Prouvabilité intuitionniste en Logique Linéaire

Vincent Balat

Rapport de stage
du DEA *Sémantique, Preuves et Programmation*, effectué au Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA, Nancy)
sous la direction de M. Didier Galmiche

septembre 1998

Remerciements

Je tiens à remercier tout particulièrement mon directeur de stage, Didier Galmiche, pour l'intérêt qu'il a porté à mon travail, son aide précieuse ainsi que pour l'expérience qu'il m'a aidé à acquérir.

Je voudrais également exprimer ma reconnaissance à tous les membres du LORIA et invités qui m'ont aidé directement ou indirectement dans mon travail, et à ceux qui ont contribué à rendre mon séjour à Nancy agréable, en particulier à Dominique Bleusse, Iadine Chades, Joshua Hodas et Dominique Larchey.

Table des matières

Introduction **4**

1 La Logique Linéaire Intuitionniste **7**

1.1 La Logique Linéaire Classique . . . 7
1.2 **ILL** . . . 9
1.3 **FILL** . . . 11

2 Prouvabilités classique et intuitionniste **14**

2.1 Logique classique . . . 15
2.1.1 Preuves annotées . . . 15
2.1.2 Exemples . . . 17
2.1.3 Extraction de la preuve intuitionniste . . . 17
2.2 Logique linéaire . . . 17
2.2.1 Un calcul des séquents annoté pour **FILL** . . . 18
2.2.2 Simplification du critère . . . 20
2.2.3 Preuve du théorème . . . 22
2.2.4 Exemples . . . 23
2.2.5 Prouvabilité intuitionniste . . . 24

3 Prouvabilité et réseaux de preuve **25**

3.1 Définitions . . . 25
3.2 Orientation des réseaux . . . 26
3.2.1 Définition des réseaux orientés . . . 26
3.2.2 Des réseaux pour **ILL** . . . 28
3.3 Des réseaux pour **FILL** . . . 28
3.3.1 Définition du critère . . . 28
3.4 Algorithme de construction de réseaux pour **FILL** . . . 28
3.4.1 L'algorithme pour **MLL** . . . 29
3.4.2 Extension à **FILL** . . . 31
3.4.3 Exemple . . . 32
3.4.4 Correction de l'algorithme . . . 33

3.5 Relation avec le calcul des séquents 34
3.6 Extension avec la règle **MIX** 34

Conclusion **35**

A Différents systèmes de calcul des séquents **38**
A.1 Calcul des séquents pour **CL** 38
A.2 Calcul des séquents mono-conclusion pour **IL** 39
A.3 Calcul des séquents multi-conclusion pour **IL** 39

Introduction

La logique linéaire, introduite par J.Y. Girard [21, 22], est une logique puissante et expressive dont les applications en informatique sont nombreuses [2]. Elle permet par exemple de concevoir et analyser de manière très précise des programmes, selon les paradigmes *proofs-as-programs* [26] et *proof-search-as-computation* [27]. Le premier de ces paradigmes utilise l'isomorphisme de Curry-Howard ; les types représentent des formules et les programmes leurs preuves. Le processus de normalisation correspond au calcul. Le deuxième utilise la recherche de preuve comme méthode de calcul et est utilisé en programmation logique, notamment concurrente. L'utilisation de différents fragments de la logique linéaire permet dans les deux cas d'exprimer très finement les propriétés des programmes par exemple pour spécifier des processus concurrents [1, 6]. La logique linéaire a également eu une influence sur l'étude de la logique classique en donnant une définition plus précise de ses connecteurs et en fournissant par exemple une nouvelle approche de la recherche de preuves grâce aux réseaux [14]. Il est donc important de développer des méthodes de construction et de vérification de preuves pour les fragments de la logique linéaire [20]. Nous allons nous intéresser ici aux preuves en logique linéaire intuitionniste.

Depuis l'introduction de la logique linéaire, de nombreux travaux essayent d'en déterminer les propriétés et les utilisations possibles. Un angle d'attaque consiste à essayer de voir si les résultats connus et importants de la logique (comme l'élimination des coupures, les méthodes de preuves,...) ont un pendant intéressant et utile en logique linéaire. La logique linéaire est un raffinement de la logique classique (**CL**) avec une gestion rigoureuse des formules en tant que ressources. Elle a d'abord été présentée comme une alternative à la logique intuitionniste (**IL**) pour fournir une autre logique constructive, avec des applications à la concurrence. Cependant ces logiques ont toutes deux des propriétés propres intéressantes pour certaines applications, d'où l'idée de créer une logique linéaire intuitionniste.

Au sein de la logique linéaire, des travaux ont mené à une distinction entre logique linéaire classique (**CLL**) et logique linéaire intuitionniste (**ILL**). Même lorsqu'une logique est fixée, le choix des systèmes formels peut dépendre des objectifs visés, par exemple la recherche de preuves. En logique linéaire, les systèmes utilisés sont le calcul des séquents linéaire, mais également les réseaux de preuve, notion nouvelle qui tient compte du parallélisme intrinsèque du calcul des séquents [23]. Dans le cadre du calcul des séquents, le choix du fragment intuitionniste de la logique linéaire n'est pas évident [30]. Deux systèmes ont en fait été proposés : **ILL** et **FILL**, correspondant à des idées différentes de l'intuitionnisme. Mais leur sémantique et leur utilisation ne sont pas aussi naturelles que celles de **IL**, en particulier parce que leurs propriétés sont encore mal connues ; d'où l'intérêt de proposer des outils permettant de travailler avec ces systèmes logiques. L'étude de la logique linéaire intuitionniste permet en outre de réutiliser et étendre les nombreux résultats et méthodes connus pour la logique intuitionniste.

Le travail présenté ici propose une étude de certains liens entre la logique linéaire classique et la logique linéaire intuitionniste, en particulier du point de vue de la construction automatique de preuves. Nous nous limiterons aux fragments multiplicatifs de ces logiques. Tout comme les méthodes de la logique classique et intuitionniste ont été adaptées pour la logique linéaire, les techniques de recherche de preuve de **IL** sont parfois tirées de celles de **CL**. La prouvabilité intuitionniste est alors vue comme une perturbation de la prouvabilité classique. Dans cette optique, le principe retenu ici consiste à aborder la prouvabilité intuitionniste par rapport à la prouvabilité classique [29]. Le problème de base consiste à rechercher l'aspect intuitionniste des preuves classiques, ce qui permet d'affirmer que certaines preuves classiques contiennent ou non une ou des preuves intuitionnistes. Une première idée pourrait consister à utiliser un λ-calcul adapté pour coder les preuves et les analyser. Ritter, Pym et Wallen [29] ont par exemple utilisé les travaux récents sur le codage des preuves classiques grâce au $\lambda\mu$-calcul de Parigot [28] pour détecter les preuves intuitionnistes dans **CL**. Une approche similaire pour la logique linéaire imposerait d'utiliser un λ-calcul linéaire [10], et conduirait à une analyse très compliquée par rapport à l'apport réel de la méthode. Nous avons donc adopté une autre approche utilisant des systèmes de preuves annotés. Avant d'étudier le cas de la logique linéaire, nous proposons tout d'abord une alternative aux travaux de [29] pour **IL** et **CL** grâce à cette approche : elle consiste à associer à chacune des formules de la preuve classique en calcul des séquents un label pour prendre en compte les différences entre le calcul des séquents multi-conclusion pour **CL** et une version multi-conclusion du calcul des séquents **IL**. Nous définissons donc un calcul des séquents annoté, puis un critère sur les étiquettes (ou labels) des formules des séquents initiaux (axiomes) permettant de détecter si la preuve est intuitionniste. Nous aborderons le problème de la complétude de cette méthode, ainsi que celui de l'extraction de la preuve intuitionniste.

On souhaite adapter cette méthode à la logique linéaire. Le choix du système **FILL**paraît le plus adéquat pour cela. En effet, **FILL** est un calcul multi-conclusion, qui peut être défini à partir de **CLL**, en limitant juste l'application de la règle d'implication à droite grâce à une notion de dépendance entre formules [11]. Nous définissons un calcul des séquents annotés pour **CLL**, adapté du précédent, de façon à prendre en compte le concept de dépendances grâce aux étiquettes, ce qui permet de donner une nouvelle définition de **FILL** et d'avoir une caractérisation des preuves **FILL** dans **CLL**. Un raffinement par un système plus éloigné de celui de **CL** permet de simplifier ces résultats. L'emploi de labels présente un intérêt du point de vue algorithmique par rapport à une approche basée sur un λ-calcul, notamment pour la recherche de preuve. Mais il ne permet pas de résoudre le problème de complétude, puisque dans certains cas, la preuve intuitionniste existe, mais ne peut pas être détectée de cette façon.

Une approche utilisant la notion de réseaux de preuves va nous permettre de résoudre ce problème. Après avoir étudié la définition des réseaux de preuve pour **FILL** [5], nous en proposons un algorithme de construction, reposant sur les principes de ceux développés dans [19] pour la logique linéaire multiplicative (**MLL**) et la logique linéaire multiplicative non-commutative. À partir d'un séquent, l'algorithme construit un réseau de preuve **FILL** (s'il existe) en partant du haut et donc la preuve **FILL**. Alors que la première approche consiste à construire une preuve du but vers les axiomes et analyser ensuite ces axiomes étiquetés, nous partons ici des axiomes pour construire directement un réseau **FILL**, s'il en existe un. L'étude de ces questions en utilisant successivement le calcul des séquents et les réseaux de preuve permet de bien comprendre ce que peut apporter chacune des deux approches et de voir les rapports entre les deux. L'approche utilisant les réseaux présente au moins l'avantage par rapport à l'autre d'être complète.

Dans le chapitre 1, nous présenterons les différents systèmes logiques étudiés, et nous donnerons quelques résultats généraux permettant de les positionner les uns par rapport aux autres. Le chapitre 2 propose le calcul des séquents annoté pour la logique classique puis montre l'application du même principe pour la logique linéaire, qui aboutit à une nouvelle présentation de **FILL**. Le chapitre 3 présente une extension de l'algorithme de construction des réseaux de preuves de Didier Galmiche pour la logique linéaire intuitionniste, et montre que l'on peut en déduire un algorithme de construction de preuves **FILL** en calcul des séquents. Nous terminons en présentant les conclusions et extensions possibles de ces travaux.

Notations :
Dans toute la suite :

- A, B, C,... représentent des formules quelconques,
- $\Gamma, \Gamma', \Delta, \Delta'$ représentent des ensembles, multi-ensembles ou des listes de formules suivant le contexte.

Chapitre 1

La Logique Linéaire Intuitionniste

1.1 La Logique Linéaire Classique

La logique linéaire permet de donner à la logique une gestion plus rigoureuse des ressources en interdisant la duplication et l'élimination d'hypothèses et de conclusions [22]. La présentation habituelle de la logique linéaire utilise le calcul des séquents classique, duquel on a retiré les règles de contraction et d'affaiblissement. Cette modification des règles a pour effet de faire disparaître l'équivalence entre les présentations additives et multiplicatives des connecteurs $\vee$ et $\wedge$, d'où la nécessité de distinguer un $\vee$ multiplicatif (« par », noté $⅋$), un $\vee$ additif (« avec », noté &), un $\wedge$ multiplicatif (« tenseur », noté $\otimes$), et un $\wedge$ additif (« plus », noté $\oplus$). Les atomes sont de la forme A ou $A^\perp$. La définition de la négation $^\perp$ est étendue à toutes les formules par les règles suivantes :

$$(A ⅋ B)^\perp = (A^\perp \otimes B^\perp) \qquad (A \otimes B)^\perp = (A^\perp ⅋ B^\perp)$$

$$(A \oplus B)^\perp = (A^\perp \& B^\perp) \qquad (A \& B)^\perp = (A^\perp \oplus B^\perp)$$

Les symboles $\mathbf{1}$, $\mathbf{0}$, $\top$ et $\perp$ représentent respectivement les éléments neutres des connecteurs $\otimes$, $\oplus$, & et $⅋$. L'implication linéaire ($\multimap$) peut être définie par $A \multimap B = A^\perp ⅋ B$.

