Tableaux et déduction modulo

Richard Bonichon
Université Paris VI Pierre et Marie Curie

17 octobre 2006

Sommaire

Introduction

Logique et démonstration

Fiat lux !

Vers la fin du XIX$^{\text{ème}}$ siècle, des philosophes et mathématiciens ont ressenti la nécessité de définir la notion de démonstration de manière plus profonde que ce que la Grèce avait légué, notamment à travers la logique d'Aristote.

À l'époque[1], c'est en particulier la formalisation de l'arithmétique (et notamment la notion d'entier) qui motive les avancées de la logique. Par ailleurs, la (re)découverte de paradoxes par Burali-Forti et Russell ne fait que renforcer ce renouveau de la logique.

En 1879, Frege apporta le premier système logique formel grâce auquel on pouvait inférer des démonstrations rigoureuses. Après lui, Hilbert développa un système déductif comportant un certain nombre d'axiomes logiques admis (tel $A \Rightarrow B \Rightarrow A$) auxquels on pouvait rajouter d'autres axiomes propres à la théorie considérée. La preuve d'une proposition P dans ce système est faite au moyen d'axiomes, de propositions déjà prouvées et de deux règles dites de *modus ponens* et de *generalisation*.

L'axiomatisation forcée sous-jacente aux systèmes de Frege et de Hilbert poussa à rechercher un traitement plus « naturel » de la logique. Ce que Gentzen baptisa plus tard la *déduction naturelle* fut fondé par Jaskowski : ce système déductif, voulu plus intuitif, comporte certes un plus grand nombre de règles de déduction que ses prédécesseurs mais n'a pas d'axiomes logiques.

Le calcul des séquents

Nous sommes encore en 1929 lorsque la déduction naturelle apparaît. Quatre ans plus tard [2], Gentzen apporte une pierre fondamentale à la compréhension des preuves grâce aux calculs des séquents.

Dans son système de séquents, Gentzen peut travailler sur les hypothèses *et* les conclusions, alors que les systèmes prédécesseurs devaient trouver la conclusion en travaillant sur les hypothèses uniquement.

[1] Les articles fondateurs sont publiés sous forme de collection dans ([58])
[2] en 1933 donc

Le calcul de Gentzen possède aussi une règle spécifique dite de *coupure*, une notion mal définie jusque-là.

Cette règle correspond dans les preuves mathématiques « faites à la main « à l'utilisation dans le raisonnement de lemmes ou de théorèmes déjà démontrés par ailleurs.

Plus intéressant encore, Gentzen démontre comment cette règle peut toujours être *éliminée* d'une preuve. Tout formule démontrable est donc démontrable *sans détour*.

Le calcul des séquents possède en particulier la *propriété de la sous-formule*. Toute formule d'une preuve sans coupure est une sous-formule de celle que l'on veut démontrer.

Ces caractéristiques permettent de dire que le calcul des séquents est une avancée majeure pour ce qui sera plus tard la démonstration automatique. Il y a en effet dans le calcul des séquents tout le contenu syntaxique des méthodes de tableaux.

Démontrer automatiquement : les tableaux

Les motivations de Gentzen étaient orientées vers la théorie de la démonstration et l'analyse de la structure des preuves. D'autres travaux furent orientés par des motivations sémantiques. Les tableaux sont nés de ces préoccupations.

Beth

Beth introduit en 1955 la notion de tableau sémantique. Il définit sa méthode de tableau sémantique comme une tentative systématique de recherche de contre-exemple.

En fait, Beth arrange sa recherche de contre-exemple sous la forme d'un tableau (un vrai ! et non pas un arbre comme ce qui sera utilisé plus tard). Ce tableau a deux colonnes, nommées *valide* et *invalide*.

Pour déterminer si Y est une conséquence de $X_1, ..., X_n$, on commence par placer les X_i dans la colonne valide et Y dans l'autre. On commence donc avec la conjecture que Y n'est *pas* une conséquence des X_i, comme si on utilisait le calcul des séquents à l'envers. Si $P \wedge Q$ apparaît dans la colonne *valide*, on y ajoute P et Q. De la même manières si $A \vee B$ apparaît dans la colonne *invalide*, on ajoute A et B à celle-ci.

Mais si $A \vee B$ apparaît dans la colonne *valide* il faut pouvoir diviser en deux le tableau : en effet A *ou* B peut être valide. Donc on le divise en deux sous-tableaux, l'un avec A, l'autre avec B. Comme on peut diviser encore et encore le tableau, on a besoin d'étiqueter les différentes colonnes pour garder ensemble les colonnes *valide* et *invalide* qui doivent rester ensemble. En pratique, la représentation peut vite devenir confuse, même si le principe est assez clair.

Hintikka

En même temps que Beth, Hintikka est lui aussi concerné par des questions sémantiques. Pour lui, la recherche d'une preuve de X est en fait une tentative systématique de construire un modèle dans lequel la négation de X ($\neg X$) est vraie ; si cette tentative échoue, alors X est une tautologie.

Comme chez Beth, les preuves d'Hintikka déstructurent les formules pour arriver à leurs constituants de base. Hintikka apporte cependant un outil important dans les preuves de complétude de tableau : les ensembles d'Hintikka. Ces ensembles saturés simplifient ces preuves par abstraction des propriétés de satisfiabilité des formules en s'éloignant des détails du processus de construction d'un tableau.

