,\rho : \varphi}\ \varphi \equiv \psi_1 \Rightarrow \psi_2$$

$$(,)\ \frac{\Gamma \vdash \pi : \psi_1 \qquad \Gamma \vdash \rho : \psi_2}{\Gamma \vdash (\pi, \rho) : \varphi}\ \varphi \equiv \psi_1 \wedge \psi_2$$

$$\mathsf{fst}()\ \frac{\Gamma \vdash \pi : \varphi}{\Gamma \vdash \mathsf{fst}(\pi) : \psi_1}\ \varphi \equiv \psi_1 \wedge \psi_2 \qquad \mathsf{snd}()\ \frac{\Gamma \vdash \pi : \varphi}{\Gamma \vdash \mathsf{snd}(\pi) : \psi_2}\ \varphi \equiv \psi_1 \wedge \psi_2$$

$$i()\ \frac{\Gamma \vdash \pi : \psi_1}{\Gamma \vdash i(\pi) : \varphi}\ \varphi \equiv \psi_1 \vee \psi_2 \qquad j()\ \frac{\Gamma \vdash \pi : \psi_2}{\Gamma \vdash j(\pi) : \varphi}\ \varphi \equiv \psi_1 \vee \psi_2$$

$$\delta(,,)\ \frac{\Gamma \vdash \pi : \varphi \qquad \Gamma, x : \psi_1 \vdash \rho : \psi' \qquad \Gamma, y : \psi_2 \vdash \tau : \psi'}{\Gamma \vdash \delta(\pi, x.\rho, y.\tau) : \psi'}\ \varphi \equiv \psi_1 \vee \psi_2$$

$$\lambda^\neg\ \frac{\Gamma, x : \psi \vdash \pi : \bot}{\Gamma \vdash \lambda^\neg x.\pi : \varphi}\ \varphi \equiv \neg\psi \qquad \delta_\neg(,)\ \frac{\Gamma \vdash \pi : \varphi \qquad \Gamma \vdash \rho : \psi}{\Gamma \vdash \delta_\neg(\pi, \rho) : \bot}\ \varphi \equiv \neg\psi$$

$$\textsc{Abs}'\ \frac{\Gamma \vdash \pi : \psi}{\Gamma \vdash \lambda \mathsf{x}.\pi : \forall \mathsf{x}.\psi}\ \begin{cases} \mathsf{x} \notin \mathcal{FV}(\Gamma) \\ \varphi \equiv \forall \mathsf{x}.\psi \end{cases} \qquad \textsc{App}'\ \frac{\Gamma \vdash \pi : \varphi}{\Gamma \vdash \pi\,\mathsf{t} : \psi[\mathsf{t}/\mathsf{x}]}\ \varphi \equiv \forall \mathsf{x}.\psi$$

$$\langle,\rangle\ \frac{\Gamma \vdash \pi : \psi[\mathsf{t}/\mathsf{x}]}{\Gamma \vdash \langle \mathsf{t}, \pi \rangle : \varphi}\ \varphi \equiv \exists \mathsf{x}.\psi$$

$$\delta_\exists(,)\ \frac{\Gamma \vdash \pi : \varphi \qquad \Gamma, x : \psi \vdash \rho : \psi'}{\Gamma \vdash \delta_\exists(\pi, x.\mathsf{x}.\rho) : \psi'}\ \begin{cases} \mathsf{x} \notin \mathcal{FV}(\Gamma, \psi) \\ \varphi \equiv \exists \mathsf{x}.\psi \end{cases}$$

FIG. 5.8 – Typage du lambda-calcul étendu pour la déduction modulo

sur la structure des démonstrations n'est pas suffisant : il faut utiliser par exemple des candidats de réductibilité comme en sous-section 3.1.1. Plus précisément nous utilisons une généralisation de la définition 3.1.1.

Définition 5.3.1 (Candidats de réductibilité). *Un ensemble R de lambda-termes étendus est un candidat de réductibilité si*

- *si $\pi \in R$, alors π normalise fortement ;*
- *si $\pi \in R$ et $\pi \to \pi'$, alors $\pi' \in R$;*
- *si π est neutre et si pour tout π', $\pi \to \pi'$ implique $\pi' \in R$, alors $\pi \in R$.*

Nous noterons l'ensemble des candidats de réductibilité $\mathfrak{C}$.

Pour démontrer que tout lambda-terme étendu bien typé normalise fortement, l'idée est de construire un candidat de réductibilité R_φ pour toute formule φ, puis de montrer que si $\Gamma \vdash \pi : \varphi$ est dérivable dans $\mathrm{NJ}_\equiv$, alors $\pi \in R_\varphi$. Pour démontrer cela, les candidats de réductibilité doivent vérifier certaines propriétés de clôture, comme par exemple pour l'implication : *si $\pi \in R_{\varphi\Rightarrow\psi}$ se réduit en $\lambda x.\pi_1$, alors pour tout $\pi_2 \in R_\varphi$, $\pi_1[\pi_2/x] \in R_\psi$*. Dans ce cas et comme l'ont remarqué Dowek et Werner (2003); Dowek (2006), la construction des candidats de réductibilité ressemble alors fortement à la construction d'un modèle. Cette remarque amène alors la définition de prémodèle suivante.

Définition 5.3.2 (Prémodèle). *Un prémodèle est la donnée de* **1**. *un ensemble T ;* **2**. *pour chaque fonction f d'arité n de la signature α, une fonction $\hat{f} : T^n \to T$;* **3**. *pour chaque prédicat P d'arité n de la signature $\mathcal{A}$, une fonction $\hat{P} : T^n \to \mathfrak{C}$. Alors si $\mathtt{t}$ est un terme du premier ordre et ϑ est une fonction associant à chaque variable libre de $\mathtt{t}$ un élément de T, l'objet $|\mathtt{t}|_\vartheta$ est défini inductivement par $|\mathtt{x}|_\vartheta = \vartheta(\mathtt{x})$ et $|f(t_1,\dots,t_n)|_\vartheta = \hat{f}(|\mathtt{t}_1|_\vartheta,\dots,|\mathtt{t}_n|_\vartheta)$. Enfin si φ est une formule et ϑ est une fonction associant à chaque variable libre de φ un élément de T, l'ensemble de lambda-termes étendus $|\varphi|_\vartheta$ est défini inductivement comme suit.*

Proposition atomique *Si φ est une proposition atomique $P(\mathtt{t}_1,\dots,\mathtt{t}_n)$, alors $|\varphi|_\vartheta = \hat{P}(|\mathtt{t}_1|_\vartheta,\dots,|\mathtt{t}_n|_\vartheta)$.*

Implication *Si φ est $\psi_1 \Rightarrow \psi_2$, alors $|\varphi|_\vartheta$ est l'ensemble des lambda-termes étendus $\pi \in \mathcal{SN}$ tels que si $\pi \to^* \lambda x.\pi_1$ et si $\pi_2 \in |\psi_1|_\vartheta$, alors $\pi_1[\pi_2/x] \in |\psi_2|_\vartheta$.*

Conjonction *Si φ est $\psi_1 \wedge \psi_2$, alors $|\varphi|_\vartheta$ est l'ensemble des lambda-termes étendus π fortement normalisants tels que si $\pi \to^* (\pi_1,\pi_2)$, alors $\pi_1 \in |\psi_1|_\vartheta$ et $\pi_2 \in |\psi_2|_\vartheta$.*

Disjonction *Si φ est $\psi_1 \vee \psi_2$, alors $|\varphi|_\vartheta$ est l'ensemble des lambda-termes étendus π fortement normalisants tels que si $\pi \to^* i(\pi_1)$, alors $\pi_1 \in |\psi_1|_\vartheta$ et si $\pi \to^* j(\pi_2)$, alors $\pi_2 \in |\psi_2|_\vartheta$.*

Faux *Si φ est $\bot$, alors $|\varphi|_\vartheta$ est l'ensemble des lambda-termes étendus fortement normalisants.*

Négation *Si φ est $\neg\psi$, alors $|\varphi|_\vartheta$ est l'ensemble des lambda-termes π fortement normalisants tels que si $\pi \to^* \lambda^\neg x.\pi_1$ et si $\pi_2 \in |\psi|_\vartheta$, alors $\pi_1[\pi_2/x] \in |\bot|_\vartheta$.*

Quantification universelle *Si φ est $\forall \mathtt{x}.\psi$, alors $|\varphi|_\vartheta$ est l'ensemble des lambda-termes π fortement normalisants tels que si $\pi \to^* \lambda \mathtt{x}.\pi_1$, si $\mathtt{t}$ est un terme du premier ordre et si $e \in T$, alors $\pi_1[\mathtt{t}/\mathtt{x}] \in |\psi|_{\vartheta+(\mathtt{x},e)}$ ou $\vartheta + (\mathtt{x}, e)$ est la fonction égale à ϑ excepté pour $\mathtt{x}$ auquel elle associe e.*

Quantification existentielle *Si φ est $\exists \mathtt{x}.\psi$, alors $|\varphi|_\vartheta$ est l'ensemble des lambda-termes π fortement normalisants tels que si $\pi \to^* \langle \pi_1, \mathtt{t} \rangle$, alors il existe $e \in T$ tel que $\pi_1 \in |\psi|_{\vartheta+(\mathtt{x},e)}$.*

Pour définir un prémodèle, le point crucial est le choix de l'interprétation $\hat{P}$ pour chaque prédicat. Par exemple pour démontrer la normalisation de NJ, comme nous l'avons (partiellement) fait dans en sous-section 3.1.1, on peut choisir d'interpréter tout prédicat par l'ensemble des lambda-termes fortement normalisants. Après avoir démontré que pour toute formule φ et pour toute fonction $\vartheta : X \to T$, $|\varphi|_\vartheta$ est un candidat de réductibilité, on prouve alors que si π est de type φ, alors $\pi \in |\varphi|_\vartheta \subseteq \mathcal{SN}$ (pour tout ϑ). Cette approche n'est néanmoins plus valable dans le cas de $\mathrm{NJ}_\equiv$ car alors, si π est de type φ (c'est-à-dire π est une dérivation de $\Gamma \vdash \varphi$) et si $\varphi \equiv \psi$, alors π est aussi de type ψ. En particulier on ne peut plus démontrer dans tous les cas que si π est de type φ, alors $\pi \in |\varphi|_\vartheta$. Pour corriger cela, on doit supposer une condition supplémentaire : pour tous φ, ψ, ϑ, si $\varphi \equiv \psi$, alors $|\varphi|_\vartheta = |\psi|_\vartheta$. Dans ce cas, nous dirons que c'est un modèle de $\equiv$.

Définition 5.3.3. *Un prémodèle est un modèle de $\equiv$ si pour tous φ, ψ et ϑ tels que $\varphi \equiv \psi$ et tels que ϑ associe un élément de T à toute variable libre dans φ ou ψ, alors $|\varphi|_\vartheta = |\psi|_\vartheta$.*

Dowek et Werner (2003) ont prouvé le résultat suivant.

Propriété 5.3.1 (Théorème 3.1 puis corollaire 3.1 de (Dowek et Werner, 2003)). *S'il existe un prémodèle qui est un modèle de $\equiv$, alors toute dérivation de $\mathrm{NJ}_\equiv$ est fortement normalisante.*

Ils démontrent aussi que certains systèmes de règles de termes et de prédicats admettent un prémodèle comme la théorie des types simples, les systèmes de réécriture convergents et *quantifier free* (c'est-à-dire dont les membres droits des règles de réécriture ne contiennent aucun quantificateur) ou encore les systèmes de réécriture positifs (c'est-à-dire dont les membres droits des règles de réécriture ne contiennent aucune proposition atomique en position négative) et tels que toute proposition atomique admet au plus un seul réduit.

Gilles Dowek a remarqué que les prémodèles constituent en fait une classe particulière de pré-algèbres de Heyting. Ces dernières sont définis comme suit.

Définition 5.3.4 (Pré-algèbre de Heyting). *Soit $\mathcal{B}$ un ensemble, $\leqslant$ une relation sur $\mathcal{B}$, $\mathcal{A}$ et $\mathcal{E}$ des sous-ensembles de l'ensemble des parties de $\mathcal{B}$, $\mathfrak{f}$ et $\mathfrak{t}$ des éléments de $\mathcal{B}$ ainsi que des fonctions*

$$\mathfrak{imp}, \mathfrak{and}, \mathfrak{or} : \mathcal{B} \times \mathcal{B} \to \mathcal{B}\,, \qquad \mathfrak{A} : \mathcal{A} \to \mathcal{B} \quad \textit{et} \quad \mathfrak{E} : \mathcal{E} \to \mathcal{B}\,.$$

Alors la structure $(\mathcal{B}, \leqslant, \mathcal{A}, \mathcal{E}, \mathfrak{f}, \mathfrak{t}, \mathfrak{imp}, \mathfrak{and}, \mathfrak{or}, \mathfrak{A}, \mathfrak{E})$ *est une pré-algèbre de Heyting si*

- $\leqslant$ *est une préordre, c'est-à-dire est réflexif et transitif sur* $\mathcal{B}$ *;*
- $\mathfrak{t}$ *est un plus grand élément, c'est-à-dire pour tout* $a \in \mathcal{B}$, $a \leqslant \mathfrak{t}$ *;*
- $\mathfrak{or}(a, b)$ *est une borne supérieure de* a *et* b *et* $\mathfrak{and}(a, b)$ *est une borne inférieure de* a *et* b *;*
- *si* $a \in \mathcal{B}$ *et* $A \in \mathcal{A}$, *alors* $\mathfrak{imp}(a, A) \in \mathcal{A}$ *;*
- *si* $a \in \mathcal{B}$ *et* $E \in \mathcal{E}$, *alors* $\mathfrak{imp}(E, a) \in \mathcal{A}$ *;*
- $\mathfrak{A}$ *est une borne inférieure infinie, c'est-à-dire si* $a \in A \in \mathcal{A}$, *alors* $\mathfrak{A}(A) \leqslant a$ *et si pour tout* $a \in A$, $b \leqslant a$, *alors* $b \leqslant \mathfrak{A}(A)$ *;*
- $\mathfrak{E}$ *est une borne supérieure infinie, c'est-à-dire si* $a \in E \in \mathcal{E}$, *alors* $a \leqslant \mathfrak{E}(E)$ *et si pour tout* $a \in E$, $a \leqslant b$, *alors* $\mathfrak{E}(E) \leqslant b$.
- $a \leqslant \mathfrak{imp}(b, c)$ *si et seulement si* $\mathfrak{and}(a, b) \leqslant c$.

Nous dirons qu'une pré-algèbre de Heyting est pleine *si* $\mathcal{E}$ *et* $\mathcal{A}$ *sont tous les deux l'ensemble de toutes les parties de* $\mathcal{B}$. *Nous dirons qu'une pré-algèbre de Heyting est* ordonnée *si elle est munie d'un ordre* $\sqsubseteq$ *pour lequel* $\mathfrak{t}$ *et* $\mathfrak{f}$ *sont respectivement le plus grand et le plus petit l'élement, pour lequel les fonctions* $\mathfrak{or}$, $\mathfrak{and}$, $\mathfrak{E}$ *et* $\mathfrak{A}$ *sont croissantes, la fonction* $\mathfrak{imp}$ *est croissante en sa première variable et décroissante en sa deuxième, et contenant* $\leqslant$ *(ici, supposer que* $(\leqslant) \subseteq (\sqsubseteq)$ *est équivalent à supposer que* $\mathcal{B}^+ = \{x / \mathfrak{t} \leqslant x\}$ *est clos par le haut pour* $\sqsubseteq$*). Nous dirons qu'une pré-algèbre de Heyting est* complète *si elle est ordonnée par l'ordre* $\sqsubseteq$ *et si tout sous-ensemble de* $\mathcal{B}$ *admet une borne inférieure pour* $\sqsubseteq$. *Cela implique d'ailleurs que tout sous-ensemble de* $\mathcal{B}$ *a aussi une borne supérieure.*

Elles peuvent servir de support à la définition de modèles de la déduction modulo.

Définition 5.3.5 ($\mathcal{B}$-structure). *Si* $\mathcal{B}$ *est une pré-algèbre de Heyting, une* $\mathcal{B}$*-structure est la donnée d'un ensemble* T, *de fonctions* $\hat{f} : T^n \to T$ *pour chaque symbole de fonction d'arité* n *de* α *et de fonctions* $\hat{P} : T^n \to \mathcal{B}$ *pour chaque symbole de prédicat d'arité* n *de* $\mathcal{A}$. *Si* ϑ *est une fonction associant à chaque variable libre de* $\mathtt{t}$ *un élément de* T, *alors* $\|\mathtt{t}\|_\vartheta$ *est défini inductivement par* $\|\mathtt{x}\|_\vartheta = \vartheta(\mathtt{x})$ *et* $\|f(\mathtt{t}_1, \ldots, \mathtt{t}_n)\|_\vartheta = \hat{f}(\|\mathtt{t}_1\|_\vartheta, \ldots, \|\mathtt{t}_n\|_\vartheta)$. *Si* ϑ *est une fonction associant à chaque variable libre de* φ *un élément de* T, *alors* $\|\varphi\|_\vartheta$ *est défini inductivement en figure 5.9. Notons que* $\|\varphi\|_\vartheta$ *peut être non défini lorsque* φ *contient des quantificateurs. Néanmoins lorsque* $\mathcal{B}$ *est une pré-algèbre de Heyting pleine, alors* $\|\varphi\|_\vartheta$ *est toujours bien défini.*

Les $\mathcal{B}$-structures permettent alors de définir une notion de modèle comme suit.

Définition 5.3.6. *Une* $\mathcal{B}$*-structure est un modèle de la congruence* $\equiv$ *si, pour toutes formules* φ *et* ψ *telles que* $\varphi \equiv \psi$, *pour toute fonction* ϑ *associant à chaque variable libre de* φ *et de* ψ *un élément de* T, $\|\varphi\|_\vartheta$ *et* $\|\psi\|_\vartheta$ *sont effectivement définis et* $\|\varphi\|_\vartheta = \|\psi\|_\vartheta$.

Gilles Dowek remarque que les candidats de réducibilité forment une pré-algèbre de Heyting pleine, ordonnée (par l'inclusion des ensembles) et complète. En outre,

$$
\begin{aligned}
\|P(\mathtt{t}_1,\dots,\mathtt{t}_n)\|_\vartheta &= \hat{P}(\|\mathtt{t}_1\|_\vartheta,\dots,\|\mathtt{t}_n\|_\vartheta)\,;\\
\|\bot\|_\vartheta &= \mathfrak{f}\,;\\
\|\top\|_\vartheta &= \mathfrak{t}\,;\\
\|\psi_1 \wedge \psi_2\|_\vartheta &= \mathfrak{and}(\|\psi_1\|_\vartheta,\|\psi_2\|_\vartheta)\,;\\
\|\psi_1 \vee \psi_2\|_\vartheta &= \mathfrak{or}(\|\psi_1\|_\vartheta,\|\psi_2\|_\vartheta)\,;\\
\|\psi_1 \Rightarrow \psi_2\|_\vartheta &= \mathfrak{imp}(\|\psi_1\|_\vartheta,\|\psi_2\|_\vartheta)\,;\\
\|\forall x.\psi\|_\vartheta &= \mathfrak{A}(\{\|\psi\|_{\vartheta+(x,e)/e\in T}\})\,;\\
\|\exists x.\psi\|_\vartheta &= \mathfrak{E}(\{\|\psi\|_{\vartheta+(x,e)/e\in T}\})\,.
\end{aligned}
$$

FIG. 5.9 – Valuation dans une $\mathcal{B}$-structure

l'ensemble des $\mathfrak{C}$-structures concorde exactement avec l'ensemble des prémodèles. Cette remarque mène à proposer la notion suivante.

Définition 5.3.7 (Superconsistance). *Une congruence $\equiv$ est super-consistante si pour toute pré-algèbre de Heyting $\mathcal{B}$, il existe une $\mathcal{B}$-structure qui en est un modèle.*

Alors on obtient le corollaire de la propriété 5.3.1 suivant.

Propriété 5.3.2 (Dowek, 2006, théorème 3). *Si $\equiv$ est super-consistant, alors $N\mathcal{J}_\equiv$ est normalisant.*

Démonstration. Considérons l'ensemble des candidats de réductibilité $\mathfrak{C}$. Cet ensemble définit une pré-algèbre de Heyting pleine, ordonnée et complète. Puisque $\equiv$ est super-consistant, il existe une $\mathfrak{C}$-structure qui en est un modèle, c'est à dire un prémodèle qui est un modèle de $\equiv$. Alors la propriété 5.3.1 démontre la normalisation de $\mathrm{NJ}_\equiv$. □

Notons finalement que Dowek et Werner (2003) portent ce résultat au calcul des séquents intuitionniste modulo puis au calcul des séquents classique modulo. Pour le premier, ils étudient des coupures commutatives dans $\mathrm{NJ}_\equiv$ correspondant aux coupures commutatives du calcul des séquents. Leur résultat est le suivant.

Propriété 5.3.3 (Dowek et Werner, 2003, corollaire 4.1). *Tout comme dans le cas de $N\mathcal{J}_\equiv$, s'il existe un prémodèle qui est modèle de $\equiv$, alors l'élimination des coupures dans $L\mathcal{J}_\equiv$ normalise.*

Pour le calcul des séquents classique, ils utilisent une « *non non* » traduction définie comme suit.

Définition 5.3.8 (Non non traduction). *Si φ est une formule, alors la $\neg\neg$-traduction*

de φ, notée $(\varphi)^{\neg\neg}$, et la $\neg\neg$-traduction légère de φ, notée $(\varphi)^{l\neg\neg}$, sont définies par

$$
\begin{array}{lcllcl}
(P)^{\neg\neg} & = & \neg\neg P & (P)^{l\neg\neg} & = & P \\
(\varphi \wedge \psi)^{\neg\neg} & = & \neg\neg((\varphi)^{\neg\neg} \wedge (\psi)^{\neg\neg}) & (\varphi \wedge \psi)^{l\neg\neg} & = & (\varphi)^{\neg\neg} \wedge (\psi)^{\neg\neg} \\
(\varphi \vee \psi)^{\neg\neg} & = & \neg\neg((\varphi)^{\neg\neg} \vee (\psi)^{\neg\neg}) & (\varphi \vee \psi)^{l\neg\neg} & = & (\varphi)^{\neg\neg} \vee (\psi)^{\neg\neg} \\
(\varphi \Rightarrow \psi)^{\neg\neg} & = & \neg\neg((\varphi)^{\neg\neg} \Rightarrow (\psi)^{\neg\neg}) & (\varphi \Rightarrow \psi)^{l\neg\neg} & = & (\varphi)^{\neg\neg} \Rightarrow (\psi)^{\neg\neg} \\
(\bot)^{\neg\neg} & = & \neg\neg\bot & (\bot)^{l\neg\neg} & = & \bot \\
(\top)^{\neg\neg} & = & \neg\neg\top & (\top)^{l\neg\neg} & = & \top \\
(\forall x.\varphi)^{\neg\neg} & = & \neg\neg\forall x.(\varphi)^{\neg\neg} & (\forall x.\varphi)^{l\neg\neg} & = & \forall x.(\varphi)^{\neg\neg} \\
(\exists x.\varphi)^{\neg\neg} & = & \neg\neg\exists x.(\varphi)^{\neg\neg} & (\exists x.\varphi)^{l\neg\neg} & = & \exists x.(\varphi)^{\neg\neg}
\end{array}
$$

La non non traduction d'un ensemble $\mathcal{R}$ de règles de termes et de prédicats, notée $(\mathcal{R})^{l\neg\neg}$, est obtenue à partir de $\mathcal{R}$ en transformant toute règle de prédicat $P \longrightarrow \varphi$ en la règle $P \longrightarrow (\varphi)^{l\neg\neg}$.

Gilles Dowek et Benjamin Werner prouvent le résultat suivant concernant $\mathrm{LK}_{\equiv}$.

Propriété 5.3.4 (Dowek et Werner, 2003, théorème 4.1). *Si la congruence engendrée par $(\mathcal{R})^{l\neg\neg}$ est super-consistante, alors le calcul des séquents modulo associé à $\mathcal{R}$ normalise.*

Méthodes sémantiques

Nous allons à présent décrire une approche proposée par Olivier Hermant pour démontrer l'admissibilité des coupures en déduction modulo (Hermant, 2005; Bonichon et Hermant, 2006a,b). Il montre qu'il est possible de s'affranchir des candidats de réducibilité pour démontrer non pas la normalisation mais l'admissibilité des coupures en déduction modulo. En fonction de la logique que l'on considère (logique classique ou logique intuitionniste), on utilise la notion de modèle correspondante. Les systèmes déductifs (déduction naturelle ou calcul des séquents) sont alors corrects et complets par rapport à ces modèles (voir section 2.4). Pour chaque système déductif et pour la notion de modèle correspondante :

la correction exprime le fait que s'il existe une preuve du séquent $\Gamma \vdash \Delta$, alors tout modèle $\mathcal{M}$ le valide ;

la complétude exprime le fait inverse, c'est-à-dire si tout modèle $\mathcal{M}$ valide le séquent $\Gamma \vdash \Delta$, alors il en existe une dérivation.

Pour obtenir une preuve d'admissibilité des coupures, Hermant (2005) utilise une version de la complétude plus forte, qu'il appelle complétude forte. C'est la complétude du système sans coupure : si tout modèle $\mathcal{M}$ valide le séquent $\Gamma \vdash \Delta$, alors il en existe une preuve *sans coupure*. La correction associée à la complétude forte implique finalement l'admissibilité des coupures puisque le cas échéant, s'il existe une dérivation de $\Gamma \vdash \Delta$, par correction tout modèle valide ce séquent, puis par complétude forte il en existe une dérivation sans coupure.

Bien sûr, les définitions de correction, de complétude et de complétude forte d'un système de déduction modulo associé à une certaine congruence $\equiv$ diffèrent des

définitions pour NJ, NK, LJ ou LK : les modèles y sont restreints aux modèles validant la congruence $\equiv$. Tout comme les $\mathcal{B}$-structures, un modèle valide une congruence $\equiv$ si pour toutes formules φ et ψ telles que $\varphi \equiv \psi$, le modèle associe à φ et ψ une même dénotation.

La correction et la complétude des systèmes de déduction modulo sont acquises : elles reposent sur l'existence d'une théorie compatible $\mathcal{T}h$ pour toute congruence $\equiv$ et sur la correction des systèmes NJ, NK, LJ et NK. En revanche c'est la complétude forte qui pose problème. Elle constitue le point crucial d'une preuve d'admissibilité des coupures. Elle n'est pas vérifiée pour toute congruence, comme le démontrent les contre-exemples de la section 5.3.1. Néanmoins, Olivier Hermant a prouvé qu'un certain nombre de classes de systèmes de règles de termes et de prédicats la vérifient (Hermant, 2005; Bonichon et Hermant, 2006a,b). Pour ce faire, il raisonne par contraposée : en supposant qu'un certain séquent n'admette aucune dérivation sans coupure en déduction modulo, il construit un modèle de la congruence qui ne valide pas le séquent en question. Cette approche lui a permis d'obtenir d'intéressants critères d'admissibilité des coupures, tant dans le cadre classique qu'intuitionniste.

Élimination des coupures par complétion

Guillaume Burel et Claude Kirchner proposent une autre approche à l'élimination des coupures (Burel, 2009; Burel et Kirchner, 2007, 2009). Au lieu d'étudier des critères permettant d'établir qu'un système de règles de termes et de prédicats admet la propriété de normalisation ou d'admissibilité des coupures, ils proposent une méthode permettant de compléter un système de règles de termes et de prédicats ne satisfaisant pas l'admissibilité des coupures en rajoutant de nouvelles règles de prédicats. Le but est d'obtenir finalement un nouveau système de règles équivalent et correspondant à un système de déduction modulo admettant les coupures. Cette méthode de complétion se place dans le cadre des systèmes canoniques abstraits (Bonacina et Dershowitz, 2008; Dershowitz et Kirchner, 2006) qui, par exemple, contiennent la complétion à la Knuth et Bendix (1970).

Ils utilisent une variante du calcul des séquents classique modulo : l'ensemble $\mathcal{R}$ des règles de prédicats est divisé en deux ensembles $\mathcal{R}^+$ et $\mathcal{R}^-$ puis les règles de conversion ConvR et ConvL sont remplacées par les règles

$$\textsc{FoldR}\ \frac{\Gamma \vdash P\sigma, \varphi\sigma, \Delta}{\Gamma \vdash P\sigma, \Delta}\ P \to \varphi \in \mathcal{R}^+ \qquad \textsc{FoldL}\ \frac{\Gamma, P\sigma, \varphi\sigma \vdash \Delta}{\Gamma, P\sigma \vdash \Delta}\ P \to \varphi \in \mathcal{R}^-$$

En fait, le système obtenu est une restriction du calcul des séquents modulo polarisé (Dowek, 2002), la restriction appliquée par Guillaume Burel et Claude Kirchner consistant à remplacer les règles de conversion par des règles de *folding* à la Prawitz (1965).

Afin d'obtenir l'admissibilité des coupures du calcul des séquents obtenus ainsi à partir de $\mathcal{R}^+$ et $\mathcal{R}^-$, leur approche consiste à considérer ce qu'ils appellent des *dérivations critiques* (à l'instar des *paires critiques* des systèmes de réécriture de

termes). Une dérivation critique est une dérivation minimale qui n'est pas en forme normale qui ne contient strictement que des preuves en forme normale.

Idéalement, la complétion du système défini par les ensembles de règles $\mathcal{R}^+$ et $\mathcal{R}^-$ consiste à identifier les séquents associés aux racines des dérivations critiques puis à rajouter des règles de prédicats leur correspondant. Malheureusement, identifier ces séquents est indécidable pour la logique des prédicats (Burel et Kirchner, 2007, 2009). Pour contourner ce problème, on peut utiliser le fait que les dérivations critiques ont toujours la forme

$$\textsc{Cut}\ \dfrac{\textsc{FoldR}\ \dfrac{\overset{\pi}{\Gamma \vdash \varphi, P, \Delta}}{\Gamma \vdash P, \Delta} \qquad \textsc{FoldL}\ \dfrac{\overset{\pi'}{\Gamma, \psi, P \vdash \Delta}}{\Gamma, P \vdash \Delta}}{\Gamma \vdash \Delta}$$

où π et π' sont des preuves sans coupure où l'on ne peut pas leur retirer d'inférence inutile et où l'une des conversions $P \to \varphi$ et $P \to \psi$ n'est pas relative à une règle de termes. Ce critère caractérise une classe strictement plus grande que la classe des démonstrations critiques. Néanmoins cette classe est identifiable, en utilisant par exemple TaMeD sur les séquents $\vdash \varphi, P$ et $\psi, P \vdash$ et en rajoutant au fur et à mesure des formules dans les contextes Γ et Δ afin de bien obtenir finalement des dérivations (sans coupures) de $\Gamma \vdash \varphi, P, \Delta$ et $\Gamma, \psi, P \vdash \Delta$. L'itération de cette méthode permet finalement d'obtenir un système de règles complété induisant un système de déduction n'admettant plus de preuves critiques. Un tel système vérifie la propriété d'élimination des coupures.

5.4 Automatisation de la déduction modulo

Nous allons à présent présenter des méthodes de recherche de preuve adaptée à la déduction modulo, c'est-à-dire des procédures automatiques construisant permettant d'affirmer ou d'infirmer l'existence d'une dérivation d'un certain séquent en déduction modulo. L'approche de la déduction modulo ne remet aucunement en cause l'indécidabilité de la logique du premier ordre. Ces procédures ne pourront donc pas être à la fois complètes et terminantes. Ces procédures automatiques restent néanmoins l'une des raisons d'être de la déduction modulo : utiliser la réécriture pour modéliser certains axiomes calculatoires permet de les orienter et si le système de réécriture obtenu est convergent, on obtient alors un processus décidant la congruence engendrée de façon bien plus efficace que ce que pourrait faire une processus utilisant les axiomes non orientés. Un exemple très convainquant (Dowek *et al.*, 2003) consiste à considérer un axiome d'associativité

$$(a + b) + c = a + (b + c)\ .$$

Non orienté, cet axiome peut malheureusement générer des réarrangements infinis de parenthèses. Par contre si on l'oriente, on obtient la règle de réécriture

$$(a + b) + c \to a + (b + c)$$

Extended Resolution

$$\frac{\{P_1, \ldots, P_n, Q_1, \ldots, Q_m\}[C_1] \qquad \{\neg R_1, \ldots, \neg R_p, S_1, \ldots, S_q\}[C_2]}{\{Q_1, \ldots, Q_m, S_1, \ldots, S_q\}[\, C_1 \cup C_2 \cup \{P_1 = \cdots = P_n = R_1 = \cdots = R_p\}\,]}$$

Extended Narrowing

$$\frac{v[C] \qquad l \to r \in \mathcal{R} \qquad v_{|w} \text{ proposition atomique} \qquad v' \in \mathsf{clF}(v[r]_w)}{v'[C \cup \{v_{|w} = l\}]}$$

Fig. 5.10 – Règles de ENAR

induisant un système de réécriture convergent. L'une des motivations de la déduction modulo est donc d'éviter ce genre de désagréments à l'utilisateur de procédures de recherche de preuves.

Nous allons tout d'abord présenter ENAR (*Extended Narrowing And Resolution*), extension de la résolution et du *narrowing* au systèmes de déduction modulo proposée par Dowek *et al.* (2003). Ensuite nous décrirons TaMeD (*Tableaux Method for Deduction Modulo*), extension de la méthode des tableaux à la déduction modulo mise au point par Richard Bonichon (Bonichon, 2004; Bonichon et Hermant, 2006b; Bonichon, 2006).

5.4.1 Narrowing étendu et résolution : ENAR

La première méthode de démonstration automatique basée sur la déduction modulo est ENAR (Dowek *et al.*, 2003). Comme l'indique son nom, elle se base sur la méthode de résolution exposée en sous-section 2.5.1 et le *narrowing*. Le système ENAR travaille donc sur des formes clausales Υ. Chaque clause v dans Υ est associée à un ensemble C de contraintes de la forme $P = Q$ où P et Q sont des propositions atomiques. Une clause v associée à un ensemble de contraintes C est donc notée $v[C]$. Un ensemble de contraintes C est satisfaisable s'il existe une substitution σ tels que, pour chaque couples $P = Q \in C$, $P\sigma$ et $Q\sigma$ sont effectivement des formules égales. ENAR, tout comme la déduction modulo, est paramétré par un système de réécriture $\mathcal{R}$. ENAR comprend les deux règles d'inférence décrites par la figure 5.10.

Extended Resolution La première correspond au mécanisme de *résolution* : à partir de deux clauses de Υ, elle permet d'en générer une troisième.

Extended Narrowing La deuxième correspond au mécanisme de *narrowing* et encode la congruence correspondant au système de réécriture $\mathcal{R}$: l'une des propositions atomiques $v_{|w}$ d'une clause v de Υ est remplacée par le membre droit r d'une règle de réécriture $l \to r$ de $\mathcal{R}$. On obtient un nouvel ensemble de formules $v[r]_w$ qu'il faut remettre en forme clausale grâce au système de réécriture de la figure 2.16. La contrainte $v_{|w} = l$ est ajoutée aux clauses de cette forme clausale qui sont elles-mêmes ajoutées à Υ.

Les deux règles de ENAR sont utilisées de la façon suivante. Le point de départ est une formule, mise en forme clausale grâce au système de réécriture de mise en forme clausale. On applique alors de manière itérée les règles de ENAR afin de générer le maximum de nouvelles clauses, enrichissant ainsi la forme clausale sur laquelle on travaille. Si l'on obtient une forme clausale contenant la clause vide (c'est-à-dire un ensemble vide de formules) dont la contrainte est satisfaisable, nous obtenons une réfutation de la formule de départ : celle-ci est fausse. Au contraire si l'on obtient une forme clausale ne contenant pas la clause vide, mais à laquelle ENAR ne peut plus rajouter de clauses, on ne peut ni confirmer ni infirmer la validité de la formule.

Gilles Dowek, Thérèse Hardin et Claude Kirchner, après avoir introduit la déduction modulo ainsi que ENAR, démontrent la correction et la complétude de ce dernier système par rapport au calcul des séquents classique modulo $\mathrm{LK}_{\equiv}$ dans le cadre des systèmes de réécriture confluents.

Définition 5.4.1 (Correction et complétude de ENAR). *Soit $\mathcal{R}$ un système de règles de termes et de prédicats confluent et soit $\equiv$ la congruence qu'il engendre.*

Correction (Hermant, 2005) *Si ENAR paramétré par $\mathcal{R}$ génère la clause vide munie d'une contrainte satisfaisable à partir de $\mathsf{clF}(\Gamma, \neg\Delta)$ en un nombre fini d'étapes, alors $\Gamma \vdash \Delta$ est dérivable dans $LK_{\equiv}$ sans coupure.*

Complétude (Dowek *et al.*, 2003, théorème 3.1) *S'il existe une dérivation* sans coupure *de $\Gamma \vdash \Delta$ dans $LK_{\equiv}$, alors ENAR paramétré par $\mathcal{R}$ génère en un nombre fini d'étapes la clause vide munie d'une contrainte satisfaisable à partir de $\mathsf{clF}(\Gamma, \neg\Delta)$.*

déduction modulo *sans coupure* .