Des connecteurs ! et ? sont également utilisés pour réintroduire de manière contrôlée l'affaiblissement et la contraction. L'utilisation des connecteurs de la logique linéaire permet de décomposer l'implication intuitionniste en deux opérations plus simples : $A \to B$ correspond en fait à $!A \multimap B$. En fait, $!A$ signifie que l'on a A autant de fois que l'on veut. Il peut être réutilisé ; ce qui correspond à l'utilisation d'une hypothèse en mathématiques, ou à une mise en mémoire en informatique. Lorsque l'on écrit $A \multimap B$, cela signifie que l'on a B, mais en utilisant A. Ensuite A disparaît.

Les règles de la logique linéaire propositionnelle sont rappelées sur la figure 1.1. Nous noterons cette logique **CLL** (*Classical Linear Logic*) pour la distinguer avec la logique linéaire intuitionniste que nous allons étudier plus loin. Nous noterons **MLL** le fragment constitué des seuls connecteurs multiplicatifs et **MALL** le fragment sans les exponentielles.

La logique intuitionniste (**IL**) est étudiée pour son aspect constructif et pour ses relations naturelles avec le λ-calcul et la programmation (isomorphisme de Curry-Howard). Elle peut être définie notamment grâce à des règles de déduction naturelle sans le Tiers-Exclu ou bien en calcul des séquent. Le choix du système formel est important pour le recherche de

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ cut$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \multimap B, \vdash \Delta, \Delta'}\ \multimap_L \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \multimap_R$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\ \parr_L \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ \parr_R$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \otimes_L \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\ \otimes_R$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A\&B \vdash \Delta}\ \&_{L1} \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A\&B \vdash \Delta}\ \&_{L2} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A\&B, \Delta}\ \&_R$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}\ \oplus_L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ \oplus_{R1} \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ \oplus_{R2}$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, A^{\perp} \vdash \Delta}\ Neg_L \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^{\perp}, \Delta}\ Neg_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta}\ \mathbf{1}_L \qquad \frac{}{\vdash \mathbf{1}}\ \mathbf{1}_R$$

$$\frac{}{\perp \vdash}\ \perp_R \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \perp, \Delta}\ \perp_L$$

$$\frac{}{\Gamma, \mathbf{0} \vdash \Delta}\ \mathbf{0}_L \qquad \frac{}{\Gamma \vdash \top, \Delta}\ \top_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\ \text{Affaiblissement}_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta}\ \text{Affaiblissement}_R$$

$$\frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}\ \text{Contraction}_L \qquad \frac{\Gamma \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta}\ \text{Contraction}_R$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}\ \text{Dérelicion}_L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta}\ \text{Dérelicion}_R$$

$$\frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}\ \text{Promotion}_L \qquad \frac{!\Gamma \vdash !A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}\ \text{Promotion}_R$$

FIG. 1.1 – **CLL** *(calcul propositionnel)*

preuves. L'isomorphisme de Curry-Howard permet d'assurer une correspondance entre les termes de **IL** et les λ-termes (déduction naturelle). En calcul des séquents, autoriser pour la partie droite les mêmes règles structurelles que pour la partie gauche conduit au calcul des séquents multi-conclusion définissant la validité des formules propositionnelles de la logique classique (**CL**) (annexe A.1). On obtient **IL** en restreignant la partie droite à une formule unique (calcul mono-conclusion) (annexe A.2). On peut également donner une version multi-conclusion de **IL** (annexe A.3).

1.2 ILL

Il est naturel de vouloir étudier cette distinction entre logique classique et intuitionniste pour la logique linéaire, ce qui pourra permettre de réutiliser des outils et méthodes développés pour **IL**. Mais la définition du fragment intuitionniste n'est pas aussi naturelle que celle de **IL**. La logique linéaire est présentée généralement grâce au calcul des séquents, ce qui a conduit à une définition naturelle de la logique linéaire intuitionniste **ILL** à partir d'un calcul des séquents mono-conclusion (voir figure 1.2).

Contrairement à ce qui se passe pour **IL** et **CL**, **ILL** utilise une grammaire différente de celle de **CLL**. En effet, les règles du $\parr$, de son élément neutre $\perp$, de la négation $^\perp$ et du $?$ obligent à travailler en multi-conclusion. La grammaire des formules de **ILL** est :

$$\phi ::= p \mid \phi \otimes \phi \mid \phi \multimap \phi \mid \phi \& \phi \mid \phi \oplus \phi \mid !\phi$$

où p est une formule atomique ou une constante $\top$, $\mathbf{0}$ ou $\mathbf{1}$.

Il est possible d'associer à une formule de la logique linéaire un λ-terme linéaire [26]. Dans l'article fondateur de **ILL** [24] Girard et Lafont montrent que la logique linéaire apporte des avantages par rapport à **IL** pour représenter les types de λ-termes. Dans ce contexte, le $\otimes$ est un $\wedge$ strict alors que & est un $\wedge$ paresseux. Contrairement à ces connecteurs, le $\parr$ n'apporte rien a priori ; il est même difficile de lui donner un sens dans l'interprétation des preuves comme fonctions. Cette propriété justifie l'appellation de Logique Linéaire Intuitionniste.

Le résultat suivant précise le lien entre **CLL** et **ILL** :

Théorème 1.2.1 *Soit un séquent construit en utilisant la grammaire de* **ILL** *sans constante. S'il est prouvable dans* **CLL**, *alors il est prouvable dans* **ILL**.

Démonstration : Considérons un séquent prouvable en **CLL**, qui n'utilise que la grammaire de **ILL**.

- Montrons d'abord que tous les séquents intervenant dans la preuve **CLL** sont mono-conclusion. Il suffit de remarquer que les axiomes sont mono-conclusion et que, pour chaque règle de **CLL**, si les prémisses sont mono-conclusion, la conclusion est aussi mon-conclusion.

- Enfin, il faut montrer que l'application d'une règle **CLL** dans le cas d'un séquent mono-conclusion donne exactement les mêmes prémisses que la règle **ILL**, en utilisant le point précédent.

□

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A \quad \Gamma', A \vdash B}{\Gamma, \Gamma' \vdash B}\ cut$$

$$\frac{\Gamma \vdash A \quad \Gamma', B \vdash C}{\Gamma, \Gamma', A \multimap B, \vdash C}\ \multimap_L \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \multimap_R$$

$$\frac{\Gamma, A \vdash C}{\Gamma, A\&B \vdash C}\ \&_{L1} \qquad \frac{\Gamma, B \vdash C}{\Gamma, A\&B \vdash C}\ \&_{L2} \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A\&B}\ \&_R$$

$$\frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus_L \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus_{R1} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus_{R2}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ \otimes_L \qquad \frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \otimes B}\ \otimes_R$$

$$\frac{\Gamma \vdash A}{\Gamma, \mathbf{1} \vdash A}\ \mathbf{1}_L \qquad \frac{}{\vdash \mathbf{1}}\ \mathbf{1}_R$$

$$\frac{}{\Gamma, \mathbf{0} \vdash A}\ \mathbf{0}_L \qquad \frac{}{\Gamma \vdash \top}\ \top_R$$

$$\frac{\Gamma \vdash B}{\Gamma, !A \vdash B}\ \text{Affaiblissement} \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}\ \text{Contraction}$$

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}\ \text{Déreliction} \qquad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\ \text{Promotion}$$

FIG. 1.2 – **ILL** *(Calcul des séquents mono-conclusion)*

Remarquons d'ailleurs que la preuve **ILL** est même syntaxiquement identique. Pour le résultat soit vrai, il faut bien sûr utiliser une version de **CLL** avec des règles pour le connecteur $\multimap$, et non le définir à l'aide de la négation.

La différence entre **ILL** et **CLL** est donc exprimée uniquement dans la grammaire. Pourtant on ne peut pas affirmer que l'aspect classique de **CLL** est entièrement et uniquement contenu dans les connecteurs $⅋$ et ?. Même si l'étude de **ILL** présente un intérêt certain, puisqu'elle correspond bien à ce que l'on connaît du λ-calcul [24, 8, 10], l'absence du $⅋$ est motivée essentiellement par des raisons techniques, ce qui laisse entendre que **ILL** n'est pas la notion la plus large d'intuitionnisme en logique linéaire. La solution réside dans l'étude de la sémantique du $⅋$ qui est apparemment le connecteur le moins bien compris de la logique linéaire. Une interprétation naturelle semble être liée à l'exécution en parallèle de processus [1, 15].

1.3 FILL

Une logique linéaire intuitionniste multi-conclusion, appelée *Full Intuitionistic Linear Logic* (**FILL**) a été introduite par Hyland et de Paiva [25]. Elle autorise l'utilisation restreinte de tous les connecteurs de la logique linéaire classique. Bierman [9] puis Braüner et de Paiva [11] ont ensuite donné d'autres présentations de **FILL** pour lesquelles la propriété d'élimination des coupures n'est pas trop difficile à démontrer. On utilisera ici celle de Braüner et de Paiva qui présente la particularité de ne pas utiliser le lambda-calcul, ce qui est intéressant pour notre travail.

Tout comme la présentation multi-conclusion de **IL** [31], **FILL** est définie à partir d'une interprétation plus restrictive de l'implication. Revenons un instant au cas classique. Voici les règles de l'implication à droite en **CL** et **IL** respectivement (versions multi-conclusions) :

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

Contrairement à ce qu'il se passe en logique classique, le Δ disparaît en **IL** lors du passage du séquent conclusion au séquent prémisse. En logique linéaire, la gestion des formules est plus stricte et l'affaiblissement contenu dans cette règle est interdit (par exemple le séquent $\vdash B \multimap B,\ C$ serait **FILL** mais pas **CLL**). On introduit donc une notion de dépendance entre formules, et la règle droite de l'implication n'est autorisée que lorsque Δ ne dépend pas de A. Le calcul des séquents **FILL** de Braüner et de Paiva est présenté à la figure 1.3. Elle définit pour chaque formule C à droite des séquents de la preuve τ l'ensemble $Dep_\tau(C)$ des formules de gauche dont elle dépend. Dans les règles, τ représente la preuve dont la racine est la conclusion de la règle, τ_1 la sous-preuve dont la racine est la première prémisse et τ_2 la sous-preuve éventuelle dont la racine est la seconde prémisse. Cette notion de dépendance permet de définir la notion de *preuve* **FILL**.

