Un calcul des séquents extensible

Paul Brauner

Un calcul des séquents extensible

Rapport de stage

27 Juin 2006

Master informatique
Spécialité Maîtrise du logiciel

Paul Brauner

Encadrants :
Claude Kirchner et Benjamin Wack

Composition du jury : Dominique Mery
Noëlle Carbonell
Bertrand Gaiffe
Didier Galmiche
Claude Godart

Laboratoire Lorrain de Recherche en Informatique et ses Applications

Résumé

Ce rapport présente un calcul des séquents où les théories exprimées sous la forme de systèmes de réécriture sont traduites en règles *ad hoc* pour le calcul des séquents. Cela permet à la fois de réduire la taille des démonstrations en vue de la mise en œuvre d'un assistant et de raisonner dans la théorie vide, ce qui ramène la preuve de la cohérence d'une théorie à une démonstration d'élimination des coupures dans le système de déduction dérivé de cette théorie.

Le rapport expose différentes versions classiques et intuitionnistes du système, ainsi que l'ébauche d'un langage de termes de preuve. Après quelques exemples d'application du système, un prototype d'assistant à la démonstration fondé sur la version classique y est présenté.

1 Introduction

Logique et informatique sont intimement liées de nos jours. D'une part, certains théorèmes, comme celui des quatres couleurs, n'ont été prouvés qu'à l'aide d'un ordinateur et on est incapable aujourd'hui d'en fournir une preuve "classique" écrite à la main. D'autre part, la présence croissante de l'informatique dans les domaines critiques de la société : transports, transactions financières, chirurgie, *etc.* requiert des méthodes formelles de certification des programmes auxquelles on doit pouvoir faire confiance. Ce dernier aspect prend son sens à la lumière de l'isomorphisme de Curry-Howard qui met en relation toute démonstration avec un programme informatique. Une proposition logique peut alors être vue comme la spécification mathématique d'une famille de programmes et l'on peut en extraire un représentant à partir de la démonstration de la validité logique de cette proposition.

Malheureusement, la complexité algorithmique des méthodes de démonstration automatique et la non-décidabilité de la logique du premier ordre interdisent l'automatisation complète de ce procédé. C'est pourquoi apparaissent depuis vingt ans des assistants informatiques à la démonstration tels que les systèmes Coq [Coq06], PVS [ORS92] ou Isabelle [Pau94]. Ces programmes fournissent un cadre formel à la démonstration et permettent dans une certaine mesure d'en automatiser les aspects triviaux ou répétitifs. L'assistant Coq propose par exemple une procédure de décision pour l'arithmétique de Presburger. Ces derniers sont à présent utilisés activement pour la certification de programmes et il se profile un passage de la démonstration en tant que discipline mathématique "noble" à une ère d'ingénierie de la preuve. Historiquement cependant, ces assistants sont fondés sur les systèmes classiques de déduction logique qui s'avèrent souvent peu expressifs et mal adaptés au raisonnement mathématique : prouver un théorème revient alors la majeure partie du temps à jongler avec la syntaxe de la logique sous-jacente plutôt qu'à fournir les arguments clés de la démonstration comme on le fait naturellement en mathématiques. Un des défis à venir pour ces assistants est donc de proposer des cadres logiques plus expressifs et finalement plus proches du raisonnement de l'utilisateur.

D'un point de vue plus technique, on est confronté à la question suivante en théorie de la démonstration : sachant que l'on raisonne dans une théorie mathématique (c'est à dire en présence d'un ensemble d'axiomes), vaut-il mieux

reléguer le raisonnement logique à un noyau immuable de règles de déduction ou faut-il intégrer cette théorie au système de déduction sous la forme de nouvelles règles ? La seconde solution présente certains avantages séduisants, elle permet entre autres de ramener la preuve de la cohérence d'une théorie mathématique (*i.e.* on ne peut en dériver à la fois une proposition et sa négation) à un critère que l'on maîtrise bien. Cependant, l'ajout de nouvelles règles à un système déductif nécessite de prouver à nouveau toutes les bonnes propriétés de ce système, en particulier la correction et la complétude du nouveau système ainsi obtenu par rapport à l'ancien. Ce résultat est en général long et fastidieux à démontrer. On voudrait pouvoir disposer d'une méthode automatique qui traduise une théorie en un nouveau système déductif dont on sait qu'il possède ces propriétés.

C'est dans ce cadre que s'inscrit la déduction modulo [DHK03] proposée par Gilles Dowek, Thérèse Hardin et Claude Kirchner. En déduction modulo, les propositions logiques sont considérées modulo une congruence. Typiquement, une proposition est considérée équivalente à sa définition. La fameuse règle du *modus ponens* par exemple ne s'écrit alors plus

$$\Rightarrow_E \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B} \qquad \text{mais} \qquad \Rightarrow_E \frac{\Gamma \vdash C \qquad \Gamma \vdash A}{\Gamma \vdash B}\, C \equiv A \Rightarrow B$$

Cette congruence est généralement définie par un système de réécriture qui réécrit les termes en termes mais également les propositions en propositions. Si le système de réécriture est normalisant, on peut voir la déduction modulo comme un moyen de séparer les étapes calculatoires des véritables étapes de déduction lors d'un raisonnement logique. Les travaux autour de la déduction modulo ont mis en évidence des classes de systèmes de réécriture pour lesquels la cohérence de la logique modulo associée est assurée.

Toutefois, bien qu'elle règle en partie le problème de l'encombrement syntaxique des preuves, en particulier en réduisant notablement la taille de celles-ci, la puissance de la déduction modulo lui confère un certain manque de discernement entre les règles de réécriture qui ont une signification déductive et celles qui relèvent de la "bureaucratie". En effet, alors que les étapes de déduction suivantes illustrent bien la capacité de la déduction modulo à s'abstraire à la fois du calcul sur les termes et des propriétés habituellement passées sous silence,

$$\Rightarrow_E \frac{\vdash P(n+4-2) \Rightarrow P(n) \qquad \vdash P(n+2)}{\vdash P(n)}\, P(n+4-2) \equiv P(n+2)$$

$$\Rightarrow_E \frac{\vdash a = b \Rightarrow P \qquad \vdash b = a}{\vdash P}\, (a = b) \equiv (b = a)$$

sa pertinence est plus discutable lorsque la congruence traduit des axiomes ou des définitions chargés d'un contenu déductif :

$$\Rightarrow_E \frac{\vdash (x = 0 \vee y = 0) \Rightarrow P \qquad \vdash x \times y = 0}{\vdash P}\, (x \times y = 0) \equiv (x = 0 \vee y = 0)$$

Il faut ici faire en quelque sorte un effort de rétro-conception pour comprendre la preuve car elle encode un raisonnement déductif non trivial. De plus, du point

de vue d'un assistant à la démonstration, la déduction modulo introduit une forte composante indéterministe dans la construction des preuves puisqu'il faut fournir le bon représentant de la classe d'équivalence (ici $x \times y = 0$) à chaque étape de la déduction.

Ce rapport présente une extension de la déduction modulo qui résout ce problème : on se propose de conserver la congruence sur les termes mais de traduire les axiomes en règles de déduction *ad hoc* plutôt que de considérer les propositions modulo. Ces nouvelles règles traduisent l'équivalence entre une proposition et sa définition mais permettent également de s'abstraire des étapes de déduction systématiquement associées à ces définitions. Ainsi on obtient des règles proches du raisonnement du mathématicien, telles la règle suivante concernant l'inclusion d'ensembles :

$$\subseteq_{def_E} \frac{\Gamma \vdash X \subseteq Y \qquad \Gamma \vdash t \in X}{\Gamma \vdash t \in Y}$$

En plus de proposer un mode de raisonnement naturel, ce système résout le problème de l'indéterminisme lié à la déduction modulo et semble donc adapté à la mise en œuvre d'un assistant à la démonstration.

Un tel système a été proposé pour la déduction naturelle par Benjamin Wack [Wac05] dans sa thèse sous le nom de *déduction surnaturelle* et est présenté en section 3.2. Cette première version présente plusieurs problèmes qui freinent son adoption dans un assistant : le choix de la déduction naturelle comme support à cette extension limite la traduction des définitions en règles de déduction à un nombre restreint de formules et la volonté de conserver certaines bonnes propriétés du système (comme la complétude par rapport à la déduction naturelle) limite également les apports de cette traduction par rapport au système d'origine.

La section 4 présente nos contributions à ce mode de déduction.

Dans un premier temps, nous en présentons une version transposée au calcul des séquents classique qui résout certains de ces problèmes. Nous en montrons les principales propriétés : correction et complétude par rapport au calcul des séquents et cohérence du système dans la théorie vide. Ces démonstrations constituent le point central de ce rapport. Nous discutons alors l'apport de ces théorèmes à la recherche de la cohérence de théories axiomatiques.

Nous en verrons ensuite une amélioration qui pousse plus loin la notion d'automatisation des étapes de déduction triviales en utilisant des techniques issues de la démonstration automatisée. Nous montrons à nouveau l'équivalence du système par rapport au calcul de séquents en prenant en compte les modifications apportées. S'ensuit un exemple complet d'instanciation du système mettant en avant une application élégante de ce calcul extensible : l'encodage du principe d'induction dans une règle déduction.

Après une brève discussion sur les liens entre la nature intuitionniste de la déduction naturelle et les limitations de la déduction surnaturelle, nous présentons l'état de nos réflexions concernant une version intuitionniste de ce cacul

des séquents extensible utilisant un système de déduction logique mis en avant par la recherche en démonstration automatique.

Nous exposons également l'ébauche d'un langage de termes de preuve pour la version classique du système élaboré en collaboration avec Clément Houtmann, qui effectue son stage de master sur les termes de preuve de la déduction surnaturelle.

Enfin, nous présentons un prototype d'assistant à la preuve fondé sur notre système et écrit dans le langage TOM. Après une brève présentation du langage et de ses possibilités, nous en illustrons l'utilisation dans le code du prototype. La retransription d'une session d'utilisation de l'assistant est fournie et commentée en annexe C.

2 Définitions

Commençons par introduire les notions essentielles utilisées tout au long de ce document. Elles concernent à la fois la logique du premier ordre et la réécriture. Le lecteur qui y est familier peut passer cette partie sans problème.

2.1 Logique du premier ordre

Nous nous intéresserons majoritairement aux logiques du premier ordre. Nous commençons par rappeller les notions de base communes à toutes les logiques abordées ici.

Définition 2.1 (Langage du premier ordre)**.** *Un langage du premier ordre est la donnée d'une famille de symboles (finie ou non) parmi lesquels on distingue trois sortes :*

- *Les symboles de variables.*
- *Les symboles de fonctions. A chaque symbole de fonction on associe un entier naturel appellé* arité *de la fonction. Il correspond au nombre d'arguments de la fonction. On appelle* constante *une fonction d'arité nulle.*
- *Les symboles de relations. De la même manière que pour les fonctions, on associe une arité à chaque symbole de relation.*

Pour une fonction d'arité un on parle de fonction *unaire*, de fonction *binaire* pour les fonctions d'arité deux, *etc.* Il en va de même pour les relations. Une relation d'arité nulle est parfois appellée *variable propositionnelle.*

Traditionnellement, *sin* et $+$ sont des symboles de fonction respectivement unaire et binaire, π un symbole de constante (ou fonction zéro-aire), et *Pair* un symbole de relation unaire. On notera généralement les constantes a, b, c et les fonctions f, g, h.

Par la suite, on se donne un ensemble infini $\mathcal{V}$ de variables x, y, z ...

Définition 2.2 (Termes algébriques)**.** *Soit $\mathcal{L}$ un langage du premier ordre. L'ensemble $\mathcal{T}$ des termes engendré par $\mathcal{L}$ est le plus petit ensemble contenant les variables de $\mathcal{L}$ et stable par application des symboles de fonctions de $\mathcal{L}$ à des termes.*

Concrètement, $sin(x)$, π et $f(g(a,b))$ sont des termes dans le langage aproprié. On désignera généralement les termes par les lettres s, t et u.

A partir de ces termes, on définit l'ensemble des formules qui constituent les objets de base manipulés par la logique.

Définition 2.3 (Formules du premier ordre)**.** *Soit $\mathcal{L}$ un langage du premier ordre.*

- *On appelle* formule atomique *l'application d'une relation n-aire de $\mathcal{L}$ à n variables de $\mathcal{V}$. On note $\mathcal{A}tom$ l'ensemble des formule atomiques de $\mathcal{L}$.*
- *Le langage des formules (appellées aussi propositions) de $\mathcal{L}$ est défini par la grammaire suivante :*

$$\Phi \ ::= \ \mathcal{A}tom \mid \bot \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi \Rightarrow \Phi \mid \forall x.\Phi \mid \exists x.\Phi$$

où x parcourt l'ensemble des variables de $\mathcal{L}$.

Dans le langage des mathématiques, $\forall x.(x \in A \cup B \Rightarrow x \in A \vee x \in B)$ et $lim(s) = L \Leftrightarrow \forall \epsilon.(\epsilon > 0 \Rightarrow \exists n_0.(\forall n.(n > n_0 \Rightarrow |s(n) - \epsilon| < L)))$ sont deux exemples de formules. On a utilisé ici la notation courante $A \Leftrightarrow B$ pour $A \Rightarrow B \wedge B \Rightarrow A$. On appellera généralement ϕ, ψ ou σ les formules.

Au cours de nos raisonnements, nous invoquerons souvent la notion de *sous-terme* qui désigne simplement une composante d'un terme. Plus formellement, on la définit comme suit.

Définition 2.4 (Sous-terme)**.** *Soit t un terme algébrique d'un langage quelconque. L'ensemble des sous-termes de t, noté $\mathcal{ST}(t)$ est défini de la manière suivante :*

- *Si t est une variable, $\mathcal{ST}(t) = \{t\}$.*
- *Si $t = f(t_1, \ldots, t_n)$ avec $n \geq 0$, $\mathcal{ST}(t) = \{t\} \bigcup_{i=1\ldots n} \mathcal{ST}(t_i)$.*

Il en va de même pour la notion de *sous-formule*, définie ci-dessous.

Définition 2.5 (Sous-formule)**.** *Soit ϕ une formule du premier ordre. L'ensemble des sous-formules de ϕ, noté $\mathcal{SF}(\phi)$ est défini de la manière suivante :*

- *Si ϕ est atomique, $\mathcal{SF}(\phi) = \{\phi\}$.*
- *Si ϕ est de la forme* $\begin{cases} \phi_1 \wedge \phi_2 \\ \phi_1 \vee \phi_2 \\ \phi_1 \Rightarrow \phi_2 \end{cases}$ *, $\mathcal{SF}(\phi) = \{\phi\} \cup \mathcal{SF}(\phi_1) \cup \mathcal{SF}(\phi_2)$.*
- *Si ϕ est de la forme* $\begin{cases} \forall x.\phi_1 \\ \exists x.\phi_1 \end{cases}$ *, $\mathcal{SF}(\phi) = \{\phi\} \cup \mathcal{SF}(\phi_1)$.*

Les logiques qui manipulent des formules contenant des quantificateurs font fréquemment appel à la notion de *variable libre*.

Définition 2.6 (Variables libres)**.** *Soit ϕ une formule du premier ordre. L'ensemble des variables libres de ϕ noté $\mathcal{FV}(\phi)$ est défini comme suit.*

- *Si $\phi = R(t_1, \ldots, t_n)$ est atomique, $\mathcal{FV}(\phi)$ est l'ensemble des variables apparaissant dans les termes $t_1, \ldots, t_n$.*
- *Si ϕ est de la forme* $\begin{cases} \phi_1 \wedge \phi_2 \\ \phi_1 \vee \phi_2 \\ \phi_1 \Rightarrow \phi_2 \end{cases}$ *, $\mathcal{FV}(\phi) = \mathcal{FV}(\phi_1) \cup \mathcal{FV}(\phi_2)$.*

– *Si* ϕ *est de la forme* $\begin{cases} \forall x.\phi_1 \\ \exists x.\phi_1 \end{cases}$, $\mathcal{FV}(\phi) = \mathcal{FV}(\phi_1) \setminus \{x\}$.

Intuitivement, les variables libres d'une formules sont les variables qui ne sont pas "concernées" par un quantificateur. Les variables non libres d'une formule sont dites *liées*. Dans la formule $\forall x.R(x, y)$, x est liée et y est libre.

On introduit également la notion de *contexte*, qui est utilisée par la plupart des systèmes déductifs.

Définition 2.7 (Contexte)**.** *Un contexte est un multi-ensemble fini de propositions du premier ordre (ou encore une liste non-ordonnée de fonctions du premier ordre). On le représente sous la forme* $P_1, P_2, \ldots, P_n$ *où l'ordre des propositions n'a pas de signification.*

On note généralement Γ ou Δ les contextes. La liste suivante constitue un exemple concret de ce qu'est un contexte : $Pair(0), \forall n.(Pair(n) \Rightarrow Pair(n+2))$.

Voyons enfin la notion centrale des systèmes logiques : les règles de déduction.

Définition 2.8 (Règle de déduction et système déductif)**.**
Une règle de déduction *est la donnée d'un ensemble éventuellement vide d'hypothèses (ou prémisses)* $H_1, \ldots, H_n$*, d'une conclusion* C*, d'une condition d'application* $\mathcal{C}$ *et éventuellement d'une étiquette* e*. On note une telle règle*

$$e\ \frac{H_1 \qquad \ldots \qquad H_n}{C}\ \mathcal{C}$$

La condition d'application rassemble des prémisses dont la validité n'est pas décidée par le système d'inférence.

Un système déductif *est un ensemble de règles de déduction. Le plus souvent, on donnera des schémas de règles, autrement dit des règles valables pour toute instanciation de leurs paramètres. Une conclusion* C *est* dérivable *dans un système donné si :*

- *il existe une règle n'ayant aucune hypothèse, dont la condition d'application est vérifiée, et dont la conclusion est* C *;*
- *ou il existe une règle dont la conclusion est* C*, dont la condition d'application est vérifiée et dont toutes les hypothèses sont dérivables.*

Un enchaînement de règles permettant de dériver une conclusion C *est appelé* arbre de dérivation *pour cette conclusion.*

2.2 Réécriture

Nous présentons brièvement quelques notions de base liées à la réécriture qui nous servira de support en filigrane tout au long de cet exposé.

Définition 2.9 (Règle de réécriture)**.** *Une règle de réécriture sur les termes est une paire de termes* $l \to r$*. On impose que les variables libres de* r *apparaissent dans* l*. Une règle de réécriture sur les propositions est une paire de propositions* $l \to r$*, où* l *est une proposition atomique et* r *une proposition quelconque. Les variables libres de* r *doivent apparaître dans* l*.*

Un système de réécriture est un ensemble de règle de réécriture. Un exemple de règle de réécriture sur les termes est $x + 0 \to x$, et $x * 0 = 0 \to x = 0 \vee y = 0$ constitue un exemple de règle de réécriture sur les propositions.

Définition 2.10 (Réécriture). *Étant donné un système de réécriture $\mathcal{R}$, la proposition P se réécrit en P' dans $\mathcal{R}$, ce que l'on note $P \to_{\mathcal{R}} P'$, si $P_{|\omega} = \sigma(l)$ et $P' = P[\sigma(r)]_\omega$ pour une règle $l \to r \in \mathcal{R}$, une position ω dans P et une substitution σ. Les variables quantifiées de r sont renommées lors de l'application de la substitution σ pour éviter les captures.*

La clôture réflexive-transitive de la relation $\to_{\mathcal{R}}$ est notée $\to^*_{\mathcal{R}}$. La relation $=_{\mathcal{R}}$ est alors la plus petite congruence contenant $\to^*_{\mathcal{R}}$.

Étant donné le système de réécriture suivant, où s représente intuitivement la fonction *successeur* dans l'arithmétique de Peano :

$$\mathcal{R} = \begin{cases} s(x) + y & \to & s(x + y) \\ x + 0 & \to & x \\ x * y = 0 & \to & x = 0 \vee y = 0 \end{cases}$$

On a les étapes de réécriture suivantes :

$$\begin{array}{rcl} (s(0) + 0) * 0 = 0 & \to_{\mathcal{R}} & s(0 + 0) * 0 = 0 \\ s(0 + 0) * 0 = 0 & \to_{\mathcal{R}} & s(0) * 0 = 0 \\ s(0) * 0 = 0 & \to_{\mathcal{R}} & s(0) = 0 \vee 0 = 0 \end{array}$$

Et ces deux termes sont en relation pour $=_{\mathcal{R}}$:

$$s(s(s(0))) + s(s(0)) =_{\mathcal{R}} s(s(s(s(0)))))$$

3 Déduction naturelle, surnaturelle et modulo

3.1 Déduction naturelle

Nous rappellons brièvement les principes de la déduction naturelle intuitionniste NJ avant d'aborder la déduction surnaturelle.

Définition 3.1 (Jugement). *Un jugement est un couple $\Gamma \vdash \phi$, où Γ est un contexte et ϕ une proposition quelconque.*

Dans ce cas le contexte $\Gamma = P_1, \ldots, P_n$ doit s'interpréter comme une conjonction $P_1 \wedge \cdots \wedge P_n$ de propositions logiques et se lit "P_1,P_2 ... et P_n prouvent ϕ"

Définition 3.2 (Règles de déduction de NJ). *Les règles de déduction de NJ sont des règles de déduction (cf 2.8) dont les prémisses et conclusions sont des jugements. L'ensemble complet de ces règles se trouve en annexe A.*

Prenons pour exemple la règle d'élimination de l'implique :

$$\Rightarrow_E \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Elle se lit de la manière suivante : "si le contexte Γ prouve $A \Rightarrow B$ et si ce même Γ prouve A, on en déduit que Γ prouve B".

Définition 3.3 (Arbre de preuve dans NJ)**.** *Un arbre de preuve dans NJ est un arbre de dérivation construit à partir des règles de déduction de NJ. Il a donc un jugement pour racine et des axiomes pour feuilles.*

Enfin voyons succintement ce qu'est une coupure en déduction naturelle ce qui nous permettra de mettre en avant une notion similaire en déduction surnaturelle.

Définition 3.4 (Coupure NJ)**.** *Une coupure dans NJ consiste en une règle d'introduction d'un connecteur immédiatement suivie de la règle d'élimination pour le même connecteur.*

Une coupure correspond à un "détour" dans une preuve. Par exemple, la preuve suivante présente une coupure pour le connecteur $\Rightarrow$.

$$\dfrac{\dfrac{\dfrac{\Pi_1}{\Gamma, P \vdash Q}}{\Gamma \vdash P \Rightarrow Q}\Rightarrow_I \qquad \begin{array}{c}\Pi_2\\ \Gamma \vdash P\end{array}}{\Gamma \vdash Q}\Rightarrow_E$$

Ce genre de détour correspond au raisonnement que l'on tient habituellement en mathématiques : on commence par prouver un théorème, $\Gamma \vdash P \Rightarrow Q$, puis on l'utilise pour prouver un autre résultat. Cependant, on pourrait remplacer P par sa preuve Π_2 dans Π_1 pour obtenir une preuve sans détours. Cette opération est appellée *élmimination des coupures* et joue un rôle central en théorie de la démonstration comme nous le verrons.

3.2 Déduction surnaturelle

Comme dit dans l'introduction, raisonner en présence d'axiomes est obligatoire si l'on souhaite dériver des théorèmes dans une théorie donnée (la théorie des ensembles par exemple) et prouver autre chose que des tautologies, mais pose plusieurs problèmes : il peut y avoir une infinité d'axiomes et certaine propriétés telles que la cohérence de la logique ne sont prouvées que dans la théorie vide. De plus, une preuve en déduction naturelle faisant intervenir des axiomes est assez éloignée du raisonnement que l'on tient habituellement lorsqu'on raisonne à l'aide de définitions en mathématiques.

La déduction naturelle avec *folding* et *unfolding* proposée par Prawitz [Pra65] résout en partie ces problèmes en ajoutant à NJ deux nouvelles règles, *unfold* et *fold*, traduisant respectivement le remplacement d'une proposition atomique par sa définition et inversement. Ainsi, si l'on considère une théorie $\mathcal{T}h$, une proposition atomique P et une proposition quelconque Q telles que $(P \Leftrightarrow Q) \in \mathcal{T}h$, l'arbre de dérivation suivant

$$\dfrac{\dfrac{\dfrac{\dfrac{ax\ \dfrac{}{\mathcal{T}h, A, P \vdash P \Leftrightarrow Q}}{\mathcal{T}h, A, P \vdash P \Rightarrow Q}\wedge_E \qquad ax\ \dfrac{}{\mathcal{T}h, A, P \vdash P}}{\mathcal{T}h, A, P \vdash Q}\Rightarrow_E \qquad ax\ \dfrac{}{\mathcal{T}h, A, P \vdash A}}{\mathcal{T}h, A, P \vdash Q \wedge A}\wedge_I$$

est considérablement réduit et rendu plus naturel par l'utilisation de la règle *folding* comme le montre la preuve ci-dessous.

$$\wedge_I \dfrac{unfold \dfrac{ax \dfrac{}{A, P \vdash P}}{A, P \vdash Q} \quad ax \dfrac{}{A, P \vdash A}}{A, P \vdash Q \wedge A}$$

Ce système est correct et complet par rapport à NJ, ce qui signifie que toute preuve avec *folding* et *unfolding* est prouvable dans NJ en raisonnant dans une théorie appropriée (celle contenant les définitions) et que toute preuve dans NJ présentant une telle théorie dans ses hypothèses admet une preuve avec *folding* et *unfolding* sans cette théorie.