Lis et Smullyan

Les méthodes de Beth et Hintikka apportent à Gentzen ce qui lui manquait pour avoir toutes les parties des tableaux modernes. Malgré tout, ces deux systèmes restaient peu *conviviaux*.

La simplification des notations était donc la prochain étape. Un peu comme dans le cas précédent, deux personnes sont arrivées indépendamment dans les années 1960 à la même conclusion.

Le malheur de Lis, qui devança Smullyan, fut de publier ses recherches en polonais et d'habiter de l'autre côté du rideau de fer. Par conséquent, ses idées ont été peu diffusées.

On retrouve ses idées dans le livre de Smullyan : *First-Order Logic*[3] [50]. C'est un point d'entrée classique en ce qui concerne la méthode des tableaux. Cet ouvrage permet de faire mieux connaître les tableaux sous leur forme moderne (voir chapitre 1).

Et puis ...

Après avoir atteint une forme stable pour la logique classique, les tableaux étaient prêts à s'étendre aux logiques « exotiques ».

La logique intuitionniste fut traitée en premier par Fitting. Ce n'est pas étonnant, car les calculs de séquents pour cette logique existaient depuis Gentzen (et Beth).

Ensuite des méthodes de tableaux ont été formulées pour les logiques multivaluées, modales, temporelles, ... ce qui démontre leur capacité d'adaptation.

Les tableaux semblent un formalisme de preuve plus naturel que d'autres comme la résolution (Robinson, 1965). Cependant, les méthodes fondées sur la résolution sont maintenant en avance au niveau pratique, comme en attestent les résultats de la compétition de prouveur automatique CASC [52].

[3] Lis y est bien cité par Smullyan

Raisonnement et calcul

La formalisation du raisonnement par la logique entraîne parfois des cas « problématiques ». Prenons l'axiome de l'arithmétique $S(x) + y = x + S(y)$. Aucun humain ne répondra $21 + 21$ si on lui donne $22 + 20$ comme énoncé[4]. L'abstraction par les logiciens a mené à l'« oubli » de la solution plus évidente du calcul.

L'apparition des ordinateurs dans la deuxième moitié du siècle dernier change un peu la donne. En effet, un ordinateur n'est en essence qu'une (puissante) machine à calculer [5]. Cependant, les premiers programmes écrits pour démontrer des théorèmes automatiquement firent comprendre qu'il ne fallait pas essayer de prouver par déduction des problèmes tels que $2 + 2 = 4$

Si l'on essaie par exemple de prouver par le raisonnement la proposition

$$(a + b) + ((c + d) + e) = a + ((b + c) + (d + e))$$

à l'aide des axiomes d'associativité

$$\forall xyz\ ((x + y) + z = x + (y + z))$$

et d'identité

$$\forall x\ x = x$$

alors, un programme naïf peut arranger infiniment les parenthèses sans jamais trouver la conclusion. Ce problème semble pourtant facile à un humain car il va tenter de démontrer ce résultat par le calcul et non par la déduction. La distinction entre ces deux façons de démontrer semblent donc être d'un intérêt particulier pour la résolution de problèmes mathématiques. En effet le calcul est une exécution dirigée alors que la déduction est par nature non déterministe.

Le calcul peut être modelisé en utilisant une congruence sur les propositions de notre langage : dans l'exemple ci-dessus tout arrangement de parenthèses conduit à une proposition équivalente. Ces congruences peuvent être souvent exprimées par des systèmes de réécriture confluents qui terminent.

L'axiome d'associativité peut être supprimé et orienté

$$(x + y) + z \to x + (y + z).$$

Alors la preuve de

$$\neg(a + b) + ((c + d) + e) = a + ((b + c) + (d + e))$$

se réécrit

$$\neg((a + (b + (c + (d + e)))) = a + (b + (c + (d + e))))$$

qui peut se réfuter directement avec $x = x$.

[4] La plupart devraient en effet répondre 42...

[5] ces machines à calculer programmables sont d'ailleurs fondées sur la *logique* booléenne.

La congruence peut aussi comporter des règles de réécriture sur les propositions. Par exemple l'axiome des domaines intègres

$$\forall xy\ (x * y = 0) \Leftrightarrow (x = 0 \vee y = 0)$$

peut s'exprimer sous forme de règle de réécriture

$$x * y = 0 \rightarrow x = 0 \vee y = 0$$

Ensuite il faut pouvoir intégrer ces règles de réécriture dans les systèmes de déduction. L'une des façons de réaliser cela a abouti à la déduction modulo [30].

La déduction modulo intègre effectivement les règles de réécriture directement dans le formalisme du système déductif au lieu de les présenter comme un ensemble à part. Cela permet en particulier d'avoir des preuves compactes et plus lisibles qui se rapprochent des preuves faites à la main.

La caractéristique majeure de la déduction modulo est de pouvoir réécrire des termes (tels $2+2$) mais aussi des formules comme « x fois y égal zéro ». Cela permet d'exprimer des théories puissantes comme la logique d'ordre supérieur dans une formulation au premier ordre sans axiomes.

Le calcul des séquents modulo est une pierre angulaire de l'édifice. Il paraît donc normal de s'intéresser aux méthodes de tableaux, au vu du lien qui les unit aux séquents.