Définition 1.3.1 *[11] Une* preuve **FILL** *est une preuve en* **CLL** *telle que pour chaque règle* $\dfrac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\multimap_R$*, et pour toute formule* C *de* Δ*,* $A \notin Dep_{\tau_1}(C)$*.*

Voici par exemple une preuve non-**FILL** :

$$\frac{}{A \vdash A}\ ax \qquad Dep_\tau(A) = A$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ cut \qquad Dep_\tau(C) = \begin{cases} Dep_{\tau_1}(C) & \text{si } C \in \Delta \\ Dep_{\tau_2}(C)[A \mapsto Dep_{\tau_1}(A)] & \text{si } C \in \Delta' \end{cases}$$

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \multimap_R \qquad Dep_\tau(C) = \begin{cases} Dep_{\tau_1}(C)[A \mapsto \emptyset] & \text{si } C \in \Delta \\ Dep_{\tau_1}(B)[A \mapsto \emptyset] & \text{si } C = A \multimap B \end{cases}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \multimap B \vdash \Delta, \Delta'}\ \multimap_L \qquad Dep_\tau(C) = \begin{cases} Dep_{\tau_1}(C) & \text{si } C \in \Delta \\ Dep_{\tau_2}(C)[B \mapsto (Dep_{\tau_1}(A) \cup (A \multimap B))] & \text{si } C \in \Delta' \end{cases}$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\ \parr_L \qquad Dep_\tau(C) = \begin{cases} Dep_{\tau_1}(C)[A \mapsto (A \parr B)] & \text{si } C \in \Delta \\ Dep_{\tau_2}(C)[B \mapsto (A \parr B)] & \text{si } C \in \Delta' \end{cases}$$

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ \parr_R \qquad Dep_\tau(C) = \begin{cases} Dep_{\tau_1}(C) & \text{si } C \in \Delta \\ Dep_{\tau_1}(A) \cup Dep_{\tau_1}(B) & \text{si } C = A \parr B \end{cases}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \otimes_L \qquad Dep_\tau(C) = Dep_{\tau_1}(C)[A, B \mapsto A \otimes B]$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\ \otimes_R \qquad Dep_\tau(C) = \begin{cases} Dep_{\tau_1}(C) & \text{si } C \in \Delta \\ Dep_{\tau_2}(C) & \text{si } C \in \Delta' \\ Dep_{\tau_1}(A) \cup Dep_{\tau_2}(B) & \text{si } C = A \otimes B \end{cases}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta}\ \mathbf{1}_L \qquad Dep_\tau(C) = Dep_{\tau_1}(C)$$

$$\frac{}{\vdash \mathbf{1}}\ \mathbf{1}_R \qquad Dep_\tau(\mathbf{1}) = \emptyset$$

$$\frac{}{\bot \vdash}\ \bot_R \qquad \textit{rien à définir}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\ \bot_L \qquad Dep_\tau(C) = \begin{cases} Dep_{\tau_1}(C) & \text{si } C \in \Delta \\ \emptyset & \text{si } C = \bot \end{cases}$$

FIG. 1.3 – **FILL**

$$
\dfrac{\dfrac{\dfrac{}{C \vdash C}\,ax \quad \dfrac{}{B \vdash B}\,ax}{\underline{C \parr B} \vdash B, C}\,\parr_L \quad \dfrac{}{A \vdash A}\,ax}{C \parr B, \underline{B \multimap A} \vdash C, A}\,\multimap_L}{B \multimap A \vdash \underline{(C \parr B) \multimap C}, A}\,\multimap_R
$$

Ici, la formule A à droite du séquent $C \parr B, B \multimap A \vdash C, A$ dépend de $C \parr B$. Plus exactement $Dep(A) = \{C \parr B, B \multimap A\}$.

FILL doit être considérée comme une logique distincte de **CLL** et **ILL** et pas seulement une présentation multi-conclusions de **ILL**. Chacune des trois a son utilité et s'adapte à certains contextes particuliers. Un calcul multi-conclusion est par exemple intéressant pour la recherche de preuves parce qu'il permet de garder les différentes conclusions le plus longtemps possible et donc de faire les choix le plus tard possible.

Chapitre 2

Prouvabilités classique et intuitionniste

Nous avons vu que les liens entre **CLL** et **ILL** sont très forts et qu'une simple étude d'une formule prouvable dans **CLL** (sans constante) permet de déterminer si elle est prouvable dans **ILL**. Ce lien purement syntaxique montre que les rapports entre **CLL** et **ILL** sont très différents qu'entre **CL** et **IL**.

Afin de percevoir ces différences et avant d'aborder le cas de **FILL**, essayons de voir comment il est possible de détecter l'aspect intuitionniste dans les preuves de **CL**. Autrement dit, nous allons chercher avec quels critères on peut affirmer qu'une preuve écrite avec les règles de la logique classique est en fait une preuve intuitionniste. Cette étude va nous permettre de faire jaillir la signification du mot *intuitionnisme*, contenue de manière cachée dans les règles de déduction.

Cette approche permet une utilisation des méthodes classiques pour des preuves intuitionnistes et perçoit donc la prouvabilité intuitionniste comme une perturbation de la prouvabilité classique. Elle a déjà été utilisée par Ritter, Pym et Wallen [29] également pour **CL**. Ils ont proposé une méthode basée sur l'étude des $\lambda\mu$-termes associés à une preuve classique. Le $\lambda\mu$-calcul introduit par Parigot [28] permet d'étendre le λ-calcul afin de pouvoir y exprimer toutes les preuves classiques. Un $\lambda\mu$-terme contient donc toute l'information sur la preuve et peut donc être une base pour essayer de détecter si la preuve classique « contient » une preuve intuitionniste.

Il ne s'agit pas de donner un critère complet ; la preuve intuitionniste peut exister mais ne pas être contenue dans la preuve classique. Les preuves intuitionnistes sont la plupart du temps des « sous-dérivations » de preuves classiques. Le critère est capable de repérer dans une preuve classique si elle contient une sous-dérivation intuitionniste.

Considérons par exemple la preuve classique suivante :

$$\cfrac{\cfrac{\cfrac{}{B, A, C \vdash B, B}\,ax}{B, C \vdash \underline{A \to B}, B}\,\to_R}{B \vdash A \to B, \underline{C \to B}}\,\to_R$$

On peut voir facilement que malgré l'utilisation de la règle $\to_R$ dans sa forme classique, cette preuve contient les deux preuves intuitionnistes suivantes :

$$\dfrac{\overline{B, A \vdash B}\ ax}{B \vdash \underline{A \to B}, C \to B} \to_R \qquad \dfrac{\overline{B, C \vdash B}\ ax}{B \vdash A \to B, \underline{C \to B}} \to_R$$

Cependant pour détecter cela, passer par l'intermédiaire des $\lambda\mu$-termes ne fait que compliquer le critère et la caractérisation proposée se prête mal à un calcul automatique.

Nous allons donc proposer une méthode basée sur l'analyse directe de l'arbre de preuve classique d'un séquent, en utilisant des systèmes de déduction annotés. Il s'agit de placer des informations sur les formules permettant de mémoriser l'information nécessaire à la détection de la preuve intuitionniste.

Le système proposé dans la section 2.1 concerne **IL**. Nous allons ensuite essayer de voir dans quelle mesure cette idée peut-être étendue à la logique linéaire, ce qui va nous permettre de voir si l'intuitionnisme dans **FILL** est lié aux mêmes idées.

2.1 Logique classique

Nous allons présenter un système d'annotations simple pour les preuves classiques pour permettre de détecter leur caractère intuitionniste avec un critère très simple. On ajoute simplement au calcul des séquents classique une annotation sur chaque formule par un mot. Les annotations sont calculées de bas en haut sur la preuve, et gardent la mémoire de l'ordre d'application de certaines règles. Il suffit alors d'observer les annotations aux feuilles de l'arbre de preuve qui contiennent l'information voulue.

2.1.1 Preuves annotées

Dans sa version multi-conclusion, **IL** diffère de **CL** uniquement par les règles $\to_R$ et $\neg_R$ que nous appellerons *règles spéciales* (cas propositionnel) [13, 31]. Considérons d'abord la règle $\to_R$ dont voici la version **CL** :

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}$$

et la version **IL** multi-conclusion :

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B, \Delta}$$

À chaque application de cette règle, nous allons marquer toutes les formules de Δ avec la lettre 0 et les formules A et B avec la lettre x. La position des lettres dans les mots servent à repérer les différentes applications de la règle. Cette information va nous servir à interdire d'associer dans un axiome une (sous)-formule de A avec une (sous)-formule de l'une des formules de Δ. Le principe est le même pour la règle $\neg_R$.

Nous travaillons sur l'alphabet $\Sigma = \{1, x, 0\}$. Les séquents sont de la forme

$$A_1^{v_1}, \ldots A_k^{v_k} \vdash B_1^{w_1}, \ldots B_m^{w_m}$$

où $v_1, \ldots v_k, w_1, \ldots, w_m$ sont des mots sur Σ qui ont tous la même longueur.
Notations :
$u[p]$ désigne la lettre du mot u située à la position p, les lettres étant numérotées à partir de

$$\frac{}{A^v \vdash A^w}\ ax \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ cut$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A^v \vdash \Delta}\ Aff_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A^v}\ Aff_R$$

$$\frac{\Gamma, A^v, A^v \vdash \Delta}{\Gamma, A^v \vdash \Delta}\ Con_L \qquad \frac{\Gamma \vdash \Delta, A^v, A^v}{\Gamma \vdash \Delta, A^v}\ Con_R$$

$$\frac{\Gamma \vdash A^v, \Delta \quad \Gamma, B^v \vdash \Delta}{\Gamma, (A \to B)^v \vdash \Delta}\ \to_L \qquad \frac{\Gamma^{1u}, A^{xv} \vdash B^{xv}, \Delta^{0w}}{\Gamma^u \vdash (A \to B)^v, \Delta^w}\ \to_R$$

$$\frac{\Gamma, A^v \vdash \Delta \quad \Gamma, B^v \vdash \Delta}{\Gamma, (A \vee B)^v \vdash \Delta}\ \vee_L \qquad \frac{\Gamma \vdash A^v, \Delta}{\Gamma \vdash (A \vee B)^v, \Delta}\ \vee_{R1} \qquad \frac{\Gamma \vdash B^v, \Delta}{\Gamma \vdash (A \vee B)^v, \Delta}\ \vee_{R2}$$

$$\frac{\Gamma, A^v, B^v \vdash \Delta}{\Gamma, (A \wedge B)^v \vdash \Delta}\ \wedge_L \qquad \frac{\Gamma \vdash A^v, \Delta \quad \Gamma \vdash B^v, \Delta}{\Gamma \vdash (A \wedge B)^v, \Delta}\ \wedge_R$$

$$\frac{\Gamma \vdash A^v, \Delta}{\Gamma, \neg A^v \vdash \Delta}\ \neg_L \qquad \frac{\Gamma^{1u}, A^{xv} \vdash \Delta^{0w}}{\Gamma^u \vdash \neg A^v, \Delta^w}\ \neg_R$$

Lorsque les annotations sont omises, elles restent inchangées.

FIG. 2.1 – *Calcul des séquents* **CL** *annoté*

1 de droite à gauche.
$|u|$ désigne la longueur du mot u.
ε désigne le mot vide.
Si $a \in \Sigma$ et $v = v[n]...v[1] \in \Sigma^*$, av désigne le mot $av[n]...v[1]$.
Si $\Gamma = A_1, ..., A_k$ est une liste de formules, et $v = v_1, ..., v_k$ une liste de mots, Γ^v représente l'ensemble de formules annotées $A_1^{v_1}, ...A_k^{v_k}$, et Γ^{av} représente $A_1^{av_1}, ...A_k^{av_k}$.

Au départ, toutes les formules du séquent à prouver reçoivent le mot vide comme annotation. Les règles de déduction sont données sur la figure 2.1.

Définition 2.1.1 *Soient S un séquent, u un mot étiquetant une formule de gauche de S, v un mot étiquetant une occurrence de la même formule à droite de S, et p un entier vérifiant $0 < p \leq min(|u|, |v|)$. Le couple $(u[p], v[p])$ est appelé une* correspondance à la position p *sur le séquent S.*

Définition 2.1.2 *Un ensemble de séquents axiomes est dit* intuitionniste *s'il ne comporte aucune correspondance $(x, 0)$ ni correspondance $(0, x)$.*

Théorème 2.1.3 *Si un séquent a une preuve dans le système de la figure 2.1 dont l'ensemble d'axiomes est intuitionniste, alors il en existe une preuve dans* **IL**.

Le preuve n'est pas détaillée ici.