La déduction modulo [DHK03] proposée par Gilles Dowek, Thérèse Hardin et Claude Kirchner propose un autre traitement de ces axiomes et va plus loin en considérant les propositions logiques modulo une congruence. Cette congruence est en général définie par un système de réécriture $\mathcal{R}$ qui réécrit

– les termes en termes ;

– les propositions atomiques en propositions quelconques.

La congruence est la plus petite congruence contenant la relation de réécriture. Les règles $\wedge_I$ et $\Rightarrow_E$ par exemple deviennent alors les suivantes.

$$\wedge_I \dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash C}\ C \equiv_{\mathcal{R}} A \wedge B \qquad\qquad \Rightarrow_E \dfrac{\Gamma \vdash C \qquad \Gamma \vdash A}{\Gamma \vdash B}\ C \equiv_{\mathcal{R}} A \Rightarrow B$$

Si le système de réécriture vérifie les bonnes propriétés, on peut prouver les propriétés classiquement désirées pour un système de déduction logique : correction, complétude, élimination des coupures, *etc.*

Toutefois, ce système n'est pas totalement approprié pour la déduction interactive. En effet, alors que la réécriture des termes capture assez naturellement les étapes de calcul, il est difficile de raisonner sur des propositions modulo une congruence : l'utilisateur doit fournir le bon représentant de la classe d'équivalence à chaque étape de déduction. De plus, les règles de déduction modulo ne concernent plus le connecteur de tête de la proposition, comme c'est le cas en déduction naturelle. Prenons par exemple la règle de réécriture $X \subseteq Y \rightarrow \forall x.(x \in X \Rightarrow x \in Y)$ qui traduit la définition de l'inclusion en théorie des ensembles. L'étape de déduction suivante est alors valide en déduction modulo :

$$\forall_I \dfrac{\vdash x \in A \Rightarrow x \in B}{\vdash A \subseteq B}\ A \subseteq B \equiv_{\mathcal{R}} \forall x.(x \in A \Rightarrow x \in B)$$

Cependant, le symbole $\forall$ n'apparaît pas en tête de la proposition $A \subseteq B$ à laquelle on applique la règle $\forall_I$. Il faut d'abord considérer la proposition $\forall x.(x \in A \Rightarrow x \in B)$ qui lui est congrue pour faire apparaître son connecteur de tête. Si le système est normalisant, on peut également raisonner sur les formes normales des propositions, mais on perd alors l'intérêt d'avoir considéré une congruence sur des formules.

Dans sa thèse [Wac05], Benjamin Wack propose une variante de la déduction modulo adaptée à la preuve interactive : au lieu de considérer les propositions

modulo une relation, on ajoute de nouvelles règles de déduction traduisant les axiomes de la forme $P \equiv \phi$ avec P atomique à celles de NJ. L'idée est de permettre le remplacement d'une proposition P par sa définition ϕ (comme en déduction naturelle avec *folding* et *unfolding*) mais également d'introduire au maximum les connecteurs de ϕ.

Prenons à nouveau pour exemple la théorie des ensembles et considérons le système de réécriture suivant, qui traduit les définitions de l'inclusion et de l'ensemble vide.

$$\mathcal{R} = \left\{ \begin{array}{lcl} \subseteq_{def} : X \subseteq Y & \rightarrow & \forall x.(x \in X \Rightarrow x \in Y) \\ \emptyset_{def} : \quad x \in \emptyset & \rightarrow & \bot \end{array} \right.$$

Nous transformons ces règles de réécriture en règles de déduction selon une procédure décrite ci-après. On obtient deux nouvelles règles de déduction pour la définition de l'inclusion et une nouvelle règle d'élimination pour celle de l'ensemble vide :

$$\emptyset_{def_E}\ \frac{\Gamma \vdash t \in \emptyset}{\Gamma \vdash \phi}$$

$$\subseteq_{def_I}\ \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y}\ x \notin \mathcal{FV}(\Gamma) \qquad \subseteq_{def_E}\ \frac{\Gamma \vdash X \subseteq Y \qquad \Gamma \vdash t \in X}{\Gamma \vdash t \in Y}$$

Remarquons que ces règles sont très naturelles : la troisième règle par exemple peut se lire "Si X est inclus dans Y et t est dans X, alors t est dans Y". On peut alors dériver la démonstration suivante de $\vdash \emptyset \subseteq A$.

$$\begin{array}{r} ax\ \dfrac{}{x \in \emptyset \vdash_+ x \in \emptyset} \\ \emptyset_{def_E}\ \dfrac{}{x \in \emptyset \vdash_+ x \in A} \\ \subseteq_{def_I}\ \dfrac{}{\vdash_+ \emptyset \subseteq A} \end{array}$$

La démonstration est concise, lisible et se fait dans la théorie vide. La déduction surnaturelle semble donc adaptée à la preuve interactive.

Voyons maintenant plus en détail comment procède l'algorithme de décomposition de ϕ pour le calcul des nouvelles règles. 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. L'algorithme a été très légèrement modifié par rapport à la version proposée par Benjamin Wack, les réflexions survenues au cours du stage ayant abouti à sa simplification.

Définition 3.5 (Calcul des règles de déduction surnaturelle)**.** *Soit une règle de réécriture $R : P \rightarrow \phi$. On initialise la procédure par le jugement $\Gamma \vdash \phi$*

1. *Pour trouver la règle d'introduction de R, on applique (de bas en haut) les règles de la figure 1 tant qu'il reste des formules auxquelles ces règles s'appliquent.*
2. *Pour trouver les règles d'élimination de R, on applique (de haut en bas) les règles de la figure 2 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.

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash (\phi \wedge \psi)} \qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash (\forall x.\phi)}\ x \notin \mathcal{FV}(\Gamma)$$

$$\frac{\Gamma, \psi \vdash \phi}{\Gamma \vdash (\psi \Rightarrow \phi)}$$

FIG. 1 – Calcul de la règle d'introduction

$$\frac{\Gamma \vdash (\phi \wedge \psi)}{\Gamma \vdash \phi} \qquad \frac{\Gamma \vdash (\phi \wedge \psi)}{\Gamma \vdash \psi} \qquad \frac{\Gamma \vdash (\forall x.\phi)}{\Gamma \vdash (\phi[x/t])}$$

$$\frac{\Gamma \vdash (\psi \Rightarrow \phi) \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash \vartheta}$$

FIG. 2 – Calcul des règles d'élimination

On remarque que l'on obtient systématiquement une règle d'introduction par règle de réécriture. En revanche, en raison de la présence de la règle de calcul inspirée de $\wedge_E$, on peut obtenir plusieurs règles d'élimination pour la même règle de réécriture.

Nous pouvons maintenant définir formellement la notion de *système de déduction surnaturelle* puis en présenter la propriété principale.

Définition 3.6 (Système de déduction surnaturelle)**.** *Étant donné un système de réécriture $\mathcal{R} = \mathcal{R}_t \cup \mathcal{R}_P$ tel que $\mathcal{R}_t$ réécrit des termes et $\mathcal{R}_P$ réécrit des propositions atomiques, le* système de déduction surnaturelle *associé à $\mathcal{R}$ est donné par :*

- *les règles de la logique du premier ordre ;*
- *les règles construites à partir de $\mathcal{R}_P$;*

où toutes les propositions sont considérées modulo $=_{\mathcal{R}_t}$. On notera $\Gamma \vdash_+ \phi$ les jugements dérivés dans un tel système.

Voyons maintenant le théorème principal de la déduction surnaturelle : la correction et la complétude de ce système par rapport à NJ.

Theorème 1 (Équivalence)**.** *Pour tout système déduction surnaturelle associé à un TRS $\mathcal{R}$, il existe une théorie $\mathcal{T}h$ telle que $\Gamma \vdash_+ \phi$ si et seulement si $\Gamma, \mathcal{T}h \vdash \phi$.*

Démonstration. Pour construire la théorie $\mathcal{T}h$ on procède comme suit. S'il n'existe pas déjà un prédicat d'égalité $=$, on en ajoute un ainsi que les axiomes correspondants. Chaque règle de réécriture $l \to r$ se traduit alors en un axiome :

- $\forall \overline{x}.(l = r)$ si $l \to r \in \mathcal{R}_t$
- $\forall \overline{x}.(l \Leftrightarrow r)$ si $l \to r \in \mathcal{R}_P$

où $\overline{x}$ est l'ensemble des variables libres dans l et r.

Il suffit alors d'exhiber un procédé pour transformer une preuve surnaturelle en une preuve en déduction naturelle avec cette théorie et inversement. La preuve complète est présente dans [Wac05]. □

3.3 Applications

Les applications de la déduction surnaturelle, au-delà de la preuve interactive, sont prometteuses. Un premier exemple intéressant est la simulation des logiques d'ordre supérieur . En déduction modulo, on l'exprime à travers la règle

$$\epsilon(\alpha(\dot{\forall}, x)) \quad \rightarrow \quad \forall y \epsilon(\alpha(x, y))$$

Après transformation en règles de déduction surnaturelles, on obtient exactement les règles de déduction de la logique d'ordre supérieure :

$$\frac{\Gamma \vdash_+ \epsilon(\alpha(x, y))}{\Gamma \vdash_+ \epsilon(\alpha(\dot{\forall}, x))}\, y \notin \mathcal{FV}(\Gamma) \qquad \frac{\Gamma \vdash_+ \epsilon(\alpha(\dot{\forall}, x))}{\Gamma \vdash_+ \epsilon(\alpha(x, \vartheta))}$$

On pourrait donc imaginer un assistant de preuve proposant de manière transparente la manipulation de logiques d'ordre supérieur dont la correction serait assurée par les résultats théoriques sur la déduction surnaturelle. On peut imaginer le même procédé pour la simulation d'autres systèmes de déduction comme les logiques modales.

L'utilisation du ρ-calcul [CLW03] a été proposée dans [Wac05] pour mettre au point un langage de termes de preuve adapté à la déduction surnaturelle. Il faut alors générer de nouvelles règles de typage *ad hoc* pour chaque règle de réécriture. Il n'est pas exposé ici, cependant la partie 4.5 présente les résultats d'une première réflexion sur le langage de termes de notre système.

Un autre domaine où la déduction surnaturelle trouve une application élégante est la représentation des prédicats inductifs. En effet, si l'on utilise la règle suivante (proposée dans [DW05]) pour définir les entiers naturels :

$$n \in N \rightarrow \forall P.(0 \in P \Rightarrow \forall m.(m \in P \Rightarrow S(m) \in P) \Rightarrow n \in P)$$

on obtient la règle d'élimination suivante :

$$\frac{\Gamma \vdash_+ n \in N \qquad \Gamma \vdash_+ 0 \in P \qquad \Gamma \vdash_+ \forall m.(m \in P \Rightarrow S(m) \in P)}{\Gamma \vdash_+ n \in P}\, P \notin \mathcal{FV}(\Gamma)$$

On peut "tricher" et pousser la décomposition de la règle un peu plus loin en introduisant les connecteurs de $\forall m.(m \in P \Rightarrow S(m) \in P)$. La possibilité d'étendre la décomposition des formules de cette manière constitue en partie la motivation du passage au calcul des séquents qui fait l'objet de ce stage, comme nous le verrons plus loin.

$$\frac{\Gamma \vdash_+ n \in N \qquad \Gamma \vdash_+ 0 \in P \qquad \Gamma, m \in P \vdash_+ S(m) \in P}{\Gamma \vdash_+ n \in P}\, P,m \notin \mathcal{FV}(\Gamma)$$

On obtient alors une règle concise pour exprimer le raisonnement habituellement observé sur les ensembles inductifs. Encore une fois, on voit le bénéfice que l'on pourrait tirer d'une telle règle dans un environnement de preuve interactif.

4 Une extension du calcul des séquents

4.1 Problématique

La déduction surnaturelle semble donc apporter une solution au problème de la mise en œuvre d'un assistant de preuve modulo et ses applications potentielles sont variées. Cependant, il subsiste plusieurs points à éclaircir ou à améliorer.

Décomposition des règles Tout d'abord, les règles d'élimination obtenues lors de la décomposition de formules contenant une implication présentent encore des prémisses dont la décomposition pourrait être poursuivie. En effet, la règle de calcul des nouvelles règles délimination inspirée de $\Rightarrow_E$ introduit une nouvelle prémisse qui n'est pas décomposée puisque l'on applique les règles de haut en bas. C'est le cas dans l'exemple de la section 3.3 concernant la traduction du principe d'induction.

Prenons par exemple la règle de réécriture $R : P \rightarrow (A \Rightarrow B) \Rightarrow C$ à laquelle on applique l'algorithme décrit en 3.5. On obtient les nouvelles règles de déduction suivantes :

$$R_I \ \frac{\Gamma, A \Rightarrow B \vdash_+ C}{\Gamma \vdash_+ P} \qquad R_E \ \frac{\Gamma \vdash_+ P \qquad \Gamma \vdash_+ A \Rightarrow B}{\Gamma \vdash_+ C}$$

On voudrait pouvoir décomposer à nouveau la proposition $A \Rightarrow B$ dans les prémisses de R_E pour obtenir la règle :

$$R'_E \ \frac{\Gamma \vdash_+ P \qquad \Gamma, A \vdash_+ B}{\Gamma \vdash_+ C}$$

Cette règle est toujours correcte par rapport à NJ mais on perd la notion de coupure que l'on avait avec R_I et R_E :

$$\begin{array}{l} R_I \ \dfrac{\Gamma, A \Rightarrow B \vdash_+ C}{\Gamma \vdash_+ P} \qquad \Gamma \vdash_+ A \Rightarrow B \\ R_E \ \hline \Gamma \vdash_+ C \end{array}$$

Pour obtenir une coupure avec R'_E il faudrait pouvoir décomposer $A \Rightarrow B$ à gauche du jugement qui forme la prémisse de R_I comme on le ferait en calcul des séquents.

Connecteurs utilisés Ensuite, seulement une partie des règles de NJ est utilisée pour la décomposition des propositions lors du calcul des nouvelles règles. En effet les règles de calcul proposées figures 1 et 2 n'incluent pas les règles de NJ concernant les connecteurs $\vee$ et $\exists$. Cela s'explique par les phénomènes qui se produisent lorsque l'on autorise la décomposition des formules présentant ces derniers connecteurs en tête. Voyons le cas du $\vee$ et considérons les règles de réécriture $R : P \rightarrow \forall x.(A(x) \vee B(x))$ et $R' : P' \rightarrow \forall x.A(x) \vee \forall x.B(x)$. Supposons que l'on autorise l'utilisation de la règle $\vee_I$ dans l'algorithme 3.5 (remarquons au passage que l'algorithme produit alors une ou plusieurs règles d'introduction),

on obtient exactement les mêmes règles d'introduction pour R et R' :

$$R_{I1}\ \frac{\Gamma \vdash_+ A}{\Gamma \vdash_+ P}\ x \notin \mathcal{FV}(\Gamma) \qquad R_{I2}\ \frac{\Gamma \vdash_+ B}{\Gamma \vdash_+ P}\ x \notin \mathcal{FV}(\Gamma)$$

$$R'_{I1}\ \frac{\Gamma \vdash_+ A}{\Gamma \vdash_+ P'}\ x \notin \mathcal{FV}(\Gamma) \qquad R'_{I2}\ \frac{\Gamma \vdash_+ B}{\Gamma \vdash_+ P'}\ x \notin \mathcal{FV}(\Gamma)$$

On imagine bien que cela pose un problème, ne serait-ce que pour la complétude du sytème. Essayons par exemple de prouver que la définition de la proposition P implique P elle-même :

$$R_{I1}\ \frac{\forall x.(A(x) \vee B(x))) \vdash_+ A(x)}{\forall x.(A(x) \vee B(x)) \vdash_+ P} \qquad R_{I2}\ \frac{\forall x.(A(x) \vee B(x))) \vdash_+ B(x)}{\forall x.(A(x) \vee B(x)) \vdash_+ P}$$

Dans les deux cas, on ne peut dériver la preuve en commençant par une des deux nouvelles règles d'introduction. La preuve commençant par une élimination du $\vee$ échoue pour une autre raison :

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash_+ \forall x.(A(x) \vee B(x))}\,ax}{\Gamma \vdash_+ A(x) \vee B(x)}\,\forall_E \qquad \Gamma, A(x) \vdash_+ P \qquad \Gamma, B(x) \vdash_+ P}{\Gamma = (\forall x.(A(x) \vee B(x))) \vdash_+ P}\,\vee_E$$

En effet, on ne peut pas appliquer R_{I1} ou R_{I2} aux feuilles de l'arbre sans violer leurs conditions d'application ($x \notin \mathcal{FV}(\Gamma)$).

Le calcul des séquents présentant plus de symétries, il nous paraissait plausible que son utilisation en lieu et place de la déduction naturelle comme cadre de travail pour la traduction des axiomes en règles de déduction permette d'aborder la question avec plus de succès. Il s'est en fait avéré que le système déductif n'est pas en cause mais qu'il s'agit d'un problème sémantique plus profond comme nous le verrons par la suite.

Propriété de la sous-formule Enfin, la déduction naturelle, et *a fortiori* la déduction surnaturelle, sont proches du raisonnement classique des mathématiciens mais ne sont pas très adaptées à la recherche automatique de preuves, les règles ne respectant pas la propriété de la sous-formule (*cf* 4.2).

Ces différents points nous ont poussé à nous pencher sur une version de la déduction surnaturelle exprimée dans le calcul des séquents. Dans la suite de ce document, nous introduisons un tel système ainsi que ses propriétés, un langage de termes de preuve appropriés et une mise en œuvre du système à travers l'écriture d'un assistant de preuve.

4.2 Calcul des séquents - rappels

Commençons par rappeller quelques notions concernant le calcul des séquents, support de notre calcul extensible. Nous présentons d'abord LK, le calcul des séquents classique (en opposition à intuitionniste), ce caractère étant nécessaire au bon fonctionnement de notre système. Nous présentons ensuite brièvement LJ, son homologue intuitionniste.

4.2.1 Séquents classiques

Définition 4.1 (Séquent classique)**.** *Un séquent classique est un couple* $\Gamma \vdash \Delta$ *où* Γ *est un contexte et* Δ *un contexte non vide.*

Un séquent $P_1, \ldots, P_n \vdash \phi_1, \ldots, \phi_m$ doit s'interpréter intuitivement de la sorte : $P_1 \wedge \cdots \wedge P_n \vdash \phi_1 \vee \cdots \vee \phi_m$. On la lit ainsi : "$P_1$, P_2 ... et P_n prouvent ϕ_1, ϕ_2 ... ou ϕ_m".

Définition 4.2 (Règles de déduction de LK)**.** *Les règles de déduction de LK sont des règles de déduction (cf 2.8) dont les prémisses et conclusions sont des séquents. L'ensemble complet de ces règles est représenté figure 3.*

Nous pouvons formuler plusieurs remarques à leur sujet :

- Les règles droites correspondent aux règles d'introduction de la déduction naturelle. En revanche, les règles gauches se lisent moins naturellement que les règles d'élimination de NJ, en particulier $\Rightarrow_{\mathcal{L}}$. Jean-Yves Girard donne un procédé de traduction systématique de NJ vers LJ et inversement dans [GLT89].
- Toutes les règles logiques gauches possèdent une règle droite complémentaire parfaitement symétrique.
- La règle de coupure traduit la grande majorité des raisonnements du mathématicien : prouver un résultat en faisant un détour par un théorème. Elle capture en une règle les détours potentiels inhérents aux règles de la déduction naturelle. Comme expliqué ci-dessous, la notion de preuve sans coupure (*i.e.* sans détours) est une propriété majeure losque l'on raisonne dans LK.
- Toutes les règles à l'exception de la coupure (*cut*) respectent la *propriété de la sous-formule*, c'est à dire que les formules apparaissant dans les séquents prémisses sont présentes dans le séquent conclusion. Cette propriété a des répercussions en recherche automatique de preuve puisque la recherche d'une preuve sans coupure ne nécessite pas de "deviner" de nouvelle proposition au cours de la démonstration comme ce serait le cas dans NJ avec la règle $\Rightarrow_E$ par exemple.

Au vu de la dernière remarque, on comprend que la notion de coupure joue un rôle central dans LK. En effet, si l'on peut prouver que la règle *cut* est superflue pour une théorie Γ donnée, on peut ramener toute preuve à une preuve qui possède la propriété de la sous-formule et l'on peut alors facilement démontrer que $\Gamma \vdash \bot$ n'est pas dérivable, autrement dit que la théorie est cohérente (*cf* [GLT89] à nouveau). Ce résultat est avéré pour la théorie vide, cependant il reste à prouver pour chaque nouvelle théorie.

La notion d'arbre de preuve découle naturellement de la discussion précédente.

Définition 4.3 (Arbre de preuve dans LK)**.** *Un arbre de preuve dans LK est un arbre de dérivation construit à partir des règles de déduction de LK. Il a donc un séquent pour racine et ses branches se terminent par l'application des règles ax,* $\bot$ *ou* $\top$*.*

Pour exemple, l'arbre de la figure 4 est valide dans LK. Il prouve la proposition $\emptyset \subseteq A$ dans un contexte contenant les définitions de $\emptyset$ et $\subseteq$.

Axiome

$$ax\ \frac{}{\Gamma, A \vdash A, \Delta}$$

Coupure

$$cut\ \frac{\Gamma \vdash B, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma \vdash \Delta}$$

Règles logiques gauches

$$\bot_{\mathcal{L}}\ \frac{}{\Gamma, \bot \vdash \Delta}$$

$$\wedge_{\mathcal{L}}\ \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}$$

$$\vee_{\mathcal{L}}\ \frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}$$

$$\Rightarrow_{\mathcal{L}}\ \frac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}$$

$$\forall_{\mathcal{L}}\ \frac{\Gamma, A[t/x] \vdash \Delta}{\Gamma, \forall x.A \vdash \Delta}$$

$$\exists_{\mathcal{L}}\ \frac{\Gamma, A \vdash \Delta}{\Gamma, \exists x.A \vdash \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta)$$

Règles logiques droites

$$\top_{\mathcal{R}}\ \frac{}{\Gamma \vdash \top, \Delta}$$

$$\wedge_{\mathcal{R}}\ \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

$$\vee_{\mathcal{R}}\ \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}$$

$$\Rightarrow_{\mathcal{R}}\ \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}$$

$$\forall_{\mathcal{R}}\ \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall x.A, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta)$$

$$\exists_{\mathcal{R}}\ \frac{\Gamma \vdash A[t/x], \Delta}{\Gamma \vdash \exists x.A, \Delta}$$

Règles structurelles gauches

$$weak_{\mathcal{L}}\ \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}$$

$$contr_{\mathcal{L}}\ \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}$$

Règles structurelles droites

$$weak_{\mathcal{R}}\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}$$

$$contr_{\mathcal{R}}\ \frac{\Gamma, A \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$$

Pour les règles $\forall_{\mathcal{R}}$ et $\exists_{\mathcal{L}}$, la condition d'application $x \notin \mathcal{FV}(\Gamma, \Delta)$ signifie que la variable x ne doit être libre ni dans Γ, ni dans Δ. La notation $A[t/x]$, signifie que l'on substitue un terme t quelconque à toutes les occurences libres de x dans la proposition A.

FIG. 3 – Règles déductives de LK

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \in \emptyset \vdash x \in A,\ \emptyset \subseteq A,\ x \in \emptyset}\,ax \qquad \dfrac{}{\bot,\ x \in \emptyset \vdash x \in A,\ \emptyset \subseteq A}\,\bot_{\mathcal{L}}}{x \in \emptyset \Rightarrow \bot,\ x \in \emptyset \vdash x \in A,\ \emptyset \subseteq A}\,\Rightarrow_{\mathcal{L}}}{x \in \emptyset \Leftrightarrow \bot,\ x \in \emptyset \vdash x \in A,\ \emptyset \subseteq A}\,\wedge_{\mathcal{L}}}{\forall x.(x \in \emptyset \Leftrightarrow \bot),\ x \in \emptyset \vdash x \in A,\ \emptyset \subseteq A}\,\forall_{\mathcal{L}}}{\forall x.(x \in \emptyset \Leftrightarrow \bot) \vdash x \in \emptyset \Rightarrow x \in A,\ \emptyset \subseteq A}\,\Rightarrow_{\mathcal{R}}}{\forall x.(x \in \emptyset \Leftrightarrow \bot) \vdash \forall x.(x \in \emptyset \Rightarrow x \in A),\ \emptyset \subseteq A}\,\forall_{\mathcal{R}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots \qquad \dfrac{}{\forall x.(x \in \emptyset \Leftrightarrow \bot),\ \emptyset \subseteq A \vdash \emptyset \subseteq A}\,ax}{\forall x.(x \in \emptyset \Leftrightarrow \bot),\ \emptyset \subseteq A \Leftarrow \forall x.(x \in \emptyset \Rightarrow x \in A) \vdash \emptyset \subseteq A}\,\Rightarrow_{\mathcal{L}}}{\forall x.(x \in \emptyset \Leftrightarrow \bot),\ \emptyset \subseteq A \Leftrightarrow \forall x.(x \in \emptyset \Rightarrow x \in A) \vdash \emptyset \subseteq A}\,\wedge_{\mathcal{L}}}{\forall x.(x \in \emptyset \Leftrightarrow \bot),\ \forall Z.(\emptyset \subseteq Z \Leftrightarrow \forall x.(x \in \emptyset \Rightarrow x \in Z)) \vdash \emptyset \subseteq A}\,\forall_{\mathcal{L}}}{\forall x.(x \in \emptyset \Leftrightarrow \bot),\ \forall Y.\forall Z.(Y \subseteq Z \Leftrightarrow \forall x.(x \in Y \Rightarrow x \in Z)) \vdash \emptyset \subseteq A}\,\forall_{\mathcal{L}}$$

FIG. 4 – un arbre de dérivation dans LK

On remarquera que les preuves dans cette logique sont très verbeuses et contiennent beaucoup d'étapes vides d'intérêt mathématique. Nous allons voir en quoi notre système de séquents extensible : LK_+ nous permet entre autres de réduire considérablement la taille des preuves.