La rencontre des tableaux avec la déduction modulo permet d'obtenir des méthodes de preuves automatiques intégrant les théories exprimées sous forme de règle de réécriture sur les termes et les propositions. Cela permet d'étudier les questions théoriques usuelles relatives aux tableaux mais dans le cadre de la déduction modulo : correction, complétude, sémantique, élimination des coupures.

Plan de la thèse

Dans un premier chapitre, nous étudierons les évolutions des méthodes de tableaux pour la logique classique depuis Smullyan. Nous commencerons par rappeler le formalisme dans le cadre propositionnel avant de passer à la logique du premier ordre. Dans ce cadre, nous verrons les améliorations proposées de la méthode des tableaux : métavariables et skolemisation. Enfin, nous verrons les bases des tableaux intuitionnistes.

Le deuxième chapitre s'attarde sur le prédicat d'égalité. Nous présenterons les différentes méthodes permettant de traiter l'égalité dans les tableaux. Nous verrons dans ce chapitre que si le problème purement théorique du traitement de l'égalité reste assez simple, les choses se complexifient dès lors qu'on se préoccupe d'efficacité (pratique).

Le chapitre 3 traite tout d'abord de la théorie de la déduction modulo. Nous y verrons en particulier le calcul des séquents modulo. Ensuite, nous présenterons certains résultats : des plus établis comme la logique d'ordre supérieur en déduction modulo ou la méthode de résolution ENAR aux plus récents comme l'élimination

des coupures sémantique, l'élimination des coupures par complétion abstraite ou la théorie des ensembles de Zermelo modulo.

Dans le chapitre 4, nous présentons les premiers résultats sur les tableaux modulo (méthode TaMed). Nous donnons une présentation analytique des tableaux modulo avec d'une part les règles d'expansion des tableaux métavariables et d'autre part les règles de fermeture et de narrowing étendues pour la déduction modulo. Nous établissons la correction et la complétude des tableaux modulo par des preuves syntaxiques.

Le chapitre 5 permet d'introduire les méthodes sémantiques développées pour la déduction modulo. Nous reformulons tout d'abord les tableaux modulo. Ensuite, nous développons une preuve de complétude en étendant pour la déduction modulo les modèles booléens usuels. Ces preuves permettent de dégager des classes de systèmes de réécriture pour lesquels la propriété de complétude des tableaux est établie.

Dans le chapitre 6, nous abordons la logique intuitionniste modulo. Le résultat principal de ce chapitre est l'obtention d'une preuve constructive d'élimination des coupures pour la logique intuitionniste modulo. Cette preuve nécessite de formuler une méthode de tableaux intuitionniste modulo. Nous démontrons sa complétude de manière sémantique sous certaines conditions sur les systèmes de réécriture considérés. Puis nous montrons de façon syntaxique que les tableaux (modulo) correspondent effectivement à une recherche de preuve sans coupure dans le calcul des séquents (modulo) sous-jacent. Nous pouvons alors extraire de ces preuves un théorème constructif d'élimination des coupures.

Enfin le chapitre 7 décrit le prouveur Zenon du projet Focal. Nous décrivons les règles de ce démonstrateur automatique fondé sur les tableaux. Nous évoquons la traduction des preuves faites dans le langage de preuves hiérarchique Cutter vers le langage intermédiaire puis le langage bas-niveau de Zenon pour arriver à leur traduction vers Coq. Enfin, nous montrons comment certains aspect de Zenon correspondent à l'intégration d'une certaine forme de déduction modulo.

Chapitre 1

La méthode des tableaux

1.1 Qu'est-ce qu'un tableau ?

La réponse à cette question n'est pas — évidemment — le sujet de cette thèse mais elle est nécessaire pour la comprendre. Tout ce premier chapitre (et même une partie des suivants) peut être considéré comme une réponse longuement étayée par une approche par cas. Il faut quand même essayer de donner dès maintenant une première formulation de réponse. On peut tenter de donner une réponse en deux temps :

- tout d'abord, décrire en langue naturelle les différentes caractéristiques possibles d'une méthode de tableaux ;
- puis compléter en illustrant par un exemple pour rendre plus tangible la nature des tableaux.

Nous allons prendre le temps de voir ces deux aspects complémentaires. L'exemple permettra non seulement de donner l'intuition derrière la méthode mais aussi d'introduire certaines des notations utilisées par la suite.

Plusieurs définitions sont possibles

Avant tout, les tableaux (ou les méthodes de tableaux) regroupent une famille de méthodes de recherche de preuve automatisée. Les tableaux (ou encore les méthodes de tableaux) sont majoritairement présentés en tant que *méthodes de réfutation*. Dans [35] les tableaux pour la logique classique du premier ordre sont décrits de manière duale en tant que méthode de recherche de preuve directe, c'est-à-dire qu'au lieu de chercher à réfuter une formule donnée en entrée, on cherche à démontrer qu'elle est une tautologie.

La méthode des tableaux fonctionne en première approximation comme une méthode de décomposition syntaxique appliquée sur la formule à réfuter. Celle-ci est alors décomposée en sous-cas que l'on va devoir réfuter un à un. On parlera de *tableaux analytiques* lorsque ce point de vue sera mis en avant.