2.1.2 Exemples

Voici un exemple de preuve classique dans laquelle on peut voir une preuve intuitionniste :

$$
\dfrac{\dfrac{\dfrac{}{\Gamma, A^x \vdash A^1, B^x, B^0}\,ax \quad \dfrac{}{\Gamma, B^1 \vdash B^x, B^0}\,ax}{\Gamma, A^x, \underline{A \rightarrow B}^1 \vdash B^x, B^0}\,\rightarrow_L}{\Gamma, A \rightarrow B \vdash \underline{A \rightarrow B}, B}\,\rightarrow_R \quad \dfrac{}{\Gamma, B \vdash B}\,ax
$$
$$
\dfrac{\Gamma, A \rightarrow B \vdash \underline{A \rightarrow B}, B \qquad \Gamma, B \vdash B}{\Gamma, A \rightarrow B, \underline{(A \rightarrow B) \rightarrow B} \vdash B}\,\rightarrow_L
$$

Remarquons que si les règles sont appliquées dans un autre ordre, il n'est plus possible de détecter une preuve intuitionniste à l'aide du critère précédent (à cause d'une correspondance $(x, 0)$ sur la formule A d'un des axiomes) :

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, A^x \vdash B^x, A^0}\,ax}{\Gamma \vdash \underline{A \rightarrow B}, A, B}\,\rightarrow_R \quad \dfrac{}{\Gamma, B \vdash A, B}\,ax}{\Gamma, \underline{(A \rightarrow B) \rightarrow B} \vdash A, B}\,\rightarrow_L \quad \dfrac{}{\Gamma, B \vdash B}\,ax}{\Gamma, \underline{A \rightarrow B}, (A \rightarrow B) \rightarrow B \vdash B}\,\rightarrow_L
$$

Comme dans le critère donné par Ritter, Pym et Wallen [29], nous n'avons donc pas la complétude. Il paraît d'ailleurs peu probable de pouvoir trouver un critère qui indique s'il existe une preuve intuitionniste à chaque fois qu'elle existe, juste par l'étude d'une preuve classique.

2.1.3 Extraction de la preuve intuitionniste

Lorsque la preuve classique contient une preuve intuitionniste, les informations contenues dans les mots peuvent servir à extraire cette preuve intuitionniste. Le principe consiste à détecter les formules inutiles dans la preuve classique en observant les annotations et à les supprimer. Dans le premier des deux exemples précédents (page 17) on obtient la preuve suivante :

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, A \vdash A, B}\,ax \quad \dfrac{}{\Gamma, B \vdash B}\,ax}{\Gamma, A, \underline{A \rightarrow B} \vdash B}\,\rightarrow_L}{\Gamma, A \rightarrow B \vdash \underline{A \rightarrow B}, B}\,\rightarrow_R \quad \dfrac{}{\Gamma, B \vdash B}\,ax}{\Gamma, A \rightarrow B, \underline{(A \rightarrow B) \rightarrow B} \vdash B}\,\rightarrow_L
$$

Cette opération d'extraction à partir d'une preuve détectée « intuitionniste » peut être réalisée grâce à un algorithme.

2.2 Logique linéaire

Essayons maintenant de voir si ce critère peut être étendu à la logique linéaire. Dans le cas de **ILL**, on peut donner un critère très simple qui consiste à observer la présence ou non de connecteurs $\parr$ ou ?, ou encore à vérifier que la preuve est bien monoconclusion (voir théorème 1.2.1). Il n'est pas possible de donner une preuve **CLL** non **ILL** d'un séquent dont la preuve **ILL** existe.

Le cas de **FILL** est beaucoup plus intéressant. Il y a une différence essentielle avec le cas de **IL** : dans **FILL**, la structure des preuves est exactement la même que dans **CLL**. Une preuve **FILL** est définie grâce à un critère sur les preuves **CLL** (voir définition 1.3.1). En appliquant le même type de critère que précédemment, on peut donc espérer détecter exactement les preuves **FILL**. Le problème de l'extraction de la preuve intuitionniste ne se pose plus. Nous ne nous intéressons plus à savoir si une preuve **CLL** contient une preuve **FILL**, mais à savoir si une preuve **CLL** **est** une preuve **FILL**. Comme dans le cas précédent, si le critère répond non, cela ne signifie pas qu'une preuve **FILL** n'existe pas.

En fait, la définition de **FILL** donnée par Braüner et de Paiva correspond exactement à l'idée utilisée précédemment pour repérer le caractère intuitionniste des preuves classiques. Nous allons donc présenter une adaptation de cette idée pour donner une nouvelle présentation de **FILL** à l'aide d'un calcul des séquents annoté.

2.2.1 Un calcul des séquents annoté pour FILL

Nous allons présenter deux systèmes de déduction annotés pour **FILL**. Le premier est une extension directe du critère précédent. Nous verrons ensuite que ce critère peut être simplifié.

Le principe est similaire au cas de **IL**. Les formules de la preuve en séquents sont annotées en utilisant des mots ; le critère porte sur les mots des séquents axiomes. Nous avons cependant besoin d'un système un peu plus élaboré, puisqu'il ne s'agit plus seulement de simuler l'effacement de certaines formules lors d'une règle spéciale, mais de tester précisément les dépendances entre formules. Pour pouvoir traiter des chaînes transitives de dépendances qui peuvent être de longueur quelconque, nous allons utiliser un alphabet Σ infini comportant les lettres 1, x, et 0.

La position dans le mot sert toujours à repérer la règle spéciale concernée. Dans cette version, une formule est étiquetée par une liste de mots, pour autoriser plusieurs correspondances à la même position entre les deux mêmes formules.

Notations :
Une liste de tête u de queue l sera notée $u :: l$. Les listes seront notées en extension avec des point-virgules, par exemple la liste contenant les mots u, v et w sera notée $u; v; w$. La liste vide est notée $\emptyset$.

Au départ, chaque formule du séquent racine de l'arbre de preuve est étiqueté avec la liste $\varepsilon :: \emptyset$ contenant uniquement le mot vide.

Les règles de ce calcul des séquents annoté sont données par la figure 2.2.

La notion de correspondance de la définition 2.1.1 est étendue aux à ce nouveau calcul et nous permet de définir les *chaînes de dépendances.*

Définition 2.2.1 *Soit E un ensemble de séquents. Une* chaîne de dépendances sur E à la position p entre les lettres α et β *est une suite de couples de la forme $(\alpha, \alpha_1), (\alpha_1, \alpha_2), ..., (\alpha_n, \beta)$ ($n \in \mathbb{N}^*$) dont chaque élément est une correspondance à la position p sur un des séquents de E.*

Définition 2.2.2 *Un ensemble de séquents est dit* intuitionniste *s'il n'existe aucune position p avec une chaîne de dépendances entre x et 0.*

$$\frac{\Gamma^{1u::l_1}, A^{x1^n::\emptyset} \vdash B^{1v::l_2}, \Delta^{0w::l_3}}{\Gamma^{u::l_1} \vdash (A \multimap B)^{v::l_2}, \Delta^{w::l_3}} \multimap_R$$

$$\frac{\Gamma \vdash A^{a^n::\emptyset}, \Delta \quad \Gamma', A^{a^n::\emptyset} \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \, cut \qquad \text{où } a \text{ est une lettre fraîche}$$

$$\frac{\Gamma \vdash A^{a^n::\emptyset}, \Delta \quad \Gamma', B^{v::a^n::l} \vdash \Delta'}{\Gamma, \Gamma', (A \multimap B)^{v::l} \vdash \Delta, \Delta'} \multimap_L \qquad \text{où } a \text{ est une lettre fraîche}$$

$$\frac{}{A^{l_1} \vdash A^{l_2}} \, ax$$

$$\frac{\Gamma, A^l \vdash \Delta \quad \Gamma', B^l \vdash \Delta'}{\Gamma, \Gamma', (A \parr B)^l \vdash \Delta, \Delta'} \parr_L \qquad \frac{\Gamma \vdash A^l, B^l, \Delta}{\Gamma \vdash (A \parr B)^l, \Delta} \parr_R$$

$$\frac{\Gamma, A^l, B^l \vdash \Delta}{\Gamma, (A \otimes B)^l \vdash \Delta} \otimes_L \qquad \frac{\Gamma \vdash A^l, \Delta \quad \Gamma' \vdash B^l, \Delta'}{\Gamma, \Gamma' \vdash (A \otimes B)^l, \Delta, \Delta'} \otimes_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1}^l \vdash \Delta} \mathbf{1}_L \qquad \frac{}{\vdash \mathbf{1}^l} \mathbf{1}_R$$

$$\frac{}{\bot^l \vdash} \bot_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot^l, \Delta} \bot_R$$

Lorsque les annotations sont omises, elles restent inchangées.
n est la longueur du mot en tête de liste pour chaque formule de la conclusion (ces mots ont tous la même longueur).

FIG. 2.2 – *Calcul des séquents* **CLL** *annoté pour* **FILL**.

Théorème 2.2.3 *Soit P un arbre de preuve construit à partir des règles de la figure 2.2 ; P est une preuve* **FILL** *si et seulement si l'ensemble des séquents initiaux est intuitionniste.*

La démonstration de ce théorème peut être très facilement adaptée de celle donnée dans la section 2.2.3 pour le critère suivant.

Dans l'exemple suivant, la preuve est **CLL** mais pas **FILL**, à cause d'une chaîne de dépendances $(x,y),(y,0)$.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{\underline{C^x} \vdash C^z}\,ax
      \qquad
      \dfrac{}{D^x \vdash D^y}\,ax
    }{\underline{(C \parr D)^x} \vdash C^z, D^y}\,\parr_L
    \qquad
    \dfrac{}{B^{1;z} \vdash B^1}\,ax
  }{\underline{(C \multimap B)^1}, (C \parr D)^x \vdash D^y, B^1}\,\multimap_L
  \qquad
  \dfrac{}{A^{1;y} \vdash A^0}\,ax
}{
  \dfrac{(C \multimap B)^1, \underline{(D \multimap A)^1}, (C \parr D)^x \vdash B^1, A^0}{C \multimap B, D \multimap A \vdash \underline{(C \parr D) \multimap B}, A}\,\multimap_R
}\,\multimap_L
$$

2.2.2 Simplification du critère

Dans le critère précédent, la position dans le mot sert à repérer à quelle application de la règle $\multimap_R$ les lettres sont apparues, ce qui évite de mélanger les chaînes de dépendances. Mais cette technique a l'inconvénient de compliquer le système puisqu'elle oblige à conserver plusieurs mots pour chaque formule, et à créer des mots de la forme y^n, ne sachant pas à quelle position le y va être utilisé.

Une autre possibilité, plus éloignée du critère initial, consiste à utiliser un marquage différent pour chaque règle spéciale. La position dans le mot n'a alors plus aucune importance, les mots peuvent avoir des longueurs différentes, et on n'a plus besoin que d'un seul mot par formule. L'alphabet utilisé est de la forme $\Sigma = \Sigma_1 \cup \Sigma_2$, avec Σ_1 et Σ_2 infinis disjoints quelconques. Σ_1 sert pour les extrémités de chaînes de dépendances, et Σ_2 est utilisé pour les lettres intermédiaire.

On commence avec le mot vide sur chaque formule du séquent racine de l'arbre de preuve. Les règles de ce calcul des séquents annoté sont données par la figure 2.3.