4.2.2 Séquents intuitionnistes

La logique intuitionniste réfute le raisonnement par l'absurde (traduit également par le principe du tiers-exclu, $A \vee (A \Rightarrow \bot)$) car il ne permet pas la dérivation systématique de preuves constructivistes, *i.e.* qui fournissent un témoin "tangible" de la vérité d'une démonstration. Typiquement, la preuve de la proposition «Il existe des nombres irrationnels a et b, tels que a^b est un nombre rationnel» [1] s'appuie sur le principe du tiers exclu tandis ce que le procédé d'orthonormalisation de Gram-Schmidt fournit à la fois une preuve et un objet mathématique témoin de cette preuve. Cette distinction prend son sens dans le cadre de l'isomorphisme de Curry-Howard (*cf* [SU98] par exemple) qui associe un programme à toute preuve intuitionniste. On peut alors en théorie (et aussi en pratique [Let04]) extraire automatiquement un programme qui fournit un objet de la preuve intuitionniste de l'existence de cet objet.

Voyons à présent brièvement ce qui différencie le calcul des séquents classique LK de LJ, le calcul des séquents intuitionniste.

Définition 4.4 (Séquent intuitionniste)**.** *Un séquent intuitionniste est un couple* $\Gamma \vdash \sigma$ *où* Γ *est un contexte et* σ *une proposition logique quelconque.*

Définition 4.5 (Règles de déduction de LJ)**.** *Les règles de déduction de LJ sont des règles de déduction de LK (cf 2.8) dont les prémisses et conclusions sont des séquents intuitionnistes.*

[1] Posons $b = \sqrt{2}$, si $\sqrt{2}^{\sqrt{2}}$ est rationnel prenons $a = \sqrt{2}$, sinon prenons $a = \sqrt{2}^{\sqrt{2}}$ et nous avons $a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\times\sqrt{2}} = \sqrt{2}^2 = 2$ qui donne le résultat puisque 2 est évidemment rationnel

$$\top_{\mathcal{L}} \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta} \qquad\qquad \bot_{\mathcal{R}} \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}$$

FIG. 5 – Règles supplémentaires pour le calcul des règles de LK_+

La seule différence entre les séquents intuitionnistes et classiques est donc le nombre de propositions présentes dans la conclusion des séquents. Cela se comprend bien si l'on considère la virgule à droite comme un $\vee$.

4.3 LK_+, un calcul des séquents extensible

Nous présentons ici un calcul des séquents extensible calqué sur le principe de la déduction surnaturelle. Étant donné un système de réécriture $\mathcal{R}$ formé de règles réécrivant les termes en termes et les propositions atomiques en propositions quelconques, on dérive un ensemble de règles de déduction qui forment, avec les règles de LK, le système déductif LK_+.

Nous décrivons d'abord une première version "naïve" de ce système et de ses propriétés afin de mettre en évidence les principaux mécanismes qui entrent en jeu avant d'en présenter une version améliorée qui possède des propriétés plus intéressantes.

4.3.1 Une première version de LK_+

Dérivation des nouvelles règles Le procédé de dérivation des nouvelles règles de déduction à partir des règles de réécriture est similaire à celui de la déduction surnaturelle. Cependant, on ne dérive plus des règles d'introduction et d'élimination mais des règles droites et gauches, la proposition atomique se retrouvant toujours dans le séquent conclusion, tantôt à droite, tantôt à gauche du signe $\vdash_+$. Voyons en détail la procédure de calcul de ces nouvelles règles.

Définition 4.6 (Calcul des nouvelles règles de LK_+, première version)**.** *Soit Calc un ensemble de règles composé du sous-ensemble des règles de LK formé par ax, $\bot_{\mathcal{L}}$, $\top_{\mathcal{R}}$, $\vee_{\mathcal{L}}$, $\vee_{\mathcal{R}}$, $\wedge_{\mathcal{L}}$, $\wedge_{\mathcal{R}}$, $\Rightarrow_{\mathcal{L}}$, $\Rightarrow_{\mathcal{R}}$, $\forall_{\mathcal{R}}$ et $\exists_{\mathcal{L}}$ ainsi que des règles $\top_{\mathcal{L}}$ et $\bot_{\mathcal{R}}$ de la figure 5. Soit une règle de réécriture $R : P \to \phi$ où P est atomique.*

1. *Pour trouver la règle droite associée à R, on initialise la procédure par le séquent $\Gamma \vdash \phi, \Delta$. On y applique ensuite les règles de Calc jusqu'à ce qu'aucune règle ne soit plus applicable. On collecte alors l'ensemble des prémisses, les conditions d'application et la conclusion, et on remplace ϕ par P pour bien obtenir la règle $R_{\mathcal{R}}$.*
2. *Pour trouver la règle gauche associée à R, on initialise la procédure par le séquent $\Gamma, \phi \vdash \Delta$. On y applique les règles de Calc et on récupère la nouvelle règle gauche de même que pour le point précédent pour obtenir la règle $R_{\mathcal{L}}$.*

On peut relever les points suivants :

– La structure des règles $\vee_{\mathcal{R}}$ et $\wedge_{\mathcal{L}}$ dans notre présentation du système LK garantissent qu'on obtient une et une seule règle de déduction droite et une et une seule règle de déduction gauche par règle de réécriture. En effet, une manière classique de présenter ces règles est d'en proposer deux versions

par connecteur (comme pour les règle $\vee_I$ en déduction naturelle) pour éviter la redondance avec les règles structurelles. On aurait alors obtenu plusieurs règles droites et gauches à l'issue de la procédure. Cependant, de telles règles ne permettent pas d'assurer la complétude de LK_+ par rapport à LK, ce qui explique notre choix.
- Les règles $\forall_{\mathcal{L}}$ et $\exists_{\mathcal{R}}$ sont absentes de *Calc*. Cela est dû à leur caractère asynchrone, nous rediscuterons ce point par la suite.
- Les règles $\top_{\mathcal{L}}$ et $\bot_{\mathcal{R}}$ de la figure 5 ont été ajoutées aux règles de calcul pour simplifier les prémisses des nouvelles règles. En effet, la présence de $\bot$ dans la conclusion ou de $\top$ dans les prémisses d'un séquent n'apporte rien à la preuve. Ces règles sont évidemment admissibles dans LK, et on a donc la propriété suivante : il existe un arbre de dérivation dans LK de $\Gamma \vdash \phi, \Delta$ qui a pour feuilles les prémisses de $R_{\mathcal{R}}$ et un arbre de dérivation dans LK de $\Gamma, \phi \vdash \Delta$ qui a pour feuilles les prémisses de $R_{\mathcal{L}}$.

Nous pouvons à présent définir formellement la

Définition 4.7 (Système de déduction LK_+)**.** *Étant donné un système de réécriture $\mathcal{R} = \mathcal{R}_t \cup \mathcal{R}_P$ tel que $\mathcal{R}_t$ réécrit des termes et $\mathcal{R}_P$ réécrit des propositions atomiques, le* système de déduction LK_+ *associé à $\mathcal{R}$ est donné par :*
- *les règles de la logique du premier ordre ;*
- *les règles construites à partir de $\mathcal{R}_P$;*

où toutes les propositions sont considérées modulo $=_{\mathcal{R}_t}$. On notera $\Gamma \vdash_+ \phi$ les jugements dérivés dans un tel système.

Reprenons notre exemple de la théorie des ensembles pour illustrer le fonctionnement de LK_+. On rappelle que le système de réécriture est formé des deux règles suivantes, qui traduisent les définition de l'inclusion et de l'ensemble vide :

$$\mathcal{R} = \left\{ \begin{array}{rcl} \subseteq_{def} : X \subseteq Y & \rightarrow & \forall x (x \in X \Rightarrow x \in Y) \\ \emptyset_{def} : \quad x \in \emptyset & \rightarrow & \bot \end{array} \right.$$

On calcule alors ainsi les nouvelles règles droite et gauche pour la définition de l'inclusion :

$$\begin{array}{cc} \Gamma, \forall x.(x \in X \Rightarrow x \in Y) \vdash \Delta & \Rightarrow_{\mathcal{R}} \dfrac{\Gamma, x \in X \vdash x \in Y, \Delta}{\Gamma \vdash x \in X \Rightarrow x \in Y, \Delta} \\ \downarrow & \forall_{\mathcal{R}} \dfrac{}{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y), \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \\ \subseteq_{def_{\mathcal{L}}} \dfrac{\Gamma, \forall x.(x \in X \Rightarrow x \in Y) \vdash_+ \Delta}{\Gamma, X \subseteq Y \vdash_+ \Delta} & \downarrow \\ & \subseteq_{def_{\mathcal{R}}} \dfrac{\Gamma, x \in X \vdash_+ x \in Y, \Delta}{\Gamma \vdash_+ X \subseteq Y, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \end{array}$$

On remarque que la règle $\subseteq_{def_{\mathcal{L}}}$ n'est pas du tout décomposée à cause de la présence du connecteur $\forall$ en tête de la proposition atomique. En revanche, elle traduit quand même le fait que l'on remplace $X \subseteq Y$ par sa définition. On est dans le cas de la déduction avec *folding* et *unfolding* de Prawitz [Pra65], qui est en quelque sorte le degré 0 de notre système. De la même manière, on obtient les règles suivantes concernant l'ensemble vide :

$$\emptyset_{def_{\mathcal{L}}} \frac{}{\Gamma, x \in \emptyset \vdash_+ \Delta} \qquad \emptyset_{def_{\mathcal{R}}} \frac{\Gamma \vdash_+ \bot, \Delta}{\Gamma \vdash_+ x \in \emptyset, \Delta}$$

On peut remarquer qu'à l'instar des règles de la déduction surnaturelle, ces règles traduisent assez fidèlement le raisonnement mathématique. On peut alors dériver la preuve suivante de $\emptyset \in A$ dans la théorie vide :

$$\begin{array}{r} \emptyset_{def_{\mathcal{L}}} \\ \subseteq_{def_{\mathcal{R}}} \end{array} \frac{\dfrac{}{x \in \emptyset \vdash_+ x \in A}}{\vdash_+ \emptyset \in A}\ x \notin \mathcal{FV}(\Gamma, \Delta)$$

La preuve obtenue est beaucoup plus concise et expressive que celle de la figure 4. De plus on peut noter que la prise en compte des connecteurs à gauche nous a permis de décomposer un pas de plus qu'en déduction surnaturelle la formule de la règle $\emptyset_{def_{\mathcal{L}}}$, ce qui raccourcit encore plus la preuve.

Propriétés de LK$_+$ Voyons maintenant comment se comporte LK$_+$, en particulier par rapport au système LK. Le théorème suivant énonce la correction et la complétude de LK vis à vis de ce dernier, *i.e.* toute preuve dans LK est correcte dans LK$_+$, et tout séquent prouvable dans LK$_+$ est prouvable dans LK.

Theorème 2 (Équivalence)**.** *Pour tout système déductif* LK$_+$ *dérivé d'un système de réécriture* $\mathcal{R}$*, il existe une théorie* $\mathcal{T}h$ *telle que* $\Gamma \vdash_+ \phi$ *si et seulement si* $\Gamma, \mathcal{T}h \vdash \phi$*.*

Démonstration. La preuve complète est fournie en annexe B.1. L'idée générale est de fournir une procédure générique pour passer d'une preuve de LK à une preuve de LK$_+$ et inversement.

Pour construire la théorie $\mathcal{T}h$ on procède comme en déduction surnaturelle. S'il n'existe pas déjà un prédicat d'égalité $=$, on en ajoute un ainsi que les axiomes correspondants. Chaque règle de réécriture $l \to r$ se traduit alors en un axiome :

- $\forall \overline{x}.(l = r)$ si $l \to r \in \mathcal{R}_t$
- $\forall \overline{x}.(l \Leftrightarrow r)$ si $l \to r \in \mathcal{R}_P$

où $\overline{x}$ est l'ensemble des variables libres dans l et r.

Si $\Gamma \vdash_+ \phi$ **alors** $\Gamma, \mathcal{T}h \vdash \phi$: Pour prouver $\Gamma, \mathcal{T}h \vdash \phi$, on peut reproduire les règles de LK et les axiomes de Γ utilisés dans la preuve de $\Gamma \vdash_+ \phi$. Il suffit donc de traduire les nouvelles règles de déduction en preuves dans LK pour obtenir la correction du système.

Considérons une "sur-règle" droite calculée à partir de la règle de réécriture $R : P \Leftrightarrow \phi$ en utilisant les règles de la procédure présentée en 4.6.

$$R_{\mathcal{R}}\ \frac{H_1 \quad \dots \quad H_n}{\Gamma \vdash_+ P, \Delta}\ \mathcal{C}$$

Par construction de $R_{\mathcal{R}}$, il existe une dérivation dans LK dont les prémisses sont $H_1 \dots H_n$, dont la conclusion contient ϕ et qui a les mêmes conditions d'application $\mathcal{C}$. En affaiblissant (règles *weak*) Γ en $\Gamma, \mathcal{T}h$ dans la conclusion de cette dernière dérivation, on peut construire la preuve de $\Gamma, \mathcal{T}h \vdash P$ suivante.

$$\dfrac{\mathcal{Th} = \mathcal{Th}' \cup \{\phi \Leftrightarrow P\}\ \dfrac{\wedge_{\mathcal{L}}\ \dfrac{\Rightarrow_{\mathcal{L}}\ \dfrac{\begin{array}{c}H_1\quad\ldots\quad H_n\\ \ddots\quad\iddots\\ \Gamma, \mathcal{Th} \vdash \phi, \Delta\end{array} \qquad ax\ \dfrac{}{\Gamma, \mathcal{Th}, P \vdash P, \Delta}}{\Gamma, \mathcal{Th}, \phi \Rightarrow P \vdash P, \Delta}}{\Gamma, \mathcal{Th}, \phi \Leftrightarrow P \vdash P, \Delta}}{\Gamma, \mathcal{Th} \vdash P, \Delta}$$

Le cas gauche est très similaire au cas droit. Par construction de la règle gauche

$$R_{\mathcal{L}}\ \dfrac{H_1 \quad \ldots \quad H_n}{\Gamma, P \vdash_+ \Delta}\ \mathcal{C}$$

il existe une dérivation dans LK qui a les mêmes prémisses et conditions d'application que $R_{\mathcal{L}}$ et dont la conclusion contient ϕ. Nous pouvons utiliser cette preuve dans la dérivation de $\Gamma, \mathcal{Th}, P \vdash \Delta$ suivante.

$$\dfrac{\mathcal{Th} = \mathcal{Th}' \cup \{\phi \Leftrightarrow P\}\ \dfrac{\wedge_{\mathcal{L}}\ \dfrac{\Rightarrow_{\mathcal{L}}\ \dfrac{ax\ \dfrac{}{\Gamma, \mathcal{Th}, P \vdash P, \Delta} \qquad \begin{array}{c}H_1\quad\ldots\quad H_n\\ \ddots\quad\iddots\\ \Gamma, \mathcal{Th}, \phi \vdash \Delta\end{array}}{\Gamma, \mathcal{Th}, P \Rightarrow \phi, P \vdash \Delta}}{\Gamma, \mathcal{Th}, \phi \Leftrightarrow P, P \vdash \Delta}}{\Gamma, \mathcal{Th}, P \vdash \Delta}$$

On a donc prouvé la correction de LK_+ par rapport à LK et la preuve étant constructive, on a obtenu un moyen systématique de traduire une preuve dans LK_+ en une preuve dans LK. Cette traduction nous sera utile pour communiquer des preuves réalisées dans LK_+ à des assistants de preuves tels que Coq ou Isabelle.

Si $\Gamma, \mathcal{Th} \vdash \phi$ **alors** $\Gamma \vdash_+ \phi$: Soit Π une preuve de $\Gamma, \mathcal{Th} \vdash_+ \phi$. Elle est obtenue trivialement en reproduisant les règles de LK utilisées dans la preuve de $\Gamma, \mathcal{Th} \vdash \phi$. Soient $\mathcal{A}x_1, \ldots, \mathcal{A}x_n$ les axiomes contenus dans la théorie $\mathcal{Th}$ et utilisés dans la preuve de $\Gamma, \mathcal{Th} \vdash \phi$, nous pouvons dériver la preuve suivante de $\Gamma \vdash_+ \phi$.

$$\dfrac{\begin{array}{l}weak\ \dfrac{\vdash_+ \mathcal{A}x_1}{\Gamma \vdash_+ \mathcal{A}x_1}\\ cut\end{array}\ \dfrac{\begin{array}{l}weak\ \dfrac{\vdash_+ \mathcal{A}x_n}{\Gamma, \mathcal{A}x_1, \ldots, \mathcal{A}x_{n-1} \vdash_+ \mathcal{A}x_n}\\ cut\end{array}\ \dfrac{\Pi}{\Gamma, \mathcal{A}x_1, \ldots, \mathcal{A}x_n \vdash_+ \phi} \qquad cut\ \dfrac{\iddots}{\Gamma, \mathcal{A}x_1 \vdash_+ \phi}}{\Gamma \vdash_+ \phi}$$

En conséquence, il suffit de prouver les axiomes de $\mathcal{Th}$ dans LK pour obtenir une preuve de $\Gamma \vdash_+ \phi$. En d'autres termes, pour chaque axiome $P_i \Leftrightarrow \phi_i$ de $\mathcal{Th}$, nous devons prouver $P_i \vdash_+ \phi_i$ et $\phi_i \vdash_+ P_i$.

La preuve est réalisée par induction sur la structure des propositions ϕ_i.

Hypothèse 2.1 (Hypothèse d'induction)**.** *Pour chaque règle R de $\mathcal{R}$ définie par $P \to \phi$, il existe une preuve de $P \vdash_+ \phi$ et une preuve de $\phi \vdash_+ P$ dans LK_+ qui ont respectivement les formes suivantes*

$$\dfrac{\dfrac{\dfrac{}{R_{\mathcal{L}}}\,Ax \quad \ldots \quad Ax \quad \ldots \quad \dfrac{}{R_{\mathcal{L}}}\,Ax \quad \ldots \quad Ax}{R^*}}{P \vdash_+ \phi} \qquad \textit{et} \qquad \dfrac{\dfrac{}{R_{\mathcal{R}}}\,Ax \quad \ldots \quad Ax}{\Gamma, \phi \vdash_+ P}$$

où $R_{\mathcal{R}}$ et $R_{\mathcal{L}}$ sont les règles de LK_+ dérivées de R et où R^* signifie "zéro ou plusieurs applications des règles droites de LK".

Considérons la règle $R : P \to \phi$. On raisonne par cas sur le connecteur de tête de ϕ. Les cas $\forall$ et $\exists$ sont traités ici à titre d'exemple pour montrer un raisonnement sur une règle gauche et un raisonnement à droite. Les autres cas sont similaires et sont traités en annexe.

ϕ est de la forme $\forall x.\phi_1(x)$: On considère une règle de réécriture sur les propositions $R_1 : P_1(x) \to \phi_1(x)$ avec $P_1(x)$ atomique. Commençons par prouver $\phi \vdash_+ P$. Par construction de la règle $R_{\mathcal{R}}$

$$\forall_{\mathcal{R}}\,\dfrac{\begin{array}{c}\Gamma, \Gamma_1(x) \vdash \Delta_1(x), \Delta \quad \ldots \quad \Gamma, \Gamma_n(x) \vdash \Delta_n(x), \Delta \\ \ddots \quad \iddots \quad \mathcal{C} \\ \Gamma \vdash \phi_1(x), \Delta\end{array}}{\Gamma \vdash \phi, \Delta}\, x \notin \mathcal{FV}(\Gamma, \Delta)$$

la règle

$$R_{\mathcal{R}}\,\dfrac{\Gamma, \Gamma_1(x) \vdash_+ \Delta_1(x), \Delta \;\ldots\; \Gamma, \Gamma_n(x) \vdash_+ \Delta_n(x), \Delta}{\Gamma \vdash_+ P, \Delta}\, \mathcal{C} \cup \{x \notin \mathcal{FV}(\Gamma, \Delta)\}$$

et la règle

$$R_{1_{\mathcal{R}}}\,\dfrac{\Gamma, \Gamma_1(x) \vdash_+ \Delta_1(x), \Delta \quad \ldots \quad \Gamma, \Gamma_n(x) \vdash_+ \Delta_n(x), \Delta}{\Gamma \vdash_+ P_1(x), \Delta}\, \mathcal{C}$$

partagent les mêmes prémisses $\Gamma_1(x), \ldots, \Gamma_n(x)$ et $\Delta_1(x), \ldots, \Delta_n(x)$. $\mathcal{C}$ est ici un ensemble de conditions d'application concernant les variables liées de ϕ_1.

Par hypothèse d'induction, il existe une preuve de $\phi_1(x) \vdash_+ P_1(x)$ dans LK_+ de la forme

$$R_{1_{\mathcal{R}}}\,\dfrac{\begin{array}{ccc}\Pi_1(x) & & \Pi_n(x) \\ \phi_1(x), \Gamma_1(x) \vdash_+ \Delta_1(x) & \ldots & \phi_1(x), \Gamma_n(x) \vdash_+ \Delta_n(x)\end{array}}{\phi_1(x) \vdash_+ P_1(x)}$$

Nous pouvons donc utiliser ces $\Pi_{1\ldots n}(x)$ pour construire la preuve suivante de $\phi \vdash_+ P$

$$R_{\mathcal{R}}\ \dfrac{\begin{array}{cc}\Pi_1(x) & \Pi_n(x)\\ \forall_{\mathcal{L}}\ \dfrac{\phi_1(x),\Gamma_1(x)\vdash_+\Delta_1(x)}{\forall x.\phi_1(x),\Gamma_1(x)\vdash_+\Delta_1(x)} & \ldots\ \forall_{\mathcal{L}}\ \dfrac{\phi_1(x),\Gamma_n(x)\vdash_+\Delta_n(x)}{\forall x.\phi_1(x),\Gamma_n(x)\vdash_+\Delta_n(x)}\end{array}}{\underbrace{\forall x.\phi_1(x)}_{\phi}\vdash_+ P}$$

Les conditions $\mathcal{C}$ ne sont pas violées lorsqu'on applique $R_{\mathcal{R}}$ puisqu'elles ne concernent que les variables liées de ϕ_1. D'autre part x est bien liée dans $\forall x.\phi_1(x)$.