Remarquons que la complétude d'ENAR affirme que l'on peut engendrer la clause vide lorsqu'il existe une preuve *sans coupure* en calcul des séquents modulo. La méthode ENAR n'est donc complète que par rapport au calcul des séquent modulo sans coupures. Lorsque le calcul des séquents modulo en question ne satisfait pas l'admissibilité des coupures, il existe des séquents dérivables à l'aide de la règle de coupure que l'on ne peut affirmer avec ENAR, car ils n'admettent pas de dérivation sans coupure en déduction modulo. ENAR n'est donc pas complet par rapport au calcul des séquent modulo avec coupures. Il est donc important de caractériser, à l'aide des méthodes décrites en section 5.3.2, quels systèmes de règles de termes et de prédicats engendrent un système de déduction modulo vérifiant la propriété d'admissibilité des coupures : on pourra alors utiliser la méthode ENAR en garantissant sa complétude par rapport à la déduction modulo avec coupures. En particulier si $\mathcal{R}$ est un ensemble de règles de termes et de prédicats confluent tel que LK modulo $\mathcal{R}$ admet les coupures, alors ENAR paramétré par $\mathcal{R}$ permet de semi-décider l'insatisfaisabilité d'une formule dans la théorie $\mathcal{T}h_{\mathcal{R}}$: Si à partir de la clause $\mathsf{clF}(\varphi)$ on obtient une forme clausale contenant la clause vide munie d'une contrainte satisfaisable, c'est que φ est insatisfaisable dans la théorie $\mathcal{T}h_{\mathcal{R}}$. Si l'on obtient une forme clausale sur laquelle ENAR ne s'applique plus et ne contenant pas de clause vide munie d'une contrainte satisfaisable, c'est que φ est satisfaisable dans la théorie

$$
\begin{array}{rcl}
(\mathcal{B}, \varphi_1 \vee \varphi_2 \mid \mathcal{T})[C] & \rightarrow & (\mathcal{B}, \varphi_1 \mid (\mathcal{B}, \varphi_2 \mid \mathcal{T})[C] \\
(\mathcal{B}, \varphi_1 \wedge \varphi_2 \mid \mathcal{T})[C] & \rightarrow & (\mathcal{B}, \varphi_1, \varphi_2 \mid \mathcal{T})[C] \\
(\mathcal{B}, \varphi_1 \Rightarrow \varphi_2 \mid \mathcal{T})[C] & \rightarrow & (\mathcal{B}, \neg\varphi_1 \mid (\mathcal{B}, \varphi_2 \mid \mathcal{T})[C] \\
(\mathcal{B}, \bot \mid \mathcal{T})[C] & \rightarrow & \mathcal{T})[C] \\
(\mathcal{B}, \top \mid \mathcal{T})[C] & \rightarrow & (\mathcal{B} \mid \mathcal{T})[C] \\
(\mathcal{B}, \forall x.\varphi \mid \mathcal{T})[C] & \rightarrow & (\mathcal{B}, \forall x.\varphi, \varphi[y/x] \mid \mathcal{T})[C] \\
 & & \text{où } y \text{ est une variable fraîche} \\
(\mathcal{B}, \exists x.\varphi \mid \mathcal{T})[C] & \rightarrow & (\mathcal{B}, \varphi[f(y_1, \ldots, y_n)/x] \mid \mathcal{T})[C] \\
 & & \text{où } f \text{ est un symbole de fonction de Skolem frais} \\
 & & \text{et } y_1, \ldots, y_n \text{ sont les variables libres de } \mathcal{B}, \exists x.\varphi.
\end{array}
$$

(*extended branch closure*)

$$
(\mathcal{B}, \varphi_1, \neg\varphi_2 \mid \mathcal{T})[C] \rightarrow (\mathcal{T})[C \cup \{\varphi_1 = \varphi_2\}]
$$

(extended narrowing)

$$
\begin{array}{rcl}
(\mathcal{B}, \varphi \mid \mathcal{T})[C] & \rightarrow & (\mathcal{B}, \varphi[r]_w \mid \mathcal{T})[C \cup \{\varphi_{|w} = l\}] \\
 & & \text{où } l \rightarrow r \in \mathcal{R}
\end{array}
$$

FIG. 5.11 – TaMeD

$\mathcal{T}h_{\mathcal{R}}$. Enfin si ENAR ne termine pas, c'est que la formule φ est satisfaisable dans la théorie $\mathcal{T}h_{\mathcal{R}}$ mais la méthode ne permet pas de le montrer car elle ne termine pas.

5.4.2 Méthode des tableaux pour la déduction modulo : TaMeD

Nous allons à présent décrire la méthode des tableaux étendue à la déduction modulo par Richard Bonichon (Bonichon, 2004; Bonichon et Hermant, 2006b; Bonichon, 2006) et qu'il appelle TaMeD. La méthode TaMeD manipule des tableaux auxquels sont associés des ensembles de contraintes de la forme $\varphi = \psi$. Tout comme dans la sous-section précédente, un tel ensemble de contraintes C est satisfaisable s'il existe une substitution σ tels que, pour chaque couples $P = Q \in C$, $P\sigma$ et $Q\sigma$ sont effectivement des formules égales. Un tableaux $\mathcal{T}$ associé à à une contrainte C est noté $(\mathcal{T})[C]$. TaMeD se compose alors des règles suivantes.

Ces règles ont pour but la *réfutation* du tableaux, c'est-à-dire de la formule qu'il représente. TaMeD diffère de la méthode des tableaux que nous avons décrite en sous-section 2.5.2 par les deux règles suivantes.

Extended Branch Closure Celle-ci permet de fermer une branche : si la branche contient à la fois une formule et sa négation, puisqu'elle correspond à leur conjonction, cette branche est immédiatement réfutée.

Extended Narrowing Tout comme pour la méthode ENAR, la deuxième correspond au mécanisme de *narrowing* et encode la congruence correspondant au système de réécriture $\mathcal{R}$: l'une des proposition atomiques $\varphi_{|w}$ d'une branche $\mathcal{B}, \varphi$ du tableau est remplacée par le membre droit r d'une règle de réécriture

$l \longrightarrow r$ de $\mathcal{R}$. On obtient une nouvelle branche $\mathcal{B}, \varphi[r]_w$ qu'il faut continuer à traiter et la contrainte $\varphi_{|w} = l$ est ajoutée au tableau.

Richard Bonichon, après avoir introduit TaMeD, démontre la correction et la complétude de cette méthode.

Propriété 5.4.1 (Correction et complétude réfutationnelle de TaMeD (Bonichon, 2004, théorème 1)). *Soit $\mathcal{R}$ un système de règles de termes et de prédicats confluent et soit $\equiv$ la congruence qu'il engendre.*

Correction *Si TaMeD paramétré par $\mathcal{R}$ transforme en un nombre fini d'étapes le tableau $\{\{\varphi\}\}$ en un tableau vide associé à un ensemble de contraintes satisfaisable, alors $\varphi \vdash$ est dérivable dans $LK_{\equiv}$.*

Complétude réfutationnelle *S'il existe une dérivation* sans coupure *de $\varphi \vdash$ dans $LK_{\equiv}$, alors TaMeD paramétré par $\mathcal{R}$ transforme en un nombre fini d'étapes le tableau $\{\{\varphi\}\}$ en un tableau vide associé à un ensemble de contraintes satisfaisable.*

Notons que TaMeD n'est pas complet par rapport au calcul des séquents modulo avec coupure. Cela souligne encore une fois l'importance de caractériser, à l'aide des méthodes décrites en section 5.3.2, quels systèmes de règles de termes et de prédicats engendrent un système de déduction modulo vérifiant la propriété d'admissibilité des coupures : on pourra alors utiliser la méthode TaMeD en garantissant sa complétude par rapport à la déduction modulo avec coupures. Dans ce cas la méthode TaMeD (tout comme la résolution, la méthode des tableaux ou encore ENAR) permet de semi-décider l'insatisfaisabilité d'une formule : Si $\mathcal{R}$ est un ensemble de règles de termes et de prédicats confluent et tel que LK modulo $\mathcal{R}$ admet les coupures, alors pour réfuter une formule φ, on applique itérativement les règles de la figure 5.11 au tableau $\{\{\varphi\}\}$ tant que cela est possible. Lorsque l'on obtient un tableau sur lequel aucune règle de TaMeD s'applique, alors s'il s'agit d'un tableau vide dont la contrainte est satisfaisable, c'est que le φ est insatisfaisable. Au contraire s'il s'agit d'un tableau non vide ou bien dont la contrainte n'est pas satisfaisable, c'est que φ est satisfaisable. Enfin il est possible que l'application des règles de TaMeD ne termine pas et dans ce cas la formule est satisfaisable mais la méthode TaMeD ne permet pas de le montrer (puisqu'elle ne termine pas).

Chapitre 6

Superdéduction

Dans le précédent chapitre nous avons présenté le paradigme de déduction modulo qui permet de traduire la partie calculatoire d'une théorie en système de réécriture puis d'enrichir un système déductif grâce à la relation de réécriture ainsi engendrée. Dans ce chapitre, nous présentons un concept dual appelé superdéduction. La superdéduction permet de transformer la partie déductive d'une théorie en nouvelles inférences venant elles aussi enrichir le système déductif. Déduction modulo et superdéduction sont donc deux paradigmes se complétant. Tout ce qui est présenté au cours de ce chapitre constitue une contribution original de cette thèse que j'ai développée avec le concours de Paul Brauner et Claude Kirchner et qui a fait l'objet de plusieurs publications (Brauner *et al.*, 2007a,b).

Plonger une théorie comme l'arithmétique en logique du premier ordre puis l'utiliser directement dans des déductions en déduction naturelle ou en calcul des séquents n'est pas une heureuse idée. L'exemple de la figure 5.1 montre que même des formules simples deviennent ardues à dériver et que les dérivations produites sont difficilement lisibles. Le point de vue de l'approche « superdéduction » est le même que celui de l'approche « modulo » : les systèmes déductifs de Gentzen, bien que plus avancés que les systèmes à la Hilbert, constituent des langages assembleur de la déduction. La déduction modulo est la superdéduction sont des pas en avant vers des langages déductifs de haut niveau. Ces deux approches ne s'excluent pas mutuellement mais au contraire se complètent. Nous avons vu que la déduction modulo doit se rapporter à la partie calculatoire d'une théorie. De manière duale, la superdéduction se rapporte à la partie déductive d'une théorie : certains axiomes de la théorie représentent en fait un pouvoir déductif que l'on peut caractériser en termes de dérivations ouvertes. La superdéduction propose de remplacer simplement ces axiomes par les règles d'inférences correspondant directement à ces dérivations ouvertes. Le pouvoir déductif est donc le même, mais la dérivation est plus lisible. Considérons par exemple $\forall x.\forall y.(x \subseteq y) \Leftrightarrow (\forall z.z \in x \Rightarrow z \in y)$. La dérivation en

calcul des séquents suivante démontre que cet axiome implique $A \subseteq A$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots, z \in A \vdash A \subseteq A, z \in A}\,\text{Axiom}}{\cdots \vdash A \subseteq A, z \in A \Rightarrow z \in A}\,\Rightarrow\text{R}}{\cdots \vdash A \subseteq A, \forall z.(z \in A \Rightarrow z \in A)}\,\forall\text{R} \qquad \dfrac{}{\ldots, A \subseteq A \vdash A \subseteq A}\,\text{Axiom}}{\ldots, (\forall z.(z \in A \Rightarrow z \in A)) \Rightarrow A \subseteq A \vdash A \subseteq A}\,\Rightarrow\text{L}}{(A \subseteq A) \Leftrightarrow \forall z.(z \in A \Rightarrow z \in A) \vdash A \subseteq A}\,\wedge\text{L}}{\forall y.(A \subseteq y) \Leftrightarrow \forall z.(z \in A \Rightarrow z \in y) \vdash A \subseteq A}\,\forall\text{L}}{\forall x.\forall y.(x \subseteq y) \Leftrightarrow (\forall z.z \in x \Rightarrow z \in y) \vdash A \subseteq A}\,\forall\text{L}$$

L'axiome que nous venons d'utiliser et servant de définition au prédicat $\subseteq$ peut être vu comme une règle de calcul et ainsi être remplacé par la règle de réécriture $x \subseteq y \;\rightarrow\; \forall z.z \in x \Rightarrow z \in y$. La dérivation s'écrit alors en déduction modulo comme suit.

$$\dfrac{\dfrac{\dfrac{}{z \in A \vdash z \in A}\,\text{Axiom}}{\vdash z \in A \Rightarrow z \in A}\,\Rightarrow\text{R}}{\vdash A \subseteq A}\,\forall\text{R}\;\; A \subseteq A \rightarrow (\forall z.z \in A \Rightarrow z \in A)$$

Par rapport à la première dérivation que nous avons écrite, celle-ci correspond aux trois inférences les plus hautes, c'est-à-dire la décomposition de $\forall z.z \in A \Rightarrow z \in A$. Tout le reste de la première dérivation (les cinq autres inférences) a été remplacé par la réécriture de $A \subseteq A$ en $\forall z.z \in A \Rightarrow z \in A$.

La superdéduction propose d'aller plus loin que la déduction modulo précisément lorsque l'axiome que l'on recherche à traduire ressemble à la définition d'un prédicat P sous la forme d'une équivalence $\forall \bar{x}.(P \Leftrightarrow \varphi)$ ou d'une règle de prédicats $P \rightarrow \varphi$. Alors que la déduction modulo remplace l'axiome $\forall x.\forall y.(x \subseteq y) \Leftrightarrow (\forall z.z \in x \Rightarrow z \in y)$ par une règle de réécriture permettant de traduire $t \subseteq u$ par sa « définition », la superdéduction propose de rajouter à cette traduction la décomposition des connecteurs de cette définition. En déduction modulo, on utilise moralement l'inférence

$$\textsc{ConvR}\;\dfrac{\Gamma \vdash \forall z.z \in t \Rightarrow z \in u, \Delta}{\Gamma \vdash t \subseteq u, \Delta}\;\; t \subseteq u \rightarrow \forall z.z \in t \Rightarrow z \in u$$

que l'on peut faire suivre par les inférences

$$\begin{array}{l}\Rightarrow\text{R ou } \textsc{Intro}\\ \forall\text{R ou } \textsc{Intro}\end{array}\;\dfrac{\dfrac{\Gamma, z \in t \vdash z \in u, \Delta}{\Gamma \vdash z \in t \Rightarrow z \in u, \Delta}}{\Gamma \vdash \forall z.z \in t \Rightarrow z \in u, \Delta}\; z \text{ frais}\,.$$

En superdéduction, nous aurons l'inférence

$$\textsc{Inclusion}\;\dfrac{\Gamma, z \in t \vdash z \in u, \Delta}{\Gamma \vdash t \subseteq u, \Delta}\; z \text{ frais}\,.$$

Ce raccourci, dans le cas de la preuve de $A \subseteq A$, conduit directement à la dérivation

$$\textsc{Inclusion}\ \dfrac{\textsc{Axiom}\ \dfrac{}{z \in A \vdash z \in A}}{\vdash A \subseteq A}\quad .$$

Cette dernière dérivation est de loin la plus simple des trois dérivations de $A \subseteq A$ que nous avons écrites. Par ailleurs la règle d'inférence Inclusion traduit directement le raisonnement que font les mathématiciens lorsqu'ils écrivent « *Prouvons que $t \subseteq u$. Soit donc z quelconque tel que $z \in t$. Si l'on démontre que $z \in u$, alors nous aurons démontré que $t \subseteq u$.* »

La règle d'inférence Inclusion considère virtuellement le prédicat $\subseteq$ comme un connecteur construit à partir des connecteurs $\Rightarrow$ et $\forall$ de la formule $\forall z.z \in x \Rightarrow z \in y$. En ce sens et si l'on utilise la classification des inférences de la déduction naturelle, la règle Inclusion est une introduction. C'est une introduction à droite si l'on utilise la classification du calcul des séquents. Cette règle vient donc de pair avec ses duales, c'est-à-dire une règle d'élimination en déduction naturelle ou bien une règle d'introduction à gauche en calcul des séquents. Celles-ci correspondent respectivement à l'élimination et à l'introduction à gauche des connecteurs $\forall$ et $\Rightarrow$ de la « définition » de $\subseteq$. Ces règles sont

$$\textsc{Inclusion (Elim)}\ \dfrac{\Gamma \vdash t \subseteq u \qquad \Gamma \vdash v \in t}{\Gamma \vdash v \in u} \quad \text{et} \quad \textsc{Inclusion (IntroL)}\ \dfrac{\Gamma \vdash v \in t, \Delta \qquad \Gamma, v \in u \vdash \Delta}{\Gamma, t \subseteq u \vdash \Delta}\quad .$$

La première de ces deux règles, à l'instar des règles de la déduction naturelle, est intuitive. Elle représente le raisonnement « *Si t est inclus dans u et v est un élément de t, alors v est un élément de u.* » La deuxième règle est un peu moins intuitive pour le lecteur peu accoutumé au calcul des séquents mais reste néanmoins tout à fait naturelle. On peut la lire comme une réfutation de $t \subseteq u$: « *Pour réfuter le fait que t est inclus dans u, il faut trouver un élément v de t qui n'est pas un élément de u, c'est-à-dire démontrer $v \in t$ et réfuter $v \in u$* ». La règle d'introduction à gauche se reformule aussi par « *Si l'on a démontré que v est un élément de t et que l'on a aussi démontré Δ en supposant que v est un élément de u, alors en supposant que t est inclus dans u, on obtient Δ.* » Ces deux règles d'inférence correspondent bien respectivement à l'élimination ou à l'introduction à gauche des connecteurs $\forall$ et $\Rightarrow$ de la formule $\forall z.z \in t \Rightarrow z \in u$.

$$\Rightarrow\textsc{Elim}\ \dfrac{\forall\textsc{Elim}\ \dfrac{\Gamma \vdash \forall z.z \in t \Rightarrow z \in u}{\Gamma \vdash v \in t \Rightarrow v \in u} \qquad \Gamma \vdash v \in t}{\Gamma \vdash v \in u}$$

$$\forall\text{L}\ \dfrac{\Rightarrow\text{L}\ \dfrac{\Gamma \vdash v \in t, \Delta \qquad \Gamma, v \in u \vdash \Delta}{\Gamma, v \in t \Rightarrow v \in u \vdash \Delta}}{\Gamma, \forall z.z \in t \Rightarrow z \in u \vdash \Delta}$$

Cette approche peut être mécanisée. Nous appellerons les systèmes déductifs ainsi obtenus des *systèmes de superdéduction*. Le reste de ce chapitre est consacré à la présentation de ces systèmes et des développements qui en découlent. J'y démontre aussi que les preuves dans un système de superdéduction sont beaucoup plus proches de l'intuition et de la pratique humaine à l'aide d'une série d'exemples incluant l'égalité et l'induction noethérienne. Si l'idée originelle est de Wack (2005), la présentation des systèmes de superdéduction, en particulier sous la forme d'un calcul des séquents, est l'une de mes contributions personnelles mise au point avec le concours de Paul Brauner et Claude Kirchner (Brauner *et al.*, 2007a,b). En particulier nous avons introduit un langage de termes de preuve ainsi qu'une procédure d'élimination des coupures tous deux basés sur le langage d'Urban (sous-section 3.3.2) est ses procédures d'élimination des coupures (Urban et Bierman, 2001; Urban, 2001, 2000). Nous avons prouvé la normalisation forte du système sous des hypothèses appropriées et naturelles, ce qui implique alors la cohérence de la théorie sous-jacente et du système de déduction. Ces développements sont présentés en section 6.2. Enfin nous avons montré comment la superdéduction associée à la déduction modulo peuvent former la fondation formelle d'une nouvelle génération d'assistants de preuve en développant un prototype d'implémentation de la superdéduction modulo, Lemuridæ.

D'autres paradigmes proposent de rajouter des inférences spécialisées représentant des théories. En particulier l'*assertion level* de Huang (1994) est motivé par la présentation des preuves en langue naturelle. Une deuxième approche proposée par Negri et von Plato (1998, 2001) exprime des axiomes sous forme d'inférences. Néanmoins ces inférences n'agissent que sur la partie gauche des séquents et donc n'interagissent que pauvrement avec l'élimination des coupures. La superdéduction se rapproche plus de la *definitional reflection* de Schroeder-Heister (1993, 1990, 1989) qui permet d'ajouter à un calcul des séquents intuitionniste des règles d'introduction à gauche et à droite représentant ce qu'il appelle une *definitional clause* correspondant à ce que nous appelons « règle de prédicats ». En particulier des résultats d'élimination des coupures ont été prouvés pour ces logiques avec définitions et induction. Cependant nous verrons que travailler avec un calcul des séquents *classique* premet de traîter des axiomes plus généraux.

6.1 Principes de la superdéduction

Nous allons à présent décrire le calcul mécanisé des nouvelles règles d'inférence introduites par la superdéduction pour représenter des règles de prédicats. Deux présentations non équivalentes du paradigme de la superdéduction sont présentées ici : la déduction supernaturelle engrengée à partir de la déduction naturelle intuitionniste et la superdéduction engrengée à partir du calcul des séquents classique. Commençons par la déduction supernaturelle.

6.1.1 Déduction supernaturelle

Les avantages de la déduction modulo proviennent de la liberté que celle-ci laisse à l'utilisateur dans l'application du mécanisme de réécriture. Cette souplesse apporte une forte expressivité en complexifiant toutefois l'application du système d'inférence : bien que la réécriture des termes soit légitime pour capturer les étapes de calcul, il peut s'avérer difficile de raisonner sur des propositions modulo une congruence. En effet à chaque étape il est nécessaire de fournir le bon représentant de la classe d'équivalence considérée. Plus précisément, l'application des règles de déduction n'est plus dirigée par la syntaxe (en fait par le connecteur de tête) de la formule à prouver. Considérons encore une fois la règle de réécriture $x \subseteq y \longrightarrow \forall z.(z \in x \Rightarrow z \in y)$ traduisant la définition de l'inclusion. Alors on peut construire une dérivation en déduction naturelle modulo de la forme

$$\forall\textsc{Intro}\ \frac{\vdash z \in t \Rightarrow z \in u}{\vdash t \subseteq u}\ t \subseteq u \equiv \forall z.(z \in t \Rightarrow z \in u)\ .$$

Bien que nous appliquions la règle d'inférence ∀Intro, le connecteur ∀ n'apparaît pas en tête du la formule $t \subseteq u$ que l'on prouve ici. Il est nécessaire de considérer la formule $\forall z.(z \in t \Rightarrow z \in u)$ qui lui est congrue pour faire apparaître ce connecteur. Notons tout de même que si le système de réécriture définissant la congruence $\equiv$ vérifie la propriété de protection des connecteurs (propriété 5.1.1), on peut garantir que les formules de la forme $t \subseteq u$ représentent des quantifications universelles. On retrouve alors un système dirigé par la syntaxe modulo réécriture.

Dans sa thèse, Wack (2005) propose de remplacer la congruence sur les propositions par de nouvelles règles d'inférence. L'idée est de considérer des axiomes de la forme $P \equiv \varphi$ où P est atomique. Ces atomes correspondent donc à des règles de prédicats. Cette idéé n'est pas totalement neuve : déjà Prawitz (1965) avait proposé de remplacer de tels axiomes par des règles d'inférences remplaçant φ par P et inversement, appelées règles *folding* et *unfolding*. Benjamin Wack propose en outre que ces règles introduisent et éliminent le maximum de connecteurs de φ. L'idée est d'appliquer tant que possible un sous-ensemble des règles de NJ à φ pour obtenir les nouvelles prémisses des règles d'introduction et les conclusions des nouvelles règles d'élimination.

Définition 6.1.1 (Calcul des règles de déduction supernaturelle). *Soit une règle de réécriture $R : P \longrightarrow \varphi$. Pour calculer la règle d'introduction de R, on manipule un ensemble de séquents S ainsi qu'un ensemble C de couples (x, Γ) où x est une variable du premier ordre et Γ un contexte. Un tel couple représente la condition de bord $x \notin \mathcal{FV}(\Gamma)$. L'ensemble S représente les prémisses de la règle d'introduction. On initialise S et C respectivement par $S = \{\vdash \varphi\}$ et $C = \emptyset$. Puis on répète les opérations suivantes.*

∧**Intro** *Si S contient un séquent de la forme $\Gamma \vdash \psi_1 \wedge \psi_2$, on le remplace par les séquents $\Gamma \vdash \psi_1$ et $\Gamma \vdash \psi_2$.*

⇒**Intro** *Si S contient un séquent de la forme $\Gamma \vdash \psi_1 \Rightarrow \psi_2$, on le remplace par le séquent $\Gamma, \psi_1 \vdash \psi_2$.*

∀**Intro** *Si S contient un séquent de la forme $\Gamma \vdash \forall x.\psi$, on le remplace par $\Gamma \vdash \psi$ et on rajoute (x, Γ) à C.*

Finalement, si $S = \{\Gamma_i \vdash \psi_i / 1 \leqslant i \leqslant n\}$ et $C = \{(x_k, \Delta_k) / 1 \leqslant k \leqslant p\}$, la règle d'introduction associée à R est celle représentant les inférences

$$\frac{\Gamma, \Gamma_1\sigma \vdash \psi_1\sigma \qquad \dots \Gamma, \Gamma_n\sigma \vdash \psi_n\sigma}{\Gamma \vdash P\sigma}\ x_k \notin \mathcal{FV}(\Gamma, \Delta_k\sigma) \text{ POUR TOUT } 1 \leqslant k \leqslant p$$

où σ est une substitution dont le domaine est inclus dans $\mathcal{FV}(P)$.

Pour calculer les *règles d'élimination de R, on manipule un ensemble de formules S, un ensemble V de variables du premier ordre ainsi qu'une formule ψ. La prémisse principale des règles d'élimination sera $\vdash \varphi$. L'ensemble des formules S représente l'ensemble des prémisses secondaires. Enfin V représente les variables qui pourront être instanciées par la règle d'élimination. On initialise S, V et ψ respectivement par $S = \emptyset$, $V = \emptyset$ et $\psi = \varphi$. Puis on répète les opérations suivantes.*

∧**Elim** *Si ψ est de la forme $\psi_1 \wedge \psi_2$, on continue en parallèle le calcul où ψ est remplacé par ψ_1 et par ψ_2, obtenant ainsi deux ensembles de règles d'élimination dont on fait finalement l'union.*

⇒**Elim** *Si ψ est de la forme $\psi_1 \Rightarrow \psi_2$, alors on rajoute la formule ψ_1 à S et on remplace ψ par ψ_2.*

∀**Elim** *Si ψ est de la forme $\forall x.\psi'$, alors on rajoute la variable x à V puis on continue en remplaçant ψ par ψ'.*

Finalement, si $S = \{\psi_i / 1 \leqslant i \leqslant n\}$, la règle d'élimination associée à R est celle représentant les inférences

$$\frac{\Gamma \vdash P\sigma \qquad \Gamma \vdash \psi_1\sigma \qquad \dots \qquad \Gamma \vdash \psi_n\sigma}{\Gamma \vdash \psi'}$$

où σ est une substitution dont le domaine est inclus dans $V \cup \mathcal{FV}(P)$ et où ψ' est $\psi\sigma$ si $\psi = \bot$, n'importe quelle formule sinon.

On obtient systématiquement une règle d'introduction par règle de prédicats. En revanche, le cas ∧**Elim** (correspondant à la règle de d'inférence ∧ELIM) permet d'obtenir plusieurs règles d'élimination pour la même règle de prédicats. Les calculs de ces règles d'introduction et d'élimination correspondent à l'application de certaines inférences de la déduction naturelle : à partir du séquent $\Gamma \vdash \varphi$, pour trouver la règle d'introduction de R, on applique en fait du bas vers le haut les règles

$$\frac{\Gamma \vdash \psi_1 \qquad \Gamma \vdash \psi_2}{\Gamma \vdash \psi_1 \wedge \psi_2}\ , \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \forall x.\psi}\ x \notin \mathcal{FV}(\Gamma) \quad \text{et} \quad \frac{\Gamma, \psi_1 \vdash \psi_2}{\Gamma \vdash \psi_1 \Rightarrow \psi_2}$$

tant qu'il reste des formules auxquelles ces règles s'appliquent ; pour trouver les règles d'élimination de R, on applique en fait du haut vers le bas les règles

$$\frac{\Gamma \vdash \psi_1 \wedge \psi_2}{\Gamma \vdash \psi_1} \quad , \quad \frac{\Gamma \vdash \psi_1 \wedge \psi_2}{\Gamma \vdash \psi_2} \quad , \quad \frac{\Gamma \vdash \forall x.\psi}{\Gamma \vdash \psi[t/x]}$$

$$\frac{\Gamma \vdash \psi_1 \Rightarrow \psi_2 \qquad \Gamma \vdash \psi_1}{\Gamma \vdash \psi_2} \quad \text{et} \quad \frac{\Gamma \vdash \bot}{\Gamma \vdash \psi}$$

tant qu'il reste des formules auxquelles ces règles s'appliquent. Enfin on collecte toutes les prémisses ainsi formées, les conditions d'application et la conclusion, et on remplace φ par P pour bien obtenir des règles qui concernent P. Les systèmes de déduction supernaturelle sont alors définis comme suit.

Définition 6.1.2 (Système de déduction supernaturelle). *Soit un ensemble de règles de prédicats $\mathcal{R}$. Le système de déduction surpernaturelle associé à $\mathcal{R}$ est défini comme l'ensemble des inférences correspondant aux règles de la déduction naturelle et aux règles de déduction supernaturelle construites à partir de $\mathcal{R}$;*

Considérons encore un exemple de la théorie des ensembles traduisant les définitions de l'inclusion et de l'ensemble vide.

$$\mathcal{R} = \begin{cases} \subseteq_{def} : y \subseteq z & \rightarrow & \forall x.(x \in y \Rightarrow x \in z) \\ \emptyset_{def} : x \in \emptyset & \rightarrow & \bot \end{cases}$$

Ces règles de prédicats induisent une règle d'introduction et une règle d'élimination pour la définition de l'inclusion ainsi qu'une règle d'élimination pour la définition de l'ensemble vide :

$$\emptyset_{def}\textsc{Elim}\ \frac{\Gamma \vdash t \in \emptyset}{\Gamma \vdash \varphi} \quad , \quad \subseteq_{def}\textsc{Intro}\ \frac{\Gamma, x \in t \vdash x \in u}{\Gamma \vdash t \subseteq u}\ x \notin \mathcal{FV}(\Gamma)$$

$$\text{et} \quad \subseteq_{def}\textsc{Elim}\ \frac{\Gamma \vdash t \subseteq u \qquad \Gamma \vdash v \in t}{\Gamma \vdash v \in u} \quad .$$

Ces règles sont tout à fait naturelles et traduisent très bien le raisonnement des mathématiciens. Par exemple la règle $\subseteq_{def}$Elim représente le raisonnement *« Si t est inclus dans u et v est élément de t, alors v est élément de u »* . La dérivation suivante est valide dans le système de déduction supernaturelle associé à $\mathcal{R}$.

$$\dfrac{\dfrac{\textsc{Axiom}\ \dfrac{}{x \in \emptyset \vdash_{\equiv} x \in \emptyset}}{\emptyset_{def}\textsc{Elim}\ \dfrac{x \in \emptyset \vdash_{\equiv} x \in \emptyset}{x \in \emptyset \vdash_{\equiv} x \in A}}}{\subseteq_{def}\textsc{Intro}\ \dfrac{x \in \emptyset \vdash_{\equiv} x \in A}{\vdash_{\equiv} \emptyset \subseteq A}}$$

Cette démonstration est concise, lisible et utilise les inférences de la superdéduction au lieu d'un contexte déductif constitué d'axiomes correspondant à $\subseteq_{def}$ et $\emptyset_{def}$.

Les inférences des systèmes de déduction supernaturelle sont calculées à partir d'inférences de NJ. Cela se traduit par la propriété suivante.

Propriété 6.1.1 (Correction des règles de déduction supernaturelle). *Si $P \longrightarrow \varphi$ est une règle de prédicats aboutissant aux inférences*

$$\frac{\Gamma, \Gamma_1\sigma \vdash \psi_1\sigma \qquad \dots \Gamma, \Gamma_n\sigma \vdash \psi_n\sigma}{\Gamma \vdash P\sigma} \; x_k \notin \mathcal{FV}(\Gamma, \Delta_k\sigma) \text{ POUR TOUT } 1 \leqslant k \leqslant p$$

et

$$\frac{\Gamma \vdash P\sigma \qquad \Gamma \vdash \psi_1\sigma \qquad \dots \qquad \Gamma \vdash \psi_n\sigma}{\Gamma \vdash \psi'}$$

alors si dans la conclusion de l'introduction et dans la prémisse principale de l'élimination, $P\sigma$ est remplacé par $\varphi\sigma$, on obtient des inférences admissibles dans NJ.

En découle un premier résultat important en rapport à la déduction supernaturelle : sa complétude et correction en rapport à NJ. Tout comme pour la déduction modulo, une règle de prédicat $P \longrightarrow \varphi$ correspond en fait à un axiome $\forall \bar{x}.(P \Leftrightarrow \varphi)$ où $\bar{x}$ représente les variables libres de P. Rappelons que si $\mathcal{R}$ est un ensemble de règles de prédicats, la théorie $\{\forall \bar{x}.(P \Leftrightarrow \varphi)/P \longrightarrow \varphi \in \mathcal{R}\}$ est notée $\mathcal{T}h_{\mathcal{R}}$ (notation introduite en section 5.2).

Propriété 6.1.2 (Équivalence (Wack, 2005)). *Pour tout ensemble $\mathcal{R}$ de règles de prédicats, $\Gamma \vdash \varphi$ est dérivable dans le système de déduction supernaturelle associé à $\mathcal{R}$ si et seulement si $\Gamma, \mathcal{T}h_{\mathcal{R}} \vdash \varphi$ l'est dans NJ.*

6.1.2 Superdéduction

Le calcul des séquents, en particulier dans le cadre de la logique classique, admet de nombreuses propriétés attrayantes. Citons par exemple la propriété de la sous-formule (propriété 2.3.5) ou encore l'inversibilité de la plupart de ses règles d'inférence (propriété 2.3.12). Cette inversibilité est un outil puissant en particulier lorsqu'on cherche à démontrer la complétude des nouvelles inférences introduites par le paradigme de la superdéduction. Je le démontrerai d'ailleurs au chapitre 7. L'idée des systèmes de superdéduction, c'est-à-dire du paradigme de la superdéduction appliqué au calcul des séquents, est de rajouter au calcul des séquents classique pour chaque règle de prédicats $P \longrightarrow \varphi$ une règle d'introduction à gauche et une règle d'introduction à droite du prédicat P correspondant respectivement à l'introduction à gauche et à droite des connecteurs composant la formule φ. Le calcul des nouvelles règles d'inférence est décrit par la définition suivante.

Définition 6.1.3 (Calcul des règles de superdéduction). *Soit une règle de prédicats $R : P \longrightarrow \varphi$. Pour calculer les règles d'introduction à droite correspondant à R, on travaille avec un ensemble S de séquents, un ensemble C de couples (x, Γ) où x est une variable du premier ordre et Γ est un contexte ainsi qu'un ensemble de variables V. Ceux-ci sont initialisés respectivement par $S = \{\vdash \varphi\}$, $C = \emptyset$ et $V = \emptyset$. On répète alors les modifications suivantes.*

Axiom *Si S contient un séquent $\Gamma, \psi \vdash \psi, \Delta$, on l'enlève simplement de S.*

$\bot$**R** *Si S contient un séquent $\Gamma \vdash \bot, \Delta$, on l'enlève simplement de S.*

$\bot$**L** *Si S contient un séquent $\Gamma, \bot \vdash \Delta$, on le remplace par $\Gamma \vdash \Delta$.*

$\top$**L** *Si S contient un séquent $\Gamma, \top \vdash \Delta$, on l'enlève simplement de S.*

$\bot$**L** *Si S contient un séquent $\Gamma \vdash \top, \Delta$, on le remplace par $\Gamma \vdash \Delta$.*

$\vee$**R** *Si S contient un séquent $\Gamma \vdash \psi_1 \vee \psi_2, \Delta$, on le remplace par $\Gamma \vdash \psi_1, \psi_2, \Delta$.*

$\vee$**L** *Si S contient un séquent $\Gamma, \psi_1 \vee \psi_2 \vdash \Delta$, on le remplace par les séquents $\Gamma, \psi_1 \vdash \Delta$ et $\Gamma, \psi_2 \vdash \Delta$.*

$\wedge$**R** *Si S contient un séquent $\Gamma \vdash \psi_1 \wedge \psi_2, \Delta$, on le remplace par les séquents $\Gamma \vdash \psi_1, \Delta$ et $\Gamma \vdash \psi_2, \Delta$.*

$\wedge$**L** *Si S contient un séquent $\Gamma, \psi_1 \wedge \psi_2 \vdash \Delta$, on le remplace par $\Gamma, \psi_1, \psi_2 \vdash \Delta$.*

$\Rightarrow$**R** *Si S contient un séquent $\Gamma \vdash \psi_1 \Rightarrow \psi_2, \Delta$, on le remplace par $\Gamma, \psi_1 \vdash \psi_2, \Delta$.*

$\Rightarrow$**L** *Si S contient un séquent $\Gamma, \psi_1 \Rightarrow \psi_2 \vdash \Delta$, on le remplace par les séquents $\Gamma, \psi_2 \vdash \Delta$ et $\Gamma \vdash \psi_1, \Delta$.*

$\neg$**R** *Si S contient un séquent $\Gamma \vdash \neg\psi, \Delta$, on le remplace par $\Gamma, \psi \vdash \Delta$.*

$\neg$**L** *Si S contient un séquent $\Gamma, \neg\psi \vdash \Delta$, on le remplace par $\Gamma \vdash \psi, \Delta$.*

$\forall$**R** *Si S contient un séquent $\Gamma \vdash \forall x.\psi, \Delta$, on le remplace par $\Gamma \vdash \psi, \Delta$, et on rajoute $(x, (\Gamma, \Delta))$ à C.*

$\forall$**L** *Si S contient un séquent $\Gamma, \forall x.\psi \vdash \Delta$, on le remplace par $\Gamma, \psi \vdash \Delta$ et on rajoute x à V.*

$\exists$**R** *Si S contient un séquent $\Gamma \vdash \exists x.\psi, \Delta$, on le remplace par $\Gamma \vdash \psi, \Delta$ et on rajoute x à V.*

$\exists$**L** *Si S contient un séquent $\Gamma, \exists x.\psi \vdash \Delta$, on le remplace par $\Gamma, \psi \vdash \Delta$, et on rajoute $(x, (\Gamma, \Delta))$ à C.*

Finalement si $S = \{\Gamma_i \vdash \Delta_i / 1 \leqslant i \leqslant n\}$ et si $C = \{(x_k, \Gamma'_k) / 1 \leqslant k \leqslant p\}$, alors l'introduction à droite de P correspondante est la règle d'inférence représentant les inférences

$$\frac{\Gamma, \Gamma_1\sigma \vdash \Delta_1\sigma, \Delta \ \ldots \ \Gamma, \Gamma_n\sigma \vdash \Delta_n\sigma, \Delta}{\Gamma \vdash P\sigma, \Delta} \ x_k \notin \mathcal{FV}(\Gamma, \Gamma'_k\sigma, \Delta) \text{ POUR TOUT } 1 \leqslant k \leqslant p$$

où σ est une substitution dont le domaine est inclus dans $V \cup \mathcal{FV}(P)$.