Les positions n'ayant plus de signification particulière, les notions de *correspondance* et de *chaînes de dépendances* sont définies de la manière suivante :

Définition 2.2.4 *Une* correspondance *sur un séquent S est un couple (α,β), où α est une lettre d'un mot étiquetant une formule de gauche de S et β une lettre d'un mot étiquetant la même formule à droite de S.*

Définition 2.2.5 *Soit E une liste de séquents, et soit $s_0,...,s_n$ une suite extraite de E. Une* chaîne de dépendances sur E entre les lettres α et β *est une suite de couples de la forme $(\alpha_0,\alpha_1),(\alpha_1,\alpha_2),...,(\alpha_n,\alpha_{n+1})$ ($n \in \mathbb{N}^*$) telle que $\alpha_0 = \alpha$, $\alpha_{n+1} = \beta$, et (α_i,α_{i+1}) est une correspondance sur s_i.*

Définition 2.2.6 *Une suite de séquents est dite* intuitionniste *s'il n'existe aucune chaîne de dépendances reliant deux occurrences d'une même lettre de Σ_1.*

Théorème 2.2.7 *Un arbre construit à partir des règles de la figure 2.3 est une preuve* **FILL** *si et seulement si la suite de ses séquents axiomes pris de gauche à droite est intuitionniste.*

$$\frac{\Gamma, A^x \vdash B^v, \Delta^{xw}}{\Gamma \vdash (A \multimap B)^v, \Delta^w} \multimap_R \qquad \text{où } x \text{ est une lettre fraîche de } \Sigma_1$$

$$\frac{\Gamma \vdash A^y, \Delta \quad \Gamma', A^y \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \; cut \qquad \text{où } y \text{ est une lettre fraîche de } \Sigma_2$$

$$\frac{\Gamma \vdash A^y, \Delta \quad \Gamma', B^{yv} \vdash \Delta'}{\Gamma, \Gamma', (A \multimap B)^v \vdash \Delta, \Delta'} \multimap_L \qquad \text{où } y \text{ est une lettre fraîche de } \Sigma_2$$

$$\frac{}{A^v \vdash A^w} \; ax$$

$$\frac{\Gamma, A^v \vdash \Delta \quad \Gamma', B^v \vdash \Delta'}{\Gamma, \Gamma', (A \parr B)^v \vdash \Delta, \Delta'} \parr_L \qquad \frac{\Gamma \vdash A^v, B^v, \Delta}{\Gamma \vdash (A \parr B)^v, \Delta} \parr_R$$

$$\frac{\Gamma, A^v, B^v \vdash \Delta}{\Gamma, (A \otimes B)^v \vdash \Delta} \otimes_L \qquad \frac{\Gamma \vdash A^v, \Delta \quad \Gamma' \vdash B^v, \Delta'}{\Gamma, \Gamma' \vdash (A \otimes B)^v, \Delta, \Delta'} \otimes_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1}^v \vdash \Delta} \mathbf{1}_L \qquad \frac{}{\vdash \mathbf{1}^v} \mathbf{1}_R$$

$$\frac{}{\bot^v \vdash} \bot_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot^v, \Delta} \bot_R$$

Lorsque les annotations sont omises, elles restent inchangées.

FIG. 2.3 – *Calcul des séquents* **CLL** *annoté.*

L'ordre des prémisses pour chaque règle de la figure 2.3 doit évidemment être respecté lors de la construction de la preuve. La démonstration de ce théorème fait l'objet de la section suivante.

Reprenons l'exemple de la section 2.2.1 avec ce critère. La preuve est **CLL** mais pas **FILL**, à cause d'une chaîne de dépendances $(x,y),(y,x)$. Ici $x \in \Sigma_1$ et $(y,z) \in \Sigma_2^2$.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$C^x \vdash C^z$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$D^x \vdash D^y$}
\RightLabel{$\parr_L$}
\BinaryInfC{$\underline{(C \parr D)^x} \vdash C^z, D^y$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$B^z \vdash B$}
\RightLabel{$\multimap_L$}
\BinaryInfC{$\underline{C \multimap B}, (C \parr D)^x \vdash D^y, B$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A^y \vdash A^x$}
\RightLabel{$\multimap_L$}
\BinaryInfC{$C \multimap B, \underline{D \multimap A}, (C \parr D)^x \vdash B, A^x$}
\RightLabel{$\multimap_R$}
\UnaryInfC{$C \multimap B, D \multimap A \vdash \underline{(C \parr D) \multimap B}, A$}
\end{prooftree}
$$

2.2.3 Preuve du théorème

La démonstration est faite pour le critère de la section 2.2.2, mais peut-être facilement adapté pour celui de la section 2.2.1.

Commençons par donner quelques propriétés pour comprendre le comportement des annotations.

Propriété 2.2.8 *Dans un sous-arbre contenant la lettre $\alpha \in \Sigma_2$ à la racine, α apparaît toujours du même côté des séquents.*

Autrement dit, les lettres de Σ_2 ne changent jamais de côté.

Remarquons également que si (α, β) est une correspondance sur un séquent S dans une preuve, alors toutes les lettres présentes dans une chaîne de dépendances entre α et β sont introduites dans le sous-arbre de racine S.

Ces résultats préliminaires vont nous servir à démontrer le théorème 2.2.7, en prouvant l'équivalence du critère avec le calcul des dépendances. Pour cela, nous montrons les deux lemmes suivants, dont on peut facilement vérifier qu'ils correspondent chacun à une implication du théorème.

Lemme 2.2.9 *S'il existe une chaîne de dépendances aux feuilles d'un arbre de preuves entre deux occurrences d'une même lettre de Σ_1, la condition de dépendance de la définition 1.3.1 n'est pas vérifiée à l'endroit où est apparu cette lettre.*

Lemme 2.2.10 *Considérons un sous-arbre dont la dernière règle est $\multimap_R$, telle qu'il existe $A \in \Delta$ vérifiant $B \in Dep(A)$. Ce sous-arbre a une chaîne de dépendances entre deux occurrences d'une même lettre de Σ_2.*

Ces lemmes sont respectivement des conséquences assez immédiates des deux propriétés suivantes :

Propriété 2.2.11 *Soit un arbre de preuve, avec une chaîne de dépendances entre les lettres α et β aux feuilles de l'arbre. Pour tout sous-arbre contenant cette chaîne, si la racine a une correspondance (α, β) alors pour ce séquent, la formule étiquetée par β dépend de celle étiquetée par α.*

Démonstration : On montre ce résultat par récurrence sur la profondeur de l'arbre. *cas $n = 1$:* On vérifie facilement la propriété sur les règles axiomes.
Supposons l'hypothèse vérifiée pour n. On fait une étude de cas selon la règle appliquée à la racine :
– Le cas des règles à une prémisse est trivial
– Pour les règles $\parr_L$ et $\otimes_R$, on vérifie que la chaîne de dépendances est entièrement dans l'un des deux sous-arbres.
– C'est la même chose pour un Cut ou $\multimap_L$ si la nouvelle lettre introduite n'apparaît pas dans la chaîne.
– Pour un Cut ou $\multimap_L$, si la nouvelle lettre y introduite apparaît dans la chaîne, et si l'on a une correspondance (α, β) à la racine, la sous-chaîne entre α et y est entièrement dans le sous-arbre gauche et la sous-chaîne entre y et β est entièrement dans le sous-arbre droit. L'hypothèse de récurrence nous donne des dépendances entre les formules étiquetées par α et y dans la prémisse gauche et par y et β dans la prémisse droite. On déduit facilement le résultat en regardant la règle de calcul des dépendances. □

Propriété 2.2.12 *Si un arbre de preuve de racine $\Gamma, E \vdash F, \Delta$ est tel que $E \in Dep(F)$ et si les formules E et F sont le support d'une correspondance (α, β) alors il existe une chaîne de dépendances entre α et β aux feuilles de cet arbre.*

Démonstration : La démonstration se fait par récurrence sur la profondeur de l'arbre. Il suffit de le vérifier pour chaque règle. □

2.2.4 Exemples

Comme dans le cas classique, voici un exemple qui montre qu'il n'y a pas de complétude. Un même séquent peut avoir une preuve **FILL** et une preuve non-**FILL**. Nous considérerons pour montrer cela un séquent qui a une seule formule atomique, dont on différenciera les occurrences grâce à des indices. Voici deux preuves annotées différentes d'un tel séquent. La première n'est pas **FILL**, à cause d'une correspondance (x, x) dans un des axiomes :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A_1^x \vdash A_2^x}\,ax \qquad \dfrac{}{A_4 \vdash A_3^x}\,ax}{A_1^x, A_4 \vdash \underline{(A_2 \otimes A_3)^x}}\,\otimes_{R-2,3} \qquad \dfrac{}{A_5 \vdash A_6}\,ax}{A_1^x, \underline{A_4 \parr A_5} \vdash A_6, (A_2 \otimes A_3)^x}\,\parr_L \qquad \dfrac{}{A_8 \vdash A_7}\,ax}{A_1^x, A_4 \parr A_5, A_8 \vdash \underline{A_6 \otimes A_7}, (A_2 \otimes A_3)^x}\,\otimes_{R-6,7}}{A_4 \parr A_5, A_8 \vdash \underline{A_1 \multimap (A_6 \otimes A_7)}, A_2 \otimes A_3}\,\multimap_R$$

On peut vérifier que $A_1 \in Dep(A_2 \otimes A_3)$ sur la deuxième ligne en partant du bas. La seconde est une preuve dans **FILL** : il n'y a aucune correspondance aux axiomes donc pas de chaîne de dépendances.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A_1^x \vdash A_6$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A_5 \vdash A_7$}
\RightLabel{$\otimes_{R-6,7}$}
\BinaryInfC{$A_1^x, A_5 \vdash \underline{A_6 \otimes A_7}$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A_4 \vdash A_3^x$}
\RightLabel{$\parr_L$}
\BinaryInfC{$A_1^x, \underline{A_4 \parr A_5} \vdash A_3^x, A_6 \otimes A_7$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A_8 \vdash A_2^x$}
\RightLabel{$\otimes_{R-2,3}$}
\BinaryInfC{$A_1^x, A_4 \parr A_5, A_8 \vdash \underline{(A_2 \otimes A_3)^x}, A_6 \otimes A_7$}
\RightLabel{$\multimap_R$}
\UnaryInfC{$A_4 \parr A_5, A_8 \vdash \underline{A_1 \multimap (A_6 \otimes A_7)}, A_2 \otimes A_3$}
\end{prooftree}
$$

2.2.5 Prouvabilité intuitionniste

Le théorème 2.2.7 présente l'intérêt de pouvoir être mis en œuvre facilement. Étant donnée une preuve classique annotée, il suffit d'observer les annotations aux feuilles pour faire le test des chaînes de dépendances. Un algorithme pourrait être le suivant :

> Pour tous les séquents S dans lesquels un élément x de Σ_1 annote une formule de gauche, appeler `Recherche_chaine(`x`,`S`,`x`)`
>
> où `Recherche_chaine` est définie par :
>
> Procédure `Recherche_chaine(origine,séquent,lettre_courante)` :
> pour toutes les correspondances (`lettre_courante`, α),
>
> – Si $\alpha =$ `origine` alors ECHEC
>
> – sinon, pour tous les séquents axiomes S' à droite de `séquent`,
> appeler `Recherche_chaine(origine,` S'`,` α`)`

L'algorithme sort par un échec si et seulement si la preuve n'est pas **FILL**. Les exemples donnés ci-dessus montrent que la taille des mots est en général très petite et donc l'algorithme n'a pas beaucoup de cas à traiter. Une étude de la complexité de l'algorithme serait intéressante mais nécessiterait de préciser les choix d'implantation.

Cet algorithme nous permet de vérifier si une preuve déjà construite est **FILL**. Il serait intéressant de voir comment les annotations peuvent servir à développer des heuristiques permettant d'aboutir plus facilement à des preuves **FILL**. Dans les preuves de la section 2.2.4, on voit par exemple que l'application de la règle $\otimes_R$ sur une formule marquée permet d'éliminer une possibilité de correspondance. Dans le chapitre suivant, nous allons voir qu'un algorithme de construction de réseaux **FILL** peut servir à construire des preuves **FILL** en calcul des séquents.

Chapitre 3

Prouvabilité et réseaux de preuve

3.1 Définitions

Une autre approche pour étudier la prouvabilité **FILL** à partir de **CLL** est fondée sur les réseaux de preuve. Ces structures ont été introduites par Girard [21] comme transposition en logique linéaire de la déduction naturelle de **IL**. Un réseau est un graphe particulier dont les nœuds sont des formules de la logique linéaire. Pour construire le réseau associé à un séquent, il faut commencer par le transformer en un séquent, équivalent pour **CLL**, sans hypothèses, en utilisant la négation. On construit alors l'arbre de décomposition des formules. Il ne reste plus qu'à placer les liens axiomes de façon à construire un réseau bien formé.