Cependant, des arguments sémantiques justifient cette décomposition syntaxique, comme nous allons le voir dès l'exemple suivant. Les tableaux sont donc

également assez souvent présentés comme des méthodes sémantiques, qui seront nommées *tableaux sémantiques*. La recherche d'une réfutation permet en effet de construire un contre-modèle de la formule à réfuter lorsqu'une branche de l'arbre ne peut être fermée.

D'un point de vue purement structurel, une méthode de tableau est souvent présentée comme la construction d'un arbre à partir d'une formule logique initiale. Néanmoins, les méthodes de tableaux peuvent également être représentées à l'aide de multi-ensembles de formules.

Selon la logique considérée, certaines de ces caractéristiques ne sont pas valables ou pas immédiatement visibles. Dans le cadre de la logique classique, elles sont toutes valables ce qui permet de parler pour le même objet de tableau analytique ou sémantique. Par ailleurs, la logique classique permet sans doute de visualiser de manière plus aisée la relation entre les tableaux et le calcul des séquents usuel.

Un exemple

Nous allons introduire la méthode des tableaux et les notations utilisées par un exemple tiré de [31]. Il s'agit de prouver la proposition $\mathcal{P}$ suivante :

$$\mathcal{P} = (P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q)$$

La méthode des tableaux est, comme nous l'avons déjà mentionné, principalement utilisée comme méthode de réfutation. La première étape de notre preuve consiste donc à prendre la négation de la proposition $\mathcal{P}$ à prouver : nous allons en fait *réfuter* $\neg\mathcal{P}$.

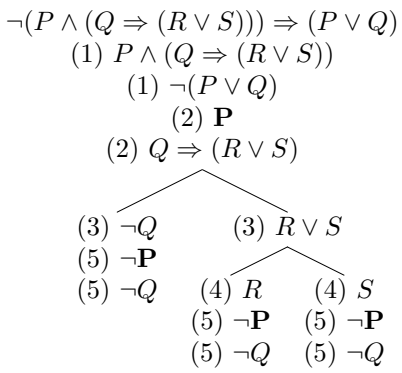
Maintenant nous allons déstructurer pas à pas $\mathcal{P}$ de façon logique en sous-formules en nous basant sur la sémantique de la logique propositionnelle pour expliquer la démarche. On choisira dans cet exemple de décomposer la formule avec le plus grand nombre de connecteurs en premier, et en cas d'égalité au niveau de la taille, on choisit la proposition la plus jeune (celle qui a été produite en dernier).

1. Si $(P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q)$ est invalide, alors il faut que $P \wedge (Q \Rightarrow (R \vee S))$ soit valide *et* $(P \vee Q)$ soit invalide.
2. Si $P \wedge (Q \Rightarrow (R \vee S))$ est valide alors il faut que P soit valide *et* que $(Q \Rightarrow (R \vee S)$ soit valide.
3. Si $(Q \Rightarrow (R \vee S)$ est valide alors il faut que Q soit invalide *ou* que $R \vee S$ soit valide.
4. Maintenant, si $R \vee S$ est valide, il faut que R *ou* S soit valide.
5. Finalement, si $(P \vee Q)$ est invalide, alors il faut que P *et* Q soient toutes deux invalides.

À la fin de cette décomposition, il ne reste plus que des propositions littérales qui ne peuvent par définition pas être décomposées plus avant.

La seule description en langue naturelle de la procédure de tableau ne facilite pas la dernière phase de notre réfutation, c'est-à-dire *montrer que les conditions*

sémantiques énoncées ne peuvent en aucun cas être remplies. Représentons donc la procédure que l'on vient d'accomplir sous forme d'arbre. Les conditions qui doivent être simultanément vérifiées (là où il y a un *et*) seront sur le même chemin, et les conditions alternatives (là où il y a un *ou*) diviseront l'arbre en deux branches. Dans la description graphique, le chiffre correspondant à l'expansion de la description verbale dont est issue la proposition est indiqué entre parenthèses.



Les chemins possibles sont très clairement représentés de cette manière. Il ne reste maintenant plus qu'à passer à l'étape de *fermeture* de tableau pour décider si la proposition initiale est fausse ou non. Si on rencontre sur un chemin de la racine à une feuille (ce chemin est appelé *branche*) une formule et sa négation, on déclare ce chemin *fermé*. Si tous les chemins d'un tableau sont fermés alors le tableau est également dit fermé[1]. Si un tableau est fermé alors on sait que la proposition initiale ne peut être fausse (c'est donc que c'est une tautologie). Dans le cas d'un tableau ouvert, chaque branche modélise un contre-exemple où la proposition initiale est fausse. Dans les deux cas, la procédure est terminée.

Revenons à l'exemple :

- sur le chemin le plus à gauche, il est dit que l'on doit avoir à la fois P valide et P invalide, on a donc une contradiction ;
- cette même contradiction est présente sur les deux autres branches ;
- par conséquent, il n'existe pas de modèle où notre proposition initiale soit fausse et nous l'avons prouvée formellement.

Les règles que nous avons utilisées implicitement au niveau sémantique peuvent aussi s'interpréter de manière strictement syntaxique (cf. section 1.2). Dans cette optique, on dira plutôt que nous avons prouvé la proposition initiale en utilisant les règles de décomposition (syntaxique) de tableau.