Pour calculer les règles d'introduction à gauche correspondant à R, on travaille de même avec un ensemble S de séquents, un ensemble C de couples (x, Γ) où x est une variable du premier ordre et Γ est un contexte ainsi qu'un ensemble de variables V. Ceux-ci sont initialisés respectivement par $S = \{\varphi \vdash\}$, $C = \emptyset$ et $V = \emptyset$. On répète alors les mêmes modifications. Puis si $S = \{\Gamma_i \vdash \Delta_i / 1 \leqslant i \leqslant n\}$ et si $C = \{(x_k, \Gamma'_k) / 1 \leqslant k \leqslant p\}$, alors l'introduction à gauche de P correspondante est la règle d'inférence représentant les inférences

$$\frac{\Gamma, \Gamma_1\sigma \vdash \Delta_1\sigma, \Delta \ \ldots \ \Gamma, \Gamma_n\sigma \vdash \Delta_n\sigma, \Delta}{\Gamma, P\sigma \vdash \Delta} \ x_k \notin \mathcal{FV}(\Gamma, \Gamma'_k\sigma, \Delta) \text{ POUR TOUT } 1 \leqslant k \leqslant p$$

où σ est une substitution dont le domaine est inclus dans $V \cup \mathcal{FV}(P)$.

Des règles d'introduction à droite ainsi que des règles d'introduction à gauche sont obtenues systématiquement à partir d'une règle de prédicats. Les calculs de ces règles d'introduction correspondent à l'application de certaines inférences du calcul des séquents : pour trouver la règle d'introduction à gauche (respectivement à droite), on applique au séquent $\Gamma, \varphi \vdash \Delta$ (respectivement $\Gamma \vdash \varphi, \Delta$) les règles d'introduction de LK, c'est-à-dire $\bot$R, $\bot$L, $\top$R, $\top$L, $\wedge$R, $\wedge$L, $\vee$R, $\vee$L, $\Rightarrow$R, $\Rightarrow$L, $\neg$R, $\neg$L, $\forall$R, $\forall$L, $\exists$R and $\exists$L, du bas vers le haut de manière à décomposer tous les connecteurs de la formule φ. Enfin on collecte toutes les prémisses ainsi formées, les conditions d'application et la conclusion, et on remplace φ par P dans celle-ci pour bien obtenir des règles qui concernent P. Les systèmes de superdéduction sont alors définis comme suit.

Définition 6.1.4 (Système de superdéduction). *Soit un ensemble de règles de prédicats $\mathcal{R}$. Le système de superdéduction associé à $\mathcal{R}$ est défini comme l'ensemble des inférences correspondant aux règles du calcul des séquents classique LK et aux règles de superdéduction construites à partir de $\mathcal{R}$;*

Reprenons l'exemple de la règle de prédicats

$$x \subseteq y \quad \rightarrow \quad \forall z.(z \in x \Rightarrow z \in y) \; .$$

Le calcul de la règle d'introduction à droite est initialisé par

$$S = \{\vdash \forall z.(z \in x \Rightarrow z \in y)\} \; , \qquad C = \emptyset \quad \text{et} \quad V = \emptyset \; .$$

La première étape consiste à traiter le connecteur $\forall$ en tête : S est remplacé par $\{\vdash z \in x \Rightarrow z \in y\}$ et C est remplacé par $\{(z, \varepsilon)\}$ (ε représente ici le contexte vide). La deuxième étape consiste à traiter le connecteur $\Rightarrow$: S est remplacé par $\{z \in x \vdash z \in y\}$. Finalement il ne reste plus aucun connecteur à traiter dans les séquents de S et la procédure termine avec

$$S = \{z \in x \vdash z \in y\} \; , \qquad C = \{(z, \varepsilon)\} \quad \text{et} \quad V = \emptyset \; .$$

La règle obtenue est donc

$$\frac{\Gamma, z \in t \vdash z \in u, \Delta}{\Gamma \vdash t \subseteq u, \Delta} \; z \notin \mathcal{FV}(\Gamma, \Delta) \quad .$$

Pour obtenir la règle d'introduction à gauche, le calcul est initialisé par

$$S = \{\forall z.(z \in x \Rightarrow z \in y) \vdash\} \; , \qquad C = \emptyset \quad \text{et} \quad V = \emptyset \; .$$

La première étape consiste à traiter le connecteur $\forall$ en tête : S est remplacé par $\{\vdash z \in x \Rightarrow z \in y\}$ et V est remplacé par $\{z\}$. La deuxième étape consiste à traiter

le connecteur $\Rightarrow$: S est remplacé par $\{(\vdash z \in x), (z \in y \vdash)\}$. Finalement il ne reste plus aucun connecteur à traiter dans les séquents de S et la procédure termine avec

$$S = \{(\vdash z \in x), (z \in y \vdash)\}\ , \qquad C = \emptyset \quad \text{et} \quad V = \{z\}\ .$$

La règle obtenue est donc

$$\frac{\Gamma \vdash v \in t, \Delta \qquad \Gamma, v \in u \vdash \Delta}{\Gamma, t \subseteq u \vdash \Delta}$$

Notons tout de même que pour une seule et même règle de prédicats, on peut obtenir plusieurs règles d'introduction ou plusieurs règles d'élimination. Considérons par exemple la règle

$$P \quad \rightarrow \quad (\exists x.R(x)) \Rightarrow (\exists y.Q(y))$$

L'algorithme calculant les règles d'introduction à droite est initialisé par

$$S = \{\vdash (\exists x.R(x)) \Rightarrow (\exists y.Q(y))\}\ , \qquad C = \emptyset \quad \text{et} \quad V = \emptyset\ .$$

La seule première étape possible décompose l'implication. On obtient alors l'état

$$S = \{(\exists x.R(x)) \vdash (\exists y.Q(y))\}\ , \qquad C = \emptyset \quad \text{et} \quad V = \emptyset\ .$$

Deux choix s'offrent alors à nous : 1. on peut décomposer d'abord le $\exists$ de droite et celui de gauche ensuite, obtenant successivement les états

$$S = \{(\exists x.R(x)) \vdash Q(y)\}\ , \qquad C = \emptyset \quad \text{et} \quad V = \{y\}$$

puis

$$S = \{R(x) \vdash Q(y)\}\ , \qquad C = \{(x, Q(y))\} \quad \text{et} \quad V = \{y\}$$

2. ou bien on peut décomposer d'abord le $\exists$ de gauche et celui de droite ensuite, otenant successivement les états

$$S = \{R(x) \vdash (\exists y.Q(y))\}\ , \qquad C = \{(x, \exists y.Q(y))\} \quad \text{et} \quad V = \emptyset$$

puis

$$S = \{R(x) \vdash Q(y)\}\ , \qquad C = \{(x, \exists y.Q(y))\} \quad \text{et} \quad V = \{y\}\ .$$

Ces calculs débouchent respectivement sur les règles d'introduction

$$\frac{\Gamma, R(x) \vdash Q(t), \Delta}{\Gamma \vdash P, \Delta}\ x \notin \mathcal{FV}(\Gamma, t, \Delta) \quad \text{et} \quad \frac{\Gamma, R(x) \vdash Q(t), \Delta}{\Gamma \vdash P, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta)\ .$$

Notons que toute instance de la première règle sera une instance de la deuxième. On peut donc se limiter ici à l'usage de cette deuxième règle. Ce n'est néanmoins pas toujours le cas. Considérons maintenant la règle

$$P \quad \rightarrow \quad (\exists x_1.\forall x_2.A(x_1, x_2)) \vee (\exists y_1.\forall y_2.B(y_1, y_2))$$

Les règles d'inférences obtenues alors ne contiennent pas une unique règle plus générale, mais deux qui sont

$$\frac{\Gamma \vdash A(t, x_1), B(u, y_2), \Delta}{\Gamma \vdash P, \Delta} \begin{cases} x_2 \notin \mathcal{FV}(\Gamma, \Delta, u) \\ y_2 \notin \mathcal{FV}(\Gamma, \Delta) \end{cases}$$

et

$$\frac{\Gamma \vdash A(t, x_1), B(u, y_2), \Delta}{\Gamma \vdash P, \Delta} \begin{cases} x_2 \notin \mathcal{FV}(\Gamma, \Delta) \\ y_2 \notin \mathcal{FV}(\Gamma, \Delta, t) \end{cases} .$$

Par conséquent lorsque l'on considère un système de superdéduction associant plusieurs règles d'introduction à gauche ou à droite à une même règle de prédicats, il est important de bien considérer l'ensemble complet de ces règles d'introduction.

Toutes les inférences rajoutées au calcul des séquents par la superdéduction sont correctes car elles correspondent à des dérivations ouvertes dans le calcul des séquents. En fait chaque étape de calcul correspond à une inférence de LK.

Propriété 6.1.3 (Correction des règles de superdéduction). *Si $P \to \varphi$ est une règle de prédicats aboutissant aux inférences*

$$\frac{\Gamma, \Gamma_1\sigma \vdash \Delta_1\sigma, \Delta \quad \dots \quad \Gamma, \Gamma_n\sigma \vdash \Delta_n\sigma, \Delta}{\Gamma \vdash P\sigma, \Delta} \begin{cases} x_k \notin \Gamma, \Gamma'_k\sigma, \Delta \quad \forall 1 \leqslant k \leqslant p \\ \mathrm{dom}(\sigma) \subseteq V \cup \mathcal{FV}(P) \end{cases}$$

et

$$\frac{\Gamma, \Gamma_1\sigma \vdash \Delta_1\sigma, \Delta \quad \dots \quad \Gamma, \Gamma_n\sigma \vdash \Delta_{n'}\sigma, \Delta}{\Gamma, P\sigma \vdash \Delta} \begin{cases} x_k \notin \Gamma, \Gamma'_k\sigma, \Delta \quad \forall 1 \leqslant k \leqslant p' \\ \mathrm{dom}(\sigma) \subseteq V' \cup \mathcal{FV}(P) \end{cases} ,$$

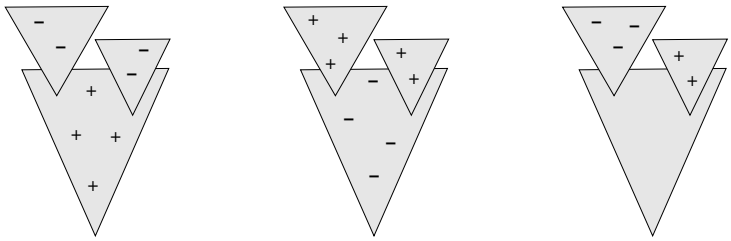
Alors si dans les conclusions, $P\sigma$ est remplacé par $\varphi\sigma$, on obtient des inférences admissibles dans LK.

Démonstration. Par induction sur le calcul des inférences de superdéduction, on démontre l'invariant établissant que tout arrêt prématur'e du calcul mène à des inférences admissible dans LK. □

De ce résultat découle la correction des systèmes de superdéduction.

Propriété 6.1.4 (Correction des systèmes de superdéduction (Brauner *et al.*, 2007a,b)). *Pour tout ensemble $\mathcal{R}$ de règles de prédicats, si $\Gamma \vdash \Delta$ est dérivable dans le système de superdéduction associé à $\mathcal{R}$, alors $\Gamma, \mathcal{Th}_\mathcal{R} \vdash \Delta$ l'est dans LK.*

La correction des nouvelles règles d'inférences introduites par la superdéduction est une propriété cruciale. Par exemple elle permet de démontrer que si la théorie $\mathcal{Th}_\mathcal{R}$ associée à un ensemble $\mathcal{R}$ de règles de prédicats est consistante, alors le système de superdéduction associé à $\mathcal{R}$ est lui-même consistant. En effet s'il existait une dérivation de $\vdash \bot$ dans le système de superdéduction, par correction il existerait une dérivation de $\mathcal{Th}_\mathcal{R} \vdash \bot$ dans LK, ce qui est impossible si $\mathcal{Th}_\mathcal{R}$ est consistant. Plus généralement, la propriété de correction permet de transformer toute dérivation



(a) Bipôle (b) Négation d'un bipôle (c) Hyperpôle

FIG. 6.1 – Définition 6.1.6

dans un système de superdéduction en dérivation dans LK. L'inverse (c'est-à-dire la *complétude* de la superdéduction) est un problème difficile (Houtmann, 2008a) et reste une question ouverte. Nous nous pencherons d'ailleurs plus largement sur ce problème au chapitre 7, en particulier pour les systèmes sans coupure. Nous sommes toutefois capable dès à présent de résoudre ce problème pour certaines classes de règles de prédicats identifiables syntaxiquement. Cette identification repose sur les notions de monopôles et de bipôles.

Définition 6.1.5 (Occurrences de connecteurs neutres, synchrones et asynchrones). *Dans une formule φ, nous dirons qu'une occurrence d'un connecteur est* **neutre** *si c'est un connecteur propositionnel ;* **synchrone** *si c'est un quantificateur universel apparaissant en position positive ou un quantificateur existentiel apparaisant en position négative ;* **asynchrone** *si c'est un quantificateur existentiel apparaissant en position négative ou un quantificateur universel apparaissant en position positive.*

Définition 6.1.6 (Monopôles, bipôles et hyperpôles). **Les monopôles** *sont des formules ne contenant que des occurrences neutres ou asynchrones de connecteurs.* **Les bipôles** *sont des formules construites à partir de monopôles en utilisant des occurrences neutres ou synchrones de connecteurs.* **Les hyperpôles** *sont des bipôles dont la négation est aussi un bipôle.*

Notons que la négation transforme les connecteurs synchrones en connecteurs asynchrones et inversement. La figure 6.1 représente un bipôle, la négation d'un bipôle ainsi qu'un hyperpôle, en utilisant les symboles « + » et « − » pour représenter respectivement les occurrences synchrones et asynchrones. Les hyperpôles permettent de définir une classe de règles de prédicats garantissant la complétude du système de superdéduction correspondant.

Propriété 6.1.5 (Complétude de la superdéduction (Brauner *et al.*, 2007a,b)). *Pour tout ensemble $\mathcal{R}$ de règles de prédicats dont les membres droits sont des hyperpôles, si $\Gamma, \mathcal{T}h_{\mathcal{R}} \vdash \Delta$ est dérivable dans LK, alors $\Gamma \vdash \Delta$ est dérivable dans le système de superdéduction associé à $\mathcal{R}$.*

Lorsqu'un ensemble de règle de prédicats contient des règles dont les membres droits ne sont pas des hyperpôles, la complétude du système de superdéduction qui en découle n'est pas assurée. Dowek (2010) a par exemple proposé le contre-exemple suivant : considérons la règle de prédicat

$$P \quad \rightarrow \quad \forall x.\exists y.(Q(x,y) \Rightarrow Q(x,f(y)))\ .$$

Alors que le séquent

$$P \Leftrightarrow \forall x.\exists y.(Q(x,y) \Rightarrow Q(x,f(y))), \forall x.\exists y.(Q(x,y) \Rightarrow Q(x,f(y))) \vdash P$$

est dérivable en calcul des séquents, le séquent $\forall x.\exists y.(Q(x,y) \Rightarrow Q(x,f(y))) \vdash P$ ne l'est pas dans le système de superdéduction associé à cette règle de prédicat : pour dériver ce séquent sans coupure il faudrait utiliser la variable x_0 libérée par la décomposition de P à droite

$$\frac{\forall x.\exists y.(Q(x,y) \Rightarrow Q(x,f(y))), Q(x_0,t)| - Q(x_0,f(t))}{\forall x.\exists y.(Q(x,y) \Rightarrow Q(x,f(y))) \vdash P}$$

pour instancier le $\forall x$ de la formule de gauche, puis libérer une variable y_0 grâce au $\exists y$ de cette même formule de gauche. C'est alors cette variable y_0 qu'il faut utiliser comme terme t dans la décomposition de P. C'est impossible : la décomposition de P doit avoir lieu avant la décomposition du $\forall x$ de gauche qui doit avoir lieu avant la décomposition du $\exists y$ de gauche qui ne peut donc pas avoir lieu avant la décomposition de P. Notons que puisqu'il n'est pas possible de dériver ce séquent sans coupure, il n'est pas non plus possible de le dériver avec coupures car la règle de prédicat satisfait l'hypothèse 6.2.1 et donc par la propriété 6.2.6 que nous démontrerons dans la section suivante, la règle de coupure est redondante dans le système de superdéduction associé.

Le prédicat P contient ici moralement la séquence $\forall x.\exists y$. En le décomposant à droite, on *force* l'application successive des règles $\forall$R puis $\exists$R. On force donc l'application de règles non inversibles. On pouvait donc s'attendre à perdre la complétude.

Afin de pouvoir traiter tout de même des règles de prédicats dont les membres droits ne sont pas des hyperpôles, nous définirons en section 7.2 une procédure permettant de transformer un tel ensemble de règles en un ensemble équivalent et ne contenant que des règles dont les membres droits sont tous des hyperpôles. On obtient alors un système de superdéduction différent mais correct et complet.

6.2 Coupures en superdéduction

Tout comme pour la déduction modulo, ni la normalisation forte ni l'admissibilité des coupures ni même la cohérence des systèmes de déduction supernaturelle ou de superdéduction ne sont garantis. En fait, tous les contre-exemples décrits en sous-section 5.3.1 s'adaptent parfaitement à ces paradigmes. Le contre-exemple de Crabbé

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,\textsc{Axiom}}{A, B \vdash A}\,R_c\textsc{Intro} \qquad \dfrac{\dfrac{}{A, B \vdash A}\,\textsc{Axiom}}{A, B \vdash \bot}\,R_c\textsc{Elim2}}{B \vdash A} \qquad \dfrac{\dfrac{}{A, B \vdash A}\,\textsc{Axiom}}{A, B \vdash \bot}\,R_c\textsc{Elim2}}{B \vdash A} \qquad \vdots}{B \vdash \bot}\,R_c\textsc{Intro}}{\vdash \neg B}\,\neg\textsc{Intro}$$

FIG. 6.2 – Contre-exemple à la normalisation en superdéduction naturelle

peut par exemple être aisément adapté (Brauner *et al.*, 2007a,b) : la règle de prédicat considérée est $R_c \;:\; A \;\to\; B \wedge (A \Rightarrow \bot)$. Dans le cadre de la déduction supernaturelle, elle mène aux règles d'inférence

$$\dfrac{\Gamma \vdash B \qquad \Gamma, A \vdash \bot}{\Gamma \vdash A}\,R_c\textsc{Intro}\;, \qquad \dfrac{\Gamma \vdash A}{\Gamma \vdash B}\,R_c\textsc{Elim1} \quad \text{et} \quad \dfrac{\Gamma \vdash A \qquad \Gamma \vdash A}{\Gamma \vdash \bot}\,R_c\textsc{Elim2}\;.$$

L'élimination des coupures, pour ces règles d'inférence, correspond aux réductions des preuves

$$\dfrac{\dfrac{\dfrac{\overset{\pi_1}{\Gamma \vdash B} \qquad \overset{\pi_2}{\Gamma, A \vdash \bot}}{\Gamma \vdash A}\,R_c\textsc{Intro}}{\Gamma \vdash B}\,R_c\textsc{Elim1} \quad \text{et} \quad \dfrac{\dfrac{\overset{\pi_3}{\Gamma \vdash B} \qquad \overset{\pi_4}{\Gamma, A \vdash \bot}}{\Gamma \vdash A}\,R_c\textsc{Intro} \qquad \overset{\pi_5}{\Gamma \vdash A}}{\Gamma \vdash \bot}\,R_c\textsc{Elim2}\;.$$

La première se réduit en π_1. La deuxième se réduit en la dérivation π_4 où les axiomes de la forme $\Gamma, \Gamma', A \vdash A$ sont remplacés par la dérivation $\Gamma' \cdot \pi_5$ de $\Gamma, \Gamma' \vdash A$. Considérons à présent la dérivation décrite par la figure 6.2. Elle contient une coupure R_cIntro/Elim2. Si l'on réduit cette coupure, on obtient exactement la même dérivation. C'est donc un contre-exemple à la normalisation en déduction supernaturelle. Dans le cadre de la superdéduction, cette même règle de prédicats mène aux règles d'inférence

$$R_c\mathrm{R}\;\dfrac{\Gamma \vdash B, \Delta \qquad \Gamma, A \vdash \Delta}{\Gamma \vdash A, \Delta} \quad \text{et} \quad R_c\mathrm{L}\;\dfrac{\Gamma, B \vdash A, \Delta}{\Gamma, A \vdash \Delta}\;.$$

Nous verrons en sous-section 6.2.2 comment ces règles permettent d'écrire une coupure dont la réduction ne termine pas. De même, les autres contre-exemples que nous avons exposés en sous-section 5.3.1 s'adaptent au cadre de la déduction supernaturelle comme au cadre de la superdéduction.

Nous allons à présent développer plusieurs critères syntaxiques permettant de garantir l'élimination des coupures ou la normalisation des systèmes de déduction supernaturelle ou de superdéduction. En sous-section 6.2.1, nous traduisons les dérivations en déduction supernaturelle ainsi qu'en superdéduction vers la déduction modulo de manière à démontrer que la normalisation en déduction modulo implique la normalisation en déduction supernaturelle et en superdéduction. En sous-section 6.2.2, nous nous basons sur le calcul d'Urban pour démontrer la normalisation en superdéduction.

6.2.1 Traduction vers la déduction modulo

Les inférences en déduction supernaturelle ainsi qu'en superdéduction sont calculées respectivement à partir des inférences de NJ et LK. Nous nous appuyons sur cette observation pour traduire les dérivations en déduction supernaturelle et en superdéduction en dérivations en déduction modulo.

De la déduction supernaturelle vers la déduction modulo

Les systèmes de déduction supernaturelle sont calculés à partir d'inférences de NJ. Soit un ensemble $\mathcal{R}$ de règles de prédicats. Comme le démontre la propriété 6.1.1, les nouvelles inférences de déduction supernaturelle correspondant à une règle $P \longrightarrow \varphi$ de $\mathcal{R}$ peuvent être transformées en inférences admissibles dans NJ : Lorsque nous considérons une introduction correspondant à $P \longrightarrow \varphi$, sa conclusion est de la forme $\Gamma \vdash P\sigma$ et si l'on y remplace $P\sigma$ par $\varphi\sigma$, on obtient une inférence admissible dans NJ. Lorsque nous considérons une élimination correspondant à $P \longrightarrow \varphi$, sa prémisse principale est de la forme $\Gamma \vdash P\sigma$ et si l'on y remplace $P\sigma$ par $\varphi\sigma$, on obtient une inférence admissible dans NJ. Notons que la congruence $\equiv$ engendrée par $\mathcal{R}$ égalise les formules $\varphi\sigma$ et $P\sigma$ (quel que soit σ). Par conséquent ce remplacement peut être directement effectué dans le système de déduction modulo $\text{NJ}_{\equiv}$. Nous traduisons donc les dérivations du système de déduction surpernaturelle en dérivations de $\text{NJ}_{\equiv}$ de la façon suivante. **(a)** Si l'inférence α apparaissant à la racine de la dérivation est en fait une inférence de NJ, il suffit de traduire récursivement les dérivations de ses prémisses dans $\text{NJ}_{\equiv}$ pour obtenir grâce à la même inférence α une dérivation du même séquent dans $\text{NJ}_{\equiv}$. **(b)** Si l'inférence α apparaissant à la racine de la dérivation est une introduction d'un prédicat P en déduction supernaturelle correspondant à une règle de prédicats $P \longrightarrow \varphi$ de $\mathcal{R}$, il suffit de traduire récursivement les dérivations de ses prémisses dans $\text{NJ}_{\equiv}$ puis d'utiliser l'inférence de NJ obtenue à partir de α par la propriété 6.1.1 pour obtenir une dérivation où l'instance de P à été remplacée par l'instance de φ correspondante. Alors $\text{ConvR}_{\equiv}$ permet de convertir cette dernière dérivation en dérivation dans $\text{NJ}_{\equiv}$ du séquent de départ. **(c)** Si l'inférence α apparaissant à la racine de la dérivation est une élimination d'un prédicat P en déduction supernaturelle correspondant à une règle de prédicats $P \longrightarrow \varphi$ de $\mathcal{R}$, les dérivations de ses prémisses se traduisent récursivement en dérivations dans $\text{NJ}_{\equiv}$. Sa prémisse principale est de la forme $\Gamma \vdash P\sigma$. La règle

ConvR$_{\equiv}$ nous permet alors d'obtenir un dérivation dans NJ$_{\equiv}$ de $\Gamma \vdash \varphi\sigma$. Enfin l'inférence de NJ obtenue à partir de α par la propriété 6.1.1 nous permet d'obtenir à partir des dérivations de $\Gamma \vdash \varphi\sigma$ ainsi que des prémisses secondaires une dérivation du séquent de départ dans NJ$_{\equiv}$.

De la superdéduction vers la déduction modulo

Les systèmes de superdéduction sont calculés à partir d'inférences de LK. Soit un ensemble $\mathcal{R}$ de règles de prédicats. Comme le démontre la propriété 6.1.3, les nouvelles inférences de superdéduction correspondant à une règle $P \to \varphi$ de $\mathcal{R}$ peuvent être transformées en inférences admissibles dans LK : il suffit de remplacer $P\sigma$ par $\varphi\sigma$ dans la conclusion lorsque nous considérons une introduction à droite ou à gauche correspondant à $P \to \varphi$. Encore une fois, la congruence $\equiv$ engendrée par $\mathcal{R}$ égalise les formules $P\sigma$ et $\varphi\sigma$ (quel que soit σ). Par conséquent ce remplacement peut être directement effectué dans le système de déduction modulo LK$_{\equiv}$. Nous traduisons donc les dérivations du système de superdéduction en dérivations de LK$_{\equiv}$ de la façon suivante : **(a)** si l'inférence α apparaissant à la racine de la dérivation est en fait une inférence de LK, il suffit de traduire récursivement les dérivations de ses prémisses dans LK$_{\equiv}$ pour obtenir grâce à la même inférence α une dérivation du même séquent dans LK$_{\equiv}$; **(b)** si l'inférence α apparaissant à la racine de la dérivation est une introduction à gauche ou à droite d'un prédicat P en déduction supernaturelle correspondant à une règle de prédicats $P \to \varphi$ de $\mathcal{R}$, il suffit de traduire récursivement les dérivations de ses prémisses dans LK$_{\equiv}$ puis d'utiliser l'inférence de LK obtenue à partir de α par la propriété 6.1.3 pour obtenir une dérivation où l'instance de P à été remplacée par l'instance de φ correspondante. Alors ConvR$_{\equiv}$ ou ConvL$_{\equiv}$ permettent de convertir cette dernière dérivation en dérivation dans LK$_{\equiv}$ du séquent de départ.

Conséquences sur l'élimination des coupures

Nous venons de définir deux traductions : la première traduit les dérivations en déduction supernaturelle vers NJ$_{\equiv}$, la deuxième traduit les dérivations en superdéduction vers LK$_{\equiv}$. Notons qu'une dérivation sans coupure est traduite par une dérivation sans coupure. Nous en déduisons le résultat suivant.

Propriété 6.2.1. *Soit $\mathcal{R}$ un ensemble de règles de prédicats et $\equiv$ la congruence qu'il engendre. 1. Tout séquent dérivable dans le système de déduction supernaturelle sans coupure associé à $\mathcal{R}$ est dérivable dans NJ$_{\equiv}$ sans coupure. 2. Tout séquent dérivable dans le système de superdéduction sans coupure associé à $\mathcal{R}$ est dérivable dans LK$_{\equiv}$ sans coupure. L'admissibilité des coupures en déduction supernaturelle et superdéduction implique donc l'admissibilité des coupures respectivement en déduction modulo NJ$_{\equiv}$ et en calcul des séquents LK$_{\equiv}$.*

Notons à présent une deuxième propriété de nos traductions relative àux coupures : que ce soit en déduction supernaturelle ou en superdéduction, si π est une

dérivation contenant une coupure et π' est un réduit direct résultant de l'élimination de cette coupure, alors la traduction $\tilde{\pi}$ de π contient aussi une coupure et se réduit en au moins un pas en la traduction $\tilde{\pi'}$ de π'. Alors une réduction infinie de π se traduit automatiquement en une réduction infinie de $\tilde{\pi}$. Nous en déduisons les résultats suivants.

Propriété 6.2.2. *Soit $\mathcal{R}$ un ensemble de règles de prédicats et $\equiv$ la congruence qu'il engendre. 1. Si une procédure d'élimination des coupures est fortement normalisante dans $N\mathcal{J}_{\equiv}$, alors la procédure correspondante pour le système de déduction supernaturelle associé à $\mathcal{R}$ est aussi fortement normalisante. 2. Si une procédure d'élimination des coupures est fortement normalisante dans $LK_{\equiv}$, alors la procédure correspondante pour le système de superdéduction associé à $\mathcal{R}$ est aussi fortement normalisante.*

6.2.2 Calcul d'Urban pour la superdéduction

Nous l'avons vu au chapitre 3, la relation entre logique classique et calcul existe sous plusieurs formes. Depuis les résultats de prouvabilité de Friedman (1978) et la correspondance de Griffin (1990) entre logique classique et programmation par continuations, de nombreux formalismes proposent d'écrire une correspondance de Curry-De Bruijn-Howard pour le calcul des séquents classique : en particulier le $\bar{\lambda}\mu\tilde{\mu}$-calcul de Curien et Herbelin (2000) que nous avons exposé en sous-section 3.3.1 ainsi que le calcul d'Urban (Urban et Bierman, 2001; Urban, 2001, 2000) que nous avons décrit en sous-section 3.3.2. Le premier est directement inspiré du $\lambda\mu$-calcul et propose un paradigme de typage *orienté* : dans un séquent, ou plutôt dans un jugement de typage, une formule est mise à l'écart comme le type du terme de preuve typé. Nous appellerons ce mécanisme le *focus*. Les autres formules sont les types des variables. Cet orientation provient en fait de l'orientation de la déduction naturelle des hypothèses (les variables) vers l'unique conclusion (le lambda-terme typé), orientation conservée par la construction successive du $\lambda\mu$-calcul puis du $\bar{\lambda}\mu\tilde{\mu}$-calcul à partir du lambda-calcul. Au contraire le calcul d'Urban ne se repose pas sur le lambda-calcul mais considère directement le calcul des séquents comme un calcul en soi : un terme de preuve est toujours typé par un séquent entier et aucune formule n'est mise à l'écart par les jugements de typage. À chaque règle d'inférence est associé un constructeur du langage. Celui-ci permet alors de représenter de façon plus concise les dérivations en calcul des séquents puis d'étudier formellement l'élimination des coupures. Notons que le calcul d'Urban peut aussi être étudié dans un cadre non-typé comme le proposent Bakel *et al.* (2005); Bakel et Lescanne (2008). Ils étudient le fragment implicatif qu'ils appellent $\mathcal{X}$ et démontrent son pouvoir expressif en y interprétant par exemple le lambda-calcul, le $\lambda\mu$-calcul ou encore le $\lambda\mathbf{x}$ de Bloo et Rose (1995).

Afin de représenter les dérivations des systèmes de superdéduction, on peut donc choisir de se baser sur le $\bar{\lambda}\mu\tilde{\mu}$-calcul ou encore sur le calcul d'Urban. Pour expliquer le choix que nous allons faire, remarquons que le calcul des nouvelles inférences de

la superdéduction contient des changements de *focus* comme par exemple dans la dérivation

$$\vee\mathrm{L}\ \dfrac{\varphi_1, \varphi_3 \vdash \varphi_4 \qquad \varphi_2, \varphi_3 \vdash \varphi_4}{\Rightarrow\mathrm{R}\ \dfrac{\varphi_1 \vee \varphi_2, \varphi_3 \vdash \varphi_4}{\varphi_1 \vee \varphi_2 \vdash \varphi_3 \Rightarrow \varphi_4}}\ .$$

Elle contient un changement de focus, du bas vers le haut, du séquent $\varphi_1 \vee \varphi_2, \varphi_3 \vdash \underline{\varphi_4}$ vers le séquent $\varphi_3, \underline{\varphi_1 \vee \varphi_2} \vdash \varphi_4$. De telles étapes implicites en calcul des séquents sont explicites en $\bar{\lambda}\mu\tilde{\mu}$-calcul : les constructeurs μ et $\tilde{\mu}$ joue effectivement le rôle de *focusers*. Puisque la construction des nouvelles règles d'inférence de la superdéduction contient de tels changements de focus qui sont finalement masqués dans l'inférence finale, il est plus simple d'utiliser un langage de termes de preuve où ces changements de focus sont implicites. C'est bel et bien le cas du calcul d'Urban. Nous avons donc choisi de l'étendre aux systèmes de superdéduction.

Termes de preuves pour la superdéduction

Quand les nouvelles inférences de superdéduction correspondant à une certaine règle de prédicats $P \to \varphi$ sont calculées, la procédure calcule en fait une dérivation *ouverte* qui, pour être transformée en inférence du système de superdéduction obtenu, doit être instanciée de trois manières : (1) en instanciant les variables du premier ordre de la règle $P \to \varphi$; (2) en instanciant les variables du premier ordre correspondant à l'ensemble V de la définition 6.1.3, c'est-à-dire en fournissant des termes du premier ordre aux décompositions correspondant aux règles ∃R et ∀L et éventuellement (3) en lui fournissant des preuves des prémisses. Afin de représenter ces dérivations ouvertes, nous utiliserons une notion formelle de *termes ouverts* : des termes qui contiennent (a) des feuilles ouvertes représentant les prémisses restant à prouver que nous noterons $\Box \triangleright \Gamma \vdash \Delta$ et (b) des marques substitutives que nous noterons $\alpha, \beta \ldots$ représentant des termes du premier ordre non instanciés. Par conséquent, nous utiliserons des substitutions sur ces marques substitutives notées $[\alpha := \mathsf{t}, \ldots]$ (où t est un terme du premier ordre) et applicables aux termes du premier ordre, aux formules, aux séquents ainsi qu'aux termes. L'ensemble de termes ouverts est alors le langage hors-contexte associé à la grammaire sous forme de Backus-Naur suivante.

$$\begin{aligned} C, D \ ::= \ & \Box \triangleright \Gamma \vdash \Delta \mid \mathsf{Ax}(x, a) \mid \mathsf{Cut}(\widehat{a}C, \widehat{x}D) \mid \mathsf{False}_L(x) \mid \mathsf{True}_R(a) \\ & \mid \mathsf{Not}_R(\widehat{x}C, a) \mid \mathsf{Not}_L(\widehat{a}C, x) \mid \mathsf{Imp}_R(\widehat{x}\widehat{a}C, b) \mid \mathsf{Imp}_L(\widehat{x}C, \widehat{a}D, y) \\ & \mid \mathsf{And}_R(\widehat{a}C, \widehat{b}D, c) \mid \mathsf{And}_L(\widehat{x}\widehat{y}C, z) \mid \mathsf{Or}_R(\widehat{a}\widehat{b}C, c) \mid \mathsf{Or}_L(\widehat{x}C, \widehat{y}D, z) \\ & \mid \mathsf{Exists}_R(\widehat{a}C, \alpha, b) \mid \mathsf{Exists}_L(\widehat{x}\widehat{\mathrm{x}}C, y) \mid \mathsf{Forall}_R(\widehat{a}\widehat{\mathrm{x}}C, b) \mid \mathsf{Forall}_L(\widehat{x}C, \alpha, y) \end{aligned}$$

La procédure d'élimination des coupures de la figure 3.11 est naturellement étendue aux termes ouverts. Le typage est aussi étendu aux termes ouverts en rajoutant les inférences

$$\dfrac{}{(\Box \triangleright \Gamma \vdash \Delta) \triangleright \Gamma \vdash \Delta}$$

au système de typage de la figure 3.10. Nous noterons d'ailleurs ces jugements de typage plus simplement $\Box \triangleright \Gamma \vdash \Delta$.