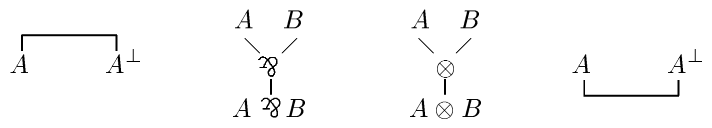
Le réseau représente la preuve du séquent. Un avantage par rapport au calcul des séquents est de supprimer tous les problèmes de permutation de règles, ce qui réduit les indéterminismes lors de la construction de la preuve.

Nous allons poursuivre notre étude de la prouvabilité intuitionniste à partir de la prouvabilité classique en utilisant les réseaux. Les notions de réseaux **ILL** et de réseaux **FILL** seront définies, puis nous présenterons un algorithme de construction pour ces réseaux qui étend celui de [19] de **CLL** à **FILL**.

Même s'il est possible d'étendre la définition, nous nous limiterons ici à la logique linéaire multiplicative sans constantes.

Définissons tout d'abord la notion de *pré-réseau.*

Définition 3.1.1 *Un* pré-réseau *est un graphe dont les sommets sont des formules, construit à partir des sous-structures suivantes (appelées* liens*) :*



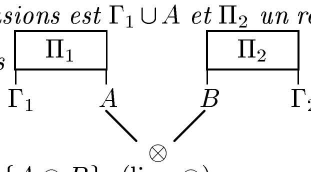
(où A et B sont des formules quelconques), et vérifiant :

- *toute formule est prémisse d'au moins un lien,*
- *toute formule est conclusion d'exactement un lien.*

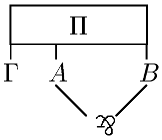
Plusieurs définitions des réseaux ont été proposées, qui utilisent pour la plupart des critères permettant de vérifier qu'un pré-réseau est bien un réseau [12, 21]. Celle que nous allons utiliser ici, introduite par Bellin [7], n'utilise pas un critère mais une définition inductive, ce qui est intéressant dans notre approche de construction de réseaux. Un réseau de preuve est donc un pré-réseau particulier défini de la manière suivante :

Définition 3.1.2 *Un* réseau de preuve **MLL** *et son ensemble de* conclusions *sont définis inductivement par*

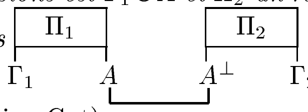
- *Si A est une formule multiplicative quelconque,* $A \qquad A^\perp$ *est un réseau, de conclusions A et $A^\perp$.* (lien axiome)
- *Si Π_1 est un réseau dont l'ensemble des conclusions est $\Gamma_1 \cup A$ et Π_2 un réseau disjoint de Π_1 dont les conclusions sont $\Gamma_2 \cup B$, alors*

  est un réseau dont l'ensemble des conclusions est $\Gamma_1 \cup \Gamma_2 \cup \{A \otimes B\}$. (lien $\otimes$)
- *Si Π est un réseau dont l'ensemble des conclusions est $\Gamma \cup A \cup B$, alors*

  est un réseau dont l'ensemble des conclusions est $\Gamma \cup \{A ⅋ B\}$. (lien ⅋)
- *Si Π_1 est un réseau dont l'ensemble des conclusions est $\Gamma_1 \cup A$ et Π_2 un réseau disjoint de Π_1 dont les conclusions sont $\Gamma_2 \cup A^\perp$, alors*

  est un réseau dont l'ensemble des conclusions est $\Gamma_1 \cup \Gamma_2$. (lien Cut)

On peut définir sur les réseaux une opération de réduction du réseau, qui correspond à la notion d'exécution dans le λ-calcul. Un théorème d'élimination des coupures dit que s'il existe un réseau de conclusions Γ alors il en existe un sans liens Cut. Tous les réseaux de preuve que nous considérerons dans la suite seront sans coupure.

Remarque 3.1.3 *On ne distinguera pas l'ordre des prémisses dans les réseaux. Autrement dit les prémisses pourront être échangées sur le dessin et le réseau obtenu sera considéré identique au premier. Dans un cadre non-commutatif, il faudrait tenir compte de l'ordre des prémisses [18].*

3.2 Orientation des réseaux

3.2.1 Définition des réseaux orientés

Pour construire les réseaux de la définition ci-dessus à partir d'une formule quelconque, on utilise les équivalences suivantes :

$$(E \otimes F)^\perp = E^\perp ⅋ F^\perp$$
$$(E ⅋ F)^\perp = E^\perp \otimes F^\perp$$
$$E \multimap F = E^\perp ⅋ F$$

En logique linéaire intuitionniste, l'implication ne peut plus être définie à partir du ⅋. On ne dispose plus de la négation. Pour construire les réseaux, nous allons cependant effectuer

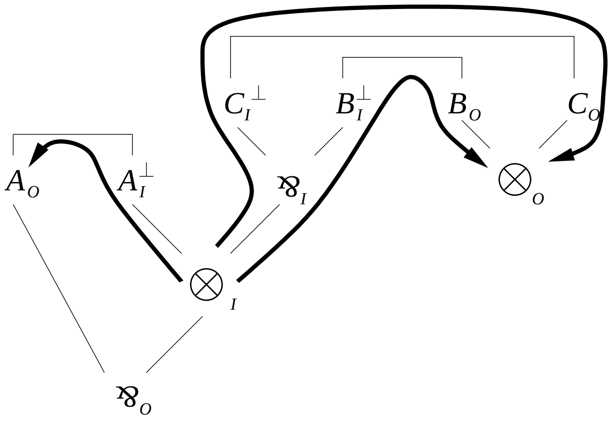


FIG. 3.1 – *Exemple de réseau orienté non-***FILL**

ces mêmes transformations, en introduisant juste une orientation. Pour cela, on annote chaque noeud avec soit la lettre I (Input) soit la lettre O (Output). Cela nous permet de garder la structure des réseaux en introduisant la négation, mais en gardant les informations nécessaires pour détecter si le réseau correspond à une preuve intuitionniste ou non. Cette annotation est effectuée au moment de la construction de l'arbre à partir d'un séquent, de la manière suivante :

- Les formules à gauche du séquent de départ reçoivent la lettre I, les formules à droite reçoivent la lettre O,
- Les formules filles d'un $⅋$ ou d'un $\otimes$ qui est réellement un $⅋$ ou un $\otimes$ dans la formule initiale reçoivent la même annotation que leur père,
- La formule à droite d'un $⅋$ ou d'un $\otimes$ représentant une implication reçoit l'orientation de son père ; la formule de gauche reçoit l'autre orientation.

Dans une preuve en calcul des séquent, les formules étiquetées I (contravariantes) apparaîtront à gauche des séquents et les formules étiquetées O (covariantes) apparaîtront à droite.

L'exemple de la figure 3.1 est le réseau orienté correspondant au séquent

$$\vdash (A ⅋ (B \otimes C)) \multimap A,\ \ B \otimes C$$

En remplaçant le $\multimap$ et en mettant les orientations, on obtient le séquent suivant (classiquement équivalent) :

$$\vdash ((A_I^\perp \otimes (B_I^\perp ⅋ C_I^\perp)_I)_I ⅋ A_O)_O,\ \ (B_O \otimes C_O)_O$$

On peut énoncer tout de suite une propriété importante des réseaux orientés :

Propriété 3.2.1 *Tout réseau orienté a au moins une conclusion O.*

Démonstration : Par induction sur la définition des réseaux. □

Remarque : On peut construire des réseaux sans conclusion I, par exemple à partir de séquents prouvables, sans formules à gauche.

3.2.2 Des réseaux pour **ILL**

Définition 3.2.2 *Un* réseau **ILL** *est un réseau orienté qui n'a pas de nœud de la forme*

$$\begin{array}{cc} O & O \\ \multicolumn{2}{c}{\parr} \\ \multicolumn{2}{c}{O} \end{array} \quad \textit{ni} \quad \begin{array}{cc} I & I \\ \multicolumn{2}{c}{\otimes} \\ \multicolumn{2}{c}{I} \end{array}$$

Théorème 3.2.3 *Un séquent est prouvable en* **ILL** *si et seulement si le réseau associé est* **ILL**.

C'est une conséquence directe du théorème 1.2.1.

3.3 Des réseaux pour **FILL**

3.3.1 Définition du critère

On peut utiliser l'orientation introduite à la section précédente pour donner une définition des réseaux **FILL**. La définition donnée ici est adaptée de celle de Bellin [5] pour **FILL** + **MIX**.

Définition 3.3.1 *Un* chemin de dépendance *est un chemin entre une conclusion I et une conclusion O d'un réseau, qui vérifie les deux règles :*

- *toujours passer sur les nœuds d'orientation I en montant,*
- *toujours passer sur les nœuds d'orientation O en descendant.*

Les chemins de dépendance possibles pour chaque type de nœud sont résumés sur la figure 3.2.

Définition 3.3.2 *Soit R un réseau et A un de ses nœuds. L'*empire de A *est le plus grand sous-réseau dont A est une conclusion.*

Définition 3.3.3 *Un* réseau **FILL** *est un réseau orienté tel que pour tout noeud* $\begin{array}{cc} E_I & F_O \\ \multicolumn{2}{c}{\parr_O} \end{array}$, *il n'y a aucun chemin de dépendance entre E et une autre une conclusion de l'empire de E que cette occurrence de F.*

La figure 3.3 présente un exemple de réseau non-**FILL** et de réseau **FILL** pour la même formule. Il s'agit des réseaux correspondants aux exemples de la page 23. Les formules atomiques ne sont pas précisées ; il s'agit de A pour les atomes étiquetés par O et de $A^{\perp}$ pour ceux étiquetés par I. Dans le premier cas, il y a un chemin de dépendance entre la prémisse I du $\parr$ et un O qui n'est pas prémisse de ce $\parr$.

3.4 Algorithme de construction de réseaux pour **FILL**

L'algorithme de construction des réseaux pour **MLL** a été introduit par Didier Galmiche dans [17, 14] où il est présenté en détail. Il a été étendu ensuite aux connecteurs additifs **MALL** [16] d'après la définition des réseaux **MALL** de Girard [23], et au cas de la logique non commutative [18].

Avant de présenter l'adaptation de l'algorithme à **FILL**, rappelons l'algorithme pour **MLL**.

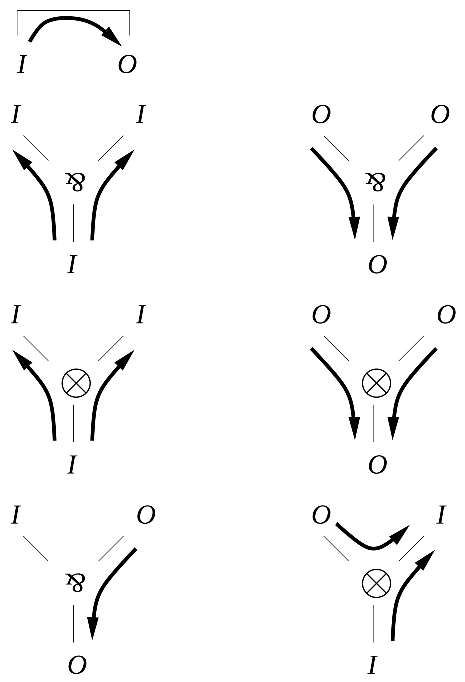


FIG. 3.2 – *Chemins de dépendance dans les réseaux* **FILL**

3.4.1 L'algorithme pour MLL

L'algorithme utilise la définition inductive des réseaux donnée ci-dessus, ce qui permet de vérifier au fur et à mesure de la construction que l'on a toujours un réseau, plutôt que de faire une vérification à partir d'un réseau entièrement construit à l'aide d'un critère.