Prouvons maintenant $P\vdash_+\phi$. Ce cas est trivial car on ne décompose pas les formules présentant $\forall$ comme connecteur de tête dans la partie gauche des séquents lors du calcul des nouvelles règles. C'est pourquoi la nouvelle règle gauche est simplement

$$R_{\mathcal{L}}\ \frac{\Gamma,\forall x.\phi_1(x)\vdash_+\Delta}{\Gamma,P\vdash_+\Delta}$$

et on obtient immédiatement la preuve suivante de $P\vdash_+\phi$

$$R_{\mathcal{L}}\ \frac{ax\ \dfrac{}{\forall x.\phi_1(x)\vdash_+\forall x.\phi_1(x)}}{P\vdash_+\forall x.\phi_1(x)}$$

ϕ **est de la forme** $\exists x.\phi_1(x)$: On considère une règle de réécriture R_1 : $P_1(x)\to\phi_1(x)$ telle que $P_1(x)$ est une proposition atomique. Commençons par prouver $P\vdash_+\phi$. Par construction de la règle $R_{\mathcal{L}}$

$$\exists_{\mathcal{L}}\ \frac{\begin{array}{c}\Gamma,\Gamma_1(x)\vdash\Delta_1(x),\Delta\quad\ldots\quad\Gamma,\Gamma_n(x)\vdash\Delta_n(x),\Delta\\ \ddots\quad\iddots\quad\mathcal{C}\\ \Gamma,\phi_1(x)\vdash\Delta\end{array}}{\Gamma,\phi\vdash\Delta}\ x\notin\mathcal{FV}(\Gamma,\Delta)$$

la règle

$$R_{\mathcal{L}}\ \frac{\Gamma,\Gamma_1(x)\vdash_+\Delta_1(x),\Delta\quad\ldots\quad\Gamma,\Gamma_n(x)\vdash_+\Delta_n(x),\Delta}{\Gamma,P\vdash_+\Delta}\ \mathcal{C}\cup\{x\notin\mathcal{FV}(\Gamma,\Delta)\}$$

et la règle

$$R_{1_{\mathcal{L}}}\ \frac{\Gamma,\Gamma_1(x)\vdash_+\Delta_1(x),\Delta\quad\ldots\quad\Gamma,\Gamma_n(x)\vdash_+\Delta_n(x),\Delta}{\Gamma,P_1(x)\vdash_+\Delta}\ \mathcal{C}$$

partagent les mêmes prémisses $\Gamma_1(x),\ldots,\Gamma_n(x)$ and $\Delta_1(x),\ldots,\Delta_n(x)$. $\mathcal{C}$ est un ensemble de conditions d'application concernant les variables liées de ϕ_1.
Par hypothèse d'induction, il existe une preuve $\phi_1(x)\vdash_+ P_1(x)$ dans LK_+ telle que

$$R^* \dfrac{\begin{array}{ccc}\Pi_1 & \ldots & \Pi_m \\ & \ddots \quad .^{\cdot}^{\cdot} & \end{array}}{P_1(x) \vdash_+ \phi_1(x)}$$

où $\Pi_1 \ldots \Pi_m$ sont les preuves ci-dessous

$$R_{1_\mathcal{L}} \dfrac{\begin{array}{cc}\Pi_{1,1}(x) & \Pi_{1,n}(x) \\ \Gamma'_1(x), \Gamma_1(x) \vdash_+ \Delta'_1(x), \Delta_1(x) & \ldots \ \Gamma'_1(x), \Gamma_n(x) \vdash_+ \Delta'_1(x), \Delta_n(x)\end{array}}{P_1(x), \Gamma'_1(x) \vdash_+ \Delta'_1(x)}$$

$$\vdots$$

$$R_{1_\mathcal{L}} \dfrac{\begin{array}{cc}\Pi_{m,1}(x) & \Pi_{m,n}(x) \\ \Gamma'_m(x), \Gamma_1(x) \vdash_+ \Delta'_m(x), \Delta_1(x) & \ldots \ \Gamma'_m(x), \Gamma_n(x) \vdash_+ \Delta'_m(x), \Delta_n(x)\end{array}}{P_1(x), \Gamma'_m(x) \vdash_+ \Delta'_m(x)}$$

On peut maintenant dériver la preuve de $\phi \vdash_+ P$ suivante

$$\begin{array}{c} \exists_\mathcal{R} \\ R_\mathcal{L} \end{array} \dfrac{\begin{array}{ccc}\dfrac{\Pi'_1}{\Gamma_1(x) \vdash_+ \Delta_1(x), \exists x.\phi_1(x)} & \ldots & \exists_\mathcal{R} \dfrac{\Pi'_n}{\Gamma_n(x) \vdash_+ \Delta_n(x), \exists x.\phi_1(x)}\end{array}}{P \vdash_+ \exists x.\phi_1(x)}$$

où les $\Pi'_1 \ldots \Pi'_n$ sont obtenus en appliquant des règles droites et gauche à ϕ_1 jusqu'à ce que les $\Pi_{i,j}(x)$ soient applicables, comme décrit ci-dessous

$$\dfrac{\begin{array}{ccc}\Pi_{1,1}(x) & & \Pi_{m,1}(x) \\ \Gamma'_1(x), \Gamma_1(x) \vdash_+ \Delta'_1(x), \Delta_1(x) & \ldots & \Gamma'_m(x), \Gamma_1(x) \vdash_+ \Delta'_m(x), \Delta_1(x) \\ & \ddots \quad .^{\cdot}^{\cdot} & \end{array}}{\Gamma_1(x) \vdash_+ \Delta_1(x), \phi_1(x)}$$

$$\vdots$$

$$\dfrac{\begin{array}{ccc}\Pi_{m,1}(x) & & \Pi_{m,n}(x) \\ \Gamma'_1(x), \Gamma_n(x) \vdash_+ \Delta'_1(x), \Delta_n(x) & \ldots & \Gamma'_m(x), \Gamma_n(x) \vdash_+ \Delta'_m(x), \Delta_n(x) \\ & \ddots \quad .^{\cdot}^{\cdot} & \end{array}}{\Gamma_n(x) \vdash_+ \Delta_n(x), \phi_1(x)}$$

On ne viole aucune condition en appliquant $R_\mathcal{L}$ car x est liée partout et les autres conditions dans $\mathcal{C}$ ne concernent que les variables liées de ϕ_1.

Prouvons maintenant $\phi \vdash_+ P$. Comme pour le cas $\forall$, c'est trivial car on ne décompose pas les $\exists$ du côté droit des séquents lors du calcul des nouvelles règles. Ainsi, la nouvelle règle est la suivante.

$$R_{\mathcal{R}} \frac{\Gamma \vdash_+ \exists x.\phi_1(x), \Delta}{\Gamma \vdash_+ \phi, \Delta}$$

Et on obtient immédiatement la preuve de $\phi \vdash_+ P$.

$$\cfrac{\cfrac{}{ax}}{R_{\mathcal{R}}} \frac{\exists x.\phi_1(x) \vdash_+ \exists x.\phi_1(x)}{\exists x.\phi_1(x) \vdash P}$$

ϕ **est de la forme ...** Les autres cas sont traités en annexe.

□

Le théorème suivant ramène la preuve de cohérence de toute théorie à une preuve d'élimination des coupures dans LK_+.

Theorème 3 (Cohérence des preuves sans coupures dans un contexte vide)**.** *Étant donné un système déductif* LK_+ *dérivé d'un système de réécriture* R*, il n'existe pas de preuve de* $\vdash_+ \bot$ *sans coupure.*

Démonstration. Une preuve de $\vdash_+ \bot$ commence nécessairement par l'application de la règle ax ou d'une règle droite de LK_+. Les règles de LK_+ sont composées des règles de LK et des nouvelles règles calculées.

- Le contexte de la règle ax n'est pas vide, donc ax ne peut être appliqué.
- La seule règle droite de LK qui ne présente pas de formule composée en conclusion est $cut_{\mathcal{R}}$, mais nous construisons une preuve sans coupures.
- Une règle droite calculée ne peut présenter $\bot$ en conclusion que si elle traduit une règle de réécriture de la forme $\bot \to \phi$, mais ce n'est pas autorisé, $\bot$ n'étant pas une formule atomique.

□

En déduction modulo, l'élimination des coupures a été prouvée pour plusieurs congruences dans [DW98], parmis lesquelles les congruences définies par un système de réécriture confluent, terminant et sans quantificateurs. Ce genre de critère fonctionne probablement pour LK_+, mais la question reste ouverte à l'heure actuelle.

4.3.2 Version améliorée

Motivation Le système LK_+ tel qu'il est présenté précdemment souffre de l'omission des règles $\forall_{\mathcal{L}}$ et $\exists_{\mathcal{R}}$ lors du calcul des nouvelles règles de déduction. Considérons par exemple la règle de réécriture présentée en 3.3 qui définit les entiers comme les objets mathématiques vérifiant l'ensemble des prédicats inductifs :

$$\in_{\mathbb{N}}\colon n \in \mathbb{N} \;\to\; \forall P.(0 \in P \Rightarrow \forall m.(m \in P \Rightarrow S(m) \in P) \Rightarrow n \in P)$$

Comme on travaille au premier ordre, on encode ici l'application d'un prédicat unaire (P ici) à un terme (ici 0, n ou m) par l'application du prédicat binaire $\in$ à une variable représentant le prédicat unaire et à une variable représentant l'argument de ce prédicat unaire. Cet encodage des prédicats sous forme de variables nous permet de quantifier sur les prédicats et ainsi d'exprimer cette proposition en pure logique du premier ordre plutôt que sous la forme d'un schéma d'axiomes par exemple. Cette approche est décrite dans [Kir06] par Florent Kirchner dans le cadre de la déduction modulo. On compte ainsi obtenir une nouvelle règle de déduction qui traduise le raisonnement par récurrence. Toutefois, la proposition présentant le connecteur $\forall$ en tête, la règle gauche obtenue à l'issue de la traduction est décevante :

$$\in_{\mathbb{N}_\mathcal{L}} \frac{\Gamma, \forall P.(0 \in P \Rightarrow \forall m.(m \in P \Rightarrow S(m) \in P) \Rightarrow n \in P) \vdash_+ \Delta}{\Gamma, n \in N \vdash_+ \Delta}$$

Nous allons voir dans un premier temps ce qui motive cette restriction du calcul des nouvelles règles avant de proposer une solution qui, en plus de contourner le problème, est chargée de sémantique.

Problématique La restriction concernant les règles $\forall_\mathcal{L}$ et $\exists_\mathcal{R}$ dans l'algorithme de calcul des nouvelles règles est motivée par les problèmes qui surviennent au niveau des quantificateurs lorsque l'on décompose tous les connecteurs sans discernement. Voyons cela sur un exemple. Considérons la règle de réécriture $R : P \rightarrow \forall x \exists y.\phi(x, y)$. Supposons que nous autorisions l'utilisation de la règle $\exists_\mathcal{R}$ lors du calcul de la règle $R_\mathcal{R}$. Nous obtenons la règle suivante :

$$R_\mathcal{R} \frac{\Gamma \vdash \phi(x, t), \Delta}{\Gamma \vdash P, \Delta}\ x \notin FV(\Gamma, \Delta)$$

Il est alors impossible de dériver une preuve de $\forall x \exists y.\phi(x, y) \vdash_+ P$ (*i.e.* P est impliqué par sa définition) dans LK_+ en utilisant cette nouvelle règle, quel que soit l'ordre dans lequel nous appliquons les règles de déduction :

$$R_\mathcal{R} \frac{\forall_\mathcal{L} \dfrac{\exists_\mathcal{L} \dfrac{\phi(x, y) \vdash_+ \phi(x, t)}{\exists y.\phi(x, y) \vdash_+ \phi(x, t)}}{\forall x \exists y.\phi(x, y) \vdash_+ \phi(x, t)}}{\forall x \exists y.\phi(x, y) \vdash_+ P} \qquad \forall_\mathcal{L} \frac{\exists_\mathcal{L} \dfrac{\phi(x, y) \vdash_+ P}{\exists y.\phi(x, y) \vdash_+ P}}{\forall x \exists y.\phi(x, y) \vdash_+ P}$$

Dans la preuve de gauche, aucune règle autre que la coupure ne peut plus être appliquée. Si nous avions instancié y par y plutôt que par t lors de l'application de la règle $R_\mathcal{R}$, nous n'aurions pas pu appliquer $\exists_\mathcal{L}$, y ayant alors été libre dans $\phi(x, y)$. La preuve de droite échoue parce qu'il est impossible d'appliquer $R_\mathcal{R}$ sans violer la condition $x \notin FV(\Gamma, \Delta)$. Une troisième solution serait l'application successive des règles $\forall_\mathcal{L}$, $\mathcal{A}x_\mathcal{R}$ puis $\exists_\mathcal{L}$. Cela nous ramène au problème de la preuve de gauche.

En raison des symétries inhérentes au calcul des séquents, autoriser l'application de la règle $\forall_\mathcal{L}$ lors du calcul des nouvelles règles remet en cause également la complétude de LK_+ par rapport à LK.

Il est apparu que l'on peut raffiner ce résultat en restreignant l'usage de la règle $\forall_{\mathcal{L}}$ aux cas où il n'y a pas de $\forall$ en position négative (ou de $\exists$ en position positive) sous le connecteur auquel on souhaite l'appliquer (*cf* définition 4.8). Il en va symétriquement de même pour la règle $\exists_{\mathcal{R}}$ qui ne peut être appliquée que si la proposition concernée ne contient pas de $\forall$ en position positive. On remarque que ce phénomène rejoint les cas de non-permutabilité des règles de déduction dans la recherche de preuve automatique en logique classique. Cette similitude paraît naturelle si l'on voit le calcul des nouvelles règles de notre système comme une automatisation partielle de la preuve. Malheureusement, les définitions mathématiques dignes d'intérêt contenant des quantificateurs semblent pour la plupart présenter ce cas de figure (définition de la limite par exemple). Ce n'est pas étonnant puisque l'intelligence d'une preuve mathématique réside en grande partie dans la mise en évidence du bon représentant d'un $\exists$ pour "nourrir" un $\forall$ (fournir le bon ϵ lors d'une preuve en analyse par exemple).

Définition 4.8 (Polarité d'une sous-formule)**.** *La polarité d'une sous-formule est positive ou négative. On appelle $\neg$ la fonction qui inverse la polarité. La polarité d'une sous-formule est alors récursivement définie comme suit :*

Soit ϕ une sous-formule et p sa polarité.

- *Si $\phi = \phi_1 \vee \phi_2$ ou $\phi = \phi_1 \wedge \phi_2$, alors ϕ_1 et ϕ_2 sont de polarité p.*
- *Si $\phi = \forall x.\phi_1(x)$ ou $\phi = \exists x.\phi_1(x)$, alors ϕ_1 est de polarité p.*
- *Si $\phi = \phi_1 \Rightarrow \phi_2$, alors ϕ_1 est de polarité $\neg p$ et ϕ_2 est de polarité p.*

Intuitivement, la polarité d'une sous-formule est obtenue en descendant dans l'arbre syntaxique d'une formule depuis la racine et en inversant la polarité dans chaque branche gauche du signe $\Rightarrow$.

Solution Ramener le traitement des connecteurs $\forall$ et $\exists$ à un problème de permutabilité des règles de déduction nous amène à considérer les solutions ordinairement adoptées dans ce domaine. La technique du *focussing*, principalement employée en logique linéaire mais également dans d'autres systèmes [ACP01] paraît ici appropriée. On se propose d'introduire un nouveau symbole de prédicat pour chaque proposition présentant le connecteur de tête $\forall$ (resp. $\exists$) en position négative (resp. positive) sous un autre $\forall$ ou le connecteur de tête $\forall$ (resp. $\exists$) en position positive (resp. négative) sous un autre $\exists$. Ainsi, la sous-formule $\forall m.(m \in P \Rightarrow S(m) \in P)$ de la règle $\in_{\mathbb{N}}$ est remplacée par le prédicat $H(P)$ qui traduit le fait que P est héréditaire :

$$\in_{\mathbb{N}}\colon n \in \mathbb{N} \rightarrow \forall P.(0 \in P \Rightarrow H(P) \Rightarrow n \in P)$$

La règle $\in_{\mathbb{N}}$ peut à présent être décomposée entièrement pour donner lieu à la règle de déduction gauche suivante :

$$\in_{\mathbb{N}_{\mathcal{L}}} \frac{\Gamma \vdash_+ 0 \in P, \Delta \qquad \Gamma \vdash_+ H(P), \Delta \qquad \Gamma, n \in P \vdash_+ \Delta}{\Gamma, n \in \mathbb{N} \vdash_+ \Delta}$$

La règle traduit maintenant bien le raisonnement par induction. On introduit évidemment une nouvelle règle de réécriture qui réécrit $H(P)$ en sa définition pour assurer la complétude du système. L'exemple de la section 4.3.3 présente une preuve complète utilisant ces règles.

Il est intéressant de noter qu'en modifiant une règle de réécriture (càd. un axiome) pour traiter un problème de non permutabilité, on a fait jaillir une notion sur laquelle les mathématiciens ont déja mis un nom : l'hérédité d'une propriété. Cela n'est pas sans rappeller les objets mathématiques mis en évidence lors du processus de complétion d'un système de réécriture [Les86] traduisant une théorie axiomatique.

L'algorithme 1 formalise ce processus. Il transforme une règle de réécriture en un système de réécriture ne contenant plus de règles comportant
- un $\forall$ (resp. $\exists$) en position négative (resp. positive) sous un $\forall$;
- un $\forall$ (resp. $\exists$) en position positive (resp. négative) sous un $\exists$.

On suppose que l'on dispose d'une procédure *positions_mauvais_quantificateurs-moins_profond* qui prend une formule en argument et qui renvoie l'ensemble des positions des quantificateurs à problème les moins profonds dans le terme. Par exemple, pour la formule $\forall x.(\forall y.P(x,y) \Rightarrow Q(x))$, la procédure renvoie la position de la sous-formule $\forall y.P(x,y)$.

Algorithm 1 *prépare* (P, ϕ)

$positions \leftarrow positions\_mauvais\_quantificateurs\_moins\_profond(\phi)$
$res \leftarrow \emptyset$
for all $\omega \in positions$ **do**
  $\phi_1 \leftarrow \phi_{|\omega}$
  $\overline{x} \leftarrow \mathcal{FV}(\phi_1)$
  $Q \leftarrow nouveau\_symbole\_de\_prédicat$
  $\phi \leftarrow \phi[Q(\overline{x})]_\omega$
  $res \leftarrow res \cup prépare(Q(\overline{x}), \phi_1)$
end for
return $res \cup \{P \rightarrow \phi\}$

On peut donc proposer une nouvelle version de LK_+ prenant en compte le *focussing*, et qui utilise maintenant l'intégralité des règles logiques de LK pour décomposer les formules.

Définition 4.9 (Calcul des nouvelles règles de LK_+)**.** *Soit Calc un ensemble de règles composé du sous-ensemble des règles de LK formé par ax, $\bot_\mathcal{L}$, $\top_\mathcal{R}$, $\vee_\mathcal{L}$, $\vee_\mathcal{R}$, $\wedge_\mathcal{L}$, $\wedge_\mathcal{R}$, $\Rightarrow_\mathcal{L}$, $\Rightarrow_\mathcal{R}$, $\forall_\mathcal{L}$, $\forall_\mathcal{R}$, $\exists_\mathcal{L}$ et $\exists_\mathcal{R}$ ainsi que des règles $\top_\mathcal{L}$ et $\bot_\mathcal{R}$ de la figure 5. Soit une règle de réécriture $R : P \rightarrow \phi$ où P est atomique.*

1. *On applique la procédure* prépare *à P et ϕ pour récupérer l'ensemble des règles "nettoyées".*
2. *Pour chaque règle $R_i : P_i \rightarrow \phi_i$ obtenue :*
   (a) *Pour trouver la règle droite de R_i, on initialise la procédure par le séquent $\Gamma \vdash \phi_i, \Delta$. On y applique ensuite les règles de Calc jusqu'à ce qu'aucune règle ne soit plus applicable. On collecte alors l'ensemble des prémisses, les conditions d'application et la conclusion, et on remplace ϕ_i par P_i pour bien obtenir la règle droite $R_{i_\mathcal{R}}$.*
   (b) *Pour trouver la règle gauche de R_i, on initialise la procédure par le séquent $\Gamma, \phi_i \vdash \Delta$. On y applique les règles de Calc et on récupère la nouvelle règle gauche de même que pour le point précédent.*

Le théorème d'équivalence est alors encore valable modulo quelques arrangements dans sa démonstration : il faut à présent traiter le cas des connecteurs $\forall$ à gauche et $\exists$ à droite sachant que les propositions ne présentent plus de cas de non permutabilité. Ce résultat est la conséquence directe des remarques 2.1 et 2.2, et est détaillé en annexe B.2.

4.3.3 Exemples

Voyons maintenant un cas concret d'utilisation du système de déduction LK_+ en montrant que pour tout entier n, n est pair ou impair. Ce genre de preuve est très verbeux dans LK. Reprenons l'exemple de la définition des entiers via les prédicats inductifs :

$$\in_{\mathbb{N}}: n \in \mathbb{N} \ \rightarrow \ \forall P.(0 \in P \Rightarrow \forall m.(m \in P \Rightarrow S(m) \in P) \Rightarrow n \in P)$$

qui donne les règles de réécriture suivante une fois la procédure *prépare* passée :

$$\begin{array}{rcrcl} \in_{\mathbb{N}} & : & n \in \mathbb{N} & \rightarrow & \forall P.(0 \in P \Rightarrow H(P) \Rightarrow n \in P) \\ hered & : & H(P) & \rightarrow & \forall m.(m \in P \Rightarrow S(m) \in P) \end{array}$$

On obtient donc les nouvelles règles de déduction suivante dans LK pour la définition des entiers :

$$\in_{\mathbb{N}_{\mathcal{L}}} \frac{\Gamma \vdash_+ 0 \in P, \Delta \qquad \Gamma \vdash_+ H(P), \Delta \qquad \Gamma, n \in P \vdash_+ \Delta}{\Gamma, n \in \mathbb{N} \vdash_+ \Delta}$$

$$\in_{\mathbb{N}_{\mathcal{R}}} \frac{0 \in P, H(P) \vdash_+ n \in P, \Delta}{\Gamma \vdash_+ n \in \mathbb{N}, \Delta}$$

La régle gauche traduit bien le raisonnement par récurrence habituellement mené sur les prédicats inductifs. La régle de réécriture concernant le caractère héréditaire d'un prédicat est elle aussi transformée en nouvelles règles de déductions.

$$hered_{\mathcal{L}} \frac{\Gamma \vdash_+ m \in P, \Delta \qquad \Gamma, S(m) \in P \vdash_+ \Delta}{\Gamma, H(P) \vdash_+ \Delta}$$

$$hered_{\mathcal{R}} \frac{\Gamma, m \in P \vdash_+ S(m) \in P, \Delta}{\Gamma \vdash_+ H(P), \Delta}$$

La règle droite est particulièrement parlante, et transcrit bien la définition de l'hérédité. On définit ensuite les prédicats *pair* et *impair* par les trois règles de réécriture suivantes :

$$\begin{array}{rcrcl} zero & : & 0 \in Even & \rightarrow & \top \\ even & : & S(n) \in Even & \rightarrow & n \in Odd \\ odd & : & S(n) \in Odd & \rightarrow & n \in Even \end{array}$$

qui donnent lieu aux six règles de déduction suivantes :

$$zero_{\mathcal{L}}\ \frac{\Gamma \vdash_+ \Delta}{\Gamma, 0 \in Even \vdash_+ \Delta} \qquad zero_{\mathcal{R}}\ \frac{}{\Gamma \vdash_+ 0 \in Even, \Delta}$$

$$even_{\mathcal{L}}\ \frac{\Gamma, n \in Odd \vdash_+ \Delta}{\Gamma, S(n) \in Even \vdash_+ \Delta} \qquad even_{\mathcal{R}}\ \frac{\Gamma \vdash_+ n \in Odd, \Delta}{\Gamma \vdash_+ S(n) \in Even, \Delta}$$

$$odd_{\mathcal{L}}\ \frac{\Gamma, n \in Even \vdash_+ \Delta}{\Gamma, S(n) \in Odd \vdash_+ \Delta} \qquad odd_{\mathcal{R}}\ \frac{\Gamma \vdash_+ n \in Even, \Delta}{\Gamma \vdash_+ S(n) \in Odd, \Delta}$$

Enfin, notre encodage des prédicats sous forme de variables nous oblige à mettre un nom (une constante) sur la propriété *pair ou impair*. Nous l'appellons $Prop$ et rajoutons la définition suivante au système de réécriture :

$$prop\ :\ n \in Prop\ \rightarrow\ n \in Odd\ \vee n \in Even$$

Ce qui entraîne la création des deux règles suivantes dans LK_+ :

$$prop_{\mathcal{L}}\ \frac{\Gamma, n \in Odd \vdash_+ \Delta \qquad \Gamma, n \in Even \vdash_+ \Delta}{\Gamma n \in Prop \vdash_+ \Delta}$$

$$prop_{\mathcal{R}}\ \frac{\Gamma \vdash_+ n \in Odd, n \in Even}{\Gamma \vdash_+ n \in Prop, \Delta}$$

On peut finalement dériver la preuve de $n \in \mathbb{N} \vdash_+ n \in Odd\ \vee n \in Even$, c'est à dire que tout entier est pair ou impair. La figure 6 présente cette démonstration sous la forme d'un arbre de dérivation dans LK_+. Contrairement à une démonstration dans LK, cette dernière, bien qu'apparemment large, est lisible : on commence par montrer que zéro est pair ou impair (Π_1) et que la propriété *pair ou impair* est héréditaire (Π_2) en utilisant les règles de éduction issues des définitions de pair, impair et zéro. On en déduit que tout entier vérifie cette propriété en utilisant le principe de récurrence exprimé par la règle $\in_{\mathbb{N}_{\mathcal{L}}}$. La sous-preuve Π_3 est axiomatique à cause de l'utilisation "limitée" de la règle $\mathrm{b}\in_{\mathbb{N}_{\mathcal{L}}}$ ici et ne présente donc pas de sens particulier.

$$weak+prop_{\mathcal{R}}\ \dfrac{zero_{\mathcal{R}}\ \dfrac{}{\vdash_+ 0 \in Odd, 0 \in Even}}{0 \in \mathbb{N} \vdash_+ 0 \in Prop, n \in Odd, n \in Even}$$

Π_1

$$\dfrac{\begin{array}{c} weak+even_{\mathcal{R}}\ \dfrac{ax\ \dfrac{}{m \in Odd \vdash_+ m \in Odd}}{m \in Odd \vdash_+ S(m) \in Odd, S(m) \in Even} \\ prop_{\mathcal{L}} \end{array} \quad weak+odd_{\mathcal{R}}\ \dfrac{ax\ \dfrac{}{m \in Even \vdash_+ m \in Even}}{m \in Even \vdash_+ S(m) \in Odd, S(m) \in Even}}{\begin{array}{r} prop_{\mathcal{R}}\ \dfrac{m \in Prop \vdash_+ S(m) \in Odd, S(m) \in Even}{m \in Prop \vdash_+ S(m) \in Prop} \\ weak+hered_{\mathcal{R}}\ \dfrac{}{n \in \mathbb{N} \vdash_+ H(Prop), n \in Odd, n \in Even} \end{array}}$$

Π_2

$$\dfrac{\begin{array}{c} ax\ \dfrac{}{n \in Odd \vdash_+ n \in Odd, n \in Even} \\ weak+prop_{\mathcal{L}} \end{array} \quad ax\ \dfrac{}{n \in Even \vdash_+ n \in Odd, n \in Even}}{n \in \mathbb{N}, n \in Prop \vdash_+ n \in Odd, n \in Even}$$

Π_3

$$\in_{\mathbb{N}_{\mathcal{L}}}\ \dfrac{\begin{array}{ccc} \Pi_1 & \Pi_2 & \Pi_3 \\ 0 \in \mathbb{N} \vdash_+ 0 \in Prop, n \in Odd, n \in Even & n \in \mathbb{N} \vdash_+ H(Prop), n \in Odd, n \in Even & n \in \mathbb{N}, n \in Prop \vdash_+ n \in Odd, n \in Even \end{array}}{\vee_{\mathcal{R}}\ \dfrac{n \in \mathbb{N} \vdash_+ n \in Odd, n \in Even}{n \in \mathbb{N} \vdash_+ n \in Odd \ \vee\ n \in Even}}$$

FIG. 6 – Preuve de $n \in \mathbb{N} \vdash_+ n \in Odd \ \vee\ n \in Even$

Les définitions de la théorie des ensembles constituent un autre exemple élégant d'application de LK_+. La définition de la différence symétrique est traitée en annexe C à travers la présentation de l'assistant à la démonstration. Il reste beaucoup de théories déjà exprimées sous formes de systèmes de réécriture dans le cardre de la recherche sur la déduction modulo à tester dans LK_+, comme l'arithmétique [DW05] ou la théorie des ensembles de Zermelo [DM06].