Une dérivation de typage *ouverte* est une dérivation de typage d'un terme ouvert. Ses feuilles ouvertes sont les feuilles correspondant au typage de $\Box$, c'est-à-dire les feuilles ouvertes du terme ouvert. Si C est un terme ouvert, nous noterons n_C le nombre d'occurrences de $\Box$ dans C. Si une substitution $\sigma = [\alpha_1 := \mathsf{t}_1, \ldots, \alpha_n := \mathsf{t}_n]$ remplace toutes les marques substitutives de C, alors nous dirons que σ *couvre* C. Si $M_1, \ldots, M_{n_C}$ sont des termes, alors le terme $\sigma C[M_1, \ldots, M_{n_C}]$ est défini comme suit.

- si $n_C = 0$, alors $\sigma C[] = \sigma C$;
- si $C = \Box \triangleright \Gamma \vdash \Delta$ et $n_C = 1$, alors $\sigma C[M] = M$;
- si $C = \mathsf{Cut}(\widehat{a}C_1, \widehat{x}C_2)$ alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Cut}(\widehat{a}\sigma C_1[M_1, \ldots, M_{n_{C_1}}], \widehat{x}\sigma C_2[M_{n_C+1}, \ldots, M_{n_C}], c) \text{ ;}$$

- si $C = \mathsf{Not}_R(\widehat{x}C_1, a)$ alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Not}_R(\widehat{x}\sigma C_1[M_1, \ldots, M_{n_C}], a) \text{ ;}$$

- si $C = \mathsf{Not}_L(\widehat{a}C_1, x)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Not}_L(\widehat{a}\sigma C_1[M_1, \ldots, M_{n_C}], x)$$

- si $C = \mathsf{Imp}_R(\widehat{x}\widehat{a}C_1, b)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Imp}_R(\widehat{x}\widehat{a}\sigma C_1[M_1, \ldots, M_{n_C}], b) \text{ ;}$$

- si $C = \mathsf{Imp}_L(\widehat{x}C_1, \widehat{a}C_2, y)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Imp}_L(\widehat{x}\sigma C_1[M_1, \ldots, M_{n_{C_1}}], \widehat{a}\sigma C_2[M_{n_{C_1}+1}, \ldots, M_{n_C}], y) \text{ ;}$$

- si $C = \mathsf{And}_R(\widehat{a}C_1, \widehat{b}C_2, c)[M_1, \ldots, M_{n_C}]$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{And}_R(\widehat{a}\sigma C_1[M_1, \ldots, M_{n_{C_1}}], \widehat{b}\sigma C_2[M_{n_{C_1}+1}, \ldots, M_{n_C}], c) \text{ ;}$$

- si $C = \mathsf{And}_L(\widehat{x}\widehat{y}C_1, z)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{And}_L(\widehat{x}\widehat{y}\sigma C_1[M_1, \ldots, M_{n_C}], z) \text{ ;}$$

- si $C = \mathsf{Or}_R(\widehat{a}\widehat{b}C_1, c)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Or}_R(\widehat{a}\widehat{b}\sigma C_1[M_1, \ldots, M_{n_C}], c) \text{ ;}$$

- si $C = \mathsf{Or}_L(\widehat{x}C_1, \widehat{y}C_2, z)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Or}_L(\widehat{x}\sigma C_1[M_1, \ldots, M_{n_{C_1}}], \widehat{y}\sigma C_2[M_{n_{C_1}+1}, \ldots, M_{n_C}], z) \text{ ;}$$

– si $C = \mathsf{Exists}_L(\widehat{x}\widehat{\mathrm{x}}C_1, y)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Exists}_L(\widehat{x}\widehat{\mathrm{x}}\sigma C_1[M_1, \ldots, M_{n_C}], y) \; ;$$

– si $C = \mathsf{Exists}_R(\widehat{a}C_1, \alpha, b)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Exists}_R(\widehat{a}\sigma C_1[M_1, \ldots, M_{n_C}], \sigma\alpha, b) \; ;$$

– si $C = \mathsf{Forall}_R(\widehat{a}\widehat{\mathrm{x}}C_1, b)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Forall}_R(\widehat{a}\widehat{\mathrm{x}}\sigma C_1[M_1, \ldots, M_{n_C}], b) \; ;$$

– si $C = \mathsf{Forall}_L(\widehat{x}C_1, \alpha, y)$, alors $\sigma C[M_1, \ldots, M_{n_C}] =$

$$\mathsf{Forall}_L(\widehat{x}\sigma C_1[M_1, \ldots, M_{n_C}], \sigma\alpha, y) \; ;$$

Une première propriété des termes ouverts est la suivante.

Propriété 6.2.3. *Soit un terme ouvert bien typé $C \triangleright \Gamma \vdash \Delta$ dont les feuilles ouvertes sont $\Box \triangleright \Gamma_i \vdash \Delta_i$ pour $1 \leqslant i \leqslant n_C$. Alors si σ couvre C et si pour tout $1 \leqslant i \leqslant n_C$, $M_i \triangleright \sigma\Gamma_i \vdash \sigma\Delta_i$ est un terme bien typé, alors $\sigma C[M_1, \ldots, M_{n_C}] \triangleright \sigma\Gamma \vdash \sigma\Delta$ est aussi un terme bien typé.*

Démonstration. Par induction sur le terme ouvert C. □

Définissons à présent les termes étendus ainsi que les règles de réduction associés à une règle de prédicats $\mathsf{R} : P \to \varphi$. Pour cela, nous définissons pour toute formule φ, nom x et conom a les termes ouverts $\langle\!|\vdash a : \varphi \,|\!\rangle$ et $\langle\!|\, x : \varphi \vdash |\!\rangle$ en figure 6.3. Tout comme le calcul des nouvelles inférences de la superdéduction, cette définition n'est pas déterministe. Chaque terme ouvert correspondant à un instance de cette définition doit donc être reliée à la règle d'inférence de superdéduction lui correspondant. Prouvons à présent la propriété suivante.

Propriété 6.2.4. *Soit $\mathsf{R} : P \to \varphi$ une règle de prédicats. 1. Soit C le terme ouvert $\langle\!|\vdash a : \varphi \,|\!\rangle$. Considérons une instance d'une règle d'inférence $\mathsf{R}R$ ayant pour conclusion $\Gamma \vdash P, \Delta$. Alors $C \triangleright \Gamma \vdash a : \varphi, \Delta$ est bien typé et il existe σ couvrant C tel que les premisses de σC concordent avec les prémisses de cette instance de $\mathsf{R}R$. 2. Soit C' le terme ouvert $\langle\!|\, x : \varphi \vdash |\!\rangle$. Considérons une instance d'une règle d'inférence $\mathsf{R}L$ ayant pour conclusion $\Gamma, P \vdash \Delta$. Alors $C' \triangleright \Gamma, x : \varphi \vdash \Delta$ est bien typé est il existe σ couvrant C' tel que les premisses de $\sigma C'$ concordent avec les prémisses de cette instance de $\mathsf{R}L$.*

Démonstration. Par construction de C, il existe bien une dérivation de typage (ouverte) de $C \triangleright \Gamma \vdash a : \varphi, \Delta$. En outre, cette dérivation de typage ouverte est une décomposition des connecteurs de φ et concorde donc par ses prémisses avec les instances de l'une des règles de superdeduction $\mathsf{R}R$, modulo remplacement des marques substitutives de C par n'importe quelle substitution σ. Le traitement de C' est tout à fait similaire. □

$$
\begin{aligned}
\langle\!| \Gamma \vdash \Delta |\!\rangle &= \Box \triangleright \Gamma \vdash \Delta \quad \text{si } \Gamma \vdash \Delta \text{ est un séquent atomique} \\
\langle\!| \Gamma, x : \varphi \vdash a : \varphi, \Delta |\!\rangle &= \mathsf{Ax}(x, a) \\
\langle\!| \Gamma \vdash a : \varphi_1 \Rightarrow \varphi_2, \Delta |\!\rangle &= \mathsf{Imp}_R(\widehat{x}\widehat{b}\langle\!| \Gamma, x : \varphi_1 \vdash b : \varphi_2, \Delta |\!\rangle, a) \\
\langle\!| \Gamma, x : \varphi_1 \Rightarrow \varphi_2 \vdash \Delta |\!\rangle &= \mathsf{Imp}_L(\widehat{y}\langle\!| \Gamma, y : \varphi_2 \vdash \Delta |\!\rangle, \widehat{a}\langle\!| \Gamma \vdash a : \varphi_1, \Delta |\!\rangle, x) \\
\langle\!| \Gamma \vdash a : \neg\varphi, \Delta) |\!\rangle &= \mathsf{Not}_R(\widehat{x}\langle\!| \Gamma, x : \varphi \vdash \Delta |\!\rangle, a) \\
\langle\!| \Gamma, x : \neg\varphi \vdash \Delta) |\!\rangle &= \mathsf{Not}_R(\widehat{a}\langle\!| \Gamma \vdash a : \varphi, \Delta |\!\rangle, x) \\
\langle\!| \Gamma \vdash a : \varphi_1 \vee \varphi_2, \Delta |\!\rangle &= \mathsf{Or}_R(\widehat{b}\widehat{c}\langle\!| \Gamma \vdash b : \varphi_1, c : \varphi_2, \Delta |\!\rangle, a) \\
\langle\!| \Gamma, x : \varphi_1 \vee \varphi_2 \vdash \Delta |\!\rangle &= \mathsf{Or}_L(\widehat{y}\langle\!| \Gamma, y : \varphi_1 \vdash \Delta |\!\rangle, \widehat{z}\langle\!| \Gamma, z : \varphi_2 \vdash \Delta |\!\rangle, x) \\
\langle\!| \Gamma \vdash a : \varphi_1 \wedge \varphi_2, \Delta |\!\rangle &= \mathsf{And}_R(\widehat{b}\langle\!| \Gamma \vdash b : \varphi_1, \Delta |\!\rangle, \widehat{c}\langle\!| \Gamma \vdash c : \varphi_2, \Delta |\!\rangle, a) \\
\langle\!| \Gamma, x : \varphi_1 \vee \varphi_2 \vdash \Delta |\!\rangle &= \mathsf{And}_L(\widehat{y}\widehat{z}\langle\!| \Gamma, y : \varphi_1, z : \varphi_2 \vdash \Delta |\!\rangle, x) \\
\langle\!| \Gamma \vdash a : \exists \mathsf{x}.\varphi, \Delta |\!\rangle &= \mathsf{Exists}_R(\widehat{b}\langle\!| \Gamma \vdash b : \varphi[\mathsf{x} := \alpha], \Delta |\!\rangle, \alpha, a) \quad \alpha \text{ est frais} \\
\langle\!| \Gamma, x : \exists \mathsf{x}.\varphi \vdash \Delta |\!\rangle &= \mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}\langle\!| \Gamma, y : \varphi \vdash \Delta |\!\rangle, x) \quad \text{si } \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta) \\
\langle\!| \Gamma \vdash a : \forall \mathsf{x}.\varphi, \Delta |\!\rangle &= \mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}\langle\!| \Gamma \vdash b : \varphi, \Delta |\!\rangle, a) \quad \text{si } \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta) \\
\langle\!| \Gamma, x : \forall \mathsf{x}.\varphi \vdash \Delta |\!\rangle &= \mathsf{Forall}_L(\widehat{y}\langle\!| \Gamma, y : \varphi[\mathsf{x} := \alpha] \vdash \Delta |\!\rangle, \alpha, x) \quad \alpha \text{ est frais}
\end{aligned}
$$

FIG. 6.3 – Definition of $\langle\!| \_ |\!\rangle$

Nous adoptons les règles de typage suivantes permettant d'introduire P à gauche où à droite et correspondant aux règles de superdéduction associées à une règle de prédicat $\mathsf{R} : P \to \varphi$. Les règles permettant d'introduire P à droite sont

$$
\mathsf{RR}\ \frac{\sigma\left(M_i \triangleright \Gamma, x_1^i : A_1^i, \ldots, x_{p_i}^i : A_{p_i}^i \vdash a_1^i : B_1^i, \ldots, a_{q_i}^i : B_{q_i}^i, \Delta \right)_{1 \leqslant i \leqslant n}}{\sigma\mathsf{R}_R\left(\widehat{\mathsf{x}_1} \ldots \widehat{\mathsf{x}_p}, \left(\widehat{x_1^i} \ldots \widehat{x_{p_i}^i}\,\widehat{a_1^i} \ldots \widehat{a_{q_i}^i} M_i\right)_{1 \leqslant i \leqslant n}, \alpha_1, \ldots, \alpha_q, a\right) \triangleright \Gamma \vdash a : P, \Delta}\ \mathcal{C}
$$

où n est le nombre de feuilles ouvertes de $\langle\!| \vdash a : \varphi |\!\rangle$, $\mathcal{C}$ est la condition d'application de la règle de superdéduction correspondante, les variables $\mathsf{x}_1, \ldots, \mathsf{x}_p$ sont celles concernées par $\mathcal{C}$ ainsi que les variables du premier ordre liées dans $\langle\!| \vdash a : \varphi |\!\rangle$, $\alpha_1, \ldots, \alpha_q$ sont les marques substitutives apparaissant dans ce même terme ouvert et σ est une substitution couvrant ce terme. Les règles permettant d'introduire P à gauche sont

$$
\mathsf{RL}\ \frac{\sigma\left(N_j \triangleright \Gamma, y_1^j : C_1^j, \ldots, y_{r_j}^j : C_{r_j}^j \vdash b_1^j : D_1^j, \ldots, b_{s_j}^j : D_{s_j}^j, \Delta \right)_{1 \leqslant j \leqslant m}}{\sigma\mathsf{R}_L\left(\widehat{\mathsf{y}_1} \ldots \widehat{\mathsf{y}_r}, \left(\widehat{y_1^j} \ldots \widehat{y_{r_j}^j}\,\widehat{b_1^j} \ldots \widehat{b_{s_j}^j} N_j\right)_{1 \leqslant j \leqslant m}, \beta_1, \ldots, \beta_s, x\right) \triangleright \Gamma, x : P \vdash \Delta}\ \mathcal{C}'
$$

où m est le nombre de feuilles ouvertes dans $\langle\!| \ x : \varphi \vdash |\!\rangle$, $\mathcal{C}'$ est la condition d'application de la règle de superdéduction correspondante, les variable $\mathsf{y}_1, \ldots, \mathsf{y}_r$ sont celles concernées par $\mathcal{C}'$ ainsi que les variables du premier ordre liées dans $\langle\!| \ x : \varphi \vdash |\!\rangle$, $\beta_1, \ldots, \beta_s$ sont les marques substitutives apparaissant dans ce même terme ouvert et σ est une substitution couvrant ce terme. Par dualité, $p = s$ et $q = r$.

Si $\mathcal{R}$ est un ensemble de règles de prédicats, en rajoutant les règles de typage correspondant à chacune des règles de $\mathcal{R}$, nous obtenons un langage de termes de

preuve étendu pour le système de superdéduction associé à $\mathcal{R}$. Les définitions de la sous-section 3.3.2, comme par exemple, la définition de symboles libres et liés de la figure 3.6, la définition de remplacement des figures 3.7 et 3.8, la définition d'alpha-conversion de la figure 3.9, la définition 3.3.1 de position et la définition 3.3.2 d'introduction fraîche sont étendues au langage des termes de preuve étendu. Les susbtitutions sur les termes de preuve définies par les figures 3.12 et 3.13 sont aussi étendues aux termes de preuve étendus. L'élimination des coupure de la sous-section 3.3.2 sera notée ici $\xrightarrow{\mathsf{cut}}$. Nous l'étendons en une élimination des coupures $\xrightarrow{\mathsf{excut}}$ pour les termes de preuve étendus comme suit. Pour chaque règle de prédicats $\mathsf{R} \ : \ P \to \varphi$, pour chaque réduction

$$\mathsf{Cut}(\widehat{a}\langle\!|\vdash a : \varphi \,|\!\rangle, \widehat{x}\langle\!| \, x : \varphi \vdash |\!\rangle) \xrightarrow{\mathsf{cut}}{}^{+} C$$

où C est une forme normale pour $\xrightarrow{\mathsf{cut}}$, nous rajoutons à la relation $\xrightarrow{\mathsf{cut}}$ la règle de réécriture

$$\sigma\mathsf{Cut}\Bigg(\widehat{a}\mathsf{R}_R\left(\widehat{\mathsf{x}_1}\dots\widehat{\mathsf{x}_p}, \left(\widehat{x_1^i}\dots\widehat{x_{p_i}^i}\,\widehat{a_1^i}\dots\widehat{a_{q_i}^i}M_i\right)_{1\leqslant i\leqslant n}, \alpha_1\dots\alpha_q, a\right),$$
$$\widehat{x}\mathsf{R}_L\left(\widehat{\mathsf{y}_1}\dots\widehat{\mathsf{y}_r}, \left(\widehat{y_1^j}\dots\widehat{y_{r_j}^j}\,\widehat{b_1^j}\dots\widehat{b_{s_j}^j}N_j\right)_{1\leqslant j\leqslant m}, \beta_1\dots\beta_s, x\right)\Bigg)$$
$$\xrightarrow{\mathsf{excut}} \quad \sigma C[M_1,\dots,N_m]$$

conditionnée par le fait que $\mathsf{R}_R(\dots)$ et $\mathsf{R}_L(\dots)$ doivent introduire fraîchement a et x. Ici σ est une substitution couvrant C. L'élimination des coupures $\xrightarrow{\mathsf{excut}}$ obtenue est *complète* : toute instance de la règle Cut est un rédexe et par conséquent, toute forme normale est sans coupure. La *subject reduction* est une conséquence des propriétés 6.2.3 et 6.2.4.

Propriété 6.2.5 (Subject Reduction). *Si* $M \xrightarrow{\mathsf{excut}^*} M'$ *et* $M \triangleright \Gamma \vdash \Delta$ *est bien typé, alors* $M' \triangleright \Gamma \vdash \Delta$ *est bien typé.*

Normalisation forte

Afin d'obtenir la normalisation forte de l'élimination des coupures sur les termes de preuve étendus correspondant à un ensemble $\mathcal{R}$ de règles de prédicats, nous définissons une relation de réécriture $\xrightarrow{\mathsf{term}}$ sur les termes de preuve comme suit : à chaque règle de prédicats $\mathsf{R} \ : \ P \to \varphi$ dans $\mathcal{R}$ correspondent les règles de réécriture

$$\sigma\mathsf{R}_R\left(\widehat{\mathsf{x}_1}\dots\widehat{\mathsf{x}_p}, \left(\widehat{x_1^i}\dots\widehat{x_{p_i}^i}\,\widehat{a_1^i}\dots\widehat{a_{q_i}^i}M_i\right)_{1\leqslant i\leqslant n}, \alpha_1\dots\alpha_q, a\right)$$
$$\xrightarrow{\mathsf{term}} \quad \sigma\langle\!|\vdash a : \varphi \,|\!\rangle[M_1,\dots,M_n]$$

où σ est une substitution pour les marques substitutives couvrant $\langle\!|\vdash a : \varphi \,|\!\rangle$ et les symboles liés de ce dernier terme ouvert sont supposés différents des symboles

libres de $\mathsf{R}_R(\dots)$ ainsi que les règles

$$\sigma\mathsf{R}_L\left(\widehat{\mathsf{y}_1}\dots\widehat{\mathsf{y}_r}, \left(\widehat{y_1^j}\dots\widehat{y_{r_j}^j}\,\widehat{b_1^j}\dots\widehat{b_{s_j}^j}\,N_j\right)_{1\leqslant j\leqslant m}, \beta_1\dots\beta_s, x\right) \xrightarrow{\mathsf{term}} \sigma\langle\!|\, x : \varphi \vdash |\!\rangle[N_1,\dots,N_m]$$

où σ est une substitution pour les marques substitutives couvrant $\langle\!|\, x : \varphi \vdash |\!\rangle$ et les symboles liés de ce dernier terme ouvert sont supposés différents des symboles libres de $\mathsf{R}_L(\dots)$.

Notons que puisque $\xrightarrow{\mathsf{term}}$ est orthogonal, il est confluent. Par conséquent s'il est faiblement normalisant, alors l'unique forme normale de tout terme de preuve M sera notée $M\downarrow^{\mathsf{t}}$. Notons que l'ensemble des règles de prédicats $\mathcal{R}$ définit une relation de réécriture sur les formules que nous noterons $\xrightarrow{\mathsf{prop}}$. Si celle-ci est confluente et faiblement normalisante, l'unique forme normale de toute formule φ sera similairement notée $\varphi\downarrow^{\mathsf{p}}$. Nous utiliserons aussi cette notation pour les contextes, les séquents et les termes ouverts puisque ceux-ci contiennent effectivement des séquents de par le constructeur $\Box \triangleright \Gamma \vdash \Delta$.

Nous allons à présent démontrer que $\xrightarrow{\mathsf{excut}}$ est fortement normalisante lorsque l'hypothèse suivante est vérifiée.

Hypothèse 6.2.1. *1. La relation de réécriture $\xrightarrow{\mathsf{prop}}$ associée à l'ensemble $\mathcal{R}$ de règles de prédicats est faiblement normalisante et confluente et 2. pour chaque règle $\mathsf{R} : P \to \varphi$ de $\mathcal{R}$, P ne contient que des variables du premier ordre (c'est-à-dire aucun symbole de fonction et aucune constante) et $\mathcal{FV}(\varphi) \subseteq \mathcal{FV}(P)$.*

La partie 2 de l'hypothèse 6.2.1 restreint l'usage des variables du premier ordre afin d'éviter les contre-exemples terminants comme ceux que nous avons décrits en section 5.3.1. L'hypothèse 6.2.1 nous permet de démontrer la normalisation forte de $\xrightarrow{\mathsf{excut}}$.

Propriété 6.2.6 (Normalisation forte de $\xrightarrow{\mathsf{excut}}$ (Brauner *et al.*, 2007a)). *Lorsque $\mathcal{R}$ satisfait l'hypothèse 6.2.1, alors l'élimination des coupures sur les termes étendus $\xrightarrow{\mathsf{excut}}$ est fortement normalisante sur les termes étendus bien typés.*

Démonstration. Tout d'abord puisque $\xrightarrow{\mathsf{prop}}$ est faiblement normalisant et confluent, alors $\xrightarrow{\mathsf{term}}$ l'est aussi et si $M \triangleright \Gamma \vdash \Delta$ et un terme bien typé, alors $M\downarrow^{\mathsf{t}} \triangleright \Gamma\downarrow^{\mathsf{p}} \vdash \Delta\downarrow^{\mathsf{p}}$ est aussi un terme bien typé. En outre si $M \xrightarrow{\mathsf{excut}} M'$, alors $M\downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}{}^{+} M'\downarrow^{\mathsf{t}}$. Alors la normalisation forte de $\xrightarrow{\mathsf{excut}}$ découle de la normalisation forte de $\xrightarrow{\mathsf{cut}}$. □

Notons que l'hypothèse 6.2.1 implique alors l'admissibilité des coupures dans le système de superdéduction sans coupure correspondant. Puisque nous avons démontré que ce système était correct et complet, on peut en déduire la cohérence de la théorie associée à l'ensemble de règles de prédicats.

Contre-exemple de Crabbé

Notre langage de termes étendu permet aussi de formaliser des contre-exemples à la normalisation. Les termes ouverts associés à la règle de prédicats $\mathsf{R} \ : \ A \rightarrow B \wedge \neg A$ sont

$$\langle\!|\vdash a : B \wedge \neg A \,|\!\rangle = \mathsf{And}_R(\widehat{b}(\square\triangleright \vdash b : B), \widehat{c}\mathsf{Not}_R(\widehat{x}(\square \triangleright x : A \vdash), c), a)$$

et

$$\langle\!| \ x : B \wedge \neg A \vdash |\!\rangle = \mathsf{And}_L(\widehat{y}\widehat{z}\mathsf{Not}_L(\widehat{a}(\square \triangleright y : B \vdash a : A), z), x) \ .$$

Leurs dérivations de typage sont

$$\wedge\mathrm{R}\ \frac{\dfrac{}{(\square\triangleright \vdash b : B)\triangleright \vdash b : B} \qquad \neg\mathrm{R}\ \dfrac{\dfrac{}{(\square \triangleright x : A \vdash) \triangleright x : A \vdash}}{\mathsf{Not}_R(\widehat{x}(\square \triangleright x : A \vdash), c)\triangleright \vdash \neg c : A}}{\mathsf{And}_R(\widehat{b}(\square\triangleright \vdash b : B), \widehat{c}\mathsf{Not}_R(\widehat{x}(\square \triangleright x : A \vdash), c), a)\triangleright \vdash a : B \wedge \neg A}$$

et

$$\wedge\mathrm{L}\ \frac{\neg\mathrm{L}\ \dfrac{\dfrac{}{(\square \triangleright y : B \vdash a : A) \triangleright \Gamma, y : B \vdash a : A, \Delta}}{\mathsf{Not}_L(\widehat{a}(\square \triangleright y : B \vdash a : A), z) \triangleright \Gamma, y : B, z : \neg A \vdash \Delta}}{\mathsf{And}_L(\widehat{y}\widehat{z}\mathsf{Not}_L(\widehat{a}(\square \triangleright y : B \vdash a : A), z), x) \triangleright \Gamma, x : B \wedge \neg A \vdash \Delta} \ .$$

Elles induisent les règles de typage

$$\mathsf{RR}\ \frac{M_1 \triangleright \Gamma \vdash b : B, \Delta \qquad M_2 \triangleright \Gamma, x : A \vdash \Delta}{\mathsf{R}_R(\widehat{b}M_1, \widehat{x}M_2, a) \triangleright \Gamma \vdash a : A, \Delta}$$

et

$$\mathsf{RL}\ \frac{M \triangleright \Gamma, y : B \vdash a : A, \Delta}{\mathsf{R}_L(\widehat{y}\widehat{a}M, x)\Gamma, x : A \vdash \Delta} \ .$$

La réduction

$$\begin{array}{ll} & \mathsf{Cut}(\widehat{a}\mathsf{And}_R(\widehat{b}M_1, \widehat{c}\mathsf{Not}_R(\widehat{x}M_2, c), a), \widehat{x'}\mathsf{And}_L(\widehat{y'}\widehat{z'}\mathsf{Not}_L(\widehat{a'}M, z'), x')) \\ \xrightarrow{\mathsf{cut}} & \quad \mathsf{Cut}(\widehat{b}M_1, \widehat{y'}\mathsf{Cut}(\widehat{c}\mathsf{Not}_R(\widehat{x}M_2, c), \widehat{z'}\mathsf{Not}_L(\widehat{a'}M, z'))) \\ \xrightarrow{\mathsf{cut}} & \quad \mathsf{Cut}(\widehat{b}M_1, \widehat{y'}\mathsf{Cut}(\widehat{a'}M, \widehat{x}M_2)) \end{array}$$

induit la réduction

$$\begin{array}{ll} & \mathsf{Cut}(\widehat{a}\mathsf{R}_R(\widehat{b}M_1, \widehat{x}\widehat{b}M_2, a), \widehat{x'}\mathsf{R}_L(\widehat{y'}\widehat{a'}M, x')) \\ \xrightarrow{\mathsf{excut}} & \mathsf{Cut}(\widehat{b}M_1, \widehat{y'}\mathsf{Cut}(\widehat{a'}M, \widehat{x}M_2)) \end{array}$$

conditionnée de la façon suivante : les termes $\mathsf{R}_R(\widehat{b}M_1, \widehat{x}\widehat{b}M_2, a)$ et $\mathsf{R}_L(\widehat{y'}\widehat{a'}M, x')$ doivent respectivement introduire fraîchement les variables a et x', c'est-à-dire que a et x' ne doivent pas apparaître libres respectivement dans M_1 ou M_2 et dans M.

Notons à présent δ le terme $\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)$ et Δ le terme $\mathsf{R}_R(\widehat{b}\mathsf{Ax}(z,b),\widehat{x}\delta,c)$. Ils sont tous les deux bien typés comme le montre la dérivation

$$\dfrac{\dfrac{}{\mathsf{Ax}(z,b)\triangleright z:B\vdash b:B}\ \textsc{Axiom}\quad \dfrac{\dfrac{}{\mathsf{Ax}(x,a)\triangleright x:A,y:B\vdash a:A}\ \textsc{Axiom}}{\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)\triangleright x:A\vdash}\ \mathsf{RL}}{\mathsf{R}_R(\widehat{b}\mathsf{Ax}(z,b),\widehat{x}\delta,c)\triangleright z:B\vdash c:A}\ \mathsf{RR}$$

Pourtant la réduction suivante ne termine pas.

$$\begin{array}{rll}
\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\delta) = & \mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)) & \\
& (\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)\text{ n'introduit pas fraîchement }x) & \\
\to & \mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)[x:=\widehat{c}\Delta] & \\
= & \mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a)[x:=\widehat{c}\Delta],x)) & \\
= & \mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\Delta[c\mapsto a],a)) & \\
\overset{\alpha}{=} & \mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{c}\Delta,a)) & \\
= & \mathsf{Cut}(\widehat{c}\mathsf{R}_R(\widehat{b}\mathsf{Ax}(z,b),\widehat{x}\delta,c),\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{c}\Delta,a)) & \\
\to & \mathsf{Cut}(\widehat{b}\mathsf{Ax}(z,b),\widehat{y}\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\delta)) & \\
& (\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\delta)\text{ n'introduit par fraîchement }y) & \\
\to & \mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\delta)[y:=\widehat{b}\mathsf{Ax}(z,b)] & \\
= & \mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\delta) & \to\quad\ldots
\end{array}$$

Encore une fois, comme en déduction modulo (section 5.3.1) et en déduction supernaturelle (début de la présente section), cette règle de prédicats (qui ne satisfait pas l'hypothèse 6.2.1) casse la normalisation.

Traduction par normalisation

Nous avons vu que les nouvelles inférences des systèmes de déduction supernaturelle ou des systèmes de superdéduction sont calculées à partir d'inférences de NJ ou de LK (propriétés 6.1.1 et 6.1.3). Remplacer ces nouvelles inférences par le calcul qu'elles représentent nous a déjà permis de démontrer la correction de ces systèmes (propriétés 6.1.2 et 6.1.4). Ce remplacement nous a aussi permis de définir une traduction des systèmes de déduction supernaturelle et de superdéduction vers la déduction modulo (sous-section 6.2.1) ou encore vers le calcul d'Urban (c'est-à-dire LK) pour obtenir des résultats d'admissibilité des coupures (propriété 6.2.1) et de normalisation (propriétés 6.2.2 et 6.2.6).

Curieusement, la normalisation de l'élimination des coupures en superdéduction nous permet d'écrire une traduction inverse remplaçant alors des inférences en déduction modulo ou dans LK par des inférences en superdéduction utilisant directement les nouvelles inférences de la superdéduction. Contrairement au premier, ce remplacement est loin d'être une triviale expansion des nouvelles inférences : lorsque l'on considère une règle de prédicats $P\to\varphi$, il s'agit de remplacer dans une dérivation un nuage d'inférences concernant φ et dispersé dans toute la preuve

par une seule inférence concernant P localisée à un seul et même endroit. En quoi la normalisation contient-elle alors une telle traduction ? Considérons par exemple la démonstration de la propriété 6.1.5. Pour tout ensemble de règles de prédicats correspondant à une théorie $\mathcal{T}h_{\mathcal{R}}$ et pour tout séquent $\Gamma \vdash \Delta$, elle nous permet de traduire toute dérivation de $\Gamma \vdash \Delta$ dans LK en une dérivation de $\Gamma, \mathcal{T}h_{\mathcal{R}} \vdash \Delta$ dans le système de superdéduction associé à $\mathcal{R}$. Cette traduction n'effectue pas le remplacement mais au contraire utilise une dérivation de $\vdash \mathcal{T}h_{\mathcal{R}}$ en superdéduction qui à travers la coupure

$$\textsc{Cut}\ \dfrac{\overset{\text{SUPERDÉDUCTION}}{\vdash \mathcal{T}h_{\mathcal{R}}} \qquad \overset{\text{LK}}{\Gamma, \mathcal{T}h_{\mathcal{R}} \vdash \Delta}}{\Gamma \vdash \Delta}$$

fournit une dérivation de $\Gamma \vdash \Delta$ à partir de celle de $\Gamma, \mathcal{T}h_{\mathcal{R}} \vdash \Delta$ *sans en modifier la structure*. L'élimination des coupures appliquée à cette dérivation renvoit une dérivation sans coupure de $\Gamma \vdash \Delta$ *en superdéduction*. La structure de la dérivation (dans LK) de départ est modifiée puisque l'utilisation des nouvelles règles d'inférence dans la dérivation de $\vdash \mathcal{T}h_{\mathcal{R}}$ est propagée par l'élimination des coupures à l'intérieur de la dérivation de $\Gamma \vdash \Delta$. L'élimination des coupures nous permet donc d'écrire une traduction des dérivations de LK (ou en déduction modulo) en dérivations en superdéduction.