On part bien entendu d'un séquent sans hypothèses dont on construit l'arbre de décomposition. Il ne reste plus qu'à placer les liens axiomes correctement. Pour cela, on construit des sous-réseaux en partant du haut, et on les assemble petit à petit en utilisant trois procédures :

- une pour rechercher dans l'arbre de décomposition une feuille pas encore traitée,
- une pour traiter les feuilles en construisant des liens axiomes,
- une pour propager la construction vers le bas en rassemblant des sous-réseaux déjà construits.

Le choix de la formule à traiter se fait toujours en partant de branches dont certaines feuilles ont déjà été traitées, pour éviter de construire des réseaux disjoints.

Que ce soit pour la recherche de la formule à traiter ou pour la construction du réseau proprement dite par propagation, respecter la règle suivante :

Toujours traiter les $\otimes$ avant les $\parr$ si possible.

En particulier, le choix de la première feuille à traiter se fait grâce à cette règle. Choisir une branche, puis ses fils successifs jusqu'à une feuille, en préférant à chaque fois les nœuds $\otimes$. Arrivé à cette feuille, construire le premier sous-réseau en essayant toutes les formules duales jusqu'à ce que l'on en trouve une qui convient.

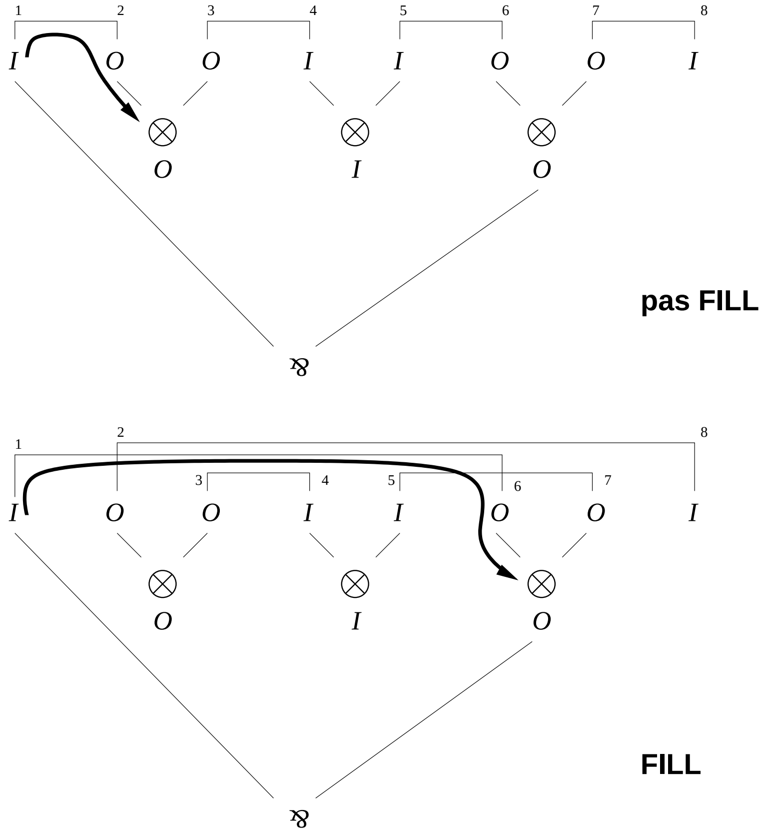


FIG. 3.3 – *Exemple de réseaux différents pour une même formule. Voir exemple page 23 pour les preuves en séquents correpondantes.*

À chaque fois qu'une partie de réseau est créée, propager vers le bas à partir de la formule courante, c'est-à-dire :

- si le noeud sous la formule courante est un $\otimes$ dont les deux fils sont dans deux sous-réseaux disjoints, fusionner les deux sous-réseaux en construisant ce nœud,
- si le noeud sous la formule courante est un $\parr$ dont les deux fils sont dans le même sous-réseau, ajouter ce nœud au réseau,
- si le noeud sous la formule courante est un $\parr$ dont les deux fils sont dans deux sous-réseaux disjoints, mettre ce nœud en attente
- si le noeud sous la formule courante est un $\otimes$ dont un seul fils est traité, traiter l'autre fils en priorité.

Poursuivre en choisissant une nouvelle formule à traiter, en traitant en premier les $\otimes$ rencontrés, puis les $\parr$, et essayer de propager sur les $\parr$ mis en attente.

À la fin, si le réseau existe, toutes les formules sont traitées et le réseau est construit.

Plus de détails sur les choix effectués par l'algorithme et l'ordre de construction du réseau ne sont pas utiles pour la suite. Il suffit de savoir que la construction suit la définition inductive des réseaux.

3.4.2 Extension à **FILL**

Pour tenir compte du critère des chemins dans les réseaux **FILL**, nous allons ajouter une information au moment de la création de chaque nœud du réseau pour mémoriser tous les chemins de dépendance entre une conclusion I et une conclusion O du réseau courant.

Pour cela, il suffit de suivre les indications données par la figure 3.2. Ce qui donne l'algorithme suivant, résumé sur la figure 3.4 :

Lors de la construction d'un lien axiome : il existe un chemin de dépendance entre la formule de polarité I et celle de polarité O.

Lors de la création d'un $\begin{array}{c} I \quad I \\ \parr \\ I \end{array}$ **ou d'un** $\begin{array}{c} I \quad I \\ \otimes \\ I \end{array}$ **:** les chemins de dépendance entre une des prémisses et une conclusion O sont tous remplacés par des chemins de dépendance entre la nouvelle conclusion I et la même conclusion O.

Lors de la création d'un $\begin{array}{c} O \quad O \\ \parr \\ O \end{array}$ **ou d'un** $\begin{array}{c} O \quad O \\ \otimes \\ O \end{array}$ **:** les chemins de dépendance entre une conclusion I et une des prémisses sont tous remplacés par des chemins de dépendance entre la même conclusion I et la nouvelle conclusion O.

Lors de la création d'un $\begin{array}{c} I \quad O \\ \parr \\ O \end{array}$ **:** s'il existe un chemin de dépendance entre la prémisse I et une conclusion O autre que le prémisse O, le réseau n'est pas **FILL**. Sinon, les chemins de dépendance entre une conclusion I et la prémisse de polarité O sont tous remplacés par des chemins de dépendance entre la même conclusion I et la nouvelle conclusion O ; les chemins de dépendance d'origine I sont supprimés.

Lors de la création d'un $\begin{array}{c} I \quad O \\ \otimes \\ I \end{array}$ **:** les chemins de dépendance entre une conclusion I et la prémisse O sont remplacés par des chemins de même origine mais allongés avec les

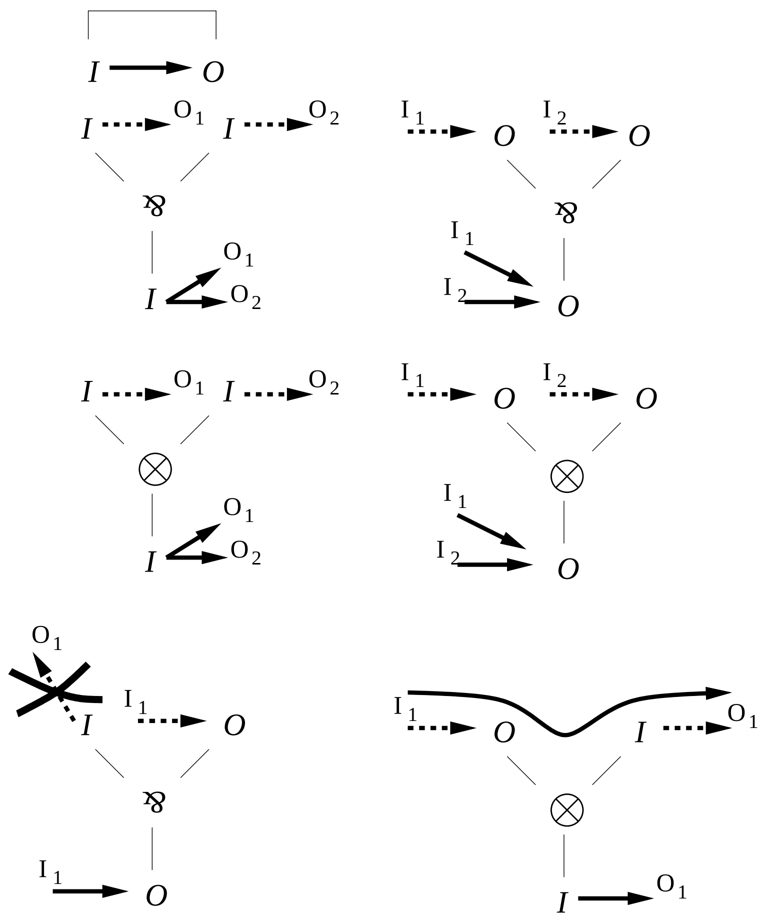


FIG. 3.4 – *Mise à jour des chemins de dépendance lors de la construction de réseaux* **FILL**. *Les flèches en pointillés sont remplacées par celles en trait plein.*

chemins partant de la prémisse I ; et les chemins de dépendance partant de la prémisse I sont remplacés par des chemins partant de la nouvelle conclusion I.

3.4.3 Exemple

Reprenons l'exemple de la figure 3.3. L'algorithme construit l'arbre de décomposition, puis crée le réseau en partant du haut. Il tente d'abord de créer un lien axiome entre un axiome I (qui correspond à $A^\perp$) et un axiome O (qui correspond à A). Il mémorise donc un chemin entre cette formule I et cette formule O. Supposons qu'il s'agit de l'axiome entre les formules 1 et 2. L'algorithme choisit de continuer sur le $\otimes$ reliés à une des formules déjà traitées. Pour pouvoir propager, il faut traiter l'autre prémisse. On crée donc le lien axiome $3-4$ et un chemin de 4 vers 3. On peut alors construire le $\otimes$. Les deux chemins mémorisés sont modifiés pour pointer vers ce $\otimes$. L'algorithme poursuit et construit ainsi tous les liens axiomes et les liens $\otimes$. Reste à construire le $\parr$.

Si l'algorithme commence par créer les liens axiomes comme sur le premier réseau (pas

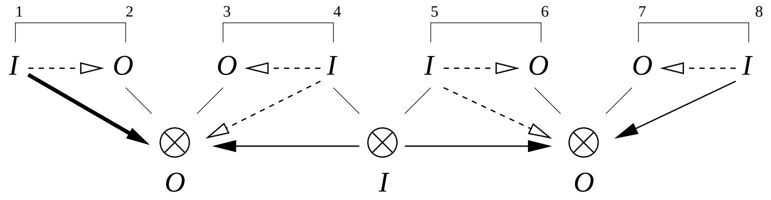


FIG. 3.5 – *Chemins de dépendances donnés par l'algorithme avant la création du lien* ⅋ *pour l'exemple de la figure 3.3.*

FILL) de la figure 3.3, la situation au moment de la création du lien ⅋ est celle donnée par la figure 3.5. Il est donc impossible de créer ce lien, à cause de la flèche représentée en gras. L'algorithme revient donc en arrière pour essayer d'autres positions des liens axiomes jusqu'à ce qu'il aboutisse à une solution qui lui permet de construire le lien ⅋.

Remarquons qu'en général, le nombre de possibilités pour les liens axiomes est assez limitée.

3.4.4 Correction de l'algorithme

Pour montrer la correction de l'algorithme obtenu, il faut montrer les propriétés 3.4.1 et 3.4.2.