La même procédure peut aussi être représentée de manière équivalente comme une succession de tableaux inférés à partir de la proposition initiale : cette représentation manipule des multi-ensembles de propositions (branches). Dans cette représentation, qui en fait infère un nouveau tableau à partir du tableau prémisse, notre exemple devient :

[1] On réservera la terminologie tableau *clos*, aussi utilisée pour cela, pour un autre cas

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg(P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q)}{P \wedge (Q \Rightarrow (R \vee S)), \neg(P \vee Q)}\,(1)}{P, (Q \Rightarrow (R \vee S)), \neg(P \vee Q)}\,(2)}{P, \neg Q, \neg(P \vee Q) \mid P, R \vee S, \neg(P \vee Q)}\,(3)}{P, \neg Q, \neg(P \vee Q) \mid P, R, \neg(P \vee Q) \mid P, S, \neg(P \vee Q)}\,(4)}{\mathbf{P}, \neg Q, \neg\mathbf{P}, \neg Q \mid \mathbf{P}, R, \neg\mathbf{P}, \neg Q \mid \mathbf{P}, S, \neg\mathbf{P}, \neg Q}\,(5)}{\odot}$$

où $\odot$ est le tableau fermé. Les branches sont des multi-ensembles de propositions. Le séparateur de branche est $\mid$ et celui des propositions est la virgule (,). On peut remarquer que cette fois-ci les propositions qui sont décomposées sont explicitement détruites (car ce n'est pas nécessaire de les garder dans le raisonnement). On peut reproduire cette destruction de propositions en annotant la première notation : par exemple $P^\star$ peut vouloir dire que l'on a déjà utilisé P et que ce n'est plus nécessaire de pouvoir à nouveau l'utiliser.

Si la représentation sous forme d'arbre est la plus claire au niveau graphique, elle est aussi moins proche du calcul des séquents sous-jacent, comme nous le verrons par la suite. Elle sera néanmoins préférée dans les preuves sous forme de tableaux que nous allons présenter.

1.2 Tableaux classiques propositionnels

Cette section introduit les notations et définitions utilisées dans la suite de cette thèse en présentant les tableaux pour la logique classique propositionnelle. Nous ne rappellerons ni la syntaxe ni la sémantique de la logique propositionnelles qui peuvent être trouvées dans [35].

1.2.1 Règles

Les règles des tableaux pour la logique classique propositionnelle sont traditionnellement divisées en deux types : les règles d'*expansion* et les règles de *fermeture*. Les règles d'expansion sont elles-mêmes subdivisées en plusieurs catégories dénommées α et β-règles qui permettent de classer les propositions selon leur connecteur logique principal comme suit :

$$\begin{array}{c} \alpha \\ \hline P \wedge Q \\ \neg(P \vee Q) \\ \neg(P \Rightarrow Q) \end{array} \qquad \begin{array}{c} \beta \\ \hline P \vee Q \\ \neg(P \wedge Q) \\ P \Rightarrow Q \end{array}$$

Une fois ces classes définies, on peut donner de façon condensée les trois règles constituant les tableaux propositionnels (figure 1.1).

La règle $\odot$ de cette figure 1.1 — dite de *fermeture* — permet de conclure lorsqu'on a effectivement trouvé une réfutation sur une branche. Il faut pouvoir l'appliquer à toutes les branches de notre tableau pour pouvoir le déclarer fermé.

$$\frac{\alpha(P,Q)}{P,Q}\,\alpha \qquad \frac{\beta(P,Q)}{P \mid Q}\,\beta \qquad \frac{P,\neg P}{\odot}\,\odot$$

FIG. 1.1 – Règles d'expansion pour les tableaux propositionnels

Le but des tableaux, comme de toute méthode de preuve, est de terminer le plus vite possible, avec le moins de travail possible. La règle de fermeture ($\odot$) ne s'applique donc pas uniquement à des propositions littérales : dans ce cas P est une proposition de taille arbitraire. Si la fermeture de branches était restreinte aux seuls littéraux, la preuve d'une formule de la forme $A \vee \neg A$, avec A contenant autant de connecteurs que souhaité, serait réellement peu performante.

1.2.2 Lien avec le calcul des séquents

Il existe un lien fort entre le calcul des séquents et les tableaux. La méthode des tableaux automatise le calcul des séquents en utilisant les propriétés de celui-ci (celle de la sous-formule notamment). Les tableaux sont en effet la formalisation de la méthode « naturelle » qui, d'un séquent-conclusion donné, permet d'obtenir la preuve correspondante lorsqu'elle existe : en effet, dans ce cas, on écrit tout d'abord la conclusion puis on remonte petit-à-petit vers les axiomes. Ainsi la preuve (légèrement raccourcie lorsque des affaiblissements sont effectués) en calcul des séquents propositionnels de la proposition de la section introductive 1.1 sera la suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash P}\,\text{ax}}{P \vdash P, Q}\,\text{aff}}{P \vdash P \vee Q, Q}\,\vee\text{-r} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash P}\,\text{ax}}{P, R \vdash P, Q}\,\text{aff}}{P, R \vdash P \vee Q}\,\vee\text{-r} \qquad \dfrac{\dfrac{\dfrac{}{P \vdash P}\,\text{ax}}{P, S \vdash P, Q}\,\text{aff}}{P, S \vdash P \vee Q}\,\vee\text{-r}}{P, R \vee S \vdash P \vee Q}\,\vee\text{-l}}{P, Q \Rightarrow (R \vee S) \vdash P \vee Q}\,\Rightarrow\text{-l}}{P \wedge (Q \Rightarrow (R \vee S)) \vdash P \vee Q}\,\wedge\text{-l}}{\vdash (P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q)}$$