Reprenons par exemple l'exemple de l'inclusion encodé par la règle de prédicats $\textsc{Inc} : x \subseteq y \longrightarrow \forall z.(z \in x \Rightarrow z \in y)$. Elle induit les règles de superdéduction

$$\textsc{IncR}\ \dfrac{\Gamma, z \in t_1 \vdash z \in t_2, \Delta}{\Gamma \vdash t_1 \subseteq t_2, \Delta}\ z \notin \mathcal{FV}(\Gamma, \Delta) \quad \text{et} \quad \textsc{IncL}\ \dfrac{\Gamma, t \in t_2 \vdash \Delta \quad \Gamma \vdash t \in t_1, \Delta}{\Gamma, t_1 \subseteq t_2 \vdash \Delta}\ .$$

Une coupure

$$\textsc{Cut}\ \dfrac{\textsc{IncR}\ \dfrac{\overset{\pi_1}{\Gamma, z \in t_1 \vdash z \in t_2, \Delta}}{\Gamma \vdash t_1 \subseteq t_2, \Delta} \qquad \textsc{IncL}\ \dfrac{\overset{\pi_2}{\Gamma, t \in t_2 \vdash \Delta} \quad \overset{\pi_3}{\Gamma \vdash t \in t_1, \Delta}}{\Gamma, t_1 \subseteq t_2 \vdash \Delta}}{\Gamma \vdash \Delta}$$

se réduit en

$$\textsc{Cut}\ \dfrac{\overset{\pi_3}{\Gamma \vdash t \in t_1, \Delta} \qquad \textsc{Cut}\ \dfrac{\overset{\pi_1[t/z]}{\Gamma, t \in t_1 \vdash t \in t_2, \Delta} \qquad \overset{\pi_2}{\Gamma, t \in t_2 \vdash \Delta}}{\Gamma, t \in t_1 \vdash \Delta}}{\Gamma \vdash \Delta}$$

et en

$$\textsc{Cut}\ \dfrac{\textsc{Cut}\ \dfrac{\overset{\pi_3}{\Gamma \vdash t \in t_1, \Delta} \qquad \overset{\pi_1[t/z]}{\Gamma, t \in t_1 \vdash t \in t_2, \Delta}}{\Gamma \vdash t \in t_2, \Delta} \qquad \overset{\pi_2}{\Gamma, t \in t_2 \vdash \Delta}}{\Gamma \vdash \Delta}\ .$$

Les inférences IncR et IncL sont traduites en langage d'Urban étendu par des constructions $\textsc{Inc}_R(\widehat{x}\widehat{b}\widehat{\mathsf{z}}M, a)$ et $\textsc{Inc}_L(\widehat{y}M_1, \widehat{a}M_2, \mathsf{t}, x)$ ainsi que par la réduction

$$\begin{array}{l}\mathsf{Cut}(\widehat{a}\textsc{Inc}_R(\widehat{x}\widehat{b}\widehat{\mathsf{z}}M, a), \widehat{x}\textsc{Inc}_L(\widehat{y}M_1, \widehat{a}M_2, \mathsf{t}, x)) \\ \quad \xrightarrow{\mathsf{excut}} \quad \left\{\begin{array}{l}\mathsf{Cut}(\widehat{a}M_2, \widehat{x}\mathsf{Cut}(\widehat{b}M[\mathsf{t}/\mathsf{z}], \widehat{y}M_1)) \\ \mathsf{Cut}(\widehat{b}\mathsf{Cut}(\widehat{a}M_2, \widehat{x}M[\mathsf{t}/\mathsf{z}]), \widehat{y}M_1))\end{array}\right. .\end{array}$$

On peut démontrer tout d'abord dans LK le séquent

$$\forall\mathsf{x}.\forall\mathsf{y}.(\mathsf{x} \subseteq \mathsf{y}) \Leftrightarrow (\forall\mathsf{z}.\mathsf{z} \in \mathsf{x} \Rightarrow \mathsf{z} \in \mathsf{y}) \vdash \mathsf{t} \subseteq \mathsf{t} \quad :$$

une dérivation possible (minimale) correspond au terme

$$\begin{array}{l}\mathsf{Forall}_L(\widehat{x_2}\mathsf{Forall}_L(\widehat{x_3}\mathsf{And}_L(\widehat{x_4}\widehat{x_5}\mathsf{Imp}_L(\widehat{x_6}\mathsf{Ax}(x_6, a), \\ \quad \widehat{b}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7, d), c), b), x_5), x_3), \mathsf{t}, x_2), \mathsf{t}, x_1)\end{array}$$

que nous noterons δ. Notre objectif est à présent de traduire cette dérivation en superdéduction. Pour cela, commençons pas démontrer l'axiome correspondant à Inc dans le système de superdéduction lui correspondant : une dérivation possible est celle représentée par le terme

$$\begin{array}{l}\mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{y}}\mathsf{And}_R(\\ \quad \widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z, g), \widehat{h}\mathsf{Ax}(y, h), \mathsf{z}, x), f), e), d), \\ \quad \widehat{i}\mathsf{Imp}_R(\widehat{x'}\widehat{j}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Forall}_L(\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'', k), \widehat{l}\mathsf{Ax}(y', l), z')\mathsf{z}, x'), j), i), \\ c), b), a)\end{array}$$

que nous noterons δ_{Inc} et qui correspond bien à une dérivation de $\vdash \forall\mathsf{x}.\forall\mathsf{y}.(\mathsf{x} \subseteq \mathsf{y}) \Leftrightarrow (\forall\mathsf{z}.\mathsf{z} \in \mathsf{x} \Rightarrow \mathsf{z} \in \mathsf{y})$ (c'est-à-dire $\vdash \mathcal{T}h_{\text{Inc}}$) dans le système de superdéduction associé à Inc. Si l'on combine δ et δ_{Inc} dans une coupure, on obtient bel et bien une dérivation $\mathsf{Cut}(\widehat{a}\delta_{\text{Inc}}, \widehat{x_1}\delta)$ du séquent $\vdash \mathsf{t} \subseteq \mathsf{t}$. Nous n'avons pas encore obtenu une *traduction* de δ. Celle-ci est obtenu par la normalisation de cette coupure comme suit. Les deux première réductions du terme

$$\begin{array}{l}\mathsf{Cut}(\widehat{a}\mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{y}}\mathsf{And}_R(\\ \qquad \widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z, g), \widehat{h}\mathsf{Ax}(y, h), \mathsf{z}, x), f), e), d), \\ \qquad \widehat{i}\mathsf{Imp}_R(\widehat{x'}\widehat{j}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Forall}_L(\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'', k), \widehat{l}\mathsf{Ax}(y', l), z')\mathsf{z}, x'), j), i), \\ \quad c), b), a), \widehat{x_1}\mathsf{Forall}_L(\widehat{x_2}\mathsf{Forall}_L(\widehat{x_3}\mathsf{And}_L(\widehat{x_4}\widehat{x_5}\mathsf{Imp}_L(\widehat{x_6}\mathsf{Ax}(x_6, a), \\ \qquad \widehat{b}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7, d), c), b), x_5), x_3), \mathsf{t}, x_2), \mathsf{t}, x_1) \quad)\end{array}$$

correspondent à des coupures logiques et retournent la dérivation

$$\begin{array}{l}\mathsf{Cut}(\widehat{c}\mathsf{And}_R(\\ \qquad \widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z, g), \widehat{h}\mathsf{Ax}(y, h), \mathsf{z}, x), f), e), d), \\ \qquad \widehat{i}\mathsf{Imp}_R(\widehat{x'}\widehat{j}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Forall}_L(\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'', k), \widehat{l}\mathsf{Ax}(y', l), z')\mathsf{z}, x'), j), i), \\ \quad c), \widehat{x_3}\mathsf{And}_L(\widehat{x_4}\widehat{x_5}\mathsf{Imp}_L(\widehat{x_6}\mathsf{Ax}(x_6, a), \\ \qquad \widehat{b}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7, d), c), b), x_5), x_3) \quad) \ .\end{array}$$

La coupure obtenue est encore une fois une coupure logique mais peut se réduire de deux manières différentes. Nous choisissons de la réduire en

$$\begin{array}{l}\mathsf{Cut}(\widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z,g),\widehat{h}\mathsf{Ax}(y,h),\mathsf{z},x),f),e),d),\\ \quad \widehat{x_4}\mathsf{Cut}(\widehat{i}\mathsf{Imp}_R(\widehat{x'}\widehat{j}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Forall}_L(\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'',k),\\ \qquad\qquad \widehat{l}\mathsf{Ax}(y',l),z')\mathsf{z},x'),j),i),\\ \quad \widehat{x_5}\mathsf{Imp}_L(\widehat{x_6}\mathsf{Ax}(x_6,a),\widehat{b}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7,d),c),b),x_5)\quad)\quad)\end{array}$$

dont la coupure interne est encore une coupure logique. On la réduit à son tour pour obtenir

$$\begin{array}{l}\mathsf{Cut}(\widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z,g),\widehat{h}\mathsf{Ax}(y,h),\mathsf{z},x),f),e),d),\\ \quad \widehat{x_4}\mathsf{Cut}(\widehat{b}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7,d),c),b),\widehat{x'}\mathsf{Cut}(\\ \qquad \widehat{j}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Forall}_L(\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'',k),\widehat{l}\mathsf{Ax}(y',l),z')\mathsf{z},x'),j),\\ \qquad \widehat{x_6}\mathsf{Ax}(x_6,a)\quad)\quad)\quad)\ .\end{array}$$

La coupure la plus profonde permet alors d'obtenir le réduit

$$\begin{array}{l}\mathsf{Cut}(\widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z,g),\widehat{h}\mathsf{Ax}(y,h),\mathsf{z},x),f),e),d),\\ \quad \widehat{x_4}\mathsf{Cut}(\widehat{b}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7,d),c),b),\\ \quad \widehat{x'}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Forall}_L(\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'',k),\widehat{l}\mathsf{Ax}(y',l),z')\mathsf{z},x'),a)\quad)\ )\ )\ .\end{array}$$

La coupure intérieure est à présent une coupure commutative qui se réduit en

$$\begin{array}{l}\mathsf{Cut}(\widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z,g),\widehat{h}\mathsf{Ax}(y,h),\mathsf{z},x),f),e),d),\\ \quad \widehat{x_4}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Cut}(\widehat{b}\mathsf{Forall}_R(\widehat{c}\widehat{\mathsf{z}'}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7,d),c),b),\\ \qquad \widehat{x'}\mathsf{Forall}_L(\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'',k),\widehat{l}\mathsf{Ax}(y',l),z')\mathsf{z},x')\quad)\quad a)\quad)\ )\end{array}$$

devenant alors une coupure logique se réduisant successivement en

$$\begin{array}{l}\mathsf{Cut}(\widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z,g),\widehat{h}\mathsf{Ax}(y,h),\mathsf{z},x),f),e),d),\\ \quad \widehat{x_4}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Cut}(\widehat{c}\mathsf{Imp}_R(\widehat{x_7}\widehat{d}\mathsf{Ax}(x_7,d),c),\widehat{z'}\mathsf{Imp}_L(\widehat{x''}\mathsf{Ax}(x'',k),\widehat{l}\mathsf{Ax}(y',l),\\ \qquad z')\quad)\quad a)\quad)\quad)\end{array}$$

puis en

$$\begin{array}{l}\mathsf{Cut}(\widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z,g),\widehat{h}\mathsf{Ax}(y,h),\mathsf{z},x),f),e),d),\\ \quad \widehat{x_4}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Cut}(\widehat{l}\mathsf{Ax}(y',l),\widehat{x7}\mathsf{Cut}(\widehat{d}\mathsf{Ax}(x_7,d),\widehat{x''}\mathsf{Ax}(x'',k)\ )\ )\ a)\ )\ .\end{array}$$

Deux réductions à l'intérieur du constructeur $\textsc{Inc}_L$ retournent le terme

$$\begin{array}{l}\mathsf{Cut}(\widehat{d}\mathsf{Imp}_R(\widehat{x}\widehat{e}\mathsf{Forall}_R(\widehat{f}\widehat{\mathsf{z}}\mathsf{Imp}_R(\widehat{y}\widehat{g}\textsc{Inc}_L(\widehat{z}\mathsf{Ax}(z,g),\widehat{h}\mathsf{Ax}(y,h),\mathsf{z},x),f),e),d),\\ \quad \widehat{x_4}\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Ax}(y',k)\ a)\ )\ .\end{array}$$

Finalement remarquons que le symbole x_4 n'apparaît pas dans le terme de droite de cette coupure. Par conséquent cette coupure commutative se réduit en

$$\textsc{Inc}_R(\widehat{y'}\widehat{k}\widehat{\mathsf{z}}\mathsf{Ax}(y',k)a)$$

qui est un terme en forme normale représentant la dérivation

$$\textsc{IncR}\ \dfrac{\textsc{Ax}\ \dfrac{}{\mathsf{z} \in \mathsf{t} \vdash \mathsf{z} \in \mathsf{t}}}{\vdash \mathsf{t} \subseteq \mathsf{t}}\ .$$

Nous avons donc obtenu cette nouvelle dérivation en superdéduction à partir de la dérivation δ dans LK, c'est-à-dire la dérivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\textsc{Axiom}\ \dfrac{}{\ldots, \mathsf{t} \subseteq \mathsf{t} \vdash \mathsf{t} \subseteq \mathsf{t}} \quad \forall\textsc{R}\ \dfrac{\Rightarrow\textsc{R}\ \dfrac{\textsc{Axiom}\ \dfrac{}{\ldots, \mathsf{z} \in \mathsf{t} \vdash \mathsf{z} \in \mathsf{t}, \mathsf{t} \subseteq \mathsf{t}}}{\cdots \vdash \mathsf{z} \in \mathsf{t} \Rightarrow \mathsf{z} \in \mathsf{t}, \mathsf{t} \subseteq \mathsf{t}}}{\cdots \vdash \forall \mathsf{z}.(\mathsf{z} \in \mathsf{t} \Rightarrow \mathsf{z} \in \mathsf{t}), \mathsf{t} \subseteq \mathsf{t}}}{\ldots, \mathsf{t} \subseteq \mathsf{t} \vdash \mathsf{t} \subseteq \mathsf{t}}}{\ldots, \forall(\mathsf{z}.\mathsf{z} \in \mathsf{t} \Rightarrow \mathsf{z} \in \mathsf{t}) \Rightarrow (\mathsf{t} \subseteq \mathsf{t}) \vdash \mathsf{t} \subseteq \mathsf{t}}}{(\mathsf{t} \subseteq \mathsf{t}) \Leftrightarrow \forall \mathsf{z}.(\mathsf{z} \in \mathsf{t} \Rightarrow \mathsf{z} \in \mathsf{t}) \vdash \mathsf{t} \subseteq \mathsf{t}}}{\forall \mathsf{y}.(\mathsf{t} \subseteq \mathsf{y}) \Leftrightarrow \forall \mathsf{z}.(\mathsf{z} \in \mathsf{t} \Rightarrow \mathsf{z} \in \mathsf{y}) \vdash \mathsf{t} \subseteq \mathsf{t}}}{\forall \mathsf{x}.\forall \mathsf{y}.(\mathsf{x} \subseteq \mathsf{y}) \Leftrightarrow \forall \mathsf{z}.(\mathsf{z} \in \mathsf{x} \Rightarrow \mathsf{z} \in \mathsf{y}) \vdash \mathsf{t} \subseteq \mathsf{t}}$$

en utilisant la dérivation δ_{Inc} en superdéduction de $\vdash \forall \mathsf{x}.\forall \mathsf{y}.(\mathsf{x} \subseteq \mathsf{y}) \Leftrightarrow (\forall \mathsf{z}.\mathsf{z} \in \mathsf{x} \Rightarrow \mathsf{z} \in \mathsf{y})$ comme un *traducteur* que l'on applique grâce à la règle de coupure et dont le calcul correspond à l'élimination de cette coupure.

6.3 Superdéduction modulo

La déduction modulo et la superdéduction sont deux paradigmes qui sont *duaux* dans le sens que le premier permet la manipulation de la partie calculatoire inhérente à une théorie donnée alors que le deuxième permet la manipulation de la partie déductive inhérente à la même théorie. Ces deux paradigmes *orthogonaux* peuvent donc être combinés pour obtenir des systèmes de superdéduction modulo.

Définition 6.3.1 (Superdéduction modulo). *Soient $\mathcal{R}_1$ et $\mathcal{R}_2$ respectivement un ensemble de règles de prédicats et de termes et un ensemble de règles de prédicats. Le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$ est formé 1. des règles de LK, 2. des règles de superdéduction obtenue à partir de $\mathcal{T}h_2$ ainsi que 3. des règles $\textsc{ConvR}_{\equiv_1}$ et $\textsc{ConvL}_{\equiv_1}$ où $\equiv_1$ est la congruence engendrée par $\mathcal{R}_1$.*

Si $\mathcal{R}_2$ est vide, alors le système est le calcul des séquents classique modulo associé à $\mathcal{R}_1$; si $\mathcal{R}_1$ est vide, alors le système est le système de superdéduction associé à $\mathcal{R}_2$. Nous avons choisi d'utiliser les règles ConvR et ConvL pour définir les systèmes de superdéduction modulo mais il est aussi possible de modifier directement les règles du système de superdéduction associé à $\mathcal{R}_2$ pour qu'elles contiennent directement la conversion par la congruence $\equiv_1$. Par exemple l'inférence

$$\textsc{IncR}\ \dfrac{\Gamma, \mathsf{z} \in \mathsf{t}_1 \vdash \mathsf{z} \in \mathsf{t}_2, \Delta}{\Gamma \vdash \mathsf{t}_1 \subseteq \mathsf{t}_2, \Delta}$$

correspondant à la règle de prédicat $(x \subseteq y) \longrightarrow \forall z(z \in x \Rightarrow z \in y)$ combinée à ConvR devient

$$\textsc{IncR}\ \frac{\Gamma, z \in t_1 \vdash z \in t_2, \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv_1 t_1 \subseteq t_2$$

Si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont des ensembles de règles définissant un système de superdéduction modulo $\mathcal{R}_1$ et $\mathcal{R}_2$ peuvent tous les deux contenir des règles de prédicats. Tout comme dans le cadre de la déduction modulo, il faut éviter les situations égalisant des connecteurs distincts. Toutefois, notons que dans le cadre de la superdéduction modulo, ce n'est pas la congruence engendrée par $\mathcal{R}_1$ seul qui doit satisfaire la propriété de protection des connecteurs (propriété 5.1.1) mais celle engendrée par $\mathcal{R}_1 \cup \mathcal{R}_2$. Nous supposerons donc ici que cette congruence satisfait bien la propriété de protection des connecteurs.

Remarquons que dans la combinaison *superdéduction modulo*, la partie *modulo* n'intervient pas dans le calcul des nouvelles inférences de superdéduction. Nous aurions pu choisir de calculer celles-ci en utilisant des règles du calcul des séquents *modulo*. On obtient un calcul des nouvelles inférences de superdéduction déterministe si $\mathcal{R}_1$ vérifie la propriété de protection des connecteurs (ce qui est le cas si $\mathcal{R}_1 \cup \mathcal{R}_2$ la vérifie lui-même) et terminant si $\mathcal{R}_1$ normalise.

La correction des systèmes de superdéduction modulo est une conséquence des corrections de la déduction modulo et de la superdéduction. La propriété est une première étape vers cette correction.

Propriété 6.3.1. *S'il existe une dérivation de $\Gamma \vdash \Delta$ dans le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$, alors il existe une dérivation de $\Gamma \vdash \Delta$ dans $LK_{\equiv}$ où $\equiv$ est la congruence engendrée par $\mathcal{R}_1 \cup \mathcal{R}_2$. En outre, si la première dérivation est sans coupure, alors la deuxième l'est aussi.*

Démonstration. Il suffit de remplacer les règles de superdéduction par les règles de LK qui lui correspondent suivies d'une instance de ConvR ou ConvL (exactement comme en sous-section 6.2.1). □

On obtient alors la correction de la superdéduction modulo.

Propriété 6.3.2 (Correction de la superdéduction modulo). *S'il existe une dérivation de $\Gamma \vdash \Delta$ dans le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$, alors il existe une dérivation de $\Gamma, \mathcal{T}h_{\mathcal{R}_1}, \mathcal{T}h_{\mathcal{R}_2} \vdash \Delta$ dans LK.*

Démonstration. La propriété 6.3.1 traduit la preuve de $\Gamma \vdash \Delta$ en superdéduction modulo en une dérivation dans $LK_{\equiv}$ où $\equiv$ est la congruence engendrée par $\mathcal{R}_1 \cup \mathcal{R}_2$. Alors par la correction de la déduction modulo (propriété 5.2.1), on obtient une dérivation de $\Gamma, \mathcal{T}h_{\mathcal{R}_1}, \mathcal{T}h_{\mathcal{R}_2} \vdash \Delta$ dans LK. □

Notons que la complétude de la déduction modulo est toujours vérifiée. Alors la complétude de la superdéduction modulo est impliquée par la complétude de la superdéduction (qui n'est pas, elle, toujours vérifiée).

Propriété 6.3.3 (Complétude de la superdéduction modulo). *Si le système de superdéduction associé à $\mathcal{R}_2$ est complet, alors le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$ l'est aussi : s'il existe une dérivation de $\Gamma, \mathcal{T}h_{\mathcal{R}_1}, \mathcal{T}h_{\mathcal{R}_2} \vdash \Delta$ dans LK, alors il existe une dérivation de $\Gamma \vdash \Delta$ dans le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$.*

Démonstration. Supposons le système de superdéduction associé à $\mathcal{R}_2$ complet. Alors pour toute formule $\varphi \in \mathcal{T}h_{\mathcal{R}_2}$, il existe une dérivation de $\vdash \varphi$ dans le système de superdéduction correspondant à $\mathcal{R}_2$. Puisque le système de déduction modulo associé à $\mathcal{R}_1$ est complet, pour toute formule $\varphi \in \mathcal{T}h_{\mathcal{R}_1}$, il existe une dérivation de $\vdash \varphi$ dans le système de déduction modulo correspondant à $\mathcal{R}_1$. Ces dérivations en superdéduction ou bien en déduction modulo sont aussi valides dans le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$. Soit à présent une dérivation de $\Gamma, \mathcal{T}h_{\mathcal{R}_1}, \mathcal{T}h_{\mathcal{R}_2} \vdash \Delta$ dans LK. Elle est aussi valide dans le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$. Par conséquent des coupures sur les formules $\varphi \in \mathcal{T}h_{\mathcal{R}_1} \cup \mathcal{T}h_{\mathcal{R}_2}$ permettent d'écrire une dérivation de $\Gamma \vdash \Delta$ dans le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$. □

Nous allons à présent démontrer l'expressivité de la superdéduction modulo par un exemple comprenant la récurrence sur les entiers de Peano. Nous utilisons le paradigme modulo pour représenter l'addition en utilisant les règles $0 + y \longrightarrow y$ et $S(x) + y \longrightarrow S(x + y)$ ainsi que la multiplication en utilisant les règles $0 * x \longrightarrow 0$ et $S(x) * y \longrightarrow y + (y * x)$. Nous définissons aussi la fonction

$$\text{sum}(x) = \sum_{k=0}^{x-1} (2 * k + 1)$$

en utilisant les règles $\text{sum}(0) \longrightarrow 0$ et $\text{sum}(S(x)) \longrightarrow S(x + (x + \text{sum}(x)))$. Le système de réécriture obtenu est convergent ce qui rend son utilisation confortable en déduction modulo.

Les nombres naturels ainsi que le principe d'induction associé sont définis par les règles $\mathbb{N}(n) \longrightarrow \forall P.(0 \in P \Rightarrow \mathfrak{H}(P) \Rightarrow n \in P)$ et $\mathfrak{H}(P) \longrightarrow \forall k.(k \in P \Rightarrow S(k) \in P)$ que nous utiliserons au travers de la superdéduction. Le prédicat $\mathfrak{H}(P)$ est introduit pour assurer la complétude des inférences de superdéduction associées à ces règles de prédicats. L'égalité de Leibniz est définie en utilisant la règle $x = y \longrightarrow \forall P.(x \in P \Rightarrow y \in P)$. Notons que dans les trois dernières règles de prédicats que nous venons d'écrire, nous avons écrit $x \in A$ au lieu d'écrire $A(x)$ (pour une formule quelconque A). Nous nous l'autorisons en rajoutant pour chaque formule φ une constante du premier ordre fraîche $\tilde{\varphi}$ ainsi que la règle de prédicats pour la déduction modulo $x \in \tilde{\varphi} \longrightarrow \varphi(x)$. Alors les nouvelles inférences de superdéduction sont celles décrites en figure 6.4. En combinant superdéduction (pour l'induction et l'égalité) et déduction modulo (pour l'addition, la multiplication et $\text{sum}(x)$), nous obtenons un système de superdéduction modulo dans lequel il est facile de prouver,

$$\mathbb{N}\mathrm{R}\ \frac{\Gamma, 0 \in P, \mathfrak{H}(P) \vdash n \in P, \Delta}{\Gamma \vdash \mathbb{N}(n), \Delta}\ P \notin \mathcal{FV}(\Gamma, \Delta)$$

$$\mathbb{N}\mathrm{L}\ \frac{\Gamma \vdash 0 \in P, \Delta \qquad \Gamma \vdash \mathfrak{H}(P), \Delta \qquad \Gamma, n \in P \vdash \Delta}{\Gamma, \mathbb{N}(n) \vdash \Delta}$$

$$\mathfrak{H}\mathrm{L}\ \frac{\Gamma \vdash m \in P, \Delta \qquad \Gamma, S(m) \in P \vdash \Delta}{\Gamma, \mathfrak{H}(P) \vdash \Delta}$$

$$\mathfrak{H}\mathrm{R}\ \frac{\Gamma, k \in P \vdash S(k) \in P, \Delta}{\Gamma \vdash \mathfrak{H}(P), \Delta}\ k \notin \mathcal{FV}(\Gamma, \Delta)$$

$$=\mathrm{R}\ \frac{\Gamma, x \in P \vdash y \in P, \Delta}{\Gamma \vdash x = y, \Delta}\ P \notin \mathcal{FV}(\Gamma, \Delta)$$

$$=\mathrm{L}\ \frac{\Gamma, y \in P \vdash \Delta \qquad \Gamma \vdash x \in P, \Delta}{\Gamma, x = y \vdash \Delta}$$

FIG. 6.4 – Règles de superdéduction pour les entiers de Peano et l'égalité de Leibniz

par exemple, l'égalité

$$\sum_{k=0}^{n-1}(2 * k + 1) = n^2 \ .$$

La preuve est celle décrite en figure 6.5. Le symbole $\tilde{P}$ y représente la constante du premier ordre associée à la proposition $\mathrm{sum}(x) = x * x$. Notons aussi que les prémisses de l'instance de =L sont $\vdash \mathrm{sum}(m) \in \tilde{Q}$ et $m * m \in \tilde{Q} \vdash S(m + (m + \mathrm{sum}(m))) = S(m+(m+(m*m)))$ où $\tilde{Q}$ est la constante du premier ordre associée à $S(m + (m + \mathrm{sum}(m))) = S(m + (m + x))$. Enfin remarquons que cette preuve n'utilise que des nouvelles règles de superdéduction en plus de la règle AXIOM.

6.4 Méthode des tableaux en superdéduction modulo

La méthode des tableaux pour la déduction modulo de Richard Bonichon que nous avons décrite en sous-section 5.4.2 s'adapte au cadre de la superdéduction modulo de la façon suivante. Considérons par exemple une règle de prédicats $P \longrightarrow \varphi$. Supposons qu'une règle d'introduction à gauche soit obtenue après avoir obtenue $S = \{\Gamma_i \vdash \Delta_i / 1 \leqslant i \leqslant n\}$ et si $C = \{(x_k, \Gamma'_k) / 1 \leqslant k \leqslant p\}$. Cette introduction à gauche de P correspondante est la règle d'inférence représentant les inférences

$$\frac{\Gamma, \Gamma_1\sigma \vdash \Delta_1\sigma, \Delta \quad \ldots \quad \Gamma, \Gamma_n\sigma \vdash \Delta_n\sigma, \Delta}{\Gamma, P\sigma \vdash \Delta}\ x_k \notin \Gamma, \Gamma'_k\sigma, \Delta \text{ POUR TOUT } 1 \leqslant k \leqslant p$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{0 \in A \vdash 0 \in A}\,\text{Axiom}}{\vdash 0 = 0}\,=\text{R}}{\vdash \text{sum}(0) = 0 * 0}\,\text{ConvR}}{\vdash 0 \in \tilde{P}}\,\text{ConvR} \qquad \vdots \qquad \dfrac{\dfrac{}{\text{sum}(n) = n * n \vdash \text{sum}(n) = n * n}\,\text{Axiom}}{n \in \tilde{P} \vdash \text{sum}(n) = n * n}\,\text{ConvL}}{\mathcal{N}(n) \vdash \text{sum}(n) = n * n}\,\mathbb{N}\text{L}}{\vdash \mathfrak{H}(\tilde{P})}\,\mathfrak{H}\text{R}}{\text{sum}(m) = m * m \vdash S(m + (m + \text{sum}(m))) = S(m + (m + (m * m)))}\,\text{ConvR}}{\text{sum}(m) = m * m \vdash \text{sum}(S(m)) = S(m) * S(m)}\,=\text{L}$$

$$\dfrac{\dfrac{}{S(m + (m + \text{sum}(m))) = S(m + (m + (m * m))) \vdash S(m + (m + \text{sum}(m))) = S(m + (m + (m * m)))}\,\text{Axiom}}{\vdash S(m + (m + \text{sum}(m))) = S(m + (m + \text{sum}(m)))}\,=\text{R} \qquad \vdots$$

FIG. 6.5 – Une preuve de $\mathbb{N}(n) \Rightarrow \sum_{k=0}^{n-1} (2 * k - 1) = n^2$ en superdéduction modulo

où σ est une substitution dont le domaine est inclus dans $V \cup \mathcal{FV}(P)$. Cette règle d'inférence se traduit en la règle pour les tableaux suivante.

$$\mathcal{B}, P\sigma \mid \mathcal{T} \quad \rightarrow \quad \mathcal{B}, P\sigma, \Gamma_1\sigma, \neg\Delta_1\sigma \mid \cdots \mid \mathcal{B}, P\sigma, \Gamma_n\sigma, \neg\Delta_n\sigma \mid \mathcal{T}$$

où σ est une substitution dont le domaine est $V \cup \mathcal{FV}(P) \cup \{x_k/1 \leqslant k \leqslant p\}$ et telle que pour toute variable $x \in V$, $x\sigma$ est une variable fraîche et pour tout x_k, $x_k\sigma$ est un terme $f(y_1, \ldots, y_m)$ où f est un symbole de fonction de Skolem frais et $\{y_1, \ldots, y_m\} = \mathcal{FV}(\mathcal{B}, P\sigma)$. Des règles pour les tableaux peuvent être similairement déduites pour les introductions à droite. En rajoutant ces règles aux règles de TaMeD (figure 5.11), nous obtenons une méthode des tableaux correcte et complète pour la superdéduction modulo sans coupure, et pour la superdéduction modulo si la propriété d'admissiblité des coupures est vérifiée.

Considérons par exemple la règle $\mathbb{N}$L définie en section 6.3. La règle des tableaux qu'elle induit est

$$\mathcal{B}, \mathbb{N}(n) \mid \mathcal{T} \quad \rightarrow \quad \mathcal{B}, \mathbb{N}(n), \neg(0 \in y) \mid \mathcal{B}, \mathbb{N}(n), \neg(\mathfrak{H}(y)) \mid \mathcal{B}, \mathbb{N}(n), (n \in y) \mid \mathcal{T}$$

Cette règle correspond à la première étape de la preuve de la figure 6.5 sauf qu'une variable fraîche y vient remplacer la constante du premier ordre $\tilde{P}$. Cette variable fraîche pourra être substituée ensuite par $\tilde{P}$ lors de l'utilisation la règle de *extended narrowing* ou encore la règle de *extended branch closure*.

6.5 Lemuridæ

Paul Brauner et moi avons développé un prototype d'assistant de preuve basé sur la superdéduction modulo que nous avons nommé Lemuridæ. Il permet d'étendre le calcul des séquents classique en utilisant des règles de termes et de propositions et implémente aussi la procédure d'hyperpolarisation que nous introduirons en section 7.2. Pour l'implémenter, nous avons choisi le langage Tom (Moreau et Reilles, 2006) qui rajoute de la réécriture et de la programmation par stratégies au langage Java. Nous verrons que le choix de ce langage a de nombreux avantages : Son expressivité nous permet d'écrire un code propre et court. C'est particulièrement important pour l'implémentation de la procédure de vérification des preuves. L'expressivité de Tom est telle que cette implémentation consiste seulement à traduire les inférences de LK en règles de réécriture pour Tom. Ainsi la fonction de vérification des preuves ne contient que 150 lignes de code. Tout utilisateur peut donc les lire puis les comparer aux inférences de LK pour finalement se convaincre de la correction de l'implémentation. Un deuxième avantage de Tom est l'expression de tactiques en utilisant des stratégies. Le langage de stratégies de Tom est inspiré du langage Elan (Visser et Benaissa, 1998) et du rho-calcul. Des combinateurs de stratégies permettent de composer des stratégies de base pour former des programmes complexes. On peut alors très facilement exprimer des tactiques de recherche de preuve complexes.

L'objet de cette section est de présenter brievement le prototype Lemuridæ et son implantation.

6.5.1 Lemuridæ par l'exemple

L'interaction avec le prototype Lemuridæ se fait soit par une interface en mode texte, soit par une interface graphique (figure 6.6). Lemuridæ propose un cadre d'expérimentation pour les systèmes de superdéduction modulo. On peut naturellement y étendre le calcul des séquents classique par les inférences correspondant à des règles de réécriture. Par exemple, le prédicat $\subseteq$ peut être défini en déclarant une règle de réécriture grâce au mot-clé `rule`.

```
> rule Subset(A,B) -> forall x, In(x,A) => In(x,B).
```

Lemuridæ affiche à titre d'information les deux schémas d'inférences induits par cette règle de réécriture.

```
The new deduction rules are :

In(x0, A) |- In(x0, B)
---------------------- (x0 fresh)
 |- Subset(A, B)

 |- In(x0, A)    In(x0, B) |-
------------------------------
Subset(A, B) |-
```

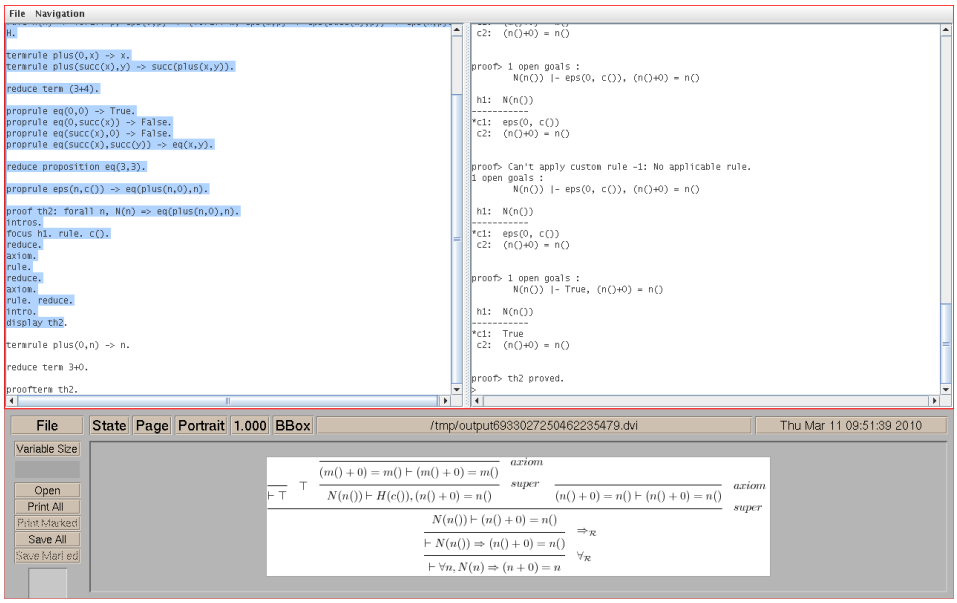


FIG. 6.6 – L'interface graphique de Lemuridæ

On peut alors s'engager dans la démonstration de $\forall x.x \subseteq x$ à l'aide de la commande `proof`. Le système entre en mode démonstration, mode dans lequel on construit interactivement une dérivation ayant pour conclusion $\vdash \forall x.x \subseteq x$. À chaque étape de cette construction, on est confronté à une dérivation partielle contenant des feuilles ouvertes qu'il faut fermer pour terminer la démonstration (comme en Coq). Dans notre cas, il n'y a qu'une feuille ouverte : $\vdash \forall x.x \subseteq x$.

```
> proof th1: forall A, Subset(A,A).

1 open goals :
         |- forall A, Subset(A, A)

-----------
*c1:  forall A, Subset(A, A)
```

Le séquent conclusion de la feuille courante est présenté horizontalement dans la section `open goals` puis verticalement. Dans le cas présent, l'information est redondante puisqu'il n'y a qu'une feuille ouverte. Dans ce cas, la tactique `auto`, qui applique toutes les règles de dérivation du calcul des séquents tant que possible, vient à bout de la démonstration.

```
proof> auto.
th1 proved.
```

On peut alors demander au système d'afficher la dérivation obtenue.

```
> display th1.
```

$$\dfrac{\dfrac{\dfrac{}{In(x,a) \vdash In(x,a)}\ axiom}{\vdash Subset(A,A)}\ super}{\vdash \forall A, Subset(A,A)}\ \forall_R$$

Considérons à présent un exemple plus intéressant : la définition des entiers de Peano, vue en section 6.3.

```
rule N(n) -> forall p, eps(0,p)
          => (forall m, eps(m,p) => eps(s(m),p))
          => eps(n,p).
```

Cette règle de réécriture peut être entrée telle quelle dans Lemuridæ ou bien générée par le système lui-même à partir du type inductif

```
inductive Nat = z() | s(m:Nat).
```

La règle de réécriture menace la complétude du système, car la proposition $\forall m.[\dots]$ se trouve en position négative sous un quantificateur universel. Pour régler ce problème, le système utilise la procédure d'hyperpolarisation que nous verrons en section 7.2. Pour continuer, l'utilisateur doit nommer la proposition $\forall m.(m \in p \Rightarrow S(m) \in p)$. On choisit par exemple `H`, pour « héréditaire ».

```
name the proposition "forall m, eps(m, p) => eps(s(m), p)" > H.
```

Cela donne lieu à quatre schémas d'inférence introduisant les prédicats `N` et `H` à droite et à gauche.

```
The new deduction rules are :

eps(0, p0), H(p0) |- eps(n, p0)
------------------------------ (p0 fresh)
 |- N(n)

|- eps(0, p0)     |- H(p0)    eps(n, p0) |-
---------------------------------------------
 N(n) |-

eps(m0, p) |- eps(s(m0), p)
------------------------------ (m0 fresh)
 |- H(p)

 |- eps(m0, p)    eps(s(m0), p) |-
--------------------------------------
 H(p) |-
```

On se propose, à titre d'exemple, de démontrer que $\forall n, N(n) \Rightarrow n + 0 = n$. Pour cela, nous définissons un ensemble de règles de réécriture de termes et de propositions, qui vont enrichir la partie *modulo* du système et qui manipuleront les additions et les égalités.

```
> termrule plus(0,x) -> x.
> termrule plus(s(x),y) -> s(plus(x,y)).
```

```
> proprule eq(0,0) -> True.
> proprule eq(0,s(x)) -> False.
> proprule eq(s(x),0) -> False.
> proprule eq(s(x),s(y)) -> eq(x,y).
> proprule eps(n,c()) -> eq(plus(n,0),n).
```

La constante `c()` représente l'ensemble $\{n/n+0=n\}$: en effet $n \in$ `c()` se réduit en $n+0=n$. Bien que nous rajoutions ici cette constante pour le besoin de la démonstration future, rajouter automatiquement de telles constantes par anticipation ne constitue pas une technique de développement viable. En réalité, il vaut mieux utiliser la théorie des classes, comme nous l'avons fait en section 6.3, sous forme d'un codage fini, comme le propose Kirchner (2006).

On peut à présent s'engager dans la démonstration de la proposition $\forall n, N(n) \Rightarrow n+0=n$.

```
> proof th2: forall n, N(n) => eq(plus(n,0),n).

1 open goals :
         |- forall n, N(n) => (n+0) = n

-----------
*c1:  forall n, N(n) => (n+0) = n
```

Remarquons que Lemuridæ affiche le symbole `plus` selon une notation infixe. C'est un comportement *ad-hoc* pour l'instant qui gagnerait à être rendu paramétrable. La tactique `intros` introduit tant que possible les connecteurs logiques. On se retrouve dans la situation suivante après son application.

```
proof> intros.

1 open goals :
        N(n) |- (n+0) = n

 h1:  N(n)
-----------
*c1:  (n+0) = n
```

On peut à présent tirer parti de l'hypothèse `h1` pour montrer `c1` par récurrence sur `n`. Pour cela, on indique à Lemuridæ que l'on souhaite porter notre attention sur `h1` (commande `focus h1`), que l'on souhaite introduire à l'aide d'une inférence de superdéduction (tactique `rule`).

```
proof> focus h1. rule.

1 open goals :
        N(n) |- (n+0) = n

*h1:  N(n)
-----------
 c1:  (n+0) = n
```

Le système requiert la classe (au sens de la théorie des classes) représentant la proposition que l'on souhaite montrer par récurrence. Il s'agit ici de `c()`.

```
proof> new term for variable p0 in rule 3 > c().

3 open goals :
        N(n) |- eps(0, c()), (n+0) = n
        N(n) |- H(c()), (n+0) = n
        eps(n, c()), N(n) |- (n+0) = n

 h1:  eps(n, c())
 h2:  N(n)
-----------
*c1:  (n+0) = n
```

Cette fois, il y a trois feuilles ouvertes, correspondant chacune aux trois prémisses de l'inférence intoduisant le prédicat N à gauche. L'hypothèse `h1` est convertible en $n+0 = n$ par la congruence du système courant. On demande donc à Lemuridæ de normaliser le séquent (commande `reduce`).

```
proof> reduce.

3 open goals :
        N(n) |- eps(0, c()), (n+0) = n
        N(n) |- H(c()), (n+0) = n
        (n+0) = n, N(n) |- (n+0) = n

 h1:  (n+0) = n
 h2:  N(n)
-----------
*c1:  (n+0) = n
```

On peut dès à présent fermer cette branche de la dérivation par un axiome, et Lemuridæ présente le prochain but à démontrer.

```
proof> axiom.

2 open goals :
        N(n) |- eps(0, c()), (n+0) = n
        N(n) |- H(c()), (n+0) = n

 h1:  N(n)
-----------
*c1:  H(c())
 c2:  (n+0) = n
```

La démonstration s'achève ensuite sans encombres à l'aide des tactiques vues jusqu'ici. On ne la détaille pas dans son intégralité.