Propriété 3.4.1 *L'algorithme mémorise tous les chemins de dépendance entre une conclusion I et une conclusion O, et uniquement ceux-ci.*

Démonstration : Il suffit de le vérifier pour chaque type de nœud. □

Propriété 3.4.2 *Considérons un nœud $A_I ⅋ B_O$. L'existence d'un chemin de dépendance entre A_I et une conclusion O du réseau différente de B_O lors de la création du nœud ⅋ équivaut à l'existence d'un chemin de dépendance entre ce nœud I et une conclusion O de l'empire de A dans le réseau final.*

Pour montrer ce résultat, nous utilisons le lemme suivant :

Lemme 3.4.3 *Soit R un réseau et A un nœud de polarité I de R. Il existe un chemin de dépendance entre A et une conclusion O de R.*

Démonstration : La propriété 3.2.1 indique qu'il existe toujours une conclusion O. Le lemme se démontre de manière similaire, par induction sur la construction du réseau. Le cas du lien axiome est trivial. Pour les liens ⅋ et $\otimes$, on peut par exemple distinguer le cas où le nœud I est le nouveau nœud introduit. Le cas le plus intéressant est celui du $\otimes$, lorsque l'hypothèse d'induction fournit un chemin entre un nœud I et l'unique prémisse O de ce $\otimes$. Il faut alors utiliser une deuxième fois l'hypothèse d'induction (sur l'autre réseau) et mettre les deux chemins bout-à-bout. □

Démonstration : *(propriété 3.4.2)* La démonstration nécessite des propriétés des réseaux qui ne seront pas détaillées ici (voir [21]). Elle est très similaire à la précédente, mais ne peut en être déduite immédiatement. Elle utilise le lemme 3.4.3 pour traiter le cas du $\otimes$ dont les prémisses n'ont pas la même orientation. □

3.5 Relation avec le calcul des séquents

L'algorithme de construction des réseaux **MLL** et son extension pour **FILL** présentent l'intérêt de fournir une méthode de construction des preuves en calcul des séquents. Il suffit construire l'arbre de preuve en partant du haut, au fur et à mesure de la construction du réseau. On a donc un algorithme pour construire des preuves en calcul des séquents **FILL** et non plus seulement un critère pour vérifier qu'une preuve est **FILL**. L'application de l'algorithme trouve la preuve **FILL** si elle existe ; on a donc maintenant la complétude qui manquait dans le chapitre 2.

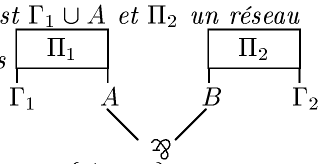
3.6 Extension avec la règle MIX

Il semble que des systèmes contenant la règle

$$\frac{\vdash \Gamma \quad \vdash \Delta}{\vdash \Gamma, \Delta}\ Mix$$

soient par exemple très adaptée pour modéliser des systèmes distribués [4]. Cela conduit a essayer d'adapter l'algorithme pour la construction de réseaux à **MLL** + **MIX** et à **FILL** + **MIX**. Pour cela, il suffit de remarquer que l'on peut donner une définition inductive des réseaux pour **MLL** + **MIX** comme on peu le faire pour les réseaux **MLL**, juste en ajoutant le cas suivant dans la définition 3.1.2 :

Si Π_1 est un réseau dont l'ensemble des conclusions est $\Gamma_1 \cup A$ et Π_2 un réseau disjoint de Π_1 dont les conclusions sont $\Gamma_2 \cup B$, alors



est un réseau dont l'ensemble des conclusions est $\Gamma_1 \cup \Gamma_2 \cup \{A \otimes B\}$.

Autrement dit, ajouter un nœud ⅋ ne nécessite plus que ses deux prémisses soient dans le même réseau. Il est facile de montrer l'équivalence de cette définition avec le critère d'acyclicité utilisé habituellement pour définir les réseaux **MLL** + **MIX**[3].

Conclusion

L'étude comparative des différents critères proposés nous montre que le lien entre **CLL** et **ILL** est plus fort que celui entre **CL** et **IL**, peut-être à cause du fait que **CLL** est déjà une logique constructive. Dans **CL**, toutes les règles permutent et on peut donc les appliquer dans l'ordre que l'on veut. L'exemple $A \vee B \vdash A \vee B$ montre que dans **IL**, l'ordre d'application des règles est important. Dans **CLL**, il en est de même ; plus précisément les règles de commutation montrent qu'il est toujours possible d'appliquer d'abord les règles $⅋$ et $\&$ et ensuite les règles $\otimes$ et $\oplus$, alors qu'une application dans un autre ordre peut entraîner un blocage [20]. Cette propriété impose déjà des restrictions sur les preuves, même pour **CLL**. La logique intuitionniste linéaire n'est donc pas caractérisée par un ordre d'application des règles différent comme c'est le cas pour **IL**.

La nature de l'intuitionnisme linéaire et son utilité sont plus difficiles à saisir à cause de cette proximité entre **CLL** et **FILL** (ou **ILL**). Cependant chacune de ses logiques est utilisée pour des applications particulières.

Enfin il serait intéressant de voir s'il est possible d'étendre ces travaux à **MALL**, voire même à **CLL**, en utilisant la définition des réseaux **MALL** de Girard [22].

Bibliographie

[1] S. Abramsky. Computational interpretations of Linear Logic. *Theoretical Computer Science*, 111(1-2):3–58, 1993.

[2] V. Alexiev. Applications of Linear Logic to Computation: An Overview. *Bulletin of the IGPL*, 2(1):77–107, 1994.

[3] A. Asperti. Causal dependencies in multiplicative linear logic with mix. *Math. Struct. in Comp. Science*, 5:351–380, 1995.

[4] A. Asperti and G. Dore. Yet another correctness criterion for multiplicative linear logic with mix. In *Logic at St Petersburg '94, Symposium on Logical Foundations of Computer Science, LNCS 813*, pages 34–46, St Petersburg, Russia, July 1994.

[5] G. Bellin. Subnets of proof-nets in multiplicative linear logic with mix. *Math. Struct. in Comp. Science*, 7:663–699, 1997.

[6] G. Bellin and P.J. Scott. On the π-calculus and linear logic. *Theoretical Computer Science*, 135(1):11–66, 1994.

[7] G. Bellin and J. van de Wiele. Subnets of proof-nets in MLL^-. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 249–260. Cambridge University Press, 1995.

[8] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In *Int. Conference on Typed Lambda Calculi and Applications, LNCS 664*, pages 75–90, Utrecht, The Netherlands, March 1993.

[9] G.M. Bierman. A note on full intuitionistic linear logic. *Annals of Pure and Applied Logic*, 79:281–287, 1996.

[10] G.M. Bierman. Towards a classical linear λ-calculus (preliminary report). *Electronic Notes in Theoretical Computer Science*, 3, 1996.

[11] T. Braüner and V. De Paiva. Cut elimination for Full Intuitionnistic Linear Logic. Unpublished draft, 1996.

[12] V. Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation (et principalement du λ-calcul)*. PhD thesis, Université de Paris VII, 1990.

[13] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57:795–807, 1992.

[14] D. Galmiche. Connection methods in Linear Logic and Proof nets Construction. *Theoretical Computer Science*, 1998/99. Accepted for publication.

[15] D. Galmiche and E. Boudinet. Proofs, concurrent objects and computations in a FILL framework. In *Workshop on Object-based Parallel and Distributed Computation, OBPDC'95, LNCS 1107*, pages 148–167, Tokyo, Japan, 1996.

[16] D. Galmiche and B. Martin. Algorithms for proof nets construction in multiplicative and additive linear logic. Technical report, CRIN-CNRS, 1996.

[17] D. Galmiche and B. Martin. Proof search and proof nets construction in linear logic. In *4th Workshop on Logic, Language, Information and Computation, Wollic'97*, Fortaleza, Brasil, August 1997. Logic Journal of IGPL, vol. 5-6, pp 883-885.

[18] D. Galmiche and B. Martin. Proof nets construction and automated deduction in non-commutative linear logic - extended abstract. In *CADE Workshop on Proof search in type-theoretic languages*, pages 51–66, Lindau, Germany, July 1998. to appear in Electonic Notes in Theoretical Computer Science, volume 17.

[19] D. Galmiche and G. Perrier. A procedure for automatic proof nets construction. In *LPAR'92, International Conference on Logic Programming and Automated Reasoning, LNAI 624*, pages 42–53, St. Petersburg, Russia, July 1992.

[20] D. Galmiche and G. Perrier. Foundations of proof search strategies design in linear logic. In *Logic at St Petersburg '94, Symposium on Logical Foundations of Computer Science, LNCS 813*, pages 101–113, St Petersburg, Russia, July 1994.

[21] J.Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[22] J.Y. Girard. Linear logic: its syntax and semantics. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995.

[23] J.Y. Girard. Proof nets: the parallel syntax for proof theory. In Ursini and Agliano, editors, *Logic and Algebra*, New York, 1995. M. Dekker.

[24] J.Y. Girard and Y. Lafont. Linear logic and lazy computation. In *TAPSOFT 87, LNCS 250*, pages 52–66, Pisa, Italia, March 1987. Springer Verlag.

[25] M. Hyland and V. de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, 64:273–291, 1993.

[26] P. Lincoln and J. Mitchell. Operational aspects of linear lambda calculus. In *7th IEEE Symposium on Logic in Computer Science*, pages 235–246, Santa-Cruz, California, 1992.

[27] D. Miller. Forum: A multiple-conclusion specification logic. *Theoretical Computer Science*, 165(1):201–232, 1996.

[28] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *LPAR'92, International Conference on Logic Programming and Automated Reasoning, LNAI 624*, pages 190–201, St. Petersburg, Russia, July 1992.

[29] E. Ritter, D. Pym, and L. Wallen. On the intuitionistic force of classical search. In *5th Int. Workshop, TABLEAUX'96*, pages 294–313, Terrasini, Palermo, Italy, May 1996.

[30] H. Schellinx. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1(4):537–559, 1991.

[31] L.A. Wallen. *Automated Proof search in Non-Classical Logics.* MIT Press, 1990.

Annexe A

Différents systèmes de calcul des séquents

A.1 Calcul des séquents pour CL

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ cut$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ Aff_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ Aff_R$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ Con_L \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ Con_R$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \to B) \vdash \Delta}\ \to_L \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash (A \to B), \Delta}\ \to_R$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \vee B) \vdash \Delta}\ \vee_L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash (A \vee B), \Delta}\ \vee_{R1} \quad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash (A \vee B), \Delta}\ \vee_{R2}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, (A \wedge B) \vdash \Delta}\ \wedge_L \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash (A \wedge B), \Delta}\ \wedge_R$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ \neg_L \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\ \neg_R$$

A.2 Calcul des séquents mono-conclusion pour **IL**

$$\frac{}{\Gamma, A \vdash A}\ ax \qquad \frac{\Gamma \vdash A \quad \Gamma', A \vdash B}{\Gamma, \Gamma' \vdash B}\ cut$$

$$\frac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, (A \to B) \vdash C}\ \to_L \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash (A \to B)}\ \to_R$$

$$\frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, (A \vee B) \vdash C}\ \vee_L \qquad \frac{\Gamma \vdash A}{\Gamma \vdash (A \vee B)}\ \vee_{R1} \quad \frac{\Gamma \vdash B}{\Gamma \vdash (A \vee B)}\ \vee_{R2}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, (A \wedge B) \vdash C}\ \wedge_L \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash (A \wedge B)}\ \wedge_R$$

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ Con \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ Aff \qquad \frac{}{\bot \vdash A}\ \bot_L$$

A.3 Calcul des séquents multi-conclusion pour **IL**

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ cut$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ Aff_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ Aff_R$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ Con_L \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ Con_R$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \to B) \vdash \Delta}\ \to_L \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash (A \to B), \Delta}\ \to_R$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \vee B) \vdash \Delta}\ \vee_L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash (A \vee B), \Delta}\ \vee_R \quad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash (A \vee B), \Delta}\ \vee_R$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, (A \wedge B) \vdash \Delta}\ \wedge_L \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash (A \wedge B), \Delta}\ \wedge_R$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ \neg_L \qquad \frac{\Gamma, A \vdash}{\Gamma \vdash \neg A, \Delta}\ \neg_R$$