Le lien avec la preuve de tableau donnée en section 1.1 est le suivant :
- les propositions précédées d'une négation $\neg$ dans le tableau sont à droite du signe $\vdash$ dans le séquent (car on travaille en réfutation dans le tableau) ;
- la création de branches, séparées par $\mid$, correspondent à l'utilisation de la règle $\vee - l$ ou $\Rightarrow - l$;
- l'extension de branche est simplement l'utilisation des règles duales $\wedge - l$ et $\vee - r$;
- la règle de fermeture, enfin, est le pendant de la règle axiome.

1.2.3 Exemple d'une heuristique simple

Dès le cas propositionnel se pose le problème de savoir s'il existe un ordre d'application des règles de tableau qui permet de trouver plus rapidement une

preuve — tout en conservant la propriété de complétude bien sûr. On peut se convaincre assez rapidement que n'importe quelle hiérarchisation des règles α et β sera complète (car toute stratégie est équitable à partir du moment où elle termine). Or, on sait que fermer un tableau nécessite de fermer toutes ses branches, donc si possible, il faut minimiser le nombre de branches à fermer à chaque instant : par conséquent on donnera la priorité à l'application de α qui allonge la branche courante sur l'application de β qui crée une branche de plus dans le tableau. Notons que la règle de fermeture est bien évidemment prioritaire sur toutes les autres : plus on ferme tôt les branches, moins il y a de travail à faire pour fermer le tableau.

En utilisant prioritairement α par rapport à β sur l'exemple de la section 1.1, on obtient la preuve suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q)}{P \wedge (Q \Rightarrow (R \vee S)), \neg(P \vee Q)}\,\alpha}{P, (Q \Rightarrow (R \vee S)), \neg(P \vee Q)}\,\alpha}{P, (Q \Rightarrow (R \vee S)), \neg P, \neg Q}\,\alpha}{\odot}\,\odot$$

Au lieu des 10 applications de règles que l'on avait à la section 1.1, on en a 4 ici en suivant une hiérarchisation des règles relativement simple.

1.2.4 Propriétés

Une méthode de preuve cherche fondamentalement à respecter deux critères : celui de *correction* et celui de *complétude*. La *correction* permet de s'assurer que la méthode ne repose que sur des éléments corrects. La *complétude*, quant à elle, établit le fait que l'on arrivera "toujours" à trouver une preuve si elle existe. Dans les deux cas, on peut choisir de démontrer les propriétés selon un axe *syntaxique* ou *sémantique*.

Dans les preuves syntaxiques, la réfutation de tableau pour $\Gamma \wedge \neg\Delta$ correspond à une preuve sans coupure du séquent $\Gamma \vdash \Delta$.

Theorème 1 (Complétude syntaxique)**.** *Si $\Gamma \vdash \Delta$ dans le calcul des séquents alors on sait produire pas-à-pas un tableau fermé équivalent pour la proposition $\Gamma \wedge \neg\Delta$.*

Démonstration. La preuve s'appuie sur la correspondance exacte entre le calcul des séquents propositionnel et les tableaux. À l'aide de l'exemple, on peut avoir l'intuition qu'il s'agit en fin de compte de retourner la preuve des séquents (les prémisses seront en bas de la preuve) pour avoir un tableau, à quelques notations syntaxiques près. □

Theorème 2 (Complétude sémantique)**.** *Si $\Gamma \wedge \neg\Delta$ est insatisfiable, alors $\Gamma \wedge \neg\Delta$ produit un tableau fermé.*

Démonstration. Il faut montrer que si un tableau pour une certaine proposition ne peut être fermé, alors on peut exhiber d'une des branches ouvertes un modèle (booléen) dans lequel cette proposition est effectivement valide.

□

Theorème 3 (Correction syntaxique)**.** *Si le tableau* $\Gamma \wedge \neg\Delta$ *est fermé alors on peut produire pas-à-pas une preuve équivalente de* $\Gamma \vdash \Delta$ *suivant la preuve donnée par le tableau.*

Démonstration. Là encore, la correspondance entre le calcul des séquents et les tableaux permet d'avoir une preuve relativement directe. Lorsque le tableau pour $\Gamma \vdash \Delta$ est fermé alors on a une preuve de $\vdash \neg(\Gamma \wedge \neg\Delta)$ soit $\Gamma \vdash \Delta$. □

Theorème 4 (Correction sémantique)**.** *Si* $\Gamma \wedge \neg\Delta$ *produit un tableau fermé, alors la formule* $\Gamma \wedge \neg\Delta$ *est insatisfiable.*

Démonstration. Il faut s'assurer que pour toute formule satisfiable en entrée du tableau, et quelles que soient les règles d'expansion appliquées, la satisfiabilité n'est pas changée. □

Les preuves sémantiques sont celles qui sont le plus souvent utilisées. On peut ainsi trouver des preuves sémantiques détaillées dans les ouvrages de Smullyan ([50]) et de Fitting ([31]), concernant aussi bien les tableaux propositionnels que les tableaux pour la logique du premier ordre que nous allons voir plus tard en section 1.3.