Il est à présent légitime (ceci n'est pas automatique dans Lemuridæ à l'heure actuelle) d'ajouter ce dernier théorème à la congruence par le biais de la règle de réécriture suivante.

```
> termrule plus(0,n) -> n.
```

Idéalement, Lemuridæ devrait vérifier, en faisant éventuellement appel à des outils tels que Cime (Contejean *et al.*, 2004), que l'ajout de cette règle ne met pas en péril la décidabilité de la congruence. Ce n'est pour l'instant pas le cas.

6.5.2 Le langage Tom

Le langage que nous avons utilisé pour programmer Lemuridæ est Tom. C'est une extension du langage Java développée par l'équipe Pareo (Balland *et al.*, 2007). Cette extension rajoute au langage Java les notions de terme, de filtrage, de réécriture et de stratégie. D'autres langages contiennent de façon plus ou moins intensive ces notions relatives à la réécriture : par exemple OCaml, Haskell, Elan ou Maude. Néanmoins le langage Tom a l'avantage de s'intégrer de façon légère au langage Java et facilite ainsi son adoption par les programmeurs non spécialisés. En outre, le fait que le compilateur Tom produise du code Java assure à la fois la pérennité des programmes produits ainsi que la possibilité de les exécuter sur une grande variété d'architectures. Les concepteurs du langage Tom adoptent donc une approche pragmatique consistant à proposer à une large communauté de programmeurs des notions jusque là réservées à des langages dont l'utilisation industrielle reste faible ou inexistante. Cette approche est couronnée de succès puisque le langage Tom est utilisé par plusieurs projets académiques et industriels, comme par exemple dans Crystal Reports, un logiciel produit par Business Objects au sein duquel le langage Tom permet la traduction de requêtes OLAP en requêtes SQL.

Termes Le langage Tom permet tout d'abord de représenter des termes en utilisant des types de données abstraits. Les entiers de Peano sont par exemple codés en Tom par la grammaire sous forme de Backus-Naur

```
Nat = Z() | S(n:Nat)
```

définissant le type de donnée abstrait `Nat`. Les arbres binaires sur les entiers de Peano (exemple 1.1.3) sont représentés par la grammaire sous forme de Backus-Naur

```
BinaryTree = Star() | Leaf(n:Nat,left:BinaryTree,right:BinaryTree)
```

Cela permet alors de construire des entiers de Peano, comme par exemple l'entier 3 représenté par `‘S(S(S(Z()))`, et des arbres binaires, comme par exemple

```
‘Leaf(S(S(Z())), Leaf(S(S(S(Z()))),Star(), Star()), Star())
```

Filtrage Le filtrage est représenté en Tom par l'instruction `%match`. On peut par exemple convertir un entier de Peano en `int` avec la méthode

```
public int peanoToInt(Nat n) {
  %match(n) {
    Z() -> { return 0; }
    S(m) -> { return 1+ peanoToInt(‘m); }
} }
```

puis calculer la somme des entiers d'un arbre binaire avec la méthode

```
public int binaryTreeToInt(BinaryTree t) {
  %match(t) {
    Star() -> { return 0; }
    Leaf(n,t1,t2) -> {
      return peanoToInt(n) + binaryTreeToInt(t1) + binaryTreeToInt(t2);
} } }
```

Le filtrage en Tom nous permet de programmer de façon très naturelle des processus manipulant des preuves (vérification, traduction ...). Le vérificateur de preuves de Lemuridæ consiste par exemple en un ensemble de règles de réécriture qui traduisent chacune un schéma d'inférence du système logique. Par exemple la règle

```
rule(
     andLeftInfo[],
     (p@rule(_,_,sequent((g1*,A,B,g2*), d),_)),
     sequent((g1*,a,g2*),d),
     a@and(A,B)
     )
-> { return proofcheck(‘p);}
```

représente le schéma $\wedge$L. Le vérificateur de preuve obtenu est extrêmement concis et lisible. Par exemple, la partie vérifiant les dérivations en calcul des séquents classique ne contient que 150 lignes de Tom.

Stratégies Afin de contrôler finement l'application des règles de réécriture, le langage Tom utilise la notion de stratégie de réécriture. Les stratégies comme `bottom-up`, `top-down`, `leftmost-innermost` sont des objets d'ordre supérieur qui décrivent comment un ensemble de règles de réécriture sera appliqué : à chaque étape, la stratégie choisit quelle règle est appliquée et à quelle position du terme réécrit.

En Tom, des stratégies de haut-niveau peuvent être définies en combinant des stratégies bas-niveau élémentaires définies par l'utilisateur. Ces stratégies bas-niveau sont en fait des ensembles de règles de réécriture. À l'aide de combinateurs tels que `TopDown`, l'utilisateur peut ensuite utiliser les stratégies bas-niveau pour créer des stratégies complexes. Tom contient une bibliothèque Java `sl` fournissant de nombreux combinateurs de stratégies.

Considérons par exemple la signature

```
Expr = Z() | S(n:Expr) | Plus(n:Expr, m:Expr) | Mult(n:Expr, m:Expr)
```

L'utilisateur peut définir la stratégie élémentaire suivante.

```
%strategy R extends Id {
  plus(x, Z())  -> x
  plus(x, S(y)) -> S(plus(x,y))
}
```

Cette définition est compilée par Tom en une classe Java `R` dont les instances représentent toutes le système de réécriture décrit dans l'exemple 1.1.3. Ces instances

n'admettent qu'une seule méthode `visit` qui représente l'application du système de réécriture : si `r` est une instance de `R`, l'application de cette stratégie à un terme `t` s'écrit `r.visit(t)`. Par exemple `r.visit(plus(S(Z()),S(Z())))` retournera `S(plus(Z(),S(Z())))` tandis que `r.visit(mult(plus(S(Z()),Z(S())),S(Z())))` retournera `mult(plus(S(Z()),Z(S())),S(Z()))` : en effet dans ce dernier cas aucune règle de réécriture ne s'applique en tête et la stratégie retourne alors le terme inchangé comme spécifié par `extends Id`.

Les combinateurs de stratégies sont aussi des classes Java dont le constructeur prend en argument des stratégies et dont les instances sont aussi des stratégies. L'exemple le plus simple est le combinateur `Seq` qui séquentialise deux stratégies. `(new Seq(r,r)).visit(t)` applique par exemple deux fois de suite `r` en tête au terme `t`. Un autre combinateur central est le combinateur `all` qui applique une stratégie à tous les sous-termes directs d'un terme. Par exemple

```
(new All(r)).visit(
  mult(
    plus(S(Z()),Z()),
    plus(S(Z()),S(Z()))
) )
```

retournera `mult(S(Z()),S(plus(S(Z()),Z())))`.

L'exemple type de stratégie obtenue à partir de ces combinateurs est la stratégie *top-down* qui applique la réécriture à un terme en recherchant les redexes en descendant depuis la racine vers les feuilles. Cette stratégie se définit comme le point fixe

$$\text{top-down}\, s = \texttt{Seq}(s, \texttt{All}(\text{top-down}\, s))$$

Ce point fixe est exprimé en Tom grâce au combinateur `Mu`.

```
Strategy TopDown(s:Strategy) = `Mu("x",Seq(s,All((MuVar("x"))(s))))
```

Les stratégies de Tom nous permettent d'exprimer très facilement des tactiques complexes de preuve. Par exemple, la tactique `intros` de Lemuridæ introduit les connecteurs logiques tant que possible dans toutes les branches d'une dérivation ouverte. Elle est simplement exprimée à l'aide de la stratégie `topdown (try intro)`.

Chapitre 7

Focalisation pour la superdéduction

La présent chapitre est dédié à **(1)** la présentation d'un résultat d'admissibilité des coupures pour la superdéduction modulo basé sur une analyse de la permutabilité des inférences (section 7.1) **(2)** suivi d'une comparaison des paradigmes de focusing et de superdéduction. Mis à part la présentation du focusing (sous-section 7.3.1), tout ce qui est exposé dans ce chapitre constitue une contribution originale de cette thèse et a fait l'objet d'une publication (Houtmann, 2008a). Le système de *focused fold* présenté en sous-section 7.3.4 ne figure cependant pas dans cette publication est constitue donc un apport totalement original de cette thèse.

Comme l'expliquent les chapitres 5 puis 6, les paradigmes de déduction modulo et de superdéduction permettent de générer des systèmes déductifs spécialisés pour une théorie donnée. La complétude des inférences de superdéduction n'est assurée que lorsque certaines conditions de synchronicité sont vérifiées par les règles de prédicats correspondant : leurs membres droits doivent être des hyperpôles (propriété 6.1.5). Nous allons le voir en section 7.1, il en va de même pour l'admissibilité des coupures : lorsque certaines conditions de synchronicité sont vérifiées, le système de superdéduction modulo sans coupure est équivalent au système de déduction modulo sans coupure et par conséquent tout résultat d'admissibilité des coupures dans l'un est valable dans l'autre et inversement. Ces conditions de synchronicité sont naturelles : elles traduisent le fait que lorsqu'on utilise des inférences de superdéduction, certains ordonnancement d'inférences sont interdits alors qu'ils seraient permis si l'on utilisait directement les inférences de LK correspondant.

Le focusing est une approche proposant une gestion des inférences dirigée par la synchronicité des connecteurs. Initialement proposée dans le cadre de la logique linéaire (Andreoli, 1992, 2001), elle séquentialise le mécanisme de preuve en phases synchrones et asynchrones. Le focusing centralise les inférences non inversibles dans les phases synchrones. On obtient alors un système bien plus favorable à la recherche de preuve : l'espace de recherche est restreint par l'alternance des phases et les points de backtracking sont explicites : ce sont les entrées des phases synchrones. Comme

le remarque Andreoli (2001), l'alternance d'une phase synchrone puis d'une phase asynchrone correspond à l'introduction d'un bipôle, c'est-à-dire à une couche synchrone englobant une couche asynchrone qui constitue ce que Girard (1999, 2000) appelle un *connecteur synthétique*. Relier superdéduction et focusing semble alors naturel : si l'on compare les connecteurs synthétiques aux règles de prédicats, les cycles synchrones/asynchrones correspondent aux inférences de superdéduction.

Mon article TYPES'08 (Houtmann, 2008a) comprend un démonstration du résultat d'admissibilité des coupures que je vais exposer en section 7.1 ainsi qu'une comparaison de la superdéduction avec le focusing. Néanmoins cette comparaison est largement incomplète : seule la gestion de l'inversibilité des inférences du focusing y est évoquée. Il manque notamment l'assignation de polarités aux propositions atomiques. Par conséquent en section 7.3 je revisite totalement le lien entre superdéduction et focusing en proposant un système de superdéduction modulo basé sur le paradigme du focusing et la notion de connecteurs synthétiques. L'alternance des phases synchrones et asynchrones y est *built-in* et la complétude n'est plus conditionnée par des hypothèses de synchronicité.

7.1 Déduction modulo vers superdéduction

Comme nous l'avons vu au chapitres 5 et 6, les propriétés de normalisation, d'élimination des coupures ou de cohérence ne sont pas toujours vérifiées pour les systèmes de déduction modulo ou de superdéduction. Pour la déduction modulo, il existe plusieurs techniques permettant de caractériser des classes de théories pour lesquelles le système de déduction modulo associé est normalisant ou bien vérifie l'admissibilité des coupures (section 5.3). Nous avons aussi déjà démontré que la normalisation d'un système de superdéduction est impliquée par la normalisation du système de déduction modulo lui correspondant. Nous allons à présent démontrer qu'il est de même pour l'admissibilité des coupures : l'admissibilité des coupures en superdéduction est impliquée par l'admissibilité des coupures en déduction modulo. Nous allons même le démontrer pour les systèmes de superdéduction modulo : si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont respectivement un ensemble de règles de termes et de prédicats et un ensemble de règles de prédicats, alors l'admissibilité des coupures dans le système de superdéduction associé à $(\mathcal{R}_1, \mathcal{R}_2)$ est impliquée par l'admissibilité des coupures dans le système de déduction modulo associé à $\mathcal{R}_1 \cup \mathcal{R}_2$. Nous obtiendrons ce résultat en prouvant que la prouvabilité en déduction modulo sans coupure concorde avec celle en superdéduction modulo sans coupure. Pour cela nous allons traduire les dérivations sans coupure en déduction modulo vers les dérivations sans coupure en superdéduction modulo (la traduction inverse étant déjà écrite en sous-section 6.2.1) de la façon suivante : Soient donc $(\mathcal{R}_1, \mathcal{R}_2)$ une paire d'ensemble de règles définissant un système de superdéduction modulo. Notre but est de démontrer l'admissibilité des règles $\textsc{ConvR}_{\mathcal{R}_2}$ et $\textsc{ConvL}_{\mathcal{R}_2}$ dans ce système de superdéduction modulo sans coupure : Tout d'abord nous montrons que pour toute règle $P \longrightarrow \varphi$ de $\mathcal{R}_2$, si un séquent $\Gamma \vdash \varphi\sigma, \Delta$ est démontrable en superdéduction modulo sans cou-

pure, alors $\Gamma \vdash P\sigma, \Delta$ l'est aussi. Nous prouvons aussi que si un séquent $\Gamma, \varphi\sigma \vdash \Delta$ est démontrable toujours en superdéduction modulo sans coupure, alors $\Gamma, P\sigma \vdash \Delta$ l'est aussi. Nous en déduisons l'admissibilité des règles $\textsc{ConvR}_{\mathcal{R}_2}$ et $\textsc{ConvL}_{\mathcal{R}_2}$ dans ce système de superdéduction modulo sans coupure ce qui nous permet de traduire alors toute dérivation sans coupure en déduction modulo en dérivation sans coupure en superdéduction modulo : il suffit de traduire les étapes de conversion correspondant à $\textsc{ConvR}_{\mathcal{R}_2}$ et $\textsc{ConvL}_{\mathcal{R}_2}$.

Le point clé de cette démonstration consiste donc à prouver que, dans le système de superdéduction modulo sans coupure associé à $(\mathcal{R}_1, \mathcal{R}_2)$, pour toute règle $P \longrightarrow \varphi$ de $\mathcal{R}_2$, si un séquent $\Gamma \vdash \varphi\sigma, \Delta$ est démontrable, alors $\Gamma \vdash P\sigma, \Delta$ l'est aussi et si $\Gamma, \varphi\sigma \vdash \Delta$ est démontrable, alors $\Gamma, P\sigma \vdash \Delta$ l'est aussi. Pour cela, l'idée est d'utiliser des permutations d'inférences : il nous faut regrouper toutes les inférences décomposant les connecteurs de $\varphi\sigma$ pour les unir en une inférence de superdéduction introduisant le prédicat P à gauche ou à droite. Les résultats de Kleene montrent que les seuls connecteurs pouvant poser problème quant aux permutations d'inférences sont les quantificateurs. Considérons par exemple une règle de prédicat $P \longrightarrow (\forall x.A(x)) \Rightarrow (\forall x.B(x))$ et la dérivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots}{A(y_0) \Rightarrow B(x_0), A(y_0) \vdash B(x_0)}}{A(y_0) \Rightarrow B(x_0), (\forall x.A(x)) \vdash B(x_0)}\,\forall\text{L}\,*}{\exists y.(A(y) \Rightarrow B(x_0)), (\forall x.A(x)) \vdash B(x_0)}\,\exists\text{R}}{\forall x.\exists y.(A(y) \Rightarrow B(x)), (\forall x.A(x)) \vdash B(x_0)}\,\forall\text{R}}{\forall x.\exists y.(A(y) \Rightarrow B(x)), (\forall x.A(x)) \vdash (\forall x.B(x))}\,\forall\text{R}\,*}{\forall x.\exists y.(A(y) \Rightarrow B(x)) \vdash (\forall x.A(x)) \Rightarrow (\forall x.B(x))}\,\Rightarrow\text{R}\,* \quad .$$

Les inférences correspondant à une décomposition d'un connecteur de $(\forall x.A(x)) \Rightarrow (\forall x.B(x))$ y sont marquées d'un $*$. Elles ne peuvent être directement regroupées car la deuxième inférence en partant du bas doit être effectuée avant la troisième pour libérer la variable x_0 et la quatrième inférence doit être effectuée avant la cinquième pour libérer la variable y_0. L'unique possibilité est d'utiliser une contraction pour obtenir la dérivation en figure 7.1. La question se pose alors de démontrer qu'un tel regroupement des inférences en utilisant des contractions est toujours possible. Nous laisserons cette question à l'état de conjecture. Par contre et lorsque les règles de prédicats utilisées par la superdéduction ont une certaine forme, les contractions ne sont pas nécessaires. Nous supposerons donc l'hypothèse suivante vérifiée.

Hypothèse 7.1.1. *Nous supposerons que pour toute règle de prédicats $P \longrightarrow \varphi$ de $\mathcal{R}_2$, φ est un monopôle ou bien la négation d'un monopôle.*

Afin de faciliter notre démonstration, nous supposerons en outre que φ est en forme prenexe. La mise en forme prenexe ne modifiant pas le résultat du calcul des inférences de la superdéduction, notre résultat final ne dépendra en fait pas de cette supposition. Par exemple la règle de prédicats $P \longrightarrow (\forall x.A(x)) \Rightarrow (\exists y.B(y))$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots}{A(y_0)\Rightarrow B(x_0), A(x_0), A(y_0)\vdash B(x_1), B(x_0)}}{A(y_0)\Rightarrow B(x_0), A(x_0), A(y_0)\vdash \forall x.B(x), B(x_0)}\forall R\,*}{A(y_0)\Rightarrow B(x_0), A(x_0), \forall x.A(x)\vdash \forall x.B(x), B(x_0)}\forall L\,*}{A(y_0)\Rightarrow B(x_0), A(x_0)\vdash \varphi, B(x_0)}\Rightarrow R\,*}{\exists y.A(y)\Rightarrow B(x_0), A(x_0)\vdash \varphi, B(x_0)}\exists L}{\forall x.\exists y.A(y)\Rightarrow B(x), A(x_0)\vdash \varphi, B(x_0)}\forall L}{\forall x.\exists y.A(y)\Rightarrow B(x), \forall x.A(x)\vdash \varphi, B(x_0)}\forall L\,*}{\forall x.\exists y.A(y)\Rightarrow B(x), \forall x.A(x)\vdash \varphi, \forall x.B(x)}\forall R\,*}{\forall x.\exists y.A(y)\Rightarrow B(x)\vdash \varphi, (\forall x.A(x))\Rightarrow(\forall x.B(x))}\Rightarrow R\,*}{\forall x.\exists y.(A(y)\Rightarrow B(x))\vdash (\forall x.A(x))\Rightarrow(\forall x.B(x))}\textsc{ContrR}\ .
$$

FIG. 7.1 – Regroupement d'inférences grâce à une contraction

vérifie l'hypothèse 7.1.1. Toutefois son membre droit n'est pas en forme prenexe mais si on le remplace par $\exists x.\exists y.(A(x) \Rightarrow A(y))$, on obtient les mêmes inférences de superdéduction, en l'occurrence

$$
\frac{\Gamma, A(t_1) \vdash B(t_2), \Delta}{\Gamma \vdash P, \Delta} \quad \text{et} \quad \frac{\Gamma, B(y) \vdash \Delta \qquad \Gamma \vdash A(x), \Delta}{\Gamma, P \vdash \Delta}\; x, y \notin \mathcal{FV}(\Gamma, \Delta) \quad .
$$

Un monopôle en forme prenexe est de la forme $\exists x_1 \dots x_n.\psi$ où ψ est une formule propositionnelle et la négation d'un monopôle en forme prenexe est de la forme $\forall x_1 \dots x_n.\psi$. De telles formules sont appelées des formules Σ_1 et Π_1. La dualité de De Morgan nous permet de ne traiter finalement que des règles de prédicats $P \to \varphi$ telles que φ est Π_1. Le traitement du cas où φ est Σ_1 y est déjà traité implicitement : il suffit d'inverser les deux côtés des séquents pour passer d'un traitement à l'autre. Par conséquent nous supposerons donc finalement que $\mathcal{R}_2$ ne contient que des règles de prédicats $P \to \varphi$ où φ est Π_1.

Nous allons à présent démontrer d'admissibilité de

$$
\textsc{ConvR}_{\mathcal{R}_2}\,\frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \psi, \Delta} \quad \text{et} \quad \textsc{ConvL}_{\mathcal{R}_2}\,\frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \psi \vdash \Delta} \qquad (\varphi \equiv_{\mathcal{R}_2} \psi)
$$

dans le système de superdéduction modulo sans coupure associé à $(\mathcal{R}_1, \mathcal{R}_2)$. Cette démonstration a fait l'objet d'une publication (Houtmann, 2008a). Elle est décrite intégralement dans la version longue de cette publication (Houtmann, 2008b). Nous allons tout d'abord avoir besoin d'adapter le résultat d'inversibilité de Kleene de LK pour la superdéduction modulo.

Propriété 7.1.1 (Lemme de Kleene pour la superdéduction modulo). *Soient des formules $A_1 \equiv_1 A_2 \equiv_1 \dots A_n \equiv_1 \varphi$. Soit $\Theta = A_1, A_2 \dots A_n$. Si $\Gamma, \Theta \vdash \Delta$ est dérivable dans le système de superdéduction modulo sans coupure, alors*

si $\varphi = \neg A$, *alors* $\Gamma \vdash A, \Delta$ *est aussi dérivable ;*
si $\varphi = A \wedge B$, *alors* $\Gamma, A, B \vdash \Delta$ *est aussi dérivable ;*
si $\varphi = A \vee B$, *alors* $\Gamma, A \vdash \Delta$ *et* $\Gamma, B \vdash \Delta$ *sont aussi dérivables ;*
si $\varphi = A \Rightarrow B$, *alors* $\Gamma, B \vdash \Delta$ *et* $\Gamma \vdash .A, \Delta$ *sont aussi dérivables ;*
si $\varphi = \exists x.Q$, *alors* $\Gamma, Q[c/x] \vdash \Delta$ *est aussi dérivable pour toute variable fraîche* c.
De même, si $\Gamma \vdash \Theta, \Delta$ *est dérivable dans le système de superdéduction modulo sans coupure, alors*
si $\varphi = \neg A$, *alors* $\Gamma, A \vdash \Delta$ *est aussi dérivable ;*
si $\varphi = A \wedge B$, *alors* $\Gamma \vdash A, \Delta$ *et* $\Gamma \vdash B, \Delta$ *sont aussi dérivables ;*
si $\varphi = A \vee B$, *alors* $\Gamma \vdash A, B, \Delta$ *est aussi dérivable ;*
si $\varphi = A \Rightarrow B$, *alors* $\Gamma, A \vdash^{cf+_2}_{\equiv_1} B, \Delta$ *est aussi dérivable.*
si $\varphi = \forall x.Q$, *alors* $\Gamma \vdash Q[c/x], \Delta$ *est aussi dérivable pour toute variable fraîche* c.

Démonstration. À chaque fois, par induction sur la dérivation. □

Cette inversibilité nous permet, lorsque $P \to \varphi$ est une règle de $\mathcal{R}_2$ et puisque φ est Π_1, de toujours pouvoir regrouper les inférences concernant $\varphi\sigma$ à la racine dans une dérivation de $\Gamma \vdash \varphi\sigma, \Delta$ en superdéduction modulo.

Propriété 7.1.2. *Soit* $\mathcal{R} \ : \ P \to \varphi$ *une règle de prédicats de* $\mathcal{R}_2$. *Puisque* φ *est* Π_1, *si* $\Gamma \vdash \varphi\sigma, \Delta$ *est dérivable en superdéduction modulo sans coupure, alors il existe des dérivations sans coupure des prémisses de* $\mathcal{R}R$ *correspondantes. Par conséquent cette règle nous donne une dérivation sans coupure de* $\Gamma \vdash P\sigma, \Delta$ *en superdéduction modulo.*

Démonstration. Par itération de la propriété 7.1.1. □

Prouver la même propriété pour les règles d'introduction à gauche est une autre affaire : les inférences ne peuvent plus être regroupées à la racine de la dérivation à cause la non-inversibilité des introductions de connecteurs synchrones. Nous choisissons donc de les permuter vers les feuilles de la dérivation. Cela nécessite d'abord quelques manipulations préliminaires sur la dérivation.

Définition 7.1.1 (Superdéduction modulo atomique). *Le système de superdéduction modulo atomique associé à* $(\mathcal{R}_1, \mathcal{R}_2)$ *est la restriction du système de superdéduction modulo où les règles de conversion*

$$\textsc{ConvR}_{\mathcal{R}_1}\ \frac{\Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi, \Delta}\ \varphi \equiv_1 \psi \qquad \textit{et} \qquad \textsc{ConvL}_{\mathcal{R}_1}\ \frac{\Gamma, \psi \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\ \varphi \equiv_1 \psi$$

sont restreintes au cas où φ *est une proposition atomique.*

Propriété 7.1.3. *Un séquent est dérivable dans le système superdéduction modulo associé à* $(\mathcal{R}_1, \mathcal{R}_2)$ *si et seulement si il l'est dans le système de superdéduction modulo atomique associé à* $(\mathcal{R}_1, \mathcal{R}_2)$.

Démonstration. Par induction sur la dérivation en utilisant la confluence de $\mathcal{R}_1$. □

Définition 7.1.2 (Décomposition totale). *Nous dirons que Γ' et Δ' sont totalement décomposés dans une preuve de $\Gamma, \Gamma' \vdash \Delta', \Delta$ (en superdéduction modulo ou en superdéduction modulo atomique) lorsque les règles* Axiom *ne sont appliquées que sur des séquents où plus un seul connecteur de Γ' ou Δ' n'apparaissent.*

Notons qu'en superdéduction modulo atomique, dans un chemin de la racine à l'une des feuilles d'une dérivation de $\Gamma, \Gamma' \vdash \Delta', \Delta$, la seule manière de faire disparaître un connecteur de Γ' ou Δ' est de le décomposer grâce à la règle d'inférence lui correspondant. Ce n'est pas le cas en superdéduction modulo (non-atomique) où une étape ConvR ou ConvL peut tout à fait effacer un connecteur pour le remplacer par une proposition atomique.

Propriété 7.1.4. *Si $\Gamma \vdash \Delta$ est dérivable en superdéduction modulo, alors il en existe une dérivation en superdéduction modulo atomique les décomposant totalement.*

Démonstration. Une preuve est décrite dans la version longue de mon article (Houtmann, 2008b). □

Définition 7.1.3. *Si $\varphi = \forall x_1 \dots x_n.\psi$ est Π_1 et σ est une substitution dont le domaine est inclus dans $\{x_1, \dots, x_{k-1}\}$, alors nous appellerons les formules de la forme $\forall x_k \dots x_n.\psi\sigma$ des décompositions partielles de φ.*

Définition 7.1.4. *Dans une dérivation de $\Gamma, \varphi \dots \varphi \vdash \Delta$ en superdéduction modulo sans coupure où φ est Π_1, nous dirons que φ est non-contracté si aucune contraction n'est effectuée sur une décomposition partielle de φ.*

Propriété 7.1.5. *S'il existe une dérivation de $\Gamma, \varphi_1 \dots \varphi_n \vdash \Delta$ où chaque φ_i est Π_1 et totalement décomposé en superdéduction modulo sans coupure, alors il existe une dérivation d'un séquent $\Gamma, \varphi_1 \dots \varphi_1, \varphi_2 \dots \varphi_2 \dots \varphi_n \vdash \Delta$ en superdéduction modulo sans coupure ou chaque φ_i est non-contracté et totalement décomposé.*

Démonstration. Par induction sur la dérivation. □

Propriété 7.1.6. *Soit $\mathcal{R} : P \to \varphi$ une règle de prédicats de $\mathcal{R}_2$. Puisque φ est Π_1, si $\Gamma, \varphi' \vdash \Delta$ est dérivable en superdéduction modulo sans coupure où φ' est une décomposition partielle de $\varphi\sigma$, alors il existe une dérivation sans coupure de $\Gamma, P \vdash \Delta$.*

Démonstration. Les propriétés 7.1.3, 7.1.4 et 7.1.5 fournissent une dérivation de $\Gamma, \varphi' \vdash \Delta$ où φ' est totalement décomposé et non-contracté. Alors on raisonne par induction sur cette dérivation. □

Nous sommes alors en mesure de démontrer la version de la propriété 7.1.2 à gauche :

Propriété 7.1.7. *Soit $\mathcal{R} : P \to \varphi$ une règle de prédicats de $\mathcal{R}_2$. Puisque φ est Π_1, si $\Gamma, \varphi\sigma \vdash \Delta$ est dérivable en superdéduction modulo sans coupure, alors il existe une dérivation sans coupure de $\Gamma, P\sigma \vdash \Delta$ en superdéduction modulo.*

Démonstration. Conséquence immédiate de la propriété 7.1.6. □

Finalement, les propriétés 7.1.2 et 7.1.7 permettent de démontrer l'admissibilité des règles $\textsc{ConvR}_{\mathcal{R}_2}$ et $\textsc{ConvL}_{\mathcal{R}_2}$ dans le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$.

Propriété 7.1.8. *Si $\equiv_2$ est la congruence engendrée par $\mathcal{R}_2$ et si $\varphi \equiv_2 \psi$, alors dans le système de superdéduction modulo sans coupure associé à $(\mathcal{R}_1, \mathcal{R}_2)$, (1) $\Gamma \vdash \varphi, \Delta$ est dérivable si et seulement si $\Gamma \vdash \psi, \Delta$ est dérivable ; (2) $\Gamma, \varphi \vdash \Delta$ est dérivable si et seulement si $\Gamma, \psi \vdash \Delta$ est dérivable.*

Démonstration. Le cas $\varphi = P\sigma$ et $\psi = \varphi'\sigma$ avec $P \to \varphi' \in \mathcal{R}_2$ provient directement de la correction des règles de superdéduction (propriété 6.1.3). Le cas inverse $\varphi = \varphi'\sigma$ et $\psi = P\sigma$ provient des propriétés 7.1.2 et 7.1.7. Pour démontrer que la prouvabilité est préservée par réécriture arbitraire, on procède par induction sur le nombre de pas de réécriture effectué puis sur la profondeur du pas de réécriture. □

En corollaire, on obtient l'équivalence entre le système de superdéduction modulo sans coupure associé à $(\mathcal{R}_1, \mathcal{R}_2)$ et le système de déduction modulo sans coupure associé à $\mathcal{R}_1 \cup \mathcal{R}_2$.

Propriété 7.1.9. *Un séquent $\Gamma \vdash \Delta$ est dérivable dans le système de superdéduction modulo sans coupure associé à $(\mathcal{R}_1, \mathcal{R}_2)$ si et seulement si il l'est dans le système de déduction modulo sans coupure associé à $\mathcal{R}_1 \cup \mathcal{R}_2$.*

Démonstration. Pour traduire la superdéduction modulo en déduction modulo, il suffit d'appliquer la traduction définie en sous-section 6.2.1. Pour la traduction inverse, il suffit d'utiliser l'admissibilité des règles $\textsc{ConvR}_{\mathcal{R}_2}$ et $\textsc{ConvL}_{\mathcal{R}_2}$ en superdéduction modulo. □

7.2 Hyperpolarisation et Monopolarisation

Parmi les résultats relatifs à la superdéduction que nous avons exposés, deux sont conditionnés par des hypothèses spécifiques. Tout d'abord nous n'avons démontré la complétude des systèmes de superdéduction (modulo) que lorsque les membres droits des règles de prédicats utilisées par le paradigme de superdéduction sont des hyperpôles. De même nous n'avons démontré la complétude des systèmes de superdéduction modulo sans coupure par rapport aux systèmes de déduction modulo sans coupure leur correspondant que lorsque les membres droits des règles de prédicats utilisées par le paradigme de superdéduction sont Π_1 ou Σ_1, ou de manière équivalente lorsque ce sont des monopôles ou des négations de monopôles.

Quand l'ensemble de règles de prédicats ne satisfait pas ces conditions, il est possible de le modifier pour obtenir un ensemble équivalent les satisfaisant : des procédures permettent d'obtenir un ensemble de règles de prédicats équivalent dont les membres droits sont des hyperpôles (Brauner *et al.*, 2007a) ou bien d'obtenir un ensemble de règles de prédicats équivalent dont les membres droits sont des monopôles ou des négations de monopôles (Houtmann, 2008a). Nous appellerons la première *hyperpolarisation* et la deuxième *monopolarisation*. Ces procédures sont inspirées de la *bipolarisation* introduite par Andreoli (2001) dans le cadre des systèmes de focusing et que nous présenterons d'ailleurs brièvement en sous-section 7.3.3.

L'hyperpolarisation est définie comme suit. Si le membre droit φ d'une règle de prédicats $P \longrightarrow \varphi$ n'est pas un hyperpôle, c'est qu'il existe un position ν ainsi qu'un préfixe ν' de ν tels que $\varphi_{|\nu}$ et $\varphi_{|\nu'}$ sont respectivement synchrones et asynchrones ou l'inverse. On remplace alors dans φ l'occurrence ν par un prédicat frais Q (que l'on paramètre par les variables libres de $\varphi_{|\nu'}$) et on rajoute la règle de prédicats $Q \longrightarrow \varphi_{|\nu'}$. On itère ainsi ces remplacements jusqu'à obtenir un ensemble de règles dont les membres droits sont tous des hyperpôles. Considérons par exemple la règle

$$\mathbb{N}(n) \longrightarrow \forall P.(0 \in P \Rightarrow (\forall k.(k \in P \Rightarrow S(k) \in P)) \Rightarrow n \in P) \ .$$

Son membre droit n'est pas un hyperpôle. L'hyperpolarisation la transforme en les règles

$$\begin{cases} \mathbb{N}(n) \longrightarrow \forall P.(0 \in P \Rightarrow \mathfrak{H}(P) \Rightarrow n \in P) \\ \mathfrak{H}(P) \longrightarrow \forall k.(k \in P \Rightarrow S(k) \in P) \ . \end{cases}$$

La monopolarisation est définie comme suit. Si le membre droit φ d'une règle de prédicats $P \longrightarrow \varphi$ n'est ni un monopôle ni la négation d'un monopôle, il en existe néanmoins un préfixe non vide qui est soit un monopôle soit la négation d'un monopôle. On remplace alors φ par ce préfixe où les sous-formules enlevées $(\psi_i)_{1 \leqslant i \leqslant n}$ sont remplacées par des prédicats frais $(Q_i)_{1 \leqslant i \leqslant n}$ (que l'on paramètre par les variables libres des ψ_i) et on rajoute les règles de prédicats $Q_i \longrightarrow \psi_i$. On itère ainsi ces remplacements jusqu'à obtenir un ensemble de règles dont les membres droits sont tous des monopôles ou des négations de monopôles. Considérons par exemple la règle $P \longrightarrow (\forall x.A(x)) \Rightarrow (\forall y.B(y))$. Son membre droit est un hyperpôle mais pas un monopôle. La monopolarisation la transforme de façon non-déterministe en les règles

$$\begin{cases} P \longrightarrow Q \Rightarrow (\forall y.B(y)) \\ Q \longrightarrow \forall x.A(x) \end{cases} \qquad \text{ou bien en} \qquad \begin{cases} P \longrightarrow (\forall x.A(x)) \Rightarrow Q \\ Q \longrightarrow \forall y.B(y) \ . \end{cases}$$

On obtient bien un ensemble de règles de prédicats logiquement équivalent et dont les membres droits sont tous des monopôles ou des négations de monopôles.

7.3 Superdéduction et Focusing

La démonstration que nous avons décrite en section 7.1 s'appuie largement sur l'inversibilité des inférences de LK : parmi les règles d'introduction de connecteurs, seules l'introduction de $\exists$ à droite et l'introduction de $\forall$ à gauche ne sont pas inversibles. Lorsqu'ils apparaissent respectivement en position positive et négative, ces connecteurs sont donc *synchrones* et leur décomposition lors de la construction d'une dérivation du bas vers le haut représente un choix sur lequel on peut vouloir revenir (*to backtrack*). Pour rester complet, il est important de ne pas forcer un tel choix. Par exemple dès que l'on considère une règle de prédicat de la forme $P \quad \rightarrow \quad \forall x.\exists y.\varphi$, l'introduction à droite correspondant à cette règle contient la séquence décomposant les deux quantificateurs. Il est alors impossible de décomposer le $\forall$ sans décomposer le $\exists$: dès que l'on décompose le premier, on choisit de décomposer aussi le deuxième sans pouvoir le décomposer plus tard. Par conséquent la règle d'introduction à droite ainsi obtenue peut ne pas être complète.