4.4 Vers une version intuitionniste

Dans cette section, nous présentons les travaux en cours concernant la mise au point d'une version intuitionniste de LK_+. Nous pensons avoir trouvé le bon système mais la démonstration du lemme d'équivalence est inachevée.

On dispose donc d'une version statisfaisante de LK_+ qui permet de prouver les séquents prouvables de LK avec plus d'aisance et éventuellement dans la théorie vide. Toutefois, le système LK (et *a fortiori* LK_+) est intrinsèquement classique et ne permet pas la construction de démonstrations intuitionnistes qui permettetraient par exemple l'extraction de programmes (*cf* 4.2). On aimerait donc mettre au point un calcul des séquents extensible "LJ_+" calqué sur LK_+ et fondé sur le calcul des séquents intuitionnistes LJ.

On a vu en 4.3.2 que les cas de de non permutabilité des règles d'un calcul des séquents, mis en évidence par la recherche en démonstration automatique, tendent à remettent en cause la complétude des systèmes extensibles qui en sont dérivés. Cela explique pourquoi le connecteur $\vee$ posait problème dans NJ (*cf* section 4.1, problème des connecteurs). En effet, si l'on reformule la règle $R : P \to \forall x.(A(x) \vee B(x))$ par une définition admissible en logique classique seulement, $P \to \forall x.((A(x) \Rightarrow \bot) \Rightarrow B(x))$, on obtient la règle d'introduction suivante :

$$R_I \ \frac{\Gamma, A(x) \Rightarrow \bot \vdash_+ B(x)}{\Gamma \vdash_+ P}$$

Et on peut cette fois dériver le jugement $\forall x.((A(x) \Rightarrow \bot) \Rightarrow B(x)) \vdash_+ P$:

$$\dfrac{\dfrac{\dfrac{ax\ \dfrac{}{\Gamma \vdash_+ \forall x.((A(x) \Rightarrow \bot) \Rightarrow B(x))}}{\forall_E\ \dfrac{\Gamma \vdash_+ (A(x) \Rightarrow \bot) \Rightarrow B(x)}{}} \qquad ax\ \dfrac{}{\Gamma \vdash_+ A(x) \Rightarrow \bot}}{\Rightarrow_E\ \dfrac{}{\Gamma = (\forall x.((A(x) \Rightarrow \bot) \Rightarrow B(x)), A(x) \Rightarrow \bot) \vdash_+ B(x)}}}{R_I\ \dfrac{}{\forall x.((A(x) \Rightarrow \bot) \Rightarrow B(x)) \vdash_+ P}}$$

Il ne s'agit donc pas d'un problème inhérent à la déduction naturelle, mais à son caractère intuitionniste.

Malheureusement, le calcul LJ compte 11 cas de non-permutabilité. On pourrait donc décrire un système LJ_+ mais la décomposition des propositions lors du calcul des nouvelles règles serait très limité et on se ramènerait à LJ avec *folding* et *unfolding*. Cependant, les cas de non permutabilité des règles de LJ ne sont pas tous la conséquence directe de son caractère intuitionniste et sont seulement dûs à la restriction syntaxique de LJ qui impose des séquents mono-conclusion. Cette restriction peut être levée dans la plupart des cas et il existe des calculs de séquents tels que LB [WW99] qui capturent l'intuitionnisme dans un sous-ensemble restreint des règles de déduction. Les règles de LB sont celles de LK

$$\forall_{\mathcal{R}} \frac{\Gamma \vdash A}{\Gamma \vdash \forall x.A, \Delta}\ x \notin \mathcal{FV}(\Gamma) \qquad \Rightarrow_{\mathcal{R}} \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B, \Delta} \qquad \bot_{\mathcal{L}} \frac{}{\bot \vdash A}$$

FIG. 7 – Règles de LB différant de celles de LK

$$\frac{\forall_{\mathcal{L}}}{\forall_{\mathcal{R}}} \qquad \frac{\forall_{\mathcal{L}} \text{ ou } \forall_{\mathcal{R}}}{\exists_{\mathcal{L}}} \qquad \frac{\Rightarrow_{\mathcal{L}}}{\Rightarrow_{\mathcal{R}} \text{ ou } \forall_{\mathcal{R}}}$$

FIG. 8 – Cas de non permutabilité dans LB

mis à part les règles $\forall_{\mathcal{R}}$, $\Rightarrow_{\mathcal{R}}$ et $\bot_{\mathcal{L}}$, représentées figure 7, qui ne conservent pas les contextes Δ. Les cas de non-permutabilité sont alors réduits au nombre de 5, et sont représentés figure 8. Les cas de non-permutabilité se lisent de bas en haut pour une preuve construite de la racine vers les feuilles. Par exemple, une preuve où l'on a appliqué $\forall_{\mathcal{R}}$ puis $\forall_{\mathcal{L}}$ n'a pas de preuve équivalente dans laquelle on applique $\forall_{\mathcal{L}}$ puis $\forall_{\mathcal{R}}$.

On peut à nouveau utiliser le focussing pour traiter ces cas et ainsi obtenir un système extensible LB_+. Il suffit d'adapter la procédure *positions_mauvais-quantificateurs_moins_profond* utilisée dans la deuxième version de LK_+ pour qu'elle prenne en compte ces cas de non permutabilité. On remarque que trois des cinq cas, ceux qui concernent la permutabilité des quantificateurs, sont les mêmes que pour LK. Les nouvelles règles renvoyées par l'algorithme *prepare* doivent donc ne pas présenter les cas supplémentaire suivants :

- un $\Rightarrow$ en position négative sous un $\Rightarrow$
- un $\Rightarrow$ sous un $\forall$

Dès lors que l'on perçoit la déduction surnaturelle et ses dérivés comme une combinaison de *folding/unfolding* et d'une automatisation des sous-preuves, on peut conjecturer sans danger que LB_+ est complet et correct par rapport à LB. Cependant la démonstration reste à faire.

4.5 Termes de preuves pour LK_+

Le stage de master qui a mené au présent document a été mené en parallèlle de celui de Cément Houtmann qui travaille à la mise au point d'un langage de termes de preuve pour la déduction surnaturelle. Comme l'assistant de preuve présenté en dernière section est fondé sur LK_+ et non la déduction surnaturelle, nous avons été amené à travailler ensemble pour décider d'un langage de termes pour LK_+. Notre collaboration ne s'est pas arrêtée là : nous avons participé ensemble à toutes les réunions avec nos encadrants et nous nous consultons quotidiennement sur l'avancée de chacun afin de synchroniser nos efforts. Ce travail commun a donné lieu à un article en cours de soumission co-écrit avec Benjamin Wack [BHW].

Ces termes sont importants à deux titres. Tout d'abord, leur intégration dans l'assistant permettra, à l'instar de Coq, de vérifier la validité d'une démonstration par une simple opération de *type checking* du terme qui pourra

être limitée à un nombre restreint de lignes de code certifiées. Ensuite, les règles de réduction du calcul associé nous permettrons de montrer l'élimination des coupures pour une théorie donnée, et donc sa cohérence, plus facilement que par des arguments structurels sur les arbres de dérivation. Les travaux de Clement, qui se limitent pour le moment au cas propositionnel, se résument de la maniere suivante.

Décrivons d'abord les termes de preuve du fragment propositionnel de LK, introduits par Hugo Herbelin [CH00], avant de les étendre à LK. LK vu comme un système de typage s'organise comme comme suit : les termes de preuve sont divisés en deux sortes, les *termes* et les *contextes*. Les termes représentent les règles droites et les contextes les règles gauches. Ils sont définis par

$$\begin{array}{lrcl} \text{terms} & v & ::= & x \mid \mathit{true} \mid (v_1, v_2) \mid \mathit{proj}(\alpha, \beta, \langle v|e\rangle) \mid \lambda x.v \mid \mu\alpha\langle v|e\rangle \\ \text{contexts} & e & ::= & \alpha \mid \mathit{false} \mid \mathit{proj}(x, y, \langle v|e\rangle) \mid (e_1, e_2) \mid v \cdot e \mid \tilde{\mu}x\langle v|e\rangle \end{array}$$

Le système de typage est alors le suivant

$$ax_{\mathcal{L}}\ \frac{}{\Gamma; \alpha : A \vdash \alpha : A, \Delta} \qquad ax_{\mathcal{R}}\ \frac{}{\Gamma, x : A \vdash x : A; \Delta}$$

$$\bot_{\mathcal{L}}\ \frac{}{\Gamma; \mathit{false} : \bot \vdash \Delta} \qquad \top_{\mathcal{R}}\ \frac{}{\Gamma \vdash \mathit{true} : \top; \Delta}$$

$$\wedge_{\mathcal{L}}\ \frac{\Gamma, x : A, y : B \vdash v : C; \Delta \qquad \Gamma, x : A, y : B; e : C \vdash \Delta}{\Gamma; \mathit{proj}(x, y, \langle v|e\rangle) : A \wedge B \vdash \Delta}$$

$$\wedge_{\mathcal{R}}\ \frac{\Gamma \vdash v_1 : A; \Delta \qquad \Gamma \vdash v_2 : B; \Delta}{\Gamma \vdash (v_1, v_2) : A \wedge B; \Delta} \qquad \vee_{\mathcal{L}}\ \frac{\Gamma; e_1 : A \vdash \Delta \qquad \Gamma; e_2 : B \vdash \Delta}{\Gamma; (e_1, e_2) : A \vee B \vdash \Delta}$$

$$\vee_{\mathcal{R}}\ \frac{\Gamma \vdash v : C; \alpha : A, \beta : B, \Delta \qquad \Gamma; e : C \vdash \alpha : A, \beta : B, \Delta}{\Gamma \vdash \mathit{proj}(\alpha, \beta, \langle v|e\rangle) : A \vee B; \Delta}$$

$$\Rightarrow_{\mathcal{L}}\ \frac{\Gamma \vdash v : A; \Delta \qquad \Gamma; e : B \vdash \Delta}{\Gamma; v \cdot e : A \Rightarrow B \vdash \Delta} \qquad \Rightarrow_{\mathcal{R}}\ \frac{\Gamma, x : A \vdash v : B; \Delta}{\Gamma \vdash \lambda x.v : A \Rightarrow B; \Delta}$$

$$\mathit{cut}_{\mathcal{L}}\ \frac{\Gamma, x : A \vdash v : B; \Delta \qquad \Gamma, x : A; e : B \vdash \Delta}{\Gamma; \tilde{\mu}x\langle v|e\rangle : A \vdash \Delta}$$

$$\mathit{cut}_{\mathcal{R}}\ \frac{\Gamma \vdash v : B; \alpha : A, \Delta \qquad \Gamma; e : B \vdash \alpha : A, \Delta}{\Gamma \vdash \mu\alpha\langle v|e\rangle : A; \Delta}$$

Remarquons d'abord que les règles de typage $ax_{\mathcal{L}}$, $ax_{\mathcal{R}}$, $\bot_{\mathcal{L}}$, $\top_{\mathcal{R}}$, $\vee_{\mathcal{L}}$, $\wedge_{\mathcal{R}}$, $\Rightarrow_{\mathcal{L}}$, $\Rightarrow_{\mathcal{R}}$, $\mathit{cut}_{\mathcal{L}}$ et $\mathit{cut}_{\mathcal{R}}$ retranscrivent exactement les règles de LK. Les règles $\vee_{\mathcal{R}}$ et $\wedge_{\mathcal{L}}$ prèsentent un léger changement mais on peut toujours exprimer une dérivation

$$\wedge_{\mathcal{L}}\ \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \quad \text{par} \quad \begin{array}{l} \top_{\mathcal{R}}\ \dfrac{}{\Gamma, x : A, y : B \vdash \mathit{true} : \top; \Delta} \\ \wedge_{\mathcal{L}}\ \dfrac{\qquad\qquad \Gamma, x : A, y : B; e : \top \vdash \Delta}{\Gamma; \mathit{proj}(x, y, \langle \mathit{true}\,|v\rangle) : A \wedge B \vdash \Delta} \end{array}$$

Le cas de $\vee_{\mathcal{R}}$ est symétrique et on aurait également pu utiliser false placé à gauche du séquent à la place de true à droite. Finalement, les réductions des termes de preuve sont définies par les règles suivantes

$$\begin{array}{lll} & (*)_{\mathcal{R}} & \mu\alpha\langle v|e\rangle \mapsto \mu\alpha\langle v'|e'\rangle \\ \text{et} & (*)_{\mathcal{L}} & \tilde{\mu}x\langle v|e\rangle \mapsto \tilde{\mu}x\langle v'|e'\rangle \end{array}$$

où $(*)$ $\langle v|e\rangle \mapsto \langle v'|e'\rangle$ peut être

$$\begin{array}{lrcl}
(\vee) & \langle proj(\alpha,\beta,\langle v|e\rangle)|(e_1,e_2)\rangle & \mapsto & \langle v|e\rangle[e_1/\alpha, e_2/\alpha] \\
(\wedge) & \langle (v_1,v_2)|\, proj(x,y,\langle v|e\rangle)\rangle & \mapsto & \langle v|e\rangle[v_1/x, v_2/y] \\
(\Rightarrow) & \langle \lambda x.v|v'\cdot e\rangle & \mapsto & \langle v[v'/x]|e\rangle \\
(\mu) & \langle \mu\alpha\langle v|e\rangle|e'\rangle & \mapsto & \langle v|e\rangle[e'/\alpha] \\
(\tilde{\mu}) & \langle v'|\tilde{\mu}x\langle v|e\rangle\rangle & \mapsto & \langle v|e\rangle[v'/x]
\end{array}$$

Présentons maintenant l'intuition derrière la construction des termes pour les nouvelles règles de LK_+ à travers un exemple. On considère la règle R : $P \to A \vee B \Rightarrow C \vee D$. Les nouvelles règles de LK_+ dérivées de R sont alors les suivantes

$$R_{\mathcal{R}}\ \frac{\Gamma, A \vdash C, D, \Delta \qquad \Gamma, B \vdash C, D, \Delta}{\Gamma \vdash P, \Delta}$$

$$R_{\mathcal{L}}\ \frac{\Gamma, C \vdash \Delta \qquad \Gamma, D \vdash \Delta \qquad \Gamma \vdash A, B, \Delta}{\Gamma, P \vdash \Delta}$$

Puisque la mise au point de termes de preuve pour ces règles nous amènera à définir de nouvelles règles de réduction exprimant l'élimination des coupures, voyons de manière informelle cette élimination des coupures : considérons la dérivation

$$cut\ \frac{\Gamma \vdash P, \Delta \qquad \Gamma, P \vdash \Delta}{\Gamma \vdash \Delta}$$

On suppose que les dérivations de $\Gamma, P \vdash \Delta$ et $\Gamma \vdash P, \Delta$ sont respectivement construites avec les règles $R_{\mathcal{R}}$ et $R_{\mathcal{L}}$. On peut alors éliminer la coupure en branchant les preuves de $\Gamma, C \vdash \Delta$ et $\Gamma, D \vdash \Delta$ dans les preuves de $\Gamma, A \vdash C, D, \Delta$ et $\Gamma, B \vdash C, D, \Delta$. On obtient donc des preuves de $\Gamma, A \vdash \Delta$ et $\Gamma, B \vdash \Delta$ qui peuvent être branchées dans les preuves de $\Gamma \vdash A, B, \Delta$ pour finalement obtenir une preuve de $\Gamma \vdash \Delta$. Cela peut être transcrit formellement à travers les règles de typage suivantes :

$$R_{\mathcal{R}}\ \frac{\Gamma; e_A : A \vdash \alpha_C : C, \alpha_D : D, \Delta \qquad \Gamma; e_B : B \vdash \alpha_C : C, \alpha_D : D, \Delta}{\Gamma \vdash \delta(\alpha_C, \alpha_D).[e_A, e_B] : P; \Delta}$$

et

$$R_{\mathcal{L}}\ \frac{\begin{array}{cc}\Gamma; e_C : C \vdash \Delta & \Gamma \vdash v : M; \alpha_A : A, \alpha_B : B, \Delta \\ \Gamma; e_D : D \vdash \Delta & \Gamma; e : M \vdash \alpha_A : A, \alpha_B : B, \Delta\end{array}}{\Gamma; [e_C, e_D].\delta(\alpha_A, \alpha_B).\langle v|e\rangle : P \vdash \Delta}$$

et les règles de réduction

$$\begin{array}{lrl}
 & (*)_{\mathcal{R}} & \mu\alpha\langle v|e\rangle \mapsto \mu\alpha\langle v'|e'\rangle \\
\text{et} & (*)_{\mathcal{L}} & \tilde{\mu}x\langle v|e\rangle \mapsto \tilde{\mu}x\langle v'|e'\rangle
\end{array}$$

où $(*)$ $\langle v|e\rangle \mapsto \langle v'|e'\rangle$ est

$$\begin{array}{ll}
(R) & \langle \delta(\alpha_C, \alpha_D).[e_A, e_B]|[e_C, e_D].\delta(\alpha_A, \alpha_B).\langle v|e\rangle\rangle \\
 & \quad \mapsto \langle v|e\rangle[e_A/\alpha_A, e_B/\alpha_B][e_C/\alpha_C, e_D/\alpha_D]
\end{array}$$

Cet exemple exprime notre intuition de la manière dont les termes pour LK_+ devraient être construits. Comme les connecteurs $\Rightarrow$ et $\vee$ y sont traités et

comme le connecteur $\wedge$ est symétrique au connecteur $\vee$, on a une bonne vision des termes de preuve du fragment intuitionniste de LK_+.

L'avancée des termes de preuve du calcul entier était bloquée par l'asymétrie de la première version de LK_+ due à l'ommission des règles $\forall_{\mathcal{L}}$ et $\exists_{\mathcal{R}}$. Cela empêchait d'exprimer proprement l'élimination des coupures. La version de LK_+ avec focussing devrait régler ce problème.

4.6 Mise en œuvre d'un assistant à la preuve

Au fur et à mesure des avancées dans l'élaboration de LK_+, les divers aspects du système ont été mis en œuvre dans un assistant à la démonstration écrit pour l'occasion avec le langage TOM. À l'heure actuelle, le système proposé par l'assistant se situe à mi-chemin entre les deux versions de LK_+ exposées dans ce rapport : les propositions non-permutables ne sont pas décomposées mais le *focussing* n'est pas encore disponible. Nous présentons ici brièvement le langage TOM avant de nous arrêter sur quelques aspects intéressants des techniques de programmation utilisées dans l'assistant. Un *log* commenté d'une session interactive de l'assistant est fourni en annexe C.

4.6.1 Le langage TOM

TOM [MR06] est un langage de programmation développé au sein de l'équipe Protheo essentiellement par Pierre-Etienne Moreau et Antoines Reilles. Il propose l'ajout de fonctionnalités de réécriture par dessus les langages Java, C et OCaml. Parmi ses fonctionnalités les plus importantes on pourra relever les suivantes :

Filtrage associatif Le langage autorise la réécriture de termes modulo les opérateurs associatifs. Concrètement, cela implique que l'on peut réécrire des listes, ce qui n'est pas possible en OCaml par exemple. Ainsi, la règle

$$[X^*, a, b, Y^*] \longrightarrow [X^*, a, b, Y^*] \text{ si } a < b, \quad [X^*, b, a, Y^*] \; \textit{sinon}$$

appliquée jusqu'à obtention d'un point fixe réalise un tri de la liste.

Filtrage non linéaire L'apparition d'une même variable plusieurs fois dans le membre gauche d'une règle de réécriture est également autorisée. Ainsi, la règle de réécriture suivante est valide :

$$f(x, g(x)) \longrightarrow h(x)$$

Cela permet d'exprimer des contraintes comme nous le verrons dans la programmation des règles de déduction.

Programmation par stratégies La programmation par stratégies constitue un point fort du langage : il est possible de définir n'importe quelle stratégie de parcours d'un arbre (terme) par combinaison de stratégies de bases, comme en Stratego [Vis01]. Par exemple, la stratégie de parcours classique qui part de la racine vers les feuilles est définie récursivement ainsi :

$$\mathit{TopDown}(S) = \mathit{Sequence}(S, \mathit{All}(\mathit{TopDown}(S)))$$

où S est la stratégie passée en argument (une fonction java qui réécrit les x en y par exemple), *All* est la stratégie qui appelle son argument

sur tous les fils du nœud courant et *Sequence* est la stratégie qui appelle son premier argument sur le nœud courant, puis le deuxième si le premier n'échoue pas.

Partage maximal de la mémoire Les termes algébriques manipulés par le langage TOM sont construits de telle sorte que les sous-termes commun à plusieurs termes ne soient jamais dupliqés en mémoire.

Transparence du langage Tom s'intègre de façon transparente au langage hôte. Il offre en particulier la possibilité de mélanger l'utilisation de termes algébriques et de fonctions Java avec une syntaxe unifiée.

Au cours du stage, j'ai été amené à collaborer avec l'équipe TOM à de nombreuses reprises, autant pour participer au développement du langage que pour me tenir informé des dernières possibilités et exprimer mes besoins en terme de fonctionnalités.

4.6.2 Implantation

Le prototype tire pleinement parti de ces spécificités, la réécriture se prêtant bien à la manipulation d'arbres de preuve. Sans surprise, la majeure partie des objets est donc représentée par l'intermédiaire de termes algébriques : propositions, termes, séquents, arbres, etc. Voyons en détail quelques utilisations judicieuses du langage dans la manipulation de ces structures.

Application des règles Il existe deux types de règles dans l'assistant : les règles de LK qui ne changent jamais et les règles de déduction dérivées des règles de réécriture fournies par l'utilisateur. Dans le premier cas, les facilités de filtrage offertes par Tom permettent l'application d'une règle à la feuille d'un arbre de manière très concise :

```
public static SeqList
  applyImpliesR(Sequent seq, Prop active) throws Exception {
  %match(Sequent seq, Prop active) {
    sequent(ctxt,(X*,act@implies(p1,p2),Y*)), act -> {
      return `concSeq(sequent(context(ctxt*,p1),
                              context(X*,p2,Y*)));
    }
  }
  throw new Exception("can't apply rule => R");
}
```

Cette fonction prend un séquent et une formule active (celle qui a le focus) en argument et renvoie la liste des prémisses après application de la règle $\Rightarrow_{\mathcal{R}}$ du calcul des séquents LK. On tire ici parti du filtrage associatif, un séquent étant représenté comme deux listes de propositions, et du filtrage non-linéaire avec l'utilisation de la variable *act* qui impose une contrainte sur la proposition $p1 \Rightarrow p2$ sélectionnée dans la liste.