1.2.5 Contenu sémantique d'un tableau ouvert

Jusqu'à présent, les formules utilisées dans les exemples produisent des tableaux fermés. Mais que se passe-t-il lorsqu'on n'arrive pas à clore le tableau produit ? Arrive-t-on à extraire une information utile de cet échec relatif ? La réponse est oui car la partie sémantique des tableaux permet d'obtenir un modèle dans lequel la proposition en entrée du tableau est satisfiable. Celle-ci est, ainsi que nous l'avons déjà dit, la négation de la proposition que l'on avait initialement donc on obtient un contre-modèle de cette proposition initiale (un modèle dans lequel elle est invalide).

Prenons comme exemple la proposition empruntée à [50] suivante :

$$(P \vee Q) \Rightarrow (P \wedge Q)$$

Dans la figure 1.2, le tableau correspondant à la négation de cette formule est construit.

Le tableau de la figure 1.2 a deux branches ouvertes. Si l'on considère la branche ouverte dont la dernière proposition est (7), alors dire que P est vraie et Q est fausse permet de donner une interprétation qui satisfait toutes les propositions de cette branche. De la même manière, on peut trouver une autre interprétation pour la branche ouverte terminant par (8) où l'on aurait Q vraie et P fausse. Dans l'une ou l'autre de ces interprétations la proposition (1) est

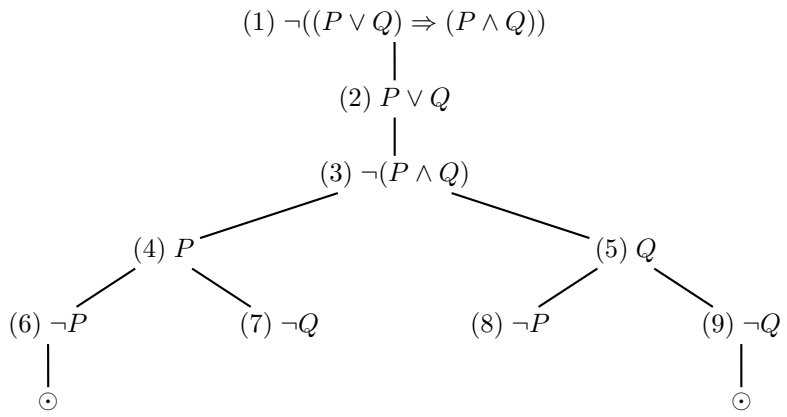


FIG. 1.2 – Tableau ouvert pour $(P \vee Q) \Rightarrow (P \wedge Q)$

vraie donc $(P \vee Q) \Rightarrow (P \wedge Q)$ est vraie. Autrement dit nous avons construit un contre-exemple à $(P \vee Q) \Rightarrow (P \wedge Q)$.

1.3 Tableaux classiques du premier ordre

Dans cette partie, nous allons présenter comment la méthode des tableaux est étendue à la logique du premier ordre. Les règles avancées de traitement des quantificateurs sont également détaillées, tout spécialement le cas du quantificateur existentiel.

1.3.1 Règles

Les tableaux pour la logique classique du premier ordre étendent les tableaux propositionnels. Ainsi, deux règles prenant en charge les quantificateurs universel et existentiel vont venir s'ajouter aux règles déjà définies précédemment pour les connecteurs logiques. Dès lors, les formules possibles sont classées en quatre groupes :

α	β	γ	δ
$P \wedge Q$	$P \vee Q$	$\forall x\ P(x)$	$\exists x\ P(x)$
$\neg(P \vee Q)$	$\neg(P \wedge Q)$	$\neg(\exists x\ P(x))$	$\neg(\forall x\ P(x))$
$\neg(P \Rightarrow Q)$	$P \Rightarrow Q$		

À partir de ce classement de formules on peut énoncer une première version des tableaux pour la logique du premier ordre en figure 1.3.

Les règles de la figure 1.3 décrivent les tableaux du premier ordre tels que formulés par Smullyan dans [50]. La règle δ utilise une « skolemisation » simplissime : on substitue une constante fraîche (par rappport à l'ensemble du tableau déjà construit) à la variable existentiellement quantifiée. L'utilisation de ces règles est illustrée sur un exemple en figure 1.4.

$$\frac{\alpha(P,Q)}{P,Q}\,\alpha \qquad \frac{\beta(P,Q)}{P \mid Q}\,1\beta \qquad \frac{P,\neg P}{\odot}\,\odot$$

$$\frac{\exists x\ P}{P[x := c]}\,\delta \text{ où } c \text{ est une constante fraîche}$$

$$\frac{\forall x\ P}{P[x := t]}\,\gamma \text{ où } t \text{ est un terme clos}$$