Le focusing propose au contraire de grouper les inférences de dérivations en se basant sur leur synchronicité. L'alternance entre phases synchrones puis asynchrones définit des règles d'introduction composites correspondant à des connecteurs synthétiques. Contrairement aux règles de superdéduction, ces règles composites sont toujours complètes. Après avoir introduit le focusing, le multifocusing et les connecteurs synthétiques en sous-sections 7.3.1, 7.3.2 et 7.3.3, nous revisiterons en sous-section 7.3.4 le paradigme de superdéduction modulo en nous basant sur le système de focusing LKF (Liang et Miller, 2007). Nous verrons alors que la complétude du système n'est alors plus problématique.

7.3.1 Focusing en logique classique

Nous présentons dans cette section le paradigme du focusing et le système de déduction avec focus LKF décrit par Liang et Miller (2007). Puisque nous souhaitons adapter l'approche du focusing à la superdéduction et que nous avons présenté les systèmes de superdéduction comme des extensions de LK, c'est naturellement vers LKF, calcul des séquents classiques avec focus, que nous nous tournons pour étudier le focusing.

Le focusing est une approche tout d'abord développée par Andreoli (1992, 2001) dans le cadre de la logique linéraire. Sa première observation consiste à diviser les connecteurs de la logique linéaire en connecteurs *asynchrones* ($\top, \bot, \&, \parr, ?, \forall$) et connecteurs *synchrones* ($1, 0, \otimes, \oplus, !, \exists$). Les premiers sont les connecteurs dont les règles d'introduction (à droite) sont inversibles. Les deuxièmes sont ceux dont les règles d'introduction (à droite) ne sont pas inversibles. En logique linéaire, la dualité de De Morgan fait correspondre à chaque connecteur synchrone un connecteur asynchrone et inversement. Dans le cadre de la logique classique, les connecteurs sont $\top, \bot, \wedge, \vee, \Rightarrow, \forall$ et $\exists$. Comme nous utiliserons dans ce chapitre des calculs des séquents à un seul côté (tout comme au chapitre 4), nous ne considérerons pas la négation comme un connecteur mais comme une fonction transformant toute formule

en sa duale de De Morgan. Notons que dans LK, seules les règles $\exists$R et $\forall$L ne sont pas inversibles. Par conséquent, seuls $\exists$ et $\forall$ *doivent* être respectivement synchrones et asynchrones. Les autres connecteurs peuvent être indifféremment considérés comme synchrones ou asynchrones. L'approche adoptée par le système LKF est de considérer une version synchrone et une version asynchrone de chaque connecteur : on obtient alors les connecteurs synchrones $\top^+, \bot^+, \wedge^+, \vee^+, \Rightarrow^+$ et $\exists$ ainsi que les connecteurs asynchrones $\top^-, \bot^-, \wedge^-, \vee^-, \Rightarrow^-$ et $\forall$. Lorsque le connecteur de tête d'une formule est un connecteur asynchrone (respectivement synchrone), on parle habituellement d'une formule négative (respectivement positive). Néanmoins et pour éviter la confusion avec les occurrences positives et négatives définies en section 2.1, nous parlerons plutôt de formules asynchrones ou synchrones.

La recherche d'une preuve focalisée (*focused proof* en anglais) utilise la division entre connecteurs synchrones et asynchrones : les inférences correspondant aux connecteurs asynchrones peuvent être appliquées du bas vers le haut sans que l'on n'ait jamais recours à un processus de *backtracking* puisque ces inférences sont précisément inversibles. Par conséquent on peut choisir de toujours décomposer les connecteurs asynchrones en premier. C'est lorsqu'il ne reste plus que des formules synchrones dans le séquent que l'on cherche à prouver qu'il faut opérer un véritable choix non-inversible. Seuls ces choix sont sujets au backtracking. Ils consistent à désigner l'une des formules synchrones disponibles dans le séquent puis à la décomposer jusqu'à l'avoir transformée en formules asynchrones. Cette décomposition est appelée une *phase focalisée* ou bien *phase synchrone* et doit être directement suivie par la décomposition gloutonne des connecteurs asynchrones des formules asynchrones qu'elle engendre, décomposition que l'on appelle *phase non-focalisée* ou bien *phase asynchrone*. Remarquons qu'après une phase synchrone relative à une certaine formule synchrone φ, les formules asynchrones décomposées par la phase asynchrone sont forcément des sous-instances de φ. Le processus de recherche de preuve alterne donc phases focalisées correspondant à la décomposition des connecteurs synchrones et phases non-focalisées correspondant à la décomposition des connecteurs asynchrones. Plus précisément, une dérivation consiste toujours en une phase asynchrone décomposant tous les connecteurs asynchrones en tête dans le séquent puis en une série de cycles enchainant une phase synchrone puis une phase asynchrone, chaque cycle correspondant au choix d'une formule synchrone précise du séquent courant. Le focusing étend aussi la division entre connecteurs synchrones et asynchrones aux formules atomiques : une polarité (synchrone ou asynchrone) est arbitrairement attribuée à chaque prédicat. La négation du prédicat reçoit quant à elle la polarité duale. Cette attribution modifie la structure des preuves focalisées sans pour autant menacer la prouvabilité de la formule.

La différence entre phase synchrone et asynchrone est rendue explicite par la syntaxe même des séquents utilisés. Nous utiliserons ici le symbole $\vdash$ pour représenter la phase asynchrone et le symbole $\mapsto$ pour représenter la phase synchrone. Dans la première, les séquents ont la forme $\vdash [\Gamma], \Delta$ où Γ contient les formules synchrones en attente et Δ contient les formules asynchrones actives lors de cette phase asynchrone. Lorsqu'il ne reste plus de formules asynchrones non-atomiques,

$$[]\ \frac{\vdash [\Gamma, C], \Delta}{\vdash [\Gamma], \Delta, C} \qquad \textsc{Focus}\ \frac{\mapsto [S, \Gamma], S}{\vdash [S, \Gamma]} \qquad \textsc{Release}\ \frac{\vdash [\Gamma], A}{\mapsto [\Gamma], A}$$

$$\textsc{Id}^+\ \frac{}{\mapsto [\neg S, \Gamma], S} \qquad \textsc{Id}^-\ \frac{}{\mapsto [A, \Gamma], \neg A}$$

$$\frac{}{\vdash [\Gamma], \Delta, \top^-} \qquad \frac{\vdash [\Gamma], \Delta}{\vdash [\Gamma], \Delta, \bot^-} \qquad \frac{\vdash [\Gamma], \Delta, \varphi \qquad \vdash [\Gamma], \Delta, \psi}{\vdash [\Gamma], \Delta, \varphi \wedge^- \psi}$$

$$\frac{\vdash [\Gamma], \Delta, \varphi, \psi}{\vdash [\Gamma], \Delta, \varphi \vee^- \psi} \qquad \frac{\vdash [\Gamma], \Delta, \neg\varphi, \psi}{\vdash [\Gamma], \Delta, \varphi \Rightarrow^- \psi} \qquad \frac{\vdash [\Gamma], \Delta, \varphi}{\vdash [\Gamma], \Delta, \forall x.\varphi}$$

$$\frac{}{\mapsto [\Gamma], \top^+} \qquad \frac{\mapsto [\Gamma], \varphi \qquad \mapsto [\Gamma], \psi}{\mapsto [\Gamma], \varphi \wedge^+ \psi} \qquad \frac{\mapsto [\Gamma], \varphi_i}{\mapsto [\Gamma], \varphi_1 \vee^+ \varphi_2}$$

$$\frac{\mapsto [\Gamma], \neg\varphi}{\mapsto [\Gamma], \varphi \Rightarrow^+ \psi} \qquad \frac{\mapsto [\Gamma], \psi}{\mapsto [\Gamma], \varphi \Rightarrow^+ \psi} \qquad \frac{\mapsto [\Gamma], \varphi[t/x]}{\mapsto [\Gamma], \exists x.\varphi}$$

FIG. 7.2 – Le système de preuve avec focus LKF (Liang et Miller, 2007)

on entre dans une phase synchrone et le séquent est de la forme $\mapsto [\Gamma], \varphi$ où Γ contient encore une fois les formules synchrones en attente et Δ contient la formule synchrone que l'on a choisi de décomposer lors de cette phase synchrone.

Nous utiliserons les symboles S et A pour représenter respectivement des formules synchrones et asynchrones. Le symbole C représentera une formule synchrone ou un littéral asynchrone. Enfin le symbole Γ représentera des contextes constitués de formules synchrones et de littéraux asynchrones alors que le symbole Δ représentera des contextes arbitraires.

Définition 7.3.1 (LKF). *Le système LKF est décrit en figure 7.2. Les séquents apparaissant à la racine d'une dérivation doivent être de la forme* $\vdash [], \Delta$.

La correction de LKF est immédiate : il suffit d'effacer les informations relatives aux polarités ainsi qu'au focus pour transformer une dérivation dans LKF en dérivation dans LK. La complétude d'un système de focusing est un résultat plus difficile et fondamental : Andreoli (1992) a tout d'abord démontré la complétude de son système de focusing Σ_3 pour la logique linéraire. La complétude de LKF s'énonce comme suit.

Propriété 7.3.1 (Complétude de LKF). *Un séquent* $\vdash \Delta$ *est classiquement valide si et seulement si il existe une dérivation de* $\vdash [], \Delta$ *dans LKF.*

Démonstration. Corollaire du théorème de Kleene (1952b) (propriété 2.3.11). □

7.3.2 Multifocusing en logique classique

Afin de rapprocher focusing et superdéduction, un dispositif va nous être particulièrement utile : il s'agit du *multifocusing*. Ce dispositif a été introduit par Chaud-

huri *et al.* (2008a) dans le but de démontrer comment le paradigme du focusing permet de répondre à la problématique posée par l'identification des preuves en calcul des séquents modulo permutation des inférences. C'est exactement la problématique que nous avons étudiée au chapitre 4 à la différence que leur solution est donnée pour le cadre de la logique linéaire. Bien que nous ne nous poserons pas la question de savoir si cette même approche appliquée à la logique classique permet d'obtenir un résultat similaire quant à cette même problématique, nous utiliserons toutefois ce même mécanisme de multifocusing pour rapprocher les paradigmes du focusing et de la superdéduction. Comme le note Andreoli (2001), L'alternance d'une phase synchrone puis d'une phase asynchrone en focusing correspond à l'introduction d'un bipôle, c'est à dire d'un connecteur composite formé à partir d'une couche asynchrone et d'une couche synchrone et forgé par le focusing en un seul et unique connecteur. C'est ce que Girard (1999, 2000) nomme les *connecteurs synthétiques*. L'intuition que nous voulons formaliser en rapprochant focusing et superdéduction suggère que les connecteurs synthétiques du focusing correspondent effectivement au traitement des règles de prédicats par la superdéduction.

Le principe du multifocusing est d'autoriser la parallélisation des phases synchrones tout en conservant la séquentialisation des phases asynchrones avec les phases synchrones. Plus précisément, une phase synchrone est autorisée à décomposer plusieurs formules synchrones en parallèle. Pour LKF, cela se traduit dans la pratique par une modification de la règle Focus de LKF en

$$\text{MFocus}\ \frac{\mapsto [\Delta, \Gamma], \Delta}{\vdash [\Delta, \Gamma]}$$

où Δ doit contenir au moins une formule positive. Les séquents focalisés ne permettent plus de ne traiter qu'une formule focalisée mais plusieurs et sont donc de la forme $\mapsto [\Gamma], \Delta$ où Δ peut contenir un nombre arbitraire de formules focalisées. Les règles d'inférence traitant des phases synchrones sont modifées en conséquence : il s'agit, en plus de celles décomposant les connecteurs synchrones, des règles Release, Id^+, Id^-. Le système ainsi obtenu à partir de LKF est le suivant.

Définition 7.3.2 (LKMF). *Le système de multifocusing LKMF est le système contenant les inférences de la figure 7.3. La règle MFocus n'est applicable que lorsque Δ contient au moins une formule synchrone. Inversement la règle MRelease n'est appliquable que lorsque Δ ne contient que des formules asynchrones.*

Le système LKMF est correct et complet par rapport à LKF.

Propriété 7.3.2. *Un séquent $\vdash [], \Delta$ est dérivable dans LKF si et seulement si il l'est dans LKMF.*

Démonstration. Chaque règle d'inférence de LKMF contient la règle d'inférence de LKF qui lui correspond. Par conséquent une dérivation dans LKF est aussi une dérivation du même séquent dans LKMF. Inversement si une dérivation d'un séquent $\vdash [], \Delta$ dans LKMF est donnée, on peut aisément la traduire en dérivation dans LK (il

$$[]\ \frac{\vdash [\Gamma, C], \Delta}{\vdash [\Gamma], \Delta, C} \qquad \text{MFocus}\ \frac{\mapsto [\Delta, \Gamma], \Delta}{\vdash [\Delta, \Gamma]} \qquad \text{MRelease}\ \frac{\vdash [\Gamma], \Delta}{\mapsto [\Gamma], \Delta}$$

$$\text{Id}^+\ \frac{}{\mapsto [\neg S, \Gamma], S, \Delta} \qquad \text{Id}^-\ \frac{}{\mapsto [A, \Gamma], \neg A, \Delta}$$

$$\frac{}{\vdash [\Gamma], \Delta, \top^-} \qquad \frac{\vdash [\Gamma], \Delta}{\vdash [\Gamma], \Delta, \bot^-} \qquad \frac{\vdash [\Gamma], \Delta, \varphi \qquad \vdash [\Gamma], \Delta, \psi}{\vdash [\Gamma], \Delta, \varphi \wedge^- \psi}$$

$$\frac{\vdash [\Gamma], \Delta, \varphi, \psi}{\vdash [\Gamma], \Delta, \varphi \vee^- \psi} \qquad \frac{\vdash [\Gamma], \Delta, \neg\varphi, \psi}{\vdash [\Gamma], \Delta, \varphi \Rightarrow^- \psi} \qquad \frac{\vdash [\Gamma], \Delta, \varphi}{\vdash [\Gamma], \Delta, \forall x.\varphi}$$

$$\frac{}{\mapsto [\Gamma], \top^+, \Delta} \qquad \frac{\mapsto [\Gamma], \varphi, \Delta \qquad \mapsto [\Gamma], \psi, \Delta}{\mapsto [\Gamma], \varphi \wedge^+ \psi, \Delta} \qquad \frac{\mapsto [\Gamma], \varphi_i, \Delta}{\mapsto [\Gamma], \varphi_1 \vee^+ \varphi_2, \Delta}$$

$$\frac{\mapsto [\Gamma], \neg\varphi, \Delta}{\mapsto [\Gamma], \varphi \Rightarrow^+ \psi, \Delta} \qquad \frac{\mapsto [\Gamma], \psi, \Delta}{\mapsto [\Gamma], \varphi \Rightarrow^+ \psi, \Delta} \qquad \frac{\mapsto [\Gamma], \varphi[t/x], \Delta}{\mapsto [\Gamma], \exists x.\varphi, \Delta}$$

FIG. 7.3 – Système de preuve avec multifocus LKMF

suffit d'enlever toute information relative au focus, tout comme pour la traduction de LKF vers LK). Alors par complétude de LKF, il existe une dérivation du même séquent dans LKF. □

Généralement, les connecteurs dont l'introduction est inversible sont des connecteurs asynchrones et ceux dont l'introduction ne l'est pas sont synchrones. C'est vrai en particulier dans le cadre de la logique linéaire, par exemple pour les systèmes originaux de focusing d'Andreoli (1992) tout comme pour le système de multifocusing MF de Chaudhuri *et al.* (2008a). Cela devient faux dans le cadre de la logique classique : bien que les connecteurs dont la règle d'introduction est non-inversible *doivent* rester synchrones et bien que, par dualité, les connecteurs asynchrones doivent être introduits par des règles inversibles, des règles d'introduction de connecteurs synchrones peuvent devenir inversibles. C'est le cas par exemple de l'introduction de $\wedge^+$ dans LKF qui est inversible mais introduit un connecteur synchrone. Notons que dans LKF tout comme dans LKMF, les connecteurs $\vee^+$ et $\vee^-$ sont prouvablement équivalents. Néanmoins, utiliser une introduction inversible pour $\vee^+$ sans multifocusing est proscrit car la phase focalisée restreint le focus à une unique formule. Ce n'est plus le cas lorsque l'on utilise le multifocusing et il est tout à fait possible d'écrire un introduction inversible de $\vee^+$ comme suit.

$$\frac{\mapsto [\Gamma], \varphi, \psi, \Delta}{\mapsto [\Gamma], \varphi \vee^+ \psi, \Delta}$$

Il en va de même pour le connecteur $\Rightarrow^+$. Cela nous amène au système de multifocusing suivant.

$$[]\ \frac{\vdash [\Gamma, C], \Delta}{\vdash [\Gamma], \Delta, C} \qquad \text{MFocus}\ \frac{\mapsto [\Delta, \Gamma], \Delta}{\vdash [\Delta, \Gamma]} \qquad \text{MRelease}\ \frac{\vdash [\Gamma], \Delta}{\mapsto [\Gamma], \Delta}$$

$$\text{Id}^+\ \frac{}{\mapsto [\neg S, \Gamma], S, \Delta} \qquad \text{Id}^-\ \frac{}{\mapsto [A, \Gamma], \neg A, \Delta}$$

$$\frac{}{\vdash [\Gamma], \Delta, \top^-} \qquad \frac{\vdash [\Gamma], \Delta}{\vdash [\Gamma], \Delta, \bot^-} \qquad \frac{\vdash [\Gamma], \Delta, \varphi \qquad \vdash [\Gamma], \Delta, \psi}{\vdash [\Gamma], \Delta, \varphi \wedge^- \psi}$$

$$\frac{\vdash [\Gamma], \Delta, \varphi, \psi}{\vdash [\Gamma], \Delta, \varphi \vee^- \psi} \qquad \frac{\vdash [\Gamma], \Delta, \neg\varphi, \psi}{\vdash [\Gamma], \Delta, \varphi \Rightarrow^- \psi} \qquad \frac{\vdash [\Gamma], \Delta, \varphi}{\vdash [\Gamma], \Delta, \forall x.\varphi}$$

$$\frac{}{\mapsto [\Gamma], \top^+, \Delta} \qquad \frac{\mapsto [\Gamma], \Delta}{\mapsto [\Gamma], \bot^+, \Delta} \qquad \frac{\mapsto [\Gamma], \varphi, \Delta \qquad \mapsto [\Gamma], \psi, \Delta}{\mapsto [\Gamma], \varphi \wedge^+ \psi, \Delta}$$

$$\frac{\mapsto [\Gamma], \varphi_1, \varphi_2, \Delta}{\mapsto [\Gamma], \varphi_1 \vee^+ \varphi_2, \Delta} \qquad \frac{\mapsto [\Gamma], \neg\varphi, \psi, \Delta}{\mapsto [\Gamma], \varphi \Rightarrow^+ \psi, \Delta} \qquad \frac{\mapsto [\Gamma], \varphi[t/x], \Delta}{\mapsto [\Gamma], \exists x.\varphi, \Delta}$$

FIG. 7.4 – Système de preuve avec multifocus LKMF1

Définition 7.3.3 (LKMF1). *Le système de multifocusing LKMF1 est le système contenant les inférences de la figure 7.4.*

LKMF1 est correct est complet par rapport à LK, LKF, et LKMF.

Propriété 7.3.3. *Un séquent* $\vdash [], \Delta$ *est dérivable dans LKMF si et seulement si il l'est dans LKMF1.*

Démonstration. La correction de LKMF1 par rapport à LK est directe : il suffit simplement d'effacer toute information relative au focus pour transformer une dérivation dans LKMF1 en dérivation dans LK. La complétude par rapport à LKMF provient de l'admissibilité de l'affaiblissement. □

Au chapitre 6 (comme d'ailleurs aux chapitres 2 et 5), nous avons toujours choisi d'utiliser au maximum des règles inversibles, en particulier pour introduire la disjonction à droite en logique classique. Par conséquent, LKMF1 est un système qui nous rapproche déjà du paradigme de la superdéduction. Nous allons donc l'utiliser comme pilier pour définir un paradigme mêlant superdéduction et (multi)focusing.

7.3.3 Connecteurs synthétiques

En écrivant une version synchrone et une version asynchrone de chaque connecteur neutre (par exemple $\wedge^-$ et $\wedge^+$ pour $\wedge$), nous obtenons un langage où il n'y a plus de connecteur neutre, mais seulement des connecteurs synchrones ou asynchrones. Néamoins les définitions de monopôles et de bipôles (définition 6.1.6) restent inchangées par rapport à la neutralité, synchronicité ou asynchronicité des connecteurs. Nous utiliserons les notations π_1 et σ_1 pour les notions de monopôles et négation

$$\dfrac{\dfrac{\dfrac{[]^* \dfrac{\vdash [\Delta, S, \Gamma_1]}{\vdash [\Delta, S], \Gamma_1} \quad \ldots \quad []^* \dfrac{\vdash [\Delta, S, \Gamma_n]}{\vdash [\Delta, S], \Gamma_n}}{\text{phase asynchrone}}}{\text{phase synchrone}}}{\text{(M)Focus}\ \dfrac{\mapsto [\Delta, S], S}{\vdash [\Delta, S]}}$$

FIG. 7.5 – Introduction d'un connecteur Synthétique correspondant à un bipôle

de monopôles relatives à l'ancienne définition de synchronicité. Pour les notions de monopôles et de négation de monopôle relatives à la nouvelle définition de synchronicité, nous utiliserons les notations $\mathfrak{A}_1$ et $\mathfrak{S}_1$. Par mimétisme avec les notations relatives à la hierarchie arithmétique, nous noterons $\mathfrak{A}_2$ les bipôles et $\mathfrak{S}_2$ les formules dont la négation est un bipôle. (Cette notation peut s'étendre à tout $n \in \mathbb{N}$. Aucune formule n'est $\mathfrak{A}_0$ ou $\mathfrak{S}_0$ puisqu'aucun connecteur n'est neutre.)

Si S est un bipôle, son traitement par un cycle synchrone/asynchrone définit en fait une inférence composite introduisant en un pas tous les connecteurs de S. Cette inference est de la forme

$$\dfrac{\vdash [\Delta, S, \Gamma_1] \quad \ldots \quad \vdash [\Delta, S, \Gamma_n]}{\vdash [\Delta, S]}\ .$$

Elle représente les dérivations décrite en figure 7.5 et correspond à l'introduction d'un connecteur composite formé des connecteurs de S et que Girard nomme *connecteur synthétique*. La complétude de LKF peut s'exprimer par la complétude de ces inférences par rapport à LK. Si S n'est pas un un bipôle mais une formule synchrone quelconque, Andreoli propose de plus de la transformer en bipôle de la façon suivante : S s'écrit $C[S_1, \ldots, S_n]$ pour un certain contexte bipolaire C et pour S_i synchrone pour tout i. La formule S est alors remplacée par le bipôle $C[P_1, \ldots, P_n]$ où les P_i sont des noms frais que l'on suppose équivalents respectivement aux S_i. On peut donc ainsi obtenir une dérivation qui n'utilise que des introductions de connecteurs synthétiques sur des propositions atomiques. Andreoli (2001) nomme cette procédure *bipolarisation*. Les procédures d'hyperpolarisation et de monopolarisation (section 7.2) sont inspirées de la bipolarisation.

7.3.4 Superdéduction avec focus

Nous allons à présent étendre LKMF1 afin d'obtenir un système similaire à la superdéduction mais basé sur le focusing et plus précisément LKMF1. Considérons un ensemble $\mathcal{R}$ de règles de prédicats de la forme $P \rightarrow \varphi$. Ici φ est une formule polarisée utilisant les connecteurs polarisés de LKMF1. Si φ est une formule synchrone, nous dirons que la règle de prédicat est synchrone alors que si φ est asynchrone, nous dirons qu'elle est asynchrone.

Définition 7.3.4 ($\mathrm{ff}_{\mathcal{R}}$). *Si $\mathcal{R}$ est un ensemble de règles de prédicats, le système déductif de focused fold associé à $\mathcal{R}$ noté $\mathit{ff}_{\mathcal{R}}$ est le système LKMF1 auquel sont rajoutées les inférences*

$$\textit{SFold}^{+}\ \frac{\mapsto [\Gamma, P\sigma], \varphi\sigma}{\vdash [\Gamma, P\sigma]} \qquad \textit{et} \qquad \textit{AFold}^{-}\ \frac{\vdash [\Delta, \neg P\sigma], \neg\varphi\sigma}{\vdash [\Delta, \neg P\sigma]}$$

pour tout $P \to \varphi \in \mathcal{R}$ synchrone et

$$\textit{SFold}^{-}\ \frac{\mapsto [\Gamma, \neg P\sigma], \neg\varphi\sigma}{\vdash [\Gamma, \neg P\sigma]} \qquad \textit{et} \qquad \textit{AFold}^{+}\ \frac{\vdash [\Delta, P\sigma], \varphi\sigma}{\vdash [\Delta, P\sigma]}$$

pour tout $P \to \varphi \in \mathcal{R}$ asynchrone.

Si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont respectivement un ensemble de règles de termes et de prédicats ainsi qu'un ensemble de règles de prédicats, nous avons défini (en section 6.3) le système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$ comme le système de superdéduction associé à $\mathcal{R}_2$ auquel sont rajoutés les règles ConvR et ConvL correspondant à $\mathcal{R}_1$. On peut aussi rajouter ces conversions à $\mathrm{ff}_{\mathcal{R}_2}$ pour obtenir un système de focusing comparable au système de superdéduction modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$. Toutefois pour que ces étapes de conversion n'interagissent pas avec le mécanisme de focusing en modifiant la polarité des formules, nous supposerons que $\mathcal{R}_1$ ne contient que des règles de termes. Puisque $\mathrm{ff}_{\mathcal{R}_2}$ est en fait un calcul des séquents à un seul côté, il suffit en fait de rajouter la règle de conversion à droite.

Définition 7.3.5 ($\mathrm{ff}^{\mathcal{R}_1}_{\mathcal{R}_2}$). *Si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont respectivement un ensemble de règles de termes ainsi qu'un ensemble de règles de prédicats, le système déductif de focused fold modulo associé à $(\mathcal{R}_1, \mathcal{R}_2)$ est le système $\mathit{ff}_{\mathcal{R}_2}$ auquel sont rajoutées les inférences*

$$\textit{Conv}\ \frac{\vdash [\Delta, \psi]}{\vdash [\Delta, \varphi]}$$

conditionnées par $\varphi \equiv \psi$ où $\equiv$ est la congruence engendrée par $\mathcal{R}_1$.

Comparaison avec la superdéduction

Lorsque $\mathcal{R}$ ne contient que des règles de prédicats dont les membres droits sont $\mathfrak{S}_1$ ou $\mathfrak{A}_1$, le système $\mathrm{ff}_{\mathcal{R}}$ associé à un ensemble de règles de prédicats $\mathcal{R}$ contient à la fois le focusing et des connecteurs synthétiques correspondant aux règles de superdéduction associées à $\mathcal{R}$. Considérons par exemple la règle de prédicats

$$x \subseteq y \ \to\ \forall z.(z \in x \Rightarrow^{-} z \in y)\ .$$

La partie droite de cette règle est bien $\mathfrak{A}_1$; la règle est donc asynchrone et les règles de superdéduction

$$\subseteq\mathrm{R}\ \frac{\Gamma, z \in t \vdash z \in u, \Delta}{\Gamma \vdash t \subseteq u, \Delta} \qquad \text{et} \qquad \subseteq\mathrm{L}\ \frac{\Gamma \vdash v \in t, \Delta \qquad \Gamma, v \in u, \Delta}{\Gamma, t \subseteq v \vdash \Delta}$$

$$\textsc{SFold}^{-}\ \dfrac{\textsc{MRelease}\ \dfrac{[]\ \dfrac{\vdash [\Gamma, \neg t \subseteq u, v \in t]}{\vdash [\Gamma, \neg t \subseteq u], v \in t}}{\mapsto [\Gamma, \neg t \subseteq u], v \in t} \qquad \textsc{MRelease}\ \dfrac{[]\ \dfrac{\vdash [\Gamma, \neg t \subseteq u, \neg v \in u]}{\vdash [\Gamma, \neg t \subseteq u], \neg v \in u}}{\mapsto [\Gamma, \neg t \subseteq u], \neg v \in u}}{\dfrac{\mapsto [\Gamma, \neg t \subseteq u], v \in t \mathbin{\wedge^{+}} \neg v \in u}{\mapsto [\Gamma, \neg t \subseteq u], \exists z.(z \in t \mathbin{\wedge^{+}} \neg z \in u)}}}{\vdash [\Gamma, \neg t \subseteq u]}$$

$$\textsc{AFold}^{+}\ \dfrac{\dfrac{\dfrac{[]\ \dfrac{[]\ \dfrac{\vdash [\Gamma, t \subseteq u, \neg z \in t, z \in u]}{\vdash [\Gamma, t \subseteq u, \neg z \in t], z \in u}}{\vdash [\Gamma, t \subseteq u], \neg z \in t, z \in u}}{\vdash [\Gamma, t \subseteq u], z \in t \Rightarrow^{-} z \in u}}{\vdash [\Gamma, t \subseteq u], \forall z.(z \in t \Rightarrow^{-} z \in u)}}{\vdash [\Gamma, t \subseteq u]}$$

FIG. 7.6 – Connecteur synthétique pour $\subseteq$

correspondent effectivement aux connecteurs synthétiques en figure 7.6. Dualement, si le membre droit d'une règle de prédicats est $\mathfrak{S}_1$, les connecteurs synthétiques correspondront de même aux règles de superdéduction associées à cette règle de prédicats.

Remarquons que si φ est une formule π_1, alors on peut la transformer en formule $\mathfrak{A}_1$ en attribuant à tous ses connecteurs la polarité asynchrone. Nous noterons la formule obtenue φ^{-}. De même si φ est σ_1, alors on peut la transformer en formule $\mathfrak{S}_1$ en attribuant à tous ses connecteurs la polarité synchrone. Nous noterons la formule obtenue φ^{+}. Par ailleurs, si $\mathcal{R}$ est un ensemble de règles de prédicats dont les membres droits sont tous π_1 ou σ_1, alors on peut toujours le transformer en un ensemble $\mathcal{R}^{\pm}$ de règles de prédicats dont les membres droits sont tous $\mathfrak{A}_1$ ou $\mathfrak{S}_1$ (en remplaçant tout membre droit φ par φ^{-} si φ est π_1 et par φ^{+} si φ est σ_1). Le système $\mathrm{ff}_{\mathcal{R}^{\pm}}$ contient alors les inférences de superdéduction associées à $\mathcal{R}$.

Propriété 7.3.4. *Si $P \rightarrow \varphi$ est une règle de prédicat telle que φ est π_1 (respectivement σ_1), alors les connecteurs synthétiques associés à la règle de prédicats $P \rightarrow \varphi^{-}$ (respectivement $P \rightarrow \varphi^{+}$) dans $\mathit{ff}_{\mathcal{R}}$ correspondent aux introductions de P respectivement à droite et à gauche en superdéduction.*

Démonstration. Par induction sur le calcul des inférences de superdéduction correspondant à la règle $P \rightarrow \varphi$. L'invariant à utiliser est qu'une étape de ce calcul correspond à une décomposition partielle du connecteur synthétique associé. □

Nous sommes alors en mesure de démontrer le résultat d'adéquation suivant.

Propriété 7.3.5. *Si les membres droits des règles d'un ensemble $\mathcal{R}$ de règles de prédicats sont tous π_1 ou σ_1, alors un séquent $\vdash \Delta$ est dérivable sans coupure dans le*

système de superdéduction associé à $\mathcal{R}$ si et seulement si $\vdash [], \Delta$ est dérivable dans le système $ff_{\mathcal{R}^{\pm}}$.

Démonstration. C'est un corollaire direct de la propriété 7.3.4. □

Enfin les systèmes de focused fold modulo sont en adéquation avec les systèmes de superdéduction modulo.

Propriété 7.3.6. *Si les membres droits des règles de l'ensemble $\mathcal{R}_2$ de règles de prédicats sont tous π_1 ou σ_1, alors un séquent est dérivable sans coupure dans le système de superdéduction associé à $(\mathcal{R}_1, \mathcal{R}_2)$ si et seulement si il est dérivable dans le système $ff^{\mathcal{R}_1}_{\mathcal{R}_2^{\pm}}$.*

Démonstration. C'est un corollaire direct de la propriété 7.3.5. □

Qu'en est-t-il du traitement des règles de prédicats dont les membres droits ne sont pas π_1 ou σ_1. En superdéduction, **(1)** on peut choisir d'utiliser la procédure de monopolarisation de la section 7.2 pour obtenir un système de règles de prédicats dont les membres droits sont tous σ_1 ou π_1 et induisant ainsi un système de superdéduction sans coupure complet par rapport à la déduction modulo sans coupure (propriété 7.1.9) ; **(2)** on peut choisir d'utiliser la procédure d'hyperpolarisation de la section 7.2 pour obtenir un système de règles de prédicats dont les membres droits sont tous des hyperpôles et induisant ainsi un système de superdéduction complet par rapport à LK (propriété 6.1.5) ou enfin **(3)** on peut choisir de calculer directement les nouvelles inférences en utilisant ces règles telles quelles. Dans le premier cas, nous retombons sur un système de règles de prédicats dont les membres droits sont tous σ_1 ou π_1. Considérons le second cas et par exemple la règle $P \quad \rightarrow \quad (\exists x.R(x)) \Rightarrow (\exists y.Q(y))$ dont le membre droit est bien un hyperpôle. Les inférences de superdéduction qui lui sont associées sont

$$\frac{\Gamma, R(x) \vdash Q(t), \Delta}{\Gamma \vdash P, \Delta}\ x \notin \mathcal{FV}(\Gamma, t, \Delta) \quad \text{et} \quad \frac{\Gamma, Q(y) \vdash \Delta \quad \Gamma \vdash R(t), \Delta}{\Gamma, P \vdash \Delta}\ y \notin \mathcal{FV}(\Gamma, \Delta)\ .$$

Pour obtenir des connecteurs synthétiques de focused fold, il faut attribuer une polarité à tous les connecteurs du membre droit de cette règle, c'est-à-dire l'implication. On obtient soit la règle synchrone $P \quad \rightarrow \quad (\exists x.R(x)) \Rightarrow^{+} (\exists y.Q(y))$, soit la règle asynchrone $P \quad \rightarrow \quad (\exists x.R(x)) \Rightarrow^{-} (\exists y.Q(y))$. Les connecteurs synthétiques de focused fold correspondant à la première sont ceux de la figure 7.7. Le premier correspond bien à la première règle de superdéduction. Par contre et contrairement aux règles de superdéduction, le deuxième ne décompose pas le quantificateur existentiel $\exists x.R(x)$ car celui-ci est synchrone dans une phase asynchrone. De même si l'on donne la polarité asynchrone à l'implication, on obtient les connecteurs synthétiques de la figure 7.8. Cette fois c'est le deuxième qui correspond bien à la deuxième règle de superdéduction. Par contre et contrairement à aux règles de superdéduction, le premier ne décompose pas le quantificateur existentiel $\exists y.Q(y)$ car celui-ci est synchrone dans une phase asynchrone.

$$\textsc{sFold}^{+}\ \dfrac{\textsc{Release}\ \dfrac{[]^{*}\ \dfrac{\vdash [\Delta, P, \neg R(x), Q(t)]}{\vdash [\Delta, P], \neg R(x), Q(t)}}{\vdash [\Delta, P], \forall x.\neg R(x), Q(t)}\ x \notin \mathcal{FV}(\Delta, t)}{\dfrac{\dfrac{\mapsto [\Delta, P], \forall x.\neg R(x), Q(t)}{\mapsto [\Delta, P], \forall x.\neg R(x), \exists y.Q(y)}}{\mapsto [\Delta, P], (\exists x.R(x)) \Rightarrow^{+} (\exists y.Q(y))}}{\vdash [\Delta, P]}$$

$$\textsc{aFold}^{-}\ \dfrac{[]\ \dfrac{\vdash [\Delta, \neg P, \exists x.R(x)]}{\vdash [\Delta, \neg P], \exists x.R(x)} \qquad []\ \dfrac{\dfrac{\vdash [\Delta, \neg P, \neg Q(y)]}{\vdash [\Delta, \neg P], \neg Q(y)}}{\vdash [\Delta, \neg P], \forall y.\neg Q(y)}\ y \notin \mathcal{FV}(\Delta)}{\vdash [\Delta, \neg P], (\exists x.R(x)) \wedge^{-} (\forall y.\neg Q(y))}}{\vdash [\Delta, \neg P]}$$

FIG. 7.7 – Connecteurs synthétiques pour $P \quad \rightarrow \quad (\exists x.R(x)) \Rightarrow^{+} (\exists y.Q(y))$

$$\textsc{aFold}^{+}\ \dfrac{[]^{*}\ \dfrac{\vdash [\Delta, P, \neg R(x), \exists y.Q(y)]}{\vdash [\Delta, P], \neg R(x), \exists y.Q(y)}}{\dfrac{\vdash [\Delta, P], \forall x.\neg R(x), \exists y.Q(y)}{\vdash [\Delta, P], (\exists x.R(x)) \Rightarrow^{-} (\exists y.Q(y))}}\ x \notin \mathcal{FV}(\Delta)}{\vdash [\Delta, P]}$$

$$\textsc{sFold}^{-}\ \dfrac{\textsc{Release}\ \dfrac{[]\ \dfrac{\vdash [\Delta, \neg P, R(t)]}{\vdash [\Delta, \neg P], R(t)}}{\mapsto [\Delta, \neg P], R(t)}}{\dfrac{\mapsto [\Delta, \neg P], \exists x.R(x)}{\mapsto [\Delta, \neg P], (\exists x.R(x)) \wedge^{+} (\forall y.\neg Q(y))}}{\vdash [\Delta, \neg P]}\ \dfrac{[]\ \dfrac{\vdash [\Delta, \neg P, \neg Q(y)]}{\vdash [\Delta, \neg P], \neg Q(y)}}{\vdash [\Delta, \neg P], \forall y.\neg Q(y)}}{\mapsto [\Delta, \neg P], \forall y.\neg Q(y)}\ y \notin \mathcal{FV}(\Delta)$$

FIG. 7.8 – Connecteurs synthétiques pour $P \quad \rightarrow \quad (\exists x.R(x)) \Rightarrow^{-} (\exists y.Q(y))$

En fait, lorsque l'on considère une règle de prédicats $P \to \varphi$ où φ est un hyperpôle, alors φ est de la forme $C[S_1, \ldots, S_n, A_1, \ldots, A_n]$ où C est un contexte neutre, les S_i sont des formules σ_1 et les A_i sont des formules π_1. Si l'on choisit d'attribuer la polariét synchrone à tous les connecteurs de C, on obtient bien un bipôle ψ (c'est-à-dire une formule $\mathfrak{S}_2$). Le connecteur synthétique introduisant P correspondra alors à l'introduction de P à droite en superdéduction mais le connecteur synthétique introduisant $\neg P$ ne décomposera que la couche asynchrone de $\neg\varphi$ contrairement à l'introduction de P à gauche en superdéduction qui décompose tous les connecteurs de $(\neg)\varphi$. Inversement si l'on choisit d'attribuer la polarité asynchrone à tous les connecteurs de C, on obtient bien la négation ψ' d'un bipôle (c'est-à-dire une formule $\mathfrak{A}_2$). Le connecteur synthétique introduisant $\neg P$ correspondra alors à l'introduction de P à gauche en superdéduction mais le connecteur synthétique introduisant P ne décomposera que la couche asynchrone de φ contrairement à l'introduction de P à droite en superdéduction qui décompose tous les connecteurs de φ.