Renommage des variables libres La puissance de la programmation par stratégies est mise en évidence par l'exemple suivant. On souhaite remplacer toutes les occurences libres d'une variable par un terme quelconque. En l'absence de la programmation par stratégie, il faudrait écrire une fonction récursive

qui descendrait dans le terme et qui discriminerait les cas selon le symbole rencontré. La majeure partie de la fonction consisterait alors à "aiguiller" les appels récursifs.

En Tom, on définit un visiteur qui traite seulement les deux cas importants : l'échec de la visite si l'on tombe sur un quantificateur qui quantife la variable à remplacer et le remplacement de la variable par le terme.

```
%strategy ReplaceFreeVars
          (old_term: Term, new_term: Term) extends `Identity() {
  visit Prop {
    relationAppl(r, tl) -> {
     TermList res = replaceTerm(`tl, old_term, new_term);
      return `relationAppl(r, res);
    }

    r@forAll(x,_) -> {
      if  (`x != old_term.getname()) {
        return `r;
      }
      else
        throw new VisitFailure();
    }

    r@exists(x,_) -> {
      if  (`x != old_term.getname())
        return `r;
      else
        throw new VisitFailure();
    }
  }
}
```

Il ne reste plus qu'à composer les stratégies de base pour former une stratégie qui applique *ReplaceFreeVars* récursivement dans le terme et qui arrête de descendre dans les branches de l'arbre pour lesquelles la visite échoue :

$$\mu x.try(sequence(ReplaceFreeVars(a,b), all(x)))$$

μ est ici un opérateur qui permet de spécifier un appel de récursif de la stratégie se trouvant sous l'opérateur. *try* est une tratégie qui renvoie l'identité en cas d'échec de l'application de son argument. *all* et *sequence* sont décrites dans la section précédente.

Construction de l'arbre de preuve La construction récursive de l'arbre de preuve au cours de la session interactive est une approche peu réaliste : on veut pouvoir changer de sous-but courant en cours de preuve, afficher l'arbre de preuve en cours de construction, reporter à plus tard la fin d'une preuve sans perdre le travail effectué, traduire une partie de l'arbre en une pure preuve dans LK, etc. On pourrait donc construire l'arbre de manière itérative en maintenant une liste de références vers les feuilles des branches ouvertes. Un problème surviendrait alors : les termes algébriques manipulés par Tom étant des structures

non-mutables (comme dans tout langage fonctionnel), on obtiendrait un nouvel objet à chaque modification de l'arbre. Les références vers les feuilles des branches ouvertes seraient alors perdues.

Une solution est de conserver les *positions* de ces feuilles, c'est à dire un chemin depuis la racine. Tom offre un tel mécanisme et permet de manipuler les positions comme des objets abstraits : on peut obtenir le sous-terme correspondant à une position dans un terme donné, remplacer un sous-terme par un terme à une position donnée, etc. Ainsi, après avoir appliqué une règle de déduction à la feuille d'un arbre de preuve, on obtient les nouvelles feuilles ouvertes sans reparcourir tout l'arbre :

```
currentPos.getOmega(`TopDown(getOpenPositions(openGoals)))
```

getOmega renvoie la stratégie qui applique le visiteur passé en argument au nœud présent à la position *currentPos*. Concrètement cela se traduit par une descente en $O(h)$ dans un arbre de hauteur h puis de l'application de la stratégie $TopDown(getOpenPositions(openGoals))$.

5 Conclusion et perspectives

5.1 Conclusion

La question de la "surdéduction" est loin d'être réglée comme le montre la section 5.2. Cependant, ce rapport en pose les bases et démontre qu'il s'agit d'une alternative viable et élégante à l'utilisation de la déduction modulo seule dans un assistant de preuve. J'espère personnellement voir un jour un assistant aussi complet que Coq mêlant modulo et surdéduction. Cela semble en bonne voie notamment grâce au précieux concours de l'équipe LogiCal du LIX.

J'ai pris beaucoup de plaisir à effectuer ce stage et à cotoyer l'équipe Prothéo. J'y ai accompli un bond à la fois dans mon intuition de la logique, mais également dans mes capacités d'abstraction. Les trois groupes de travail sur la déduction modulo auxquels j'ai participé en tant que spectateur mais aussi en tant qu'acteur à travers deux présentations concernant un article sur les techniques de *folding/unfolding* et l'avancée de mes travaux m'ont été d'une aide précieuse dans la connaissance de l'état de l'art et des enjeux du domaine. J'ai eu la chance d'y rencontrer des figures du domaine comme Frank Pfenning, Gilles Dowek, Alexandre Miquel, Thérèse Hardin et bien d'autres. Je tiens particulièrement à faire part de l'admiration que m'inspire Gilles Dowek pour son sens de la pédagogie et la clarté de ses articles.

Je remercie en premier lieu Claude Kirchner pour son encadrement, le temps qu'il a su m'accorder malgré son emploi du temps, et pour sa capacité à communiquer sa vision très synthétique de l'orientation que prend ce domaine de la recherche. Je remercie bien évidemment Benjamin Wack pour son encadrement, ses explications toujours très pédagogiques, sa disponibilité et sa patience sans limites. Je ne peux que regretter son départ du monde de la recherche et je lui souhaite de mettre son sens de l'éducation à profit avec plaisir et succès dans sa future carrière de professeur. Merci également à Clément Houtmann qui a collaboré à mes travaux avec patience et humour, et à Colin pour ses discussions

très éclairantes et son intérêt communicatif pour tout ce qui touche de près ou de loin à la logique. Merci ensuite au relecteur de ce rapport pour la patience dont il a fait preuve avant de le recevoir.

Au cours de ce stage, j'ai cotoyé quotidiennement Pierre-Etienne Moreau dont la passion pour le compilateur Tom est contaminante, et Antoine Reilles qui n'hésite pas à passer deux nuits à ajouter au langage la fonctionnalité dont j'ai besoin. Je les remercie chaudement pour m'avoir accueilli dans l'équipe comme si j'y étais depuis toujours.

Merci enfin à Florent, Émilie et Germain, Radu, Guillaume et Anderson pour leur agréable compagnie, à mes camarades de l'Esial qui se sont orientés comme moi vers la recherche et aux autres élèves du master avec qui j'ai passé de bons moments.

5.2 Perspectives

On dispose maintenant d'un calcul des séquents classique extensible et d'un prototype d'assistant à la démonstration qui repose dessus. Notre objectif à long terme est de mettre au point, en collaboration avec l'équipe LogiCal, un nouvel assistant qui marie modulo, "surdéduction" et le savoir-faire accumulé sur Coq. Cependant, il reste beaucoup de travail à effectuer sur la déduction surnaturelle et ses homologues avant d'en maîtriser tous les aspects, tant sur le plan théorique que technique.

Ce stage s'arrête au moment où notre compréhension de LK_+ s'éclaircit considérablement, notemment grâce au parallèlle avec la démonstration automatique, Les objectifs à court terme concernant LK_+ et LB_+ sont donc nombreux et on devrait y voir d'importantes avancées dans les cemaines à venir :

Raffiner la version actuelle de $\mathbf{LK}_+$ En effet, nous traitons quatre cas de non-permutabilité concernant les quantificateurs alors que LK n'en présente que trois. On pourrait donc traiter de manière encore plus fine les quantificateurs dans le focussing.

Montrer la correction et la complétude de $\mathbf{LB}_+$ par rapport à LB La démonstration du théorème d'équivalence de LK_+ peut être reprise en grande partie mais il faut traiter plus finement les cas de $\forall$ et $\Rightarrow$.

Étudier l'élimination des coupures pour des classes de théories Les récents travaux de Benjamin Wack et Gilles Dowek sur la déduction naturelle les amènent à penser que les critères que l'on connaît sur les systèmes de réécriture en déduction modulo qui assurent l'élimination des coupures s'appliquent également en déduction surnaturelle. Il faudra cependant transposer ces résultats à LK_+.

Mettre au point un langage de termes de preuve complet L'ébauche de langage présentée en 4.5 est en cours d'étude dans le cadre du stage de master de Clément Houtmann. Ces termes de preuve devraient faciliter les démonstrations d'élimination des coupures.

Tester $\mathbf{LK}_+$ sur les théories étudiées en déduction modulo En particulier, il serait intéressant de regarder les systèmes de réécritures codant l'axiomatisation de l'arithmétique [DW05] et de la théorie des ensembles de Zermelo [DM06].

En ce qui concerne le prototype d'assistant à la démonstration, ces avancés devraient nous permettre de mettre à jour ses fonctionnalités. Nous comptons étudier les points suivants dans les semaines à venir :

Gestion de l'égalité par un système de réécriture sur les termes Pour un système de réécriture sur les termes, il est envisagé de proposer à la fois la normalisation d'un sous terme et la décision de l'égalité. En fait, cette fonctionnalité est déja programmée mais il faut trouver un moyen de choisir interactivement un sous-terme.

Gestion du focussing La version actuelle de l'assistant est située à mi-chemin entre la première et la seconde version de LK_+ : les formules comportant des quantificateurs sont décomposées dans les limites des cas de non-permutabilité, mais on ne transforme pas une règle de réécriture en un système de réécriture selon ces critères. Il s'agit donc juste ici de réorganiser le code pour rajouter cette fonctionnalité.

Termes de preuves Il est prévu de construire un terme de preuve en même temps que l'arbre de dérivation dès que le langage sera fixé. Le code a été pensé avec cette contrainte en vue.

Exportation vers d'autres assistants Ces termes de preuve nous permettraient d'exporter les preuves construites dans l'assistant vers d'autres prouveurs comme Fellowship [Kir] ou Coq, modulo la conversion vers une preuve classique décrite dans la preuve de correction en annexe B.1. Fellowship lit les $\bar{\lambda}\mu\tilde{\mu}$-termes et est capable de les convertir en termes de preuve lisibles par Coq ou en stratégies pour PVS. On n'aurait donc pas besoin d'encoder nos preuves dans un λ-terme (càd. traduire nos preuves en calcul des séquents classiques vers des preuves en déduction naturelle avec des règles classiques) dans un premier temps.

Enfin, les objectifs suivants sont envisageables à plus long terme, mais ne seront réalisables qu'après une série de travaux qui est loin d'être engagée. Cependant, cela donne une bonne idée de la direction vers laquelle on s'oriente avec la déduction surnaturelle.

Simulation de la déduction naturelle dans l'assistant Le calcul des séquents présente de multiples avantages pour l'élaboration d'un système déductif extensible. Cependant, les preuves en déduction (sur)naturelle sont plus proches du raisonnement mathématique classique pour un utilisateur peu familier avec le formalisme de LK. C'est pourquoi il serait possible de proposer un mode qui simule la déduction surnaturelle dans l'assistant par l'intermédiaire de coupures. Par exemple, la preuve

$$cut\ \dfrac{\Gamma \vdash A \Rightarrow B \qquad \Rightarrow_{\mathcal{L}} \dfrac{\Gamma \vdash A \qquad ax\ \dfrac{}{\Gamma, B \vdash B}}{\Gamma, A \Rightarrow B \vdash B}}{\Gamma \vdash B}$$

simule la règle

$$\Rightarrow_E \dfrac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Bien que l'on sache traduire une preuve dans NJ en une preuve dans LK, il reste à trouver un moyen de traduire une règle de déduction surnaturelle en une règle de LK_+.

Recherche de preuve automatique L'ajout de règles *ad hoc* pour la recherche automatique de preuves peut s'avérer fructueuse [Dol06]. La méthode des tableaux ou toute autre méthode de preuve automatique transposée à LK_+ pourrait accélérer la recherche de preuve dans une théorie donnée. Grâce à [Bon04], on sait déja que c'est possible en déduction modulo, et donc en particulier modulo une relation de réécriture sur les termes comme c'est le cas dans LK_+.

Dérivation automatique d'un sur-système L'intuition qui s'installe en fin de stage est que l'on peut dériver un système de type LK_+ à partir de n'importe calcul des séquents (LJ, LB, logique linéaire, ...) pour peu que l'on connaisse les cas de non permutabilité des règles. On peut imaginer un assistant qui prenne la descripition d'un calcul des séquents en entrée avec la liste des cas de non-permutabilité et qui permette directement de construire des preuves dans le sur-système associé.

Ce stage ouvre donc beaucoup de voies à explorer, tant du point de vue théorique que pratique. Les facilités offertes par la déduction surnaturelle et LK_+ pour la mise au point d'un assistant à la démonstration en font une piste privilégiée pour la réalisation d'un prouveur modulo.

Références

[ACP01] A. Abel, B.-Y. E. Chang et F. Pfenning, « Human-readable, machine-verifiable proofs for teaching constructive logic », dans *Proceedings of the Workshop on Proof Transformations, Proof Presentations and Complexity of Proofs (PTP'01)*, Siena, Italy, juin 2001.

[BHW] P. Brauner, C. Houtmann et B. Wack. « An extendible sequent calculus ». `http://www.loria.fr/~brauner/extendible.ps.gz`.

[Bon04] R. Bonichon, « Tamed : A tableau method for deduction modulo. », dans *IJCAR*, p. 445–459, 2004.

[CH00] P.-L. Curien et H. Herbelin, « The duality of computation », dans *Proceedings of ICFP '00*, coll. « SIGPLAN Notices 35(9) », p. 233–243. ACM, 2000.

[CLW03] H. Cirstea, L. Liquori et B. Wack, « Rewriting calculus with fixpoints : Untyped and first-order systems », dans *Proceedings of TYPES*, vol. 3085. Springer, 2003.

[Coq06] Projet LogiCal, INRIA Rocquencourt, France, *The Coq proof assistant*, 2006. `http://coq.inria.fr`.

[DHK03] G. Dowek, T. Hardin et C. Kirchner, « Theorem proving modulo, revised version ». Rapport de Recherche n° 4861, Institut National de Recherche en Informatique et en Automatique, July 2003. `http://www.inria.fr/rrrt/rr-4861.html`.

[DM06] G. Dowek et A. Miquel. « Cut elimination for zermelo's set theory ». `http://www.pps.jussieu.fr/~miquel/publis/zmod-long.ps.gz`, 2006.

[Dol06] D. Doligez. « Zenon : un prouveur extensible », 2006. `http://modulogic.inria.fr/exposes-2006-06-07/doligez.pdf`.

[DW98] G. Dowek et B. Werner, « Proof normalization modulo ». Rapport de Recherche n° 3542, Institut National de Recherche en Informatique et en Automatique, novembre 1998.

[DW05] G. Dowek et B. Werner, « Arithmetic as a theory modulo », dans J. Giesl, éditeur, *Proceedings of RTA'05*, vol. 3467 (coll. *Lecture Notes in Computer Science*), p. 423–437. Springer, 2005.

[GLT89] J.-Y. Girard, Y. Lafont et P. Taylor, *Proofs and Types*, vol. 7 (coll. *Cambridge Tracts in Theoretical Computer Science*). Cambridge University Press, 1989.

[Kir] F. Kirchner, *Fellowship manual.* `http://www.lix.polytechnique.fr/Labo/Florent.Kirchner/fellowship/doc/documentation.ps`.

[Kir06] F. Kirchner. « A finite first-order theory of classes ». 2006.

[Les86] P. Lescanne, « Reve a rewrite rule laboratory », dans J. Siekmann, éditeur, *Proceedings 8th International Conference on Automated Deduction, Oxford (UK)*, coll. « Lecture Notes in Computer Science », p. 696–697. Springer-Verlag, 1986.

[Let04] P. Letouzey, *Programmation fonctionnelle certifiée – L'extraction de programmes dans l'assistant Coq.* Thèse de doctorat, Université Paris-Sud, juillet 2004.

[MR06] P.-E. Moreau et A. Reilles. « The tom home page ». `http://tom.loria.fr`, 2006.

[ORS92] S. Owre, J. M. Rushby, et N. Shankar, « PVS : A prototype verification system », dans D. Kapur, éditeur, *11th International Conference on Automated Deduction (CADE)*, vol. 607 (coll. *Lecture Notes in Artificial Intelligence*), p. 748–752, Saratoga, NY, jun 1992. Springer-Verlag.

[Pau94] L. Paulson, *Isabelle : A Generic Theorem Prover*, vol. 828 (coll. *Lecture Notes in Computer Science*). Springer-Verlag, 1994.

[Pra65] D. Prawitz, *Natural Deduction, a Proof-theoretical Study.* 1965.

[SU98] M. H. Sørensen et P. Urzyczyn, « Lectures on the curry-howard isomorphism ». Available as DIKU Rapport 98/14, 1998.

[Vis01] E. Visser, « Stratego : A language for program transformation based on rewriting strategies. System description of Stratego 0.5 », dans A. Middeldorp, éditeur, *Rewriting Techniques and Applications (RTA'01)*, vol. 2051 (coll. *Lecture Notes in Computer Science*), p. 357–361. Springer-Verlag, May 2001.

[Wac05] B. Wack, *Typage et déduction dans le calcul de réécriture.* Thèse de doctorat, Université Henri Poincaré Nancy 1, 2005.

[WW99] A. Waaler et L. Wallen, « Tableaux for intuitionistic logics », dans M. D'Agostino, D. Gabbay, R. Hähnle et J. Posegga, éditeurs, *Handbook of Tableau Methods.* Kluwer, Dordrecht, 1999.

A Règles de NJ

Axiome

$$ax\ \frac{}{\Gamma, A \vdash A}$$

Règle structurelle

$$weak\ \frac{\Gamma \vdash A}{\Gamma, B \vdash A}$$

Règles d'élimination

$$\bot_E\ \frac{\Gamma \vdash \bot}{\Gamma \vdash \phi}$$

$$\wedge_{E_1}\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \wedge_{E_2}\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$\vee_E\ \frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma \vdash \Delta}$$

$$\Rightarrow_E\ \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\forall_E\ \frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[t/x]}$$

$$\exists_E\ \frac{\Gamma \vdash \exists x.A \qquad \Gamma, A \vdash C}{\Gamma \vdash C}\ x \notin \mathcal{FV}(\Gamma, C)$$

Règles d'introduction

$$\top_E\ \frac{}{\Gamma \vdash \top}$$

$$\wedge_I\ \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

$$\vee_{I_1}\ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \vee_{I_2}\ \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\Rightarrow_I\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

$$\forall_I\ \frac{\Gamma \vdash A}{\Gamma \vdash \forall x.A}\ x \notin \mathcal{FV}(\Gamma)$$

$$\exists_I\ \frac{\Gamma \vdash A[t/x], \Delta}{\Gamma \vdash \exists x.A, \Delta}$$

Pour les règles $\forall_E$ et $\exists_I$, la condition d'application $x \notin \mathcal{FV}(\Gamma, \Delta)$ signifie que la variable x ne doit être libre ni dans Γ, ni dans Δ. La notation $A[t/x]$, signifie que l'on substitue un terme t quelconque à toutes les occurences libres de x dans la proposition A.

B Preuves d'équivalence

On propose ici une notation plus concise que celle adoptée dans le rapport afin d'alléger la preuve. Ainsi l'arbre de dérivation suivant

$$\frac{\dfrac{\Gamma \vdash Q_1, R_1, \Delta \ldots \Gamma \vdash Q_1, R_m, \Delta}{\Gamma \vdash Q_1, \Delta} \quad \ldots \quad \dfrac{\Gamma \vdash Q_n, R_1, \Delta \ldots \Gamma \vdash Q_n, R_m, \Delta}{\Gamma \vdash Q_n, \Delta}}{\Gamma \vdash P, \Delta}$$

où les R_j sont indépendants des Q_i, càd sont les mêmes dans toutes les branches, sera abbrégé comme ceci :

$$\frac{\dfrac{\Gamma \vdash Q_i, R_j, \Delta}{\Gamma \vdash Q_i, \Delta}\ j = 1 \ldots m}{\Gamma \vdash P, \Delta}\ i = 1 \ldots n$$

De plus la notation R^* (resp. L^*) représente zéro ou plusieurs applications de règles droites (resp. gauches) du calcul des séquents LK.

B.1 LK_+ version 1

Theorème 2 (Equivalence)**.** *Pour tout système déductif* LK_+ *dérivé d'un système de réécriture R, il existe une théorie* $\mathcal{T}h$ *telle que* $\Gamma \vdash_+ \phi$ *si et seulement si* $\Gamma, \mathcal{T}h \vdash \phi$.

Pour construire la théorie $\mathcal{T}h$ on procède à nouveau comme en déduction modulo. S'il n'existe pas déjà un prédicat d'égalité $=$, on en ajoute un ainsi que les axiomes correspondants. Chaque règle de réécriture $l \to r$ se traduit alors en un axiome :

- $\forall \overline{x}.(l = r)$ si $l \to r \in \mathcal{R}_t$
- $\forall \overline{x}.(l \Leftrightarrow r)$ si $l \to r \in \mathcal{R}_P$

où $\overline{x}$ est l'ensemble des variables libres dans l et r.

Démonstration. Nous montrons d'abord que si $\Gamma \vdash_+ \phi$, alors $\Gamma, \mathcal{T}h \vdash \phi$ (correction), puis nous montrons la proposition réciprocque (complétude).

Si $\Gamma \vdash_+ \phi$ **alors** $\Gamma, \mathcal{T}h \vdash \phi$: Pour prouver $\Gamma, \mathcal{T}h \vdash \phi$, on peut reproduire les règles de LK et les axiomes de Γ utilisés dans la preuve de $\Gamma \vdash_+ \phi$. Il suffit donc de traduire les nouvelles règles de déduction en preuves dans LK pour obtenir la correction du système.

Considérons une "sur-règle" droite calculée à partir de la règle de réécriture $R : P \Leftrightarrow \phi$ en utilisant les règles de la procédure présentée en 4.6.

$$R_{\mathcal{R}} \; \frac{H_1 \quad \dots \quad H_n}{\Gamma \vdash_+ P, \Delta} \; \mathcal{C}$$

Par construction de $R_{\mathcal{R}}$, il existe une dérivation dans LK dont les prémisses sont $H_1 \dots H_n$, dont la conclusion contient ϕ et qui a les mêmes conditions d'application $\mathcal{C}$. En affaiblissant (règles *weak*) Γ en $\Gamma, \mathcal{T}h$ dans la conclusion de cette dernière dérivation, on peut construire la preuve de $\Gamma, \mathcal{T}h \vdash P$ suivante.

$$\mathcal{T}h = \mathcal{T}h' \cup \{\phi \Leftrightarrow P\} \; \dfrac{\wedge_{\mathcal{L}} \; \dfrac{\Rightarrow_{\mathcal{L}} \; \dfrac{\begin{array}{c} H_1 \quad \dots \quad H_n \\ \ddots \quad \iddots \end{array} \qquad \Gamma, \mathcal{T}h \vdash \phi, \Delta \qquad ax \; \dfrac{}{\Gamma, \mathcal{T}h, P \vdash P, \Delta}}{\Gamma, \mathcal{T}h, \phi \Rightarrow P \vdash P, \Delta}}{\Gamma, \mathcal{T}h, \phi \Leftrightarrow P \vdash P, \Delta}}{\Gamma, \mathcal{T}h \vdash P, \Delta}$$

Le cas gauche est très similaire au cas droit. Par construction de la règle gauche

$$R_{\mathcal{L}} \; \frac{H_1 \quad \dots \quad H_n}{\Gamma, P \vdash_+ \Delta} \; \mathcal{C}$$

il existe une dérivation dans LK qui a les mêmes prémisses et conditions d'application que $R_{\mathcal{L}}$ et dont la conclusion contient ϕ. Nous pouvons utiliser cette preuve dans la dérivation de $\Gamma, \mathcal{T}h, P \vdash \Delta$ suivante.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \mathcal{T}h, P \vdash P, \Delta}\, ax \qquad \dfrac{H_1 \quad \dots \quad H_n}{\Gamma, \mathcal{T}h, \phi \vdash \Delta}}{\Gamma, \mathcal{T}h, P \Rightarrow \phi, P \vdash \Delta}\, \Rightarrow_{\mathcal{L}}}{\Gamma, \mathcal{T}h, \phi \Leftrightarrow P, P \vdash \Delta}\, \wedge_{\mathcal{L}}}{\Gamma, \mathcal{T}h, P \vdash \Delta}\, \mathcal{T}h = \mathcal{T}h' \cup \{\phi \Leftrightarrow P\}$$

Si $\Gamma, \mathcal{T}h \vdash \phi$ **alors** $\Gamma \vdash_+ \phi$: Soit Π une preuve de $\Gamma, \mathcal{T}h \vdash_+ \phi$. Elle est obtenue trivialement en reproduisant les règles de LK utilisées dans la preuve de $\Gamma, \mathcal{T}h \vdash \phi$. Soient $\mathcal{A}x_1, \dots, \mathcal{A}x_n$ les axiomes contenus dans la théorie $\mathcal{T}h$ et utilisés dans la preuve de $\Gamma, \mathcal{T}h \vdash \phi$, nous pouvons dériver la preuve suivante de $\Gamma \vdash_+ \phi$.

$$\dfrac{\dfrac{\dfrac{\vdash_+ \mathcal{A}x_1}{\Gamma \vdash_+ \mathcal{A}x_1}\, weak \qquad \dfrac{\dfrac{\vdash_+ \mathcal{A}x_n}{\Gamma, \mathcal{A}x_1, \dots, \mathcal{A}x_{n-1} \vdash_+ \mathcal{A}x_n}\, weak \qquad \Pi}{\Gamma, \mathcal{A}x_1, \dots, \mathcal{A}x_n \vdash_+ \phi}\, cut}{\Gamma, \mathcal{A}x_1 \vdash_+ \phi}\, cut}{\Gamma \vdash_+ \phi}\, cut$$

En conséquence, il suffit de prouver les axiomes de $\mathcal{T}h$ dans LK pour obtenir une preuve de $\Gamma \vdash_+ \phi$. En d'autres termes, pour chaque axiome $P_i \Leftrightarrow \phi_i$ de $\mathcal{T}h$, nous devons prouver $P_i \vdash_+ \phi_i$ et $\phi_i \vdash_+ P_i$.