Complétude du focused fold

Nous venons de montrer que les connecteurs synthétiques des systèmes de focused fold regroupent en général moins d'inférences que les règles de superdéduction qui leur correspondent. Cette restriction assure néanmoins la complétude des systèmes de focused fold par rapport aux systèmes de déduction modulo sans coupure (la correction étant triviale).

Propriété 7.3.7. *Soit $\mathcal{R}_1$ un ensemble de règles de termes et $\mathcal{R}_2$ un ensemble de règles de prédicats. Soit $\equiv$ la congruence engendrée par $\mathcal{R}_1 \cup \mathcal{R}_2$. Alors $\vdash \Delta$ est dérivable dans $LK_\equiv$ si et seulement si $\vdash [], \Delta$ est dérivable dans $ff^{\mathcal{R}_1}_{\mathcal{R}_2}$.*

Démonstration. Par complétude de LKMF1 par rapport à LK (propriétés 7.3.1, 7.3.2 et 7.3.3) et en y rajoutant les conversions : Conv correspond à la conversion pour $\mathcal{R}_1$ et SFold$^+$, SFold$^-$, AFold$^+$ et AFold$^-$ correspondent à la conversion pour $\mathcal{R}_2$. □

La déduction modulo étant toujours correcte et complète par rapport à LK (propriété 5.2.1), nous en déduisons la correction et complétude des systèmes de focused fold.

Propriété 7.3.8. *Soit $\mathcal{R}_1$ un ensemble de règles de termes et $\mathcal{R}_2$ un ensemble de règles de prédicats. Alors $\mathcal{T}h_{\mathcal{R}_1}, \mathcal{T}h_{\mathcal{R}_2} \vdash \Delta$ est dérivable dans LK si et seulement si $\vdash [], \Delta$ est dérivable dans $ff^{\mathcal{R}_1}_{\mathcal{R}_2}$.*

Démonstration. Il suffit de combiner les propriétés 5.2.1 et 7.3.7. □

Conclusion

Mes recherches apportent un point de vue nouveau sur la représentation, la manipulation et l'interaction des preuves et ouvrent les prespectives que je vais maintenant exposer.

Admissibilité des coupures en superdéduction modulo

Au chapitre 7, nous avons démontré l'équivalence entre déduction modulo sans coupure et superdéduction modulo sans coupure sous une hypothèse de synchronicité. Si cette hypothèse est bel et bien suffisante, je ne sais pas si elle est nécessaire. En effet je n'ai pas pu conclure sur l'équivalence entre admissibilité des coupures en superdéduction modulo et en déduction modulo dans le cas général. Voici donc une première question posée par cette thèse.

Représentation canonique des preuves

Une problématique conditionne tous les développements de cette thèse : il s'agit de la représentation des preuves, enjeu principal de la théorie de la démonstration. Les systèmes de Gentzen constituent des outils puissants permettant d'étudier la structure des preuves et de formaliser leur interaction. Les démonstrations y sont néanmoins assujetties à une puissante *bureaucratie de la syntaxe*. Notamment en calcul des séquents, une procédure d'élimination des coupures ne peut être confluente tout en restant générale, à moins de payer le prix de l'identification des dérivations qui ne diffèrent que par des permutations d'inférences. Plusieurs approches répondent partiellement à cette problématique en proposant des représentations canoniques des démonstrations du calcul des séquents : Par exemple Lamarche et Straßburger (2005) définissent des réseaux de preuve pour la logique propositionnelle classique ainsi qu'une prodédure d'élimination des coupures *convergente* en restant générale. Une autre approche concluante est celle de Chaudhuri *et al.* (2008a) qui proposent d'utiliser un multi-focusing maximal pour structurer les dérivations en calcul des séquents pour la logique linéaire multiplicative/additive. Étendre ces approches à la logique des prédicats classique permettraient de répondre à cette nécessité de représentants canoniques. Plusieurs travaux en cours (Heijltjes, 2009; McKinley, 2009) font état des difficultés posées par ce problème.

Théories axiomatiques

Le traitement des axiomes se trouve largement amélioré lorsque des paradigmes comme la déduction modulo ou la superdéduction sont utilisés. Ce traitement apporte un regard nouveau sur le débat philosophique sur la position des axiomes et du calcul dans le raisonnement tout en proposant des solutions sur les plans théorique et technique. La normalisation forte ou encore l'admissibilité des coupures étaient à l'origine des propriétés se rapportant aux systèmes logiques de Gentzen : Elles sont vérifiées par les systèmes NJ, NK, LJ et LK et impliquent certaines propriétés devenues indispensables comme la propriété du témoin ou la complétude de méthodes de recherche de preuve. À la lumière de la déduction modulo et de la superdéduction, normalisation forte et élimination des coupures apparaissent comme des propriétés intrinsèques des théories logiques. Elles ne sont plus vérifiées par le système déductif seul mais par l'extension du système déductif par le pouvoir calculatoire et déductif d'une théorie. Elles sont donc promues au rang de critères discriminant et permettent de garantir la bonne utilisation de ces pouvoirs calculatoires et déductifs. La superconsistance se pose comme un critère universel (Dowek, 2006) permettant d'assurer ces propriétés fondamentales en se plaçant à la convergence des anciens univers syntaxique et sémantique : le monde syntaxique est personnifié par les candidats de réducibilité pour la déduction naturelle et le monde sémantique est incarné par le cadre des pré-algèbres de Heyting dans lequel la notion de candidats s'insère. La normalisation forte en calcul des séquents est un résultat au moins plus général que celle en déduction naturelle et qui peut aussi être démontré en utilisant des candidats de réducibilité symétrisés comme le propose Urban (2000). La syntaxe et la sémantique opérationnelle (c'est-à-dire la procédure d'élimination des coupures) diffèrent de celles de la déduction naturelle. Il faut donc étudier dans quelle mesure ces candidats s'inscrivent dans une notion de modèle en logique intuitionniste ou classique. Ils semblent proposer un cadre plus large que les candidats du lambda-calcul et permettraient donc l'étude des propriétés intrinsèques de théories dans un spectre plus étendu de systèmes déductifs.

Calculs classiques

La correspondance de Griffin entre opérateurs de contrôle et logique classique ainsi que les développements qui l'ont suivie proposent un challenge que les futurs assistants de preuve vont devoir relever : il s'agit de transposer cette correspondance à la pratique en implémentant effectivement des procédures d'extraction de programmes à partir de preuves écrites en logique classique. Le calcul d'Urban et le $\bar{\lambda}\mu\tilde{\mu}$-calcul se posent comme deux formalismes permettant de représenter les dérivations en calcul des séquents classique ou intuitionniste. Alors que le premier est extrêmement proche des dérivations du calcul des séquents, le deuxième se rapproche des langages de programmation avec opérateurs de contrôle et permet, du point de vue logique, de formaliser des dérivations *focalisées*. L'étude des liens entre super-

déduction et focusing que nous avons développée au chapitre 7 ouvre la perspective suivante : Un ingrédient primordial pour écrire des inférences similaires à celles des systèmes de superdéduction en focusing est le *multifocusing*. Par conséquent décrire une extension du $\bar{\lambda}\mu\tilde{\mu}$-calcul avec multifocusing se révélera particulièrement utile : en partant de cette extension, on pourra écrire un langage de termes de preuves basés sur le $\bar{\lambda}\mu\tilde{\mu}$-calcul pour les systèmes de focused fold permettant ainsi un traitement adéquat des axiomes déductifs similaire à celui des systèmes de superdéduction.

Unité de la logique

Tout comme les langages de programmation, les langages de preuves et les systèmes logiques voient constamment leur nombre augmenter. L'une des caractéristiques de mon travail de thèse est l'utilisation d'un nombre important de ces systèmes : déduction naturelle et calculs des séquents ; logique classique ou intuitionniste (ou linéaire) ; propositionnelle ou des prédicats (ou d'ordre supérieur) ; déduction modulo, superdéduction et focusing ; dérivations et réseaux de preuve. Les formalismes que nous avons considérés sont nombreux mais les traductions et encodages aussi : Les preuves en déduction naturelle se traduisent en preuves en calcul des séquents et inversement. Les logiques linéaire, intuitionniste et classique se combinent par exemple dans le calcul des séquents LU de Girard (1993). Liang et Miller (2009) ont proposé un système de focusing LKU contenant la logique classique, intuitionniste et linéaire-multiplicative/additive (MALL) comme fragments. Les Pure Type Systems peuvent aussi être encodés en déduction supernaturelle modulo (Burel, 2008) ou dans $\lambda\pi$ modulo (Cousineau et Dowek, 2007). J'ai fait le lien déduction modulo (sans coupure) et superdéduction modulo (sans coupure), entre focusing et superdéduction. De tels formalismes permettent à la fois une expressivité logique importante et un contrôle considérable de la structure des preuves. Ce sont là des arguments substantiels montrant que ces paradigmes pourront servir de fondements à de nouvelles approches permettant d'unifier d'autres fragments logiques dans des formalismes plus généraux.

Recherche de preuve

La déduction modulo est un paradigme qui permet d'obenir des procédures de recherche de preuve efficaces, comme par exemple ENAR (Dowek *et al.*, 2003) et TaMeD (Bonichon, 2006, 2004; Bonichon et Hermant, 2006b). Dans un premier temps, nous avons montré que TaMeD s'adaptait en une méthode des tableaux pour la superdéduction modulo que nous avons exposée en section 6.4. Bien que ce ne soit pas encore formalisé, on peut aussi étendre ENAR à la superdéduction modulo. D'autres méthodes de recherche de preuve, comme la méthode inverse qui s'accorde par exemple avec le focusing (Chaudhuri *et al.*, 2008b), doivent être adaptées à la superdéduction modulo. Ces méthodes doivent ensuite être implémentées. La pratique et les résultats expérimentaux qui seront ainsi obtenus constitueront une autre

validation de l'approche de la superdéduction modulo.

Assistants de preuve

Les multiples systèmes déductifs que j'ai étudiés dans cette thèse doivent à présent prendre part à la fondation d'une nouvelle génération d'assistants de preuve. Lors de l'implémentation d'un assistant de preuve, la représentation des preuves est un choix crucial : cela conditionne tous les aspects de l'assistant comme par exemple sa complexité, son expressivité ou ses niveaux d'automatisation des mécanismes de recherche de preuve. D'autre part, pour un assistant de preuve, disposer d'une importante librairie de théorèmes et de preuves est un atout considérable : c'est cette librairie qui fournit un pouvoir déductif plus ou moins imporant à l'utilisateur au même titre que le système déductif lui-même. La facilité d'utilisation d'un tel pouvoir est par conséquent primordiale. En ce sens la déduction modulo et la superdéduction sont deux formalismes très bien placés pour le développement futur de nouveaux assistants. L'évolution actuel des assistants de preuve tend à accroître la modularité des prouveurs. Certains permettent une meilleure automatisation de la recherche de preuve, d'autres se voient accorder plus de confiance à leur mécanisme de vérification des preuves, d'autres ont des librairies d'axiomes et de théorèmes plus développées. Comme il est difficile d'assurer toutes ces qualités pour un seul et même prouveur, un solution confortable constite à combiner différents prouveurs appréciés pour des qualités différentes voire totalement dédiés à une tâche particulière. Notons par exemple le développement actuel de Dedukti, un *proof checker* universel (The Dedukti Development Team, 2009) fondé sur le système déductif $\lambda\pi$ modulo et spécialisé dans la vérification de preuves construites par d'autres outils (assistants de preuve tels que Coq ou outils de démonstration automatique). On peut aussi chercher à combiner des prouveurs automatiques et des assistants de preuve interactifs (Bezem *et al.*, 2002; Prevosto, 2005). L'enjeu sous-jacent de tels développements consiste à savoir traduire les preuves d'un formalisme à l'autre. Encore une fois, l'expressivité et le contrôle que fournit un système déductif, comme $\lambda\pi$ modulo, la superdéduction modulo ou encore le focused fold, sont des atouts importants lui permettant d'encoder voire de contenir un maximum de formalismes et donc d'intéragir avec un maximum d'autres implémentations. Cette évolution souligne encore l'importance des avancées actuelles concernant l'unité de la logique.

Table des figures

Bibliographie

Martin Abadi, Lucas Cardelli, Pierre-Louis Curien et Jean-Jacques Lévy : Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991. 3, 6, 32

Jean-Marc Andreoli : Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992. 4, 7, 223, 231, 233, 235

Jean-Marc Andreoli : Focussing and proof construction. *Ann. Pure Appl. Logic*, 107(1-3):131–163, 2001. 4, 7, 223, 224, 230, 231, 234, 237

Jean-Marc Andreoli et Roberto Maieli : Focusing and proof-nets in linear and non-commutative logic. *In LPAR*, volume 1705 de *LNCS*, pages 321–336. Springer, 1999. 4, 7

Steffen van Bakel, Stéphane Lengrand et Pierre Lescanne : The language chi : Circuits, computations and classical logic. *In ICTCS*, pages 81–96, 2005. 119, 198

Steffen van Bakel et Pierre Lescanne : Computation with classical sequents. *Mathematical Structures in Computer Science*, 18(3):555–609, 2008. 198

Paolo Baldan, Clara Bertolissi, Horatiu Cirstea et Claude Kirchner : A rewriting calculus for cyclic higher-order term graphs. *Mathematical Structures in Computer Science*, 17(3):363–406, 2007. 45

Emilie Balland, Paul Brauner, Radu Kopetz, Pierre-Etienne Moreau et Antoine Reilles : Tom : Piggybacking rewriting on java. *In RTA*, pages 36–47, 2007. 220

Hendrik Pieter Barendregt : *The Lambda Calculus – Its Syntax and Semantics*, volume 103. North-Holland, 1984. 3, 6, 18, 33, 36, 38

Gilles Barthe, Horatiu Cirstea, Claude Kirchner et Luigi Liquori : Pure patterns type systems. *In POPL*, pages 250–261, 2003. 3, 6, 50

Bernhard Beckert, Reiner Hähnle et Peter H. Schmitt : The even more liberalized delta-rule in free variable semantic tableaux. *In Kurt Gödel Colloquium*, pages 108–119, 1993. 99

Zine-El-Abidine Benaissa, Daniel Briaud, Pierre Lescanne et Jocelyne Rouyer-Degli : lambda-nu, a calculus of explicit substitutions which preserves strong normalisation. *J. Funct. Program.*, 6(5):699–722, 1996. 3, 6, 32

Clara Bertolissi, Horatiu Cirstea et Claude Kirchner : Expressing combinatory reduction systems derivations in the rewriting calculus. *Higher-Order and Symbolic Computation*, 19(4):345–376, 2006. 45

Clara Bertolissi et Claude Kirchner : The rewriting calculus as a combinatory reduction system. *In FoSSaCS*, pages 78–92, 2007. 45

Marc Bezem, Dimitri Hendriks et Hans de Nivelle : Automated proof construction in type theory using resolution. *Journal of Automated Reasoning*, 29(3-4):253–275, 2002. 246

Roel Bloo et Kristoffer Høgsbro Rose : Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. *In CSN '95 – Computer Science in the Netherlands*, pages 62–72, November 1995. 3, 6, 32, 119, 198

Corrado Böhm : Alcune proprietà delle forme β-η-normali nel λ-k-calcolo. *Pubbl. INAC*, (696), 1968. 40

Maria Paola Bonacina et Nachum Dershowitz : Canonical inference for implicational systems. *In IJCAR*, volume 5195 de *LNAI*, pages 380–395, 2008. 175

Richard Bonichon : Tamed : A tableau method for deduction modulo. *In IJCAR*, pages 445–459, 2004. 4, 7, 177, 179, 180, 245

Richard Bonichon : *Tableaux et Déduction Modulo*. Thèse de doctorat, Université Pierre et Marie Curie - Paris 6, December 2006. 177, 179, 245

Richard Bonichon et Olivier Hermant : On constructive cut admissibility in deduction modulo. *In TYPES*, pages 33–47, 2006a. 4, 7, 165, 174, 175

Richard Bonichon et Olivier Hermant : A semantic completeness proof for tamed. *In LPAR*, pages 167–181, 2006b. 165, 174, 175, 177, 179, 245

Peter Borovanský, Horatiu Cirstea, Hubert Dubois, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, Quang-Huy Nguyen, Christophe Ringeissen et Marian Vittek : ELAN *V 3.6 User Manual*. Nancy (France), fifth édition, feb 2004. 45

Peter Borovanský, Claude Kirchner, Helene Kirchner et Christophe Ringeissen : Rewriting with strategies in elan : a functional semantics. *International Journal of Foundations of Computer Science*, 12(1):69–98, feb 2001. 45

Paul Brauner, Clément Houtmann et Claude Kirchner : Principles of superdeduction. *In LICS*, pages 41–50, 2007a. 9, 181, 184, 192, 193, 195, 204, 230

Paul Brauner, Clément Houtmann et Claude Kirchner : Superdeduction at work. *In Rewriting, Computation and Proof*, pages 132–166, 2007b. 9, 181, 184, 192, 193, 195

Guillaume Burel : A first-order representation of pure type systems using superdeduction. *In LICS*, pages 253–263, 2008. 245

Guillaume Burel : *Bonnes démonstrations en déduction modulo.* Thèse de doctorat, Université Henri Poincaré, Nancy 1, March 2009. 160, 162, 165, 175

Guillaume Burel et Claude Kirchner : Cut elimination in deduction modulo by abstract completion. *In LFCS*, pages 115–131, 2007. 4, 6, 165, 175, 176

Guillaume Burel et Claude Kirchner : Regaining cut admissibility in deduction modulo using abstract completion. *Information and Computation*, 208:140–164, 2009. 165, 175, 176

Kaustuv Chaudhuri, Dale Miller et Alexis Saurin : Canonical sequent proofs via multi-focusing. *In IFIP TCS*, pages 383–396, 2008a. 233, 235, 243

Kaustuv Chaudhuri, Frank Pfenning et Greg Price : A logical characterization of forward and backward chaining in the inverse method. *J. Autom. Reasoning*, 40 (2-3):133–177, 2008b. 245

Alonzo Church : A set of postulates for the foundation of logic. *Annals of Mathematics*, 2:33–346, 1933-1934. 14, 32

Alonzo Church : An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936. 32, 40

Alonzo Church et J. B. Rosser : Some properties of conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936. 14, 16, 38

Horatiu Cirstea : *Rewriting Calculus : Foundations and Applications.* Thèse de doctorat, Université Henri Poincaré - Nancy I, 2000. 45, 49, 50

Horatiu Cirstea, Germain Faure et Claude Kirchner : A rho-calculus of explicit constraint application. *Higher-Order and Symbolic Computation*, 20(1-2):37–72, 2007. 45

Horatiu Cirstea, Clément Houtmann et Benjamin Wack : Distributive rho-calculus. *In Proceedings of 6th International Workshop on Rewriting Logic and its Applications*, Electronic Notes in Theoretical Computer Science. Elsevier, 2006a. 8, 46, 50, 52

Horatiu Cirstea, Clement Houtmann et Benjamin Wack : Distributive rho-calculus. Rapport technique, 2006b. http://hal.inria.fr/inria-00001112/. 50

Horatiu Cirstea et Claude Kirchner : The rewriting calculus –- part i and ii. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001. 3, 6, 45

Horatiu Cirstea, Luigi Liquori et Benjamin Wack : Rewriting calculus with fixpoints : Untyped and first-order systems. *In Proceedings of TYPES*, volume 3085 de *LNCS*. Springer, 2003. 3, 6, 46, 51, 52

Evelyne Contejean, Claude Marché et Xavier Urbain : CiME3, 2004. URL http://cime.lri.fr/. Available at http://cime.lri.fr/. 220

Thierry Coquand et Gérard P. Huet : The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988. 1, 4

Denis Cousineau et Gilles Dowek : Embedding pure type systems in the lambda-pi-calculus modulo. *In TLCA*, pages 102–117, 2007. 245

Pierre-Louis Curien et Hugo Herbelin : The duality of computation. *In ICFP*, pages 233–243, 2000. 2, 5, 102, 114, 118, 198

Haskell B. Curry : Functionality in combinatory logic. *Proceedings of the National Academy of Science of the USA*, 20:584–590, 1934. 33, 101

Haskell B. Curry et Robert Feys : *Combinatory Logic*, volume 1. North-Holland, Amsterdam, 1958. 3, 6, 33, 101

Haskell B. Curry, Roger J. Hindley et Jonathan P. Seldin : *Combinatory Logic*, volume 2. North-Holland, Amsterdam, 1972. 40

Nicolaas G. De Bruijn : Automath, a language for mathematics. Rapport technique 68-WSK-05, Eindhoven University of Technology, 1968. 33

Nicolaas G. De Bruijn : Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae (Proceedings)*, 75(5):381–392, 1972. 3, 6, 32

Anatoli Degtyarev et Andrei Voronkov : The inverse method. *In Handbook of Automated Reasoning*, pages 179–272. 2001. 1, 4, 80

Nachum Dershowitz et Claude Kirchner : Abstract canonical presentations. *TCS*, 357:53–69, 2006. 175

Gilles Dowek : What is a theory ? *In STACS*, pages 50–64, 2002. 175

Gilles Dowek : Truth values algebras and proof normalization. *In TYPES*, pages 110–124, 2006. 165, 170, 173, 244

Gilles Dowek : communication privée, 2010. 194

Gilles Dowek, Thérèse Hardin et Claude Kirchner : Theorem proving modulo. *Journal of Automated Reasoning*, 31(1):33–72, Nov 2003. 2, 5, 159, 160, 176, 177, 178, 245

Gilles Dowek et Olivier Hermant : A simple proof that super-consistency implies cut elimination. *In RTA*, pages 93–106, 2007. 165

Gilles Dowek et Benjamin Werner : Proof normalization modulo. *Journal of Symbolic Logic*, 68(4):1289–1316, 2003. 4, 6, 162, 163, 164, 165, 170, 171, 173, 174

Germain Faure : *Structures et modèles de calculs de réécriture.* Thèse de doctorat, Université Henri Poincaré & Loria Inria Nancy Grand Est, 2007. 49

Gottlob Frege : *Begriffsschrift : eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.* Halle, 1879. 55

Harvey Friedman : Classically and intuitionistically provably recursive functions. *In Higher set theory (Proc. Conf., Math. Forschungsinst., Oberwolfach, 1977)*, volume 669 de *Lecture Notes in Math.*, pages 21–27. Springer, Berlin, 1978. 198

Gehrard Gentzen : Investigations into logical deductions. *In* M. E. Szabo, éditeur : *The collected papers of Gerhard Gentzen*, pages 68–131. North Holland, 1935. 1, 2, 4, 5, 56, 63, 70, 73, 75, 118

Jean-Yves Girard : Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. 75, 133

Jean-Yves Girard : Towards a geometry of interaction. *In* J. W. Gray et A. Scedrov, éditeurs : *Categories in Computer Science and Logic*, pages 69–108. American Mathematical Society, 1989. Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference, June 14–20, 1987, Boulder, Colorado ; Contemporary Mathematics Volume 92. 131

Jean-Yves Girard : On the unity of logic. *Ann. Pure Appl. Logic*, 59(3):201–217, 1993. 245

Jean-Yves Girard : On the meaning of logical rules i : syntax vs. semantics. *In* Ulrich Berger et Helmut Schwichtenberg, éditeurs : *Computational Logic*, pages 215–272. Springer-Verlag, Heidelberg, 1999. 224, 234

Jean-Yves Girard : On the meaning of logical rules ii : multiplicatives and additives. *In* Bauer et Steinbrüggen, éditeurs : *Foundation of Secure Computation*, pages 183–212. IOS Press, Amsterdam, 2000. 224, 234

Jean-Yves Girard, Paul Taylor et Yves Lafont : *Proofs and types.* Cambridge University Press, New York, NY, USA, 1989. 70, 104

Kurt Gödel : Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für Mathematik*, 38(1):173–198, 1931. 55

Georges Gonthier : The four colour theorem : Engineering of a formal proof. *In ASCM*, page 333, 2007. 14

Timothy G. Griffin : A formulae-as-types notion of control. *In POPL*, pages 47–58. ACM Press, 1990. 2, 5, 102, 107, 114, 198

Reiner Hähnle : Tableaux and related methods. *In Handbook of Automated Reasoning*, pages 100–178. Elsevier and MIT Press, 2001. 1, 4, 80

Reiner Hähnle et Peter H. Schmitt : The liberalized delta-rule in free variable semantic tableaux. *J. Autom. Reasoning*, 13(2):211–221, 1994. 99

Willem Heijltjes : Proof forests with cut based on Herbrand's theorem. Disponible sur la page web de l'auteur, 2009. 243

Hugo Herbelin : *Séquents qu'on Calcule*. Thèse de doctorat, Université Paris 7, January 1995. 114

Hugo Herbelin : *C'est maintenant qu'on calcule, au cœur de la dualité*. Thèse de doctorat, Université de Paris XI, dec 2005. 118

Olivier Hermant : *Méthodes Sémantiques en Déduction Modulo*. Thèse de doctorat, Université Paris 7, December 2005. 4, 6, 162, 165, 168, 174, 175, 178, 248

Clément Houtmann : Axiom directed focusing. *In TYPES*, pages 169–185, 2008a. 9, 193, 223, 224, 226, 230

Clément Houtmann : Axiom directed focusing. `http://hal.inria.fr/inria-00212059/`, 2008b. 226, 228

Clement Houtmann : Three Dimensional Proofnets for Classical Logic. `http://hal.inria.fr/inria-00426469/`, 2009. 8, 133, 142

William A. Howard : The formulae-as-types notion of construction. *In* J. R. Seldin et R. J. Hindley, éditeurs : *To H.B. Curry : essays on combinatory logic, lambda calculus and formalism*, pages 479–490. Academic Press, London, New York, 1980. (Original manuscript from 1969). 1, 4, 33

Xiaorong Huang : Reconstruction proofs at the assertion level. *In CADE*, pages 738–752, 1994. 184

Gérard Huet : Confluent reductions : Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, october 1980. 18, 30, 31

Stanisław Jaśkowski : On the rules of suppositions in formal logic. *Studia Logica*, 1:5–32, 1934. Reprinted in S. McCall (1967) Polish Logic 1920–1939, Oxford University Press, pp. 232–258. 63

Immanuel Kant : *Kritik der reinen Vernunft*. 1781. 79

Richard Kelsey, William Clinger et Jonathan R. (editors) : Revised^5 report on the algorithmic language scheme. *ACM SIGPLAN Notices*, 33(9):26–76, 1998. 44

Zurab Khasidashvili : On the equivalence of persistent term rewriting systems and recursive program schemes. *In ISTCS*, pages 240–249, 1993a. 32

Zurab Khasidashvili : Optimal normalization in orthogonal term rewriting systems. *In RTA*, pages 243–258, 1993b. 32

Florent Kirchner : A finite first-order theory of classes. *In TYPES*, pages 188–202, 2006. 218

Stephen C. Kleene : λ-definability and recursiveness. *Duke Math. J.*, 2(2):340–353, 1936. 41

Stephen C. Kleene : *Introduction to Metamathematics*. Van Nostrand, New York, 1952a. 73

Stephen C. Kleene : Permutability of inferences in gentzen's calculi lk and lj. *Memoires of the American Mathematical Society*, 10:1–26, 1952b. 81, 82, 233

Stephen C. Kleene et John B. Rosser : The inconsistency of certain formal logics. *Annals of Math.*, 2(36):630–636, 1936. 32

Jan Willem Klop : *Combinatory Reduction Systems*. Thèse de doctorat, CWI, 1980. 31

Donald E. Knuth et P. B. Bendix : Simple word problems in universal algebras. *In* J. Leech, éditeur : *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970. 30, 175

François Lamarche et Lutz Straßburger : Naming proofs in classical propositional logic. *In TLCA*, pages 246–261, 2005. 133, 134, 135, 136, 243

Peter J. Landin : A correspondence between algol 60 and church's lambda-notations : Part i and part ii. *Commun. ACM*, 8(2 and 3):2–89, 1965. 32

Stéphane Lengrand : Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. *Electr. Notes Theor. Comput. Sci.*, 86(4), 2003. 119

Xavier Leroy, Damien Doligez, Jacques Garrique, Didier Rémy et Jérôme Vouillon : Objective caml, release 3.11.1. http://caml.inria.fr/ocaml/, 2009. 32

Chuck Liang et Dale Miller : Focusing and polarization in intuitionistic logic. *In CSL*, pages 451–465, 2007. 231, 233, 248

Chuck Liang et Dale Miller : A unified sequent calculus for focused proofs. *In LICS*, pages 355–364. IEEE Computer Society, 2009. 245

Luigi Liquori et Bernard P. Serpette : irho : an imperative rewriting calculus. *Mathematical Structures in Computer Science*, 18(3):467–500, 2008. 45

Per Martin-Löf : *Intuitionistic Type Theory*. Studies in Proof Theory. Bibliopolis, 1984. 1, 4

Richard McKinley : The alpha-epsilon calculus. *In Proceedings of Workshop on Structures and Deduction*, 2009. 243

Dale Miller : A compact representation of proofs. *Studia Logica*, 46(4):347–370, 1987. 133

Pierre-Étienne Moreau et Antoine Reilles : The tom home page. `http://tom.loria.fr`, 2006. 215

Sara Negri et Jan von Plato : Cut elimination in the presence of axioms. *The Bulletin of Symbolic Logic*, 4(4):418–435, 1998. 184

Sara Negri et Jan von Plato : *Structural Proof Theory*. Cambridge University Press, 2001. 184

Maxwell H. A. Newman : On theories with a combinatorial definition of equivalence. *In Annals of Math*, volume 43, pages 223–243, 1942. 18

Christos H. Papadimitriou : *Computational Complexity*. Addison Wesley Longman, 1994. 40

Michel Parigot : Lambda-mu-calculus : An algorithmic interpretation of classical natural deduction. *In* Andrei Voronkov, éditeur : *Logic Programming and Automated Reasoning*, Springer, pages 190–201, St Petersburg, Russia, 1992. 102, 109, 111, 112

Simon Peyton-Jones : Special issue : Haskell 98 language and libraries. *Journal of Functional Programming*, 13(1), 2003. 32

Henri Poincaré : *La Science et l'Hypothèse*. 1902. 2, 5, 153

Dag Prawitz : *Natural Deduction. A Proof-Theoretical Study*, volume 3 de *Stockholm Studies in Philosophy*. Almqvist & Wiksell, Stockholm, 1965. 66, 175, 185

Virgile Prevosto : Certified mathematical hierarchies : the focal system. *In Mathematics, Algorithms, Proofs*, Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. 246

Edmund Robinson : Proof nets for classical logic. *J. Log. Comput.*, 13(5):777–797, 2003. 133

John Alan Robinson : A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, January 1965. 95

Peter Schroeder-Heister : Hypothetical reasoning and definitional reflection in logic programming. *In ELP*, volume 475 de *LNCS*, pages 327–339. Springer, 1989. 184

Peter Schroeder-Heister : Cut elimination for logics with definitional reflection. *In Nonclassical Logics and Information Processing*, volume 619 de *LNCS*, pages 146–171. Springer, 1990. 184

Peter Schroeder-Heister : Rules of definitional reflection. *In LICS*, pages 222–232. IEEE Computer Society, 1993. 184

Raymond M. Smullyan : *First Order Logic*. Springer, 1968. 97

William Tait : Intensional interpretation of functionals of finite type i. *The Journal of Symbolic Logic*, 32, 1967. 104

Masako Takahashi : Parallel reductions in λ-calculus. *Information and Compututation*, 118:120–127, 1995. 38

Coq Development Team : The Coq proof assistant reference manual, version 8.2, August 2009. 1, 4

Terese : *Term Rewriting Systems*. Cambridge University Press, 2003. M. Bezem, J. W. Klop and R. de Vrijer, eds. 14, 19, 104

The Dedukti Development Team : Dedukti, a universal proof checker, version 1.0. `http://www.lix.polytechnique.fr/dedukti/`, 2009. 246

Alan M. Turing : On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936. 32, 40

Alan M. Turing : Computability and lambda-definability. *J. Symb. Log.*, 2(4): 153–163, 1937. 32

Christian Urban : *Classical Logic and Computation*. Thèse de doctorat, University of Cambridge, October 2000. 3, 6, 75, 79, 114, 119, 127, 184, 198, 244

Christian Urban : Strong normalisation for a gentzen-like cut-elimination procedure. *In Proceedings of TLCA*, pages 415–430, 2001. 2, 3, 5, 6, 75, 102, 119, 184, 198

Christian Urban et Gavin M. Bierman : Strong normalisation of cut-elimination in classical logic. *Fundam. Inform.*, 45(1-2):123–155, 2001. 2, 3, 5, 6, 102, 118, 119, 122, 127, 128, 184, 198, 247

Christian Urban et Bozhi Zhu : Revisiting cut-elimination : One difficult proof is really a proof. *In RTA*, pages 409–424, 2008. 127

Eelco Visser et Zine-el-Abidine Benaissa : A core language for rewriting. *In* C. Kirchner et H. Kirchner, éditeurs : *Proceedings of WRLA*, volume 15 de *ENTCS*. Elsevier, sep 1998. 215

Benjamin Wack : *Typage et déduction dans le calcul de réécriture*. Thèse de doctorat, Université Henri Poincaré, Nancy 1, October 2005. 49, 51, 52, 184, 185, 188

Alfred N. Whitehead et Bertrand Russell : *Principia Mathematica*, volume 1. Cambridge University Press, 1925. 55

Résumé

Cette thèse propose et étudie de nouveaux systèmes déductifs mêlant calculs et déductions. La déduction modulo est un premier formalisme qui traduit un pouvoir calculatoire grâce à un système de réécriture. Nous présentons un paradigme dual appelé superdéduction qui traduit un pouvoir déductif par de nouvelles inférences. Ces pouvoirs calculatoires et déductifs modifient la représentation des preuves et leur interaction par les processus d'élimination des coupures. La normalisation forte ou l'admissibilité des coupures ne sont plus garanties et apparaissent alors comme des propriétés intrinsèques des théories représentées sous forme de systèmes de réécriture. Nous démontrons que certains critères permettent d'assurer ces propriétés, notamment en définissant un langage de termes de preuve pour la superdéduction et en étudiant la permutabilité des inférences en calcul des séquents classique.

Notre attention est focalisée sur les calculs des séquents classiques et la représentation des preuves dans de tels systèmes. D'autres formalismes connexes sont envisagés, notamment les réseaux de preuve et le focusing. Nous comparons cette dernière approche à la superdéduction, ce qui nous amène à proposer une refonte du paradigme de superdéduction basée sur un système de multifocusing pour la logique classique. Nous en montrons les effets bénéfiques en démontrant la complétude des systèmes déductifs obtenus.

Mots clefs : Théorie de la preuve, raisonnement automatique, réécriture, calcul des séquents, déduction naturelle, théorie des types

---

Abstract

In this thesis we propose and study several deduction systems that mix deduction and computation. Deduction modulo proposes to translate a computational power through a rewriting system. We present the dual concept called superdeduction. It translates a deductive power into custom inference rules that enrich the deduction system. These computational and deductive powers modify the representation of proofs as well as their interaction through cut-elimination processes. Strong normalisation or cut-admissibility may be lost and therefore appear as intrinsic properties of theories represented as rewriting systems. We prove that certain criteria imply these properties by defining a proof-term language for superdeduction and by studying the permutability of inferences in classical sequent calculus.

Our attention is focused on classical sequent calculi and on the representation of proofs in such systems. Other related paradigms are considered, namely proof-nets and focusing. We compare this latter approach with superdeduction. We consequently reforge the superdeduction paradigm on top of a multifocusing system for classical logic. We demonstrate the benefits of this approach by proving the completeness of the obtained deduction systems.

Keywords : Proof theory, automated reasoning, rewriting, sequent calculi, natural deduction, type theory