La preuve est réalisée par induction sur la structure des propositions ϕ_i.

Hypothèse 2.1 (Hypothèse d'induction)**.** *Pour chaque règle R de $\mathcal{R}$ définie par $P \to \phi$, il existe une preuve de $P \vdash_+ \phi$ et une preuve de $\phi \vdash_+ P$ dans LK_+ qui ont respectivement les formes suivantes*

$$\dfrac{\dfrac{Ax \quad \dots \quad Ax}{R_{\mathcal{L}}} \quad \dots \quad \dfrac{Ax \quad \dots \quad Ax}{R_{\mathcal{L}}}}{P \vdash_+ \phi}\, R^* \qquad et \qquad \dfrac{Ax \quad \dots \quad Ax}{\Gamma, \phi \vdash_+ P}\, R_{\mathcal{R}}$$

où $R_{\mathcal{R}}$ et $R_{\mathcal{L}}$ sont les règles de LK_+ dérivées de R et où R^* signifie "zéro ou plusieurs applications des règles droites de LK".

Considérons la règle $R : P \to \phi$. On raisonne par cas sur le connecteur de tête de ϕ.

ϕ **est atomique** : Ce cas est évident. On obtient les nouvelles règles de déduction suivantes :

$$R_{\mathcal{R}}\, \dfrac{\Gamma \vdash_+ \phi, \Delta}{\Gamma \vdash_+ P, \Delta} \qquad\qquad R_{\mathcal{L}}\, \dfrac{\Gamma, \phi \vdash_+ \Delta}{\Gamma, P \vdash_+ \Delta}$$

et on a immédiatement les preuves de $\phi \vdash_+ P$ et $P \vdash_+ \phi$.

$$R_{\mathcal{R}}\,\dfrac{ax\,\dfrac{}{\phi \vdash_+ \phi}}{\phi \vdash_+ P} \qquad\qquad R_{\mathcal{L}}\,\dfrac{ax\,\dfrac{}{\phi \vdash_+ \phi}}{P \vdash_+ \phi}$$

ϕ **est** $\bot$: Ce cas est également trivial. La preuve de $\phi \vdash_+ P$ est immédiate.

$$\bot_{\mathcal{L}}\,\frac{}{\bot \vdash_+ P}$$

Pour le cas gauche, on utilise la nouvelle règle $R_{\mathcal{L}}$ qui est une feuille pour construire $P \vdash_+ \phi$.

$$R_{\mathcal{L}}\,\frac{}{\Gamma, P \vdash_+ \Delta} \qquad \text{donc} \qquad R_{\mathcal{L}}\,\frac{}{P \vdash_+ \bot}$$

ϕ **est** $\top$: Ce cas est similaire au précédent. La preuve de $P \vdash_+ \phi$ est immédiate.

$$\top_{\mathcal{R}}\,\frac{}{P \vdash_+ \top}$$

Pour le cas droit, on utilise la nouvelle règle $R_{\mathcal{R}}$ qui est une feuille pour construire $\phi \vdash_+ P$.

$$R_{\mathcal{R}}\,\frac{}{\Gamma \vdash_+ P, \Delta} \qquad \text{donc} \qquad R_{\mathcal{R}}\,\frac{}{\top \vdash_+ P}$$

ϕ **est de la forme** $\forall x.\phi_1(x)$: On considère une règle de réécriture sur les propositions $R_1 : P_1(x) \to \phi_1(x)$ avec $P_1(x)$ atomique. Commençons par prouver $\phi \vdash_+ P$. Par construction de la règle $R_{\mathcal{R}}$

$$R^*, L^*\,\frac{\dfrac{\Gamma, \Gamma_i(x) \vdash \Delta_i(x), \Delta}{\Gamma \vdash \phi_1(x), \Delta}\,\mathcal{C},\ i = 1 \ldots n}{\forall_{\mathcal{R}}\,\dfrac{}{\Gamma \vdash \phi, \Delta}\,x \notin \mathcal{FV}(\Gamma, \Delta)}$$

la règle

$$R_{\mathcal{R}}\,\frac{\Gamma, \Gamma_i(x) \vdash \Delta_i(x), \Delta}{\Gamma \vdash_+ P, \Delta}\,\mathcal{C} \cup \{x \notin \mathcal{FV}(\Gamma, \Delta)\},\ i = 1 \ldots n$$

et la règle

$$R_{1_{\mathcal{R}}}\,\frac{\Gamma, \Gamma_i(x) \vdash \Delta_i(x), \Delta}{\Gamma \vdash_+ P_1(x), \Delta}\,\mathcal{C},\ i = 1 \ldots n$$

partagent les mêmes prémisses $\Gamma_i(x)$ et $\Delta_i(x)$ pour $i = 1 \ldots n$. $\mathcal{C}$ est ici un ensemble de conditions concernant les variables liées de ϕ_1.

Par hypothèse d'induction, il existe une preuve de $\phi_1(x) \vdash_+ P_1(x)$ dans LK_+ de la forme

$$\Pi_i(x)$$

$$R_{1_\mathcal{R}}\ \frac{\phi_i(x), \Gamma_i(x) \vdash \Delta_i(x)}{\phi_1(x) \vdash_+ P_1(x)}\ i = 1 \ldots n$$

Nous pouvons donc utiliser ces $\Pi_{1\ldots n}(x)$ pour construire la preuve suivante de $\phi \vdash_+ P$

$$\Pi_i(x)$$

$$\begin{array}{l} \forall_\mathcal{L} \\ R_\mathcal{R} \end{array} \frac{\begin{array}{c} \phi_1(x), \Gamma_i(x) \vdash_+ \Delta_i(x) \\ \hline \forall x.\phi_1(x), \Gamma_i(x) \vdash_+ \Delta_i(x) \end{array}}{\underbrace{\forall x.\phi_1(x)}_{\phi} \vdash_+ P}\ i = 1 \ldots n$$

Les conditions $\mathcal{C}$ ne sont pas violées lorsqu'on applique $R_\mathcal{R}$ puisqu'elles ne concernent que les variables liées de ϕ_1. D'autre part x est bien liée dans $\forall x.\phi_1(x)$.

Prouvons maintenant $P \vdash_+ \phi$. Ce cas est trivial car on ne décompose pas les formules présentant $\forall$ comme connecteur de tête dans la partie gauche des séquents lors du calcul des nouvelles règles. C'est pourquoi la nouvelle règle gauche est simplement

$$R_\mathcal{L}\ \frac{\Gamma, \forall x.\phi_1(x) \vdash_+ \Delta}{\Gamma, P \vdash_+ \Delta}$$

et on obtient immédiatement la preuve suivante de $P \vdash_+ \phi$

$$\begin{array}{l} ax \\ R_\mathcal{L} \end{array} \frac{\overline{\forall x.\phi_1(x) \vdash_+ \forall x.\phi_1(x)}}{P \vdash_+ \forall x.\phi_1(x)}$$

ϕ est de la forme $\exists x.\phi_1(x)$: On considère une règle de réécriture R_1 : $P_1(x) \rightarrow \phi_1(x)$ telle que $P_1(x)$ est une proposition atomique. Commençons par prouver $P \vdash_+ \phi$. Par construction de la règle $R_\mathcal{L}$

$$\begin{array}{l} R^*, L^* \\ \exists_\mathcal{L} \end{array} \frac{\begin{array}{c} \Gamma, \Gamma_i(x) \vdash \Delta_i(x), \Delta \\ \hline \Gamma, \phi_1(x) \vdash \Delta \end{array}}{\Gamma, \phi \vdash \Delta}\ \begin{array}{l} \mathcal{C},\ i = 1 \ldots n \\ x \notin \mathcal{FV}(\Gamma, \Delta) \end{array}$$

la règle

$$R_\mathcal{L}\ \frac{\Gamma, \Gamma_i(x) \vdash \Delta_i(x), \Delta}{\Gamma, P \vdash_+ \Delta}\ \mathcal{C} \cup \{x \notin \mathcal{FV}(\Gamma, \Delta)\},\ i = 1 \ldots n$$

et la règle

$$R_{1_\mathcal{L}}\ \frac{\Gamma, \Gamma_i(x) \vdash \Delta_i(x), \Delta}{\Gamma, P_1(x) \vdash_+ \Delta}\ \mathcal{C},\ i = 1 \ldots n$$

partagent les mêmes prémisses $\Gamma_i(x)$ et $\Delta_i(x)$ pour $i = 1 \dots n$. $\mathcal{C}$ est un ensemble de conditions concernant les variables liées de ϕ_1.
Par hypothèse d'induction, il existe une preuve $\phi_1(x) \vdash_+ P_1(x)$ dans LK_+ telle que

$$\Pi_{i,j}(x)$$

$$R_{1_{\mathcal{L}}} \dfrac{\dfrac{\Gamma_i(x), \Gamma_j(x) \vdash_+ \Delta_i(x), \Delta_j(x)}{P_1(x), \Gamma_j(x) \vdash_+ \Delta_j(x), \Delta}\, i = 1 \dots n}{P_1 \vdash_+ \phi_1(x)}\, j = 1 \dots m$$

On peut maintenant dériver la preuve de $\phi \vdash_+ P$ suivante

$$\Pi_{i,j}(x)$$

$$R^*, L^* \dfrac{\dfrac{\Gamma_i(x), \Gamma_j(x) \vdash_+ \Delta_i(x), \Delta_j(x)}{\Gamma_i(x) \vdash_+ \Delta_i(x), \phi_1(x)}\, j = 1 \dots m}{\exists_{\mathcal{R}} \dfrac{\Gamma_i(x) \vdash_+ \Delta_i(x), \exists x.\phi_1(x)}{P \vdash_+ \exists x.\phi_1(x)}\, i = 1 \dots n}$$

On ne viole aucune condition en appliquant $R_{\mathcal{L}}$ car x est liée partout et les autres conditions dans $\mathcal{C}$ ne concernent que les variables liées de ϕ_1.

Prouvons maintenant $\phi \vdash_+ P$. Comme pour le cas $\forall$, c'est trivial car on ne décompose pas les $\exists$ du côté droit des séquents lors du calcul des nouvelles règles. Ainsi, la nouvelle règle est la suivante.

$$R_{\mathcal{R}} \dfrac{\Gamma \vdash_+ \exists x.\phi_1(x), \Delta}{\Gamma \vdash_+ \phi, \Delta}$$

Et on obtient immédiatement la preuve de $\phi \vdash_+ P$.

$$R_{\mathcal{R}} \dfrac{ax \dfrac{}{\exists x.\phi_1(x) \vdash_+ \exists x.\phi_1(x)}}{\exists x.\phi_1(x) \vdash P}$$

ϕ **est de la forme** $\phi_1 \wedge \phi_2$: Considérons deux règles de réécriture R_1 : $P_1 \Leftrightarrow \phi_1$ et $R_2 : P_2 \Leftrightarrow \phi_2$ où P_1 et P_2 sont des propositions atomiques. Commençons par prouver $\phi \vdash_+ P$. Par construction de la règle $R_{\mathcal{R}}$

$$R^*, L^* \dfrac{\dfrac{\Gamma, \Gamma_i \vdash \Delta_i}{\Gamma \vdash \phi_1, \Delta}\, \mathcal{C}_1, i = 1 \dots n \qquad R^*, L^* \dfrac{\Gamma, \Gamma_j \vdash \Delta_j}{\Gamma \vdash \phi_2, \Delta}\, \mathcal{C}_2,\ j = 1 \dots m}{\wedge_{\mathcal{R}} \dfrac{}{\Gamma \vdash \phi, \Delta}}$$

la règle

$$R_{\mathcal{R}} \dfrac{\Gamma, \Gamma_i \vdash_+ \Delta_i, \Delta \qquad \Gamma, \Gamma_j \vdash_+ \Delta_j, \Delta}{\Gamma \vdash_+ P, \Delta}\, \mathcal{C}_1 \cup \mathcal{C}_2$$

la règle

$$R_{1_\mathcal{R}}\ \frac{\Gamma, \Gamma_i \vdash_+ \Delta_i, \Delta}{\Gamma \vdash_+ P_1, \Delta}\ \mathcal{C}_1$$

et la règle

$$R_{2_\mathcal{R}}\ \frac{\Gamma, \Gamma_j \vdash_+ \Delta_j, \Delta}{\Gamma \vdash_+ P_2, \Delta}\ \mathcal{C}_1$$

avec $i = 1 \ldots n$ et $j = 1 \ldots m$ partagent les mêmes Γ_i, Δ_i, Γ_j et Δ_j.

Par hypothèse d'induction, on obtient les preuves suivantes de $\phi_1 \vdash_+ P_1$ et $\phi_2 \vdash_+ P_2$ dans LK_+

$$\Pi_i \qquad\qquad \Pi_j$$

$$R_{1_\mathcal{R}}\ \frac{\phi_1, \Gamma_i \vdash_+ \Delta_i}{\phi_1 \vdash_+ P_1}\ i = 1 \ldots n \qquad R_{2_\mathcal{R}}\ \frac{\phi_2, \Gamma_j \vdash_+ \Delta_j}{\phi_2 \vdash_+ P_2}\ j = 1 \ldots m$$

On peut donc construire une preuve de $\phi \vdash_+ P$ comme suit.

$$\Pi_i \qquad\qquad \Pi_j$$

$$\begin{array}{c} \wedge_\mathcal{L}\ \dfrac{\phi_1, \phi_2, \Gamma_i \vdash_+ \Delta_i}{\phi_1 \wedge \phi_2, \Gamma_i \vdash_+ \Delta_i} \qquad \wedge_\mathcal{L}\ \dfrac{\phi_1, \phi_2, \Gamma_j \vdash_+ \Delta_j}{\phi_1 \wedge \phi_2, \Gamma_j \vdash_+ \Delta_j} \\ R_\mathcal{R}\ \dfrac{}{\phi_1 \wedge \phi_2 \vdash_+ P} \end{array}$$

avec $i = 1 \ldots n$ et $j = 1 \ldots m$.
La règle $R_\mathcal{R}$ peut être appliquée sans violer ses conditions d'application car $\mathcal{C}_1$ et $\mathcal{C}_2$ ne concernent que les variables liées de ϕ_1 et ϕ_2.

Prouvons maintenant $P \vdash_+ \phi$. Par construction de la règle $R_\mathcal{L}$:

$$\begin{array}{c} R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_1}, \Gamma_{i_2} \vdash \Delta, \Delta_{i_1}, \Delta_{i_2}}{\Gamma, \Gamma_{i_1}, \phi_2 \vdash \Delta, \Delta_{i_1}}\ i_2 = 1 \ldots m \\ R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_1}, \phi_2 \vdash \Delta, \Delta_{i_1}}{\Gamma, \phi_1, \phi_2 \vdash \Delta}\ i_1 = 1 \ldots n \\ \wedge_\mathcal{L}\ \dfrac{\Gamma, \phi_1, \phi_2 \vdash \Delta}{\Gamma, \phi \vdash \Delta} \end{array}$$

la règle

$$R_\mathcal{L}\ \frac{\Gamma, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_1}, \Delta_{i_2}}{\Gamma, P \vdash_+ \Delta}\ i_1 = 1 \ldots n,\ i_2 = 1 \ldots m$$

la règle

$$R_{1_\mathcal{L}}\ \frac{\Gamma, \Gamma_{i_1} \vdash_+ \Delta, \Delta_{i_1}}{\Gamma, P_1 \vdash_+ \Delta}\ i_1 = 1 \ldots n$$

et la règle

$$R_{2_\mathcal{L}}\ \frac{\Gamma, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_2}}{\Gamma, P_2 \vdash_+ \Delta}\ i_2 = 1 \ldots m$$

partagent les mêmes prémisses Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2} pour $i_1 = 1 \dots n$ et $i_2 = 1 \dots m$.

Par hypothèse d'induction, on obtient les preuves suivantes de $P_1 \vdash_+ \phi_1$ et $P_2 \vdash_+ \phi_2$

$$\Pi_{j_1,i_1}$$

$$R_{1_{\mathcal{L}}} \dfrac{\Gamma_{j_1}, \Gamma_{i_1} \vdash_+ \Delta_{j_1}, \Delta_{i_1}}{R^* \dfrac{\Gamma_{j_1}, P_1 \vdash_+ \Delta_{j_1}}{P_1 \vdash_+ \phi_1}\, j_1 = 1 \dots p}\, i_1 = 1 \dots n$$

$$\Pi_{j_2,i_2}$$

$$R_{2_{\mathcal{L}}} \dfrac{\Gamma_{j_2}, \Gamma_{i_2} \vdash_+ \Delta_{j_2}, \Delta_{i_2}}{R^* \dfrac{\Gamma_{j_2}, P_2 \vdash_+ \Delta_{j_2}}{P_2 \vdash_+ \phi_2}\, j_2 = 1 \dots q}\, i_2 = 1 \dots m$$

Finalement, on peut dériver la preuve de $P \vdash_+ \phi$ suivante,

$$\dfrac{R_{\mathcal{L}} \dfrac{R^* \dfrac{\Pi_{j_1,i_1}}{\Gamma_{j_1}, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta_{j_1}, \Delta_{i_1}, \Delta_{i_2}}{\Gamma_{j_1}, P \vdash_+ \Delta_{j_1}}}{P \vdash_+ \phi_1} \qquad R_{\mathcal{L}} \dfrac{R^* \dfrac{\Pi_{j_2,i_2}}{\Gamma_{j_2}, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta_{j_2}, \Delta_{i_2}, \Delta_{i_2}}{\Gamma_{j_2}, P \vdash_+ \Delta_{j_2}}}{P \vdash_+ \phi_2}}{P \vdash_+ \phi_1 \wedge \phi_2}$$

où $i_1 = 1 \dots n$, $i_2 = 1 \dots m$, $j_1 = 1 \dots p$ et $j_2 = 1 \dots q$. Comme on connaît déja Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2} par récurrence, on peut alpha-convertir les variables libres apparaissant dans Γ_{j_1}, Γ_{j_2}, Δ_{j_1} et Δ_{j_2} lors de l'application de bas en haut de R^* de telle manière qu'il n'y ait pas de collisions avec celles de Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2}. Ainsi, les conditions d'application de $R_{\mathcal{L}}$ qui concernent les variables libres de Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2} ne sont pas violées.

ϕ est de la forme $\phi_1 \vee \phi_2$: On considère deux règles de réécriture $R_1 : P_1 \to \phi_1$ et $R_2 : P_2 \to \phi_2$ telles que P_1 et P_2 sont atomiques. Montrons d'abord $\phi \vdash_+ P$. Par construction de la règle $R_{\mathcal{R}}$:

$$\vee_{\mathcal{R}} \dfrac{R^*, L^* \dfrac{R^*, L^* \dfrac{\Gamma, \Gamma_{i_1}, \Gamma_{i_2} \vdash \Delta, \Delta_{i_1}, \Delta_{i_2}}{\Gamma, \Gamma_{i_1} \vdash \phi_2, \Delta, \Delta_{i_1}}\, i_2 = 1 \dots m}{\Gamma \vdash \phi_1, \phi_2, \Delta}\, i_1 = 1 \dots n}{\Gamma \vdash \phi_1 \vee \phi_2, \Delta}$$

la règle

$$R_{\mathcal{R}} \dfrac{\Gamma, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_1}, \Delta_{i_2}}{\Gamma \vdash_+ P, \Delta}\, i_1 = 1 \dots n,\ i_2 = 1 \dots m$$

la règle

$$R_{1_{\mathcal{R}}}\ \frac{\Gamma, \Gamma_{i_1} \vdash_+ \Delta, \Delta_{i_1}}{\Gamma \vdash_+ P_1, \Delta}\ i_1 = 1 \ldots n$$

et la règle

$$R_{2_{\mathcal{R}}}\ \frac{\Gamma, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_2}}{\Gamma, \vdash_+ P_2, \Delta}\ i_2 = 1 \ldots m$$

partagent les mêmes Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2} pour $i_1 = 1 \ldots n$ et $i_2 = 1 \ldots m$.

Par hypothèse d'induction, on obtient les preuves suivantes de $\phi_1 \vdash_+ P_1$ et $\phi_2 \vdash_+ P_2$.

$$\Pi_{i1}$$

$$R_{1_{\mathcal{R}}}\ \frac{\phi_1, \Gamma_{i_1} \vdash_+ \Delta_{i_1}}{\phi_1 \vdash_+ P_1}\ i_1 = 1 \ldots n$$

$$\Pi_{i2}$$

$$R_{2_{\mathcal{R}}}\ \frac{\phi_2, \Gamma_{i_2} \vdash_+ \Delta_{i_2}}{\phi_2 \vdash_+ P_2}\ i_2 = 1 \ldots m$$

On peut donc dériver la preuve suivante de $\phi \vdash_+ P$

$$\vee_{\mathcal{L}}\ \frac{\begin{array}{cc}\Pi_{i_1} & \Pi_{i_2} \\ \phi_1, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta_{i_1}, \Delta_{i_2} & \phi_2, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta_{i_1}, \Delta_{i_2}\end{array}}{R_{\mathcal{R}}\ \dfrac{\phi_1 \vee \phi_2, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta_{i_1}, \Delta_{i_2}}{\phi_1 \vee \phi_2 \vdash_+ P}}$$

où $i_1 = 1 \ldots n$ et $i_2 = 1 \ldots m$. On ne viole aucune des conditions d'application collectées lors du calcul de la règle $R_{\mathcal{R}}$ car elles ne concernent que les variables liées de ϕ_1 et ϕ_2 par construction.

Prouvons $P \vdash_+ \phi$ à présent. Par construction de la règle $R_{\mathcal{L}}$

$$\vee_{\mathcal{L}}\ \frac{R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_1} \vdash \Delta, \Delta_{i_1}}{\Gamma, \phi_1 \vdash \Delta}\ i_1 = 1 \ldots n \qquad R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_2} \vdash \Delta, \Delta_{i_2}}{\Gamma, \phi_2 \vdash \Delta}\ i_2 = 1 \ldots m}{\Gamma, \phi_1 \vee \phi_2 \vdash \Delta}$$

la règle

$$R_{\mathcal{L}}\ \frac{\Gamma, \Gamma_{i_1} \vdash_+ \Delta, \Delta_{i_1} \qquad \Gamma, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_2}}{\Gamma, P \vdash_+ \Delta}\ i_1 = 1 \ldots n,\ i_2 = 1 \ldots m$$

la règle

$$R_{1_\mathcal{L}}\ \frac{\Gamma, \Gamma_{i_1} \vdash_+ \Delta, \Delta_{i_1}}{\Gamma, P_1 \vdash_+ \Delta}\ i_1 = 1 \ldots n$$

et la règle

$$R_{2_\mathcal{L}}\ \frac{\Gamma, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_2}}{\Gamma, P_2 \vdash_+ \Delta}\ i_2 = 1 \ldots m$$

partagent les mêmes prémisses Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2} pour $i_1 = 1 \ldots n$ et $i_2 = 1 \ldots m$.

Par hypothèse d'induction, on obtient les preuves suivantes de $P_1 \vdash_+ \phi_1$ et $P_2 \vdash_+ \phi_2$

$$\Pi_{j_1, i_1}$$

$$\begin{array}{c} R_{1_\mathcal{L}}\ \dfrac{\Gamma_{j_1}, \Gamma_{i_1} \vdash_+ \Delta_{j_1}, \Delta_{i_1}}{\Gamma_{j_1}, P_1 \vdash_+ \Delta_{j_1}}\ i_1 = 1 \ldots n \\ R^*\ \dfrac{\Gamma_{j_1}, P_1 \vdash_+ \Delta_{j_1}}{P_1 \vdash_+ \phi_1}\ j_1 = 1 \ldots p \end{array}$$

$$\Pi_{j_2, i_2}$$

$$\begin{array}{c} R_{2_\mathcal{L}}\ \dfrac{\Gamma_{j_2}, \Gamma_{i_2} \vdash_+ \Delta_{j_2}, \Delta_{i_2}}{\Gamma_{j_2}, P_2 \vdash_+ \Delta_{j_2}}\ i_2 = 1 \ldots m \\ R^*\ \dfrac{\Gamma_{j_2}, P_2 \vdash_+ \Delta_{j_2}}{P_2 \vdash_+ \phi_2}\ j_2 = 1 \ldots q \end{array}$$

On peut donc dériver la preuve suivante de $P \vdash_+ \phi$

$$\begin{array}{c} \Pi_{j_1, i_1} \qquad\qquad \Pi_{j_2, i_2} \\ R_\mathcal{L}\ \dfrac{\Gamma_{j_1}, \Gamma j_2, \Gamma_{i_1} \vdash_+ \Delta_{j_1}, \Delta_{j_2}, \Delta_{i_1} \qquad \Gamma_{j_1}, \Gamma j_2, \Gamma_{i_2} \vdash_+ \Delta_{j_1}, \Delta_{j_2}, \Delta_{i_2}}{\begin{array}{c} R^*\ \dfrac{\Gamma_{j_1}, \Gamma j_2, P \vdash_+ \Delta_{j_1}, \Delta_{j_2}}{\begin{array}{c} R^*\ \dfrac{\Gamma_{j_1}, P \vdash_+ \Delta_{j_1}, \phi_2}{\begin{array}{c} \vee_\mathcal{R}\ \dfrac{P \vdash_+ \phi_1, \phi_2}{P \vdash_+ \phi_1 \vee \phi_2} \end{array}} \end{array}} \end{array}} \end{array}$$

où $i_1 = 1 \ldots n$, $i_2 = 1 \ldots m$, $j_1 = 1 \ldots p$ et $j_2 = 1 \ldots q$. Pour les mêmes raisons que pour le cas $P \vdash_+ \phi_1 \wedge \phi_2$, on peut éviter la violation des conditions d'application de $R_\mathcal{L}$ en alpha-renommant les variables fraiches lors de l'application des règles $\forall$.

ϕ **est de la forme** $\phi_1 \Rightarrow \phi_2$: On considère deux règles de réécriture $R_1 : P_1 \to \phi_1$ et $R_2 : P_2 \to \phi_2$ telles que P_1 et P_2 sont atomiques. Montrons d'abord $\phi \vdash_+ P$. Par construction de la règle $R_\mathcal{R}$:

$$\begin{array}{c} R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_1}, \Gamma_{i_2} \vdash \Delta, \Delta_{i_1}, \Delta_{i_2}}{\Gamma, \Gamma_{i_1} \vdash \phi_2, \Delta, \Delta_{i_1}}\ i_2 = 1 \ldots m \\ R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_1} \vdash \phi_2, \Delta, \Delta_{i_1}}{\Gamma \phi_1 \vdash \phi_2, \Delta}\ i_1 = 1 \ldots n \\ \Rightarrow_\mathcal{R}\ \dfrac{\Gamma \phi_1 \vdash \phi_2, \Delta}{\Gamma \vdash \phi_1 \Rightarrow \phi_2, \Delta} \end{array}$$

la règle

$$R_{\mathcal{R}}\ \frac{\Gamma, \Gamma_{i_1}, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_1}, \Delta_{i_2}}{\Gamma \vdash_+ P, \Delta}\ i_1 = 1 \ldots n,\ i_2 = 1 \ldots m$$

la règle

$$R_{1_{\mathcal{L}}}\ \frac{\Gamma, \Gamma_{i_1} \vdash_+ \Delta, \Delta_{i_1}}{\Gamma, P_1 \vdash_+ \Delta}\ i_1 = 1 \ldots n$$

et la règle

$$R_{2_{\mathcal{R}}}\ \frac{\Gamma, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_2}}{\Gamma, \vdash_+ P_2, \Delta}\ i_2 = 1 \ldots m$$

partagent les mêmes Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2} pour $i_1 = 1 \ldots n$ et $i_2 = 1 \ldots m$.

Par hypothèse d'induction, on obtient les preuves suivantes de $\phi_1 \vdash_+ P_1$ et $\phi_2 \vdash_+ P_2$

$$\Pi_{j_1, i_1}$$

$$R_{1_{\mathcal{L}}}\ \frac{\Gamma_{j_1}, \Gamma_{i_1} \vdash_+ \Delta_{j_1}, \Delta_{i_1}}{R^*\ \dfrac{\Gamma_{j_1}, P_1 \vdash_+ \Delta_{j_1}}{P_1 \vdash_+ \phi_1}\ j_1 = 1 \ldots p}\ i_1 = 1 \ldots n$$

$$\Pi_{i2}$$

$$R_{2_{\mathcal{R}}}\ \frac{\phi_2, \Gamma_{i_2} \vdash_+ \Delta_{i_2}}{\phi_2 \vdash_+ P_2}\ i_2 = 1 \ldots m$$

On peut donc dériver la preuve suivante de $\phi \vdash_+ P$

$$\Rightarrow_{\mathcal{L}}\ \frac{\begin{array}{cc}\Pi_{i_2} & \Pi_{j_1, i_1} \\ \phi_2, \Gamma_{i_2}, \Gamma_{i_1} \vdash_+ \Delta_{i_2}, \Delta_{i_1} & R^*, L^*\ \dfrac{\Gamma_{i_2}, \Gamma_{i_1}, \Gamma_{j_1} \vdash_+ \Delta_{i_2}, \Delta_{i_1}, \Delta_{j_1}}{\Gamma_{i_2}, \Gamma_{i_1} \vdash_+ \phi_1, \Delta_{i_2}, \Delta_{i_1}}\end{array}}{R_{\mathcal{R}}\ \dfrac{\phi_1 \Rightarrow \phi_2, \Gamma_{i_2}, \Gamma_{i_1} \vdash_+ \Delta_{i_2}, \Delta_{i_1}}{\phi_1 \Rightarrow \phi_2 \vdash_+ P}}$$

où $i_1 = 1 \ldots n$, $i_2 = 1 \ldots m$ et $j_1 = 1 \ldots p$. Par construction de $R_{\mathcal{R}}$, les conditions d'application de cette règle ne concernent que les variables liées de ϕ_1 et ϕ_2, elle ne sont donc pas violées ici.

Prouvons maintenant $P \vdash_+ \phi$. Par construction de la règle $R_{\mathcal{L}}$

$$\vee_{\mathcal{L}}\ \frac{R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_1} \vdash \Delta, \Delta_{i_1}}{\Gamma \vdash \phi_1, \Delta}\ i_1 = 1 \ldots n \qquad R^*, L^*\ \dfrac{\Gamma, \Gamma_{i_2} \vdash \Delta, \Delta_{i_2}}{\Gamma, \phi_2 \vdash \Delta}\ i_2 = 1 \ldots m}{\Gamma, \phi_1 \Rightarrow \phi_2 \vdash \Delta}$$

la règle

$$R_{\mathcal{L}}\ \frac{\Gamma, \Gamma_{i_1} \vdash_+ \Delta, \Delta_{i_1} \qquad \Gamma, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_2}}{\Gamma, P \vdash_+ \Delta}\ i_1 = 1 \ldots n,\ i_2 = 1 \ldots m$$

la règle

$$R_{1_{\mathcal{R}}}\ \frac{\Gamma, \Gamma_{i_1} \vdash_+ \Delta, \Delta_{i_1}}{\Gamma \vdash_+ P_1, \Delta}\ i_1 = 1 \ldots n$$

et la règle

$$R_{2_{\mathcal{L}}}\ \frac{\Gamma, \Gamma_{i_2} \vdash_+ \Delta, \Delta_{i_2}}{\Gamma, P_2 \vdash_+ \Delta}\ i_2 = 1 \ldots m$$

partagent les mêmes prémisses Γ_{i_1}, Γ_{i_2}, Δ_{i_1} et Δ_{i_2} pour $i_1 = 1 \ldots n$ et $i_2 = 1 \ldots m$.

Par hypothèse d'induction, on obtient les preuves suivantes de $\phi_1 \vdash_+ P_1$ et $P_2 \vdash_+ \phi_2$

$$\Pi_{i_1}$$

$$R_{1_{\mathcal{R}}}\ \frac{\phi_1, \Gamma_{i_1} \vdash_+ \Delta_{i_1}}{\phi_1 \vdash_+ P_1}\ i_1 = 1 \ldots n$$

$$\Pi_{j_2, i_2}$$

$$\begin{array}{c} R_{2_{\mathcal{L}}}\ \dfrac{\Gamma_{j_2}, \Gamma_{i_2} \vdash_+ \Delta_{j_2}, \Delta_{i_2}}{\Gamma_{j_2}, P_2 \vdash_+ \Delta_{j_2}}\ i_2 = 1 \ldots m \\ R^*\ \dfrac{\Gamma_{j_2}, P_2 \vdash_+ \Delta_{j_2}}{P_2 \vdash_+ \phi_2}\ j_2 = 1 \ldots q \end{array}$$

Finalement, on obtient la preuve suivante de $P \vdash_+ \phi$

$$\Rightarrow_{\mathcal{R}}\ \frac{R_{\mathcal{L}}\ \dfrac{\begin{array}{c}\Pi_{i_1} \\ \phi_1, \Gamma_{i_2}, \Gamma i_1 \vdash_+ \phi_2, \Delta_{i_1}\end{array} \qquad R^*, L^*\ \dfrac{\begin{array}{c}\Pi_{j_2, i_2} \\ \phi_1, \Gamma_{i_2}, \Gamma j_2 \vdash_+ \Delta_{i_2}, \Delta_{j_2}\end{array}}{\phi_1, \Gamma_{i_2} \vdash_+ \phi_2, \Delta_{i_2}}}{P, \phi_1 \vdash_+ \phi_2}}{P \vdash_+ \phi_1 \Rightarrow \phi_2}$$

où $i_1 = 1 \ldots n$, $i_2 = 1 \ldots m$, $j_1 = 1 \ldots p$ et $j_2 = 1 \ldots q$. On ne viole pas les conditions d'application de $R_{\mathcal{L}}$ car elles concernent les variables liées de ϕ_1 et ϕ_2 par construction.

□

B.2 LK_+ version 2

La preuve d'équivalence est presque identique à la précédente, la seule différence avec la première version de LK_+ est que l'on autorise l'utilisation des règles $\forall_{\mathcal{L}}$ et $\exists_{\mathcal{R}}$ lors du calcul des nouvelles règles de déduction. Il faut donc traiter à nouveau les cas où ϕ est de la forme $\forall x.\phi_1(x)$ et $\exists x.\phi_1(x)$.

ϕ est de la forme $\forall x.\phi_1(x)$: On considère une règle de réécriture sur les propositions $R_1 : P_1(x) \rightarrow \phi_1(x)$ avec $P_1(x)$ atomique. La preuve de $\phi \vdash_+ P$ est identique à celle de la section B.1. Il reste à montrer $P \vdash_+ \phi$.

Remarque 2.1. *La règle $P \rightarrow \phi$ ayant été fournie par l'algorithme 1, la formule $\forall x.\phi_1(x)$ ne contient pas de $\forall$ en position négative ou de $\exists$ en position positive. Par conséquent, sa décomposition lors du calcul de $R_{\mathcal{L}}$ ne génère pas de conditions d'application.*

Par construction de la règle $R_{\mathcal{L}}$ ($\mathcal{SC} = \emptyset$ symbolise l'absence de conditions d'applications) :

$$R^*, L^* \; \forall_{\mathcal{L}} \; \frac{\dfrac{\Gamma, \Gamma_i(t) \vdash \Delta_i(t), \Delta}{\Gamma\phi_1(t) \vdash \Delta}\, \mathcal{SC} = \emptyset,\ i = 1 \ldots n}{\Gamma, \forall x.\phi_1(x) \vdash \Delta}$$

la règle

$$R_{\mathcal{L}} \; \frac{\Gamma, \Gamma_i(t) \vdash \Delta_i(t), \Delta}{\Gamma, P \vdash_+ \Delta}\, \mathcal{SC} = \emptyset,\ i = 1 \ldots n$$

et la règle

$$R_{1_{\mathcal{L}}} \; \frac{\Gamma, \Gamma_i(x) \vdash \Delta_i(x), \Delta}{\Gamma, P_1(x) \vdash_+ \Delta}\, \mathcal{SC} = \emptyset,\ i = 1 \ldots n$$

partagent les mêmes prémisses $\Gamma_i(t)$ et $\Delta_i(t)$ pour $i = 1 \ldots n$ au renommage $t \rightarrow x$ près.
Par hypothèse d'induction, il existe une preuve de $P_1(x) \vdash_+ \phi_1(x)$ dans LK_+ de la forme

$$\Pi_{i,j}(x)$$

$$R_{1_{\mathcal{L}}} \; \frac{\Gamma_j(x), \Gamma_i(x) \vdash_+ \Delta_j(x), \Delta_i(x)}{P_1(x), \Gamma_j(x) \vdash_+ \Delta_j(x)}\, i = 1 \ldots n$$
$$R^* \; \frac{P_1(x), \Gamma_j(x) \vdash_+ \Delta_j(x)}{P_1(x) \vdash_+ \phi_1(x)}\, j = 1 \ldots m$$

Nous pouvons donc utiliser ces $\Pi_{i,j}(x)$ pour construire la preuve suivante de $P \vdash_+ \phi$

$$\Pi_{i,j}(x)$$

$$R_{\mathcal{L}}\,\frac{\Gamma_j(x),\Gamma_i(x)\vdash_+\Delta_j(x),\Delta_i(x)}{R^*\,\dfrac{P,\Gamma_j(x)\vdash_+\Delta_j(x)}{\forall_{\mathcal{R}}\,\dfrac{P\vdash_+\phi_1(x)}{P\vdash_+\underbrace{\forall x.\phi_1(x)}_{\phi}}\,x \text{ est bien liée}}}\,t\leftarrow x$$

Aucune condition d'application n'est violée puisqu'il n'y en a pas.

ϕ **est de la forme** $\exists x.\phi_1(x)$: Ce cas est complètement symétrique au précédent. On considère une règle de réécriture sur les propositions $R_1 : P_1(x) \to \phi_1(x)$ avec $P_1(x)$ atomique. La preuve de $P \vdash_+ \phi$ est identique à celle de la section B.1. Il reste à montrer $\phi \vdash_+ P$.

Remarque 2.2. *La règle $P \to \phi$ ayant été fournie par l'algorithme 1, la formule $\exists x.\phi_1(x)$ ne contient pas de $\forall$ en position positive ou de $\exists$ en position négative. Par conséquent, sa décomposition lors du calcul de $R_{\mathcal{R}}$ ne génère pas de conditions d'application.*

Par construction de la règle $R_{\mathcal{R}}$:

$$R^*, L^*\,\frac{\Gamma,\Gamma_i(t)\vdash\Delta_i(t),\Delta}{\exists_{\mathcal{R}}\,\dfrac{\Gamma\vdash\phi_1(t),\Delta}{\Gamma\vdash\exists x.\phi_1(x)}}\,\mathcal{SC}=\emptyset,\ i=1\dots n$$

la règle

$$R_{\mathcal{L}}\,\frac{\Gamma,\Gamma_i(t)\vdash\Delta_i(t),\Delta}{\Gamma\vdash_+ P,\Delta}\,\mathcal{SC}=\emptyset,\ i=1\dots n$$

et la règle

$$R_{1_{\mathcal{L}}}\,\frac{\Gamma,\Gamma_i(x)\vdash\Delta_i(x),\Delta}{\Gamma\vdash_+ P_1(x),\Delta}\,\mathcal{SC}=\emptyset,\ i=1\dots n$$

partagent les mêmes prémisses $\Gamma_i(t)$ et $\Delta_i(t)$ pour $i = 1 \dots n$ au renommage $t \to x$ près.
Par hypothèse d'induction, il existe une preuve de $\phi_1(x) \vdash_+ P_1(x)$ dans LK_+ de la forme

$$\Pi_{i,j}(x)$$

$$R^*, L^*\,\frac{\Gamma_j(x),\Gamma_i(x)\vdash_+\Delta_j(x),\Delta_i(x)}{R_{1_{\mathcal{R}}}\,\dfrac{P_1(x),\Gamma_i(x)\vdash_+\Delta_i(x)}{\phi_1(x)\vdash_+ P_1(x)}\,i=1\dots n}\,j=1\dots m$$

Nous pouvons donc utiliser ces $\Pi_{i,j}(x)$ pour construire la preuve suivante de $\phi \vdash_+ P$

$$\Pi_{i,j}(x)$$

$$R^*, L^* \frac{\Gamma_j(x), \Gamma_i(x) \vdash_+ \Delta_j(x), \Delta_i(x)}{R_{\mathcal{R}} \dfrac{\phi_1(x), \Gamma_i(x) \vdash_+ \Delta_i(x)}{\exists_{\mathcal{L}} \dfrac{\phi_1(x) \vdash_+ P}{\exists x.\phi_1(x) \vdash_+ P}\, x \text{ est bien liée}}\, t \leftarrow x}$$

Aucune condition d'application n'est violée puisqu'il n'y en a pas.

C Session commentée d'utilisation de l'assistant

Nous présentons ici une session d'utilisation du prototype. Nous voulons prouver $x \in A \vartriangle B \vdash_+ x \in A \cap B \Rightarrow \bot$, où $\vartriangle$ est la différence symétrique définie par $\vartriangle_{def}$: $x \in A \vartriangle B \rightarrow ((x \in A \vee x \in B) \wedge ((x \in A \wedge x \in B) \Rightarrow \bot))$. Les nouvelles règles dans LK_+ sont les suivantes :

$$\vartriangle_{def_{\mathcal{R}}} \frac{\Gamma \vdash_+ x \in A, x \in B, \Delta \qquad \Gamma, x \in A, x \in B \vdash_+ \bot, \Delta}{\Gamma \vdash_+ x \in A \vartriangle B, \Delta}$$

$$\vartriangle_{def_{\mathcal{L}}} \frac{\Gamma, x \in A \vdash_+ x \in B, \Delta \qquad \Gamma, x \in B \vdash_+ x \in A, \Delta}{\Gamma, x \in A \vartriangle B \vdash_+ \Delta}$$

On calcule également les règles pour $\cap_{def}$, la définition de l'intersection :

$$\cap_{def_{\mathcal{R}}} \frac{\Gamma \vdash_+ x \in A, \Delta \qquad \Gamma \vdash_+ x \in B, \Delta}{\Gamma \vdash_+ x \in A \cap B, \Delta}$$

$$\cap_{def_{\mathcal{L}}} \frac{\Gamma, x \in A, x \in B \vdash_+ \Delta}{\Gamma, x \in A \cap B \vdash_+ \Delta}$$

En utilisant ces nouvelles règles, on peut maintenant dériver la preuve suivante dans LK_+ :

$$\dfrac{\cap_{def_{\mathcal{L}}} \dfrac{ax \dfrac{}{x \in A, x \in A, x \in B \vdash_+ x \in B, \bot}}{\Rightarrow_{\mathcal{R}} \dfrac{x \in A, x \in A \cap B \vdash_+ x \in B, \bot}{x \in A \vdash_+ x \in B, x \in A \cap B \Rightarrow \bot}} \qquad \cap_{def_{\mathcal{L}}} \dfrac{ax \dfrac{}{x \in B, x \in A, x \in B \vdash_+ x \in A, \bot}}{\Rightarrow_{\mathcal{R}} \dfrac{x \in B, x \in A \cap B \vdash_+ x \in A, \bot}{x \in B \vdash_+ x \in A, x \in A \cap B \Rightarrow \bot}}}{x \in A \vartriangle B \vdash_+ x \in A \cap B \Rightarrow \bot}$$

On peut remarquer que la preuve est plutôt "naturelle" comparée à ce qu'on aurait obtenu dans LK en raisonnant dans la théorie contenant les définition de $\vartriangle$ et $\cap$.

La session commentée suivante montre le calcul de ces nouvelles règles par l'assistant et la construction de la preuve pas à pas.

La première étape consiste à entrer les règles de réécriture. `lhs` attend la proposition atomique et `rhs` sa définition.

```
polux@betty:~/lemu/build$ java ProofBuilder
> rule
lhs. (atom) > In(x,symdiff(a,b)).
rhs. > (In(x,a) \/ In(x,b)) /\ ((In(x,a) /\ In(x,b)) => \B).
The new deduction rules are :

 |- In(x, a), In(x, b)    In(x, a), In(x, b) |- False
-----------------------------------------------------
 |- In(x, symdiff(a, b))

In(x, b) |- In(x, a)    In(x, a) |- In(x, b)
---------------------------------------------
In(x, symdiff(a, b)) |-
> rule
lhs. (atom) > In(x,inter(a,b)).
rhs. > In(x,a) /\ In(x,b).
The new deduction rules are :

 |- In(x, a)     |- In(x, b)
----------------------------
 |- In(x, inter(a, b))

In(x, a), In(x, b) |-
---------------------
In(x, inter(a, b)) |-
```

On peut alors commencer la démondtration en entrant le but que l'on veut prouver.

```
> proof
sequent. > In(x, symdiff(a(), b())) |- (In(x, inter(a(), b())) => False).
Open goals :
        In(x, symdiff(a(), b())) |- (In(x, inter(a(), b())) => False)

 h1:  In(x, symdiff(a(), b()))
-----------
*c1:  (In(x, inter(a(), b())) => False)
```

À tout moment, on peut mettre le focus sur une proposition du séquent pour la décomposer.

```
proof> focus h1
Open goals :
        In(x, symdiff(a(), b())) |- (In(x, inter(a(), b())) => False)

*h1:  In(x, symdiff(a(), b()))
-----------
 c1:  (In(x, inter(a(), b())) => False)
```

Le mot clé `rules ?` provoque l'affichage des nouvelles règles qui s'appliquent à la proposition qui a le focus dans le but courant.

```
proof> rules?

- rule 1 :

In(x, b) |- In(x, a)    In(x, a) |- In(x, b)
-----------------------------------------
In(x, symdiff(a, b)) |-

Open goals :
        In(x, symdiff(a(), b())) |- (In(x, inter(a(), b())) => False)

*h1:  In(x, symdiff(a(), b()))
-----------
 c1:  (In(x, inter(a(), b())) => False)


proof> rule 1
Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)
        In(x, a()) |- In(x, b()), (In(x, inter(a(), b())) => False)

*h1:  In(x, a())
-----------
 c1:  In(x, b())
 c2:  (In(x, inter(a(), b())) => False)


proof> focus c2
Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)
        In(x, a()) |- In(x, b()), (In(x, inter(a(), b())) => False)

 h1:  In(x, a())
-----------
 c1:  In(x, b())
*c2:  (In(x, inter(a(), b())) => False)


proof> elim
Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)
        In(x, a()), In(x, inter(a(), b())) |- In(x, b()), False

 h1:  In(x, a())
 h2:  In(x, inter(a(), b()))
-----------
*c1:  In(x, b())
 c2:  False


proof> focus h2
Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)
        In(x, a()), In(x, inter(a(), b())) |- In(x, b()), False
```

```
 h1:  In(x, a())
*h2:  In(x, inter(a(), b()))
-----------
 c1:  In(x, b())
 c2:  False


proof> rules?

- rule 3 :

In(x, a), In(x, b) |-
----------------------
In(x, inter(a, b)) |-

Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)
        In(x, a()), In(x, inter(a(), b())) |- In(x, b()), False

 h1:  In(x, a())
*h2:  In(x, inter(a(), b()))
-----------
 c1:  In(x, b())
 c2:  False


proof> rule 3
Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)
        In(x, a()), In(x, b()), In(x, a()) |- In(x, b()), False

 h1:  In(x, a())
*h2:  In(x, b())
 h3:  In(x, a())
-----------
 c1:  In(x, b())
 c2:  False
```

L'assistant peut également afficher l'arbre de preuve en construction au format latex à tout moment à l'aide de la commande display.

$$\dfrac{\dfrac{}{In(x, b()) \vdash In(x, a()), (In(x, inter(a(), b())) \Rightarrow False)}\ (open) \quad \dfrac{\dfrac{\dfrac{}{In(x, a()), In(x, b()), In(x, a()) \vdash In(x, b()), False}\ (open)}{In(x, a()), In(x, inter(a(), b())) \vdash In(x, b()), False}\ (rule\ 3)}{In(x, a()) \vdash In(x, b()), (In(x, inter(a(), b())) \Rightarrow False)}\ (\Rightarrow R)}{In(x, symdiff(a(), b())) \vdash (In(x, inter(a(), b())) \Rightarrow False)}\ (rule\ 1)$$

```
proof> axiom
Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)
```

```
 h1:  In(x, b())
-----------
*c1:  In(x, a())
 c2:  (In(x, inter(a(), b())) => False)


proof> focus c2
Open goals :
        In(x, b()) |- In(x, a()), (In(x, inter(a(), b())) => False)

 h1:  In(x, b())
-----------
 c1:  In(x, a())
*c2:  (In(x, inter(a(), b())) => False)


proof> elim
Open goals :
        In(x, b()), In(x, inter(a(), b())) |- In(x, a()), False

 h1:  In(x, b())
 h2:  In(x, inter(a(), b()))
-----------
*c1:  In(x, a())
 c2:  False


proof> focus h2
Open goals :
        In(x, b()), In(x, inter(a(), b())) |- In(x, a()), False

 h1:  In(x, b())
*h2:  In(x, inter(a(), b()))
-----------
 c1:  In(x, a())
 c2:  False


proof> rule 3
Open goals :
        In(x, a()), In(x, b()), In(x, b()) |- In(x, a()), False

 h1:  In(x, a())
*h2:  In(x, b())
 h3:  In(x, b())
-----------
 c1:  In(x, a())
 c2:  False


proof> axiom
Proof complete.
```

Table des matières