FIG. 1.3 – Règles des tableaux pour la logique du premier ordre

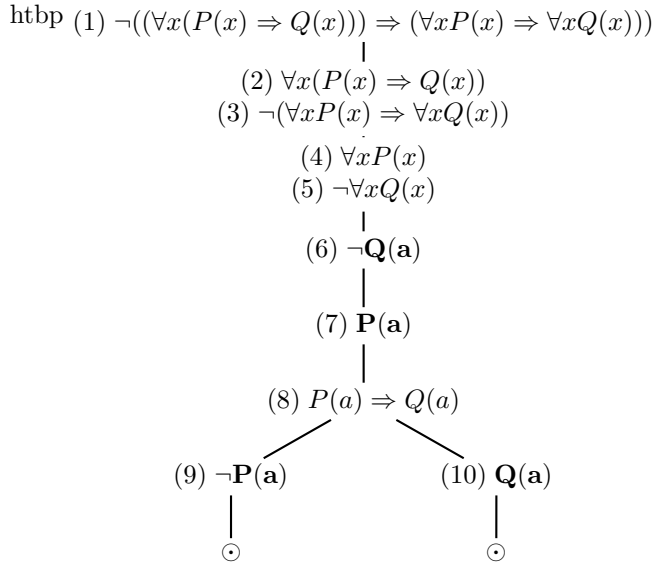


FIG. 1.4 – Un tableau clos du premier ordre pour $(\forall x(P(x) \Rightarrow Q(x))) \Rightarrow (\forall xP(x) \Rightarrow \forall xQ(x))$

Les tableaux clos montrent vite certains désavantages (auxquels on peut et va remédier) : ils sont « uniquement » l'automatisation du calcul des séquents de Gentzen dans lequel les prémisses et conséquences des règles d'inférence ont été inversées.

Dans l'exemple de la figure 1.4, la preuve escomptée est facilement obtenue : cela vient d'une combinaison d'un bon ordonnancement dans l'application des règles en prenant γ comme la moins prioritaire de toutes et d'une utilisation implicite du contexte du tableau dans l'application de ces mêmes γ-règles. Ce n'est pas par hasard que dans les formules (7) et (8) de cet exemple, l'instanciation a été faite par le terme a par ailleurs présent dans la branche. Mais comment faire le bon choix lorsque l'on a plus d'un terme clos qui semble adapté ?

En fait, Les règles γ et δ ne sont pas optimales car elles génèrent des instanciations en aveugle pour les variables liées aux quantificateurs. Les sous-sections suivantes examinent les améliorations apportées sur ces deux points.

1.3.2 Introduction de métavariables

La difficulté pratique de la méthode des tableaux provient de la γ règle. Elle est très permissive car on peut ajouter à une branche contenant une γ-formule n'importe quelle formule obtenue en substituant un terme clos à la variable universellement quantifiée. Dès lors la question qui se pose est la suivante : comment s'assurer que l'on a fait le bon choix d'instanciation ? Bien sûr, il est tout-à-fait possible d'énumérer tous les choix (possiblement infinis) de façon systématique. C'est simplement très inefficace.

Une meilleure solution au niveau de l'automatisation de preuve est de remplacer la γ formule par une formule $\gamma(X)$ avec X une nouvelle *métavariable* fraîche pour tout le tableau. L'idée est clairement développée dans [31] mais on retrouve déjà les fondements de ce principe chez Smullyan ([50] [2]). Ces métavariables sont plus communément appelées *variables libres* [31], même si cela n'a pas grand sens car elles ne sont pas de même nature. Elles sont parfois aussi nommées « *placeholders* » ou *dummies*. Elles sous-entendent par ailleurs une re-spécification (une extension par exemple) de la syntaxe utilisée dans les preuves de tableaux car elles n'existent pas telles quelles dans la syntaxe de la logique du premier ordre. On peut toujours construire des termes avec ces métavariables mais elles ne sont pas quantifiables.

Le rôle des métavariables est fondamentalement d'attendre que le contexte dans lequel se déroule la preuve fournisse suffisamment d'informations pour pouvoir tenter une instanciation raisonnée (le contraire des instanciations à l'aveugle des séquents) de notre variable universelle. En pratique, cela retarde l'instanciation des variables universellement quantifiées jusqu'à l'application de la règle de fermeture de branche ($\odot$) où l'on résoudra le problème d'instanciation par unification. Notez pour l'instant qu'il est nécessaire de garder la formule

[2] (pp.55) Suppose (...) we prove a sentence $\forall x P(x)(...)$. Then we conclude P(a). *We have not really commited "a" to being the name of any particular individual* ; $P(a)$ holds for every value of a

originelle pour éventuellement réappliquer la règle plus tard, sinon la complétude du calcul n'est plus assurée (ceci est discuté en section 1.3.3).

$$\frac{\forall x\ P}{P[x := X]}\ \gamma$$

où X est une métavariable fraîche pour le tableau

FIG. 1.5 – γ-règle pour les tableaux à métavariables

Ce traitement des formules universellement quantifiées est décrit pour la première fois dans [31]. L'introduction de métavariables dans le processus des tableaux (section 1.3.2) a également une conséquence sur les δ-règles. Avec le délai introduit dans l'instanciation des termes dans les γ-règles, comment être sûr que les paramètres utilisés dans les δ-règles sont effectivement frais ?

La solution à ce problème vient de l'introduction de techniques de skolemisation dynamique dans la règle δ (figure 1.6) afin d'éliminer les quantificateurs existentiels des formules. Les arguments du symbole de Skolem sont en fait des métavariables dont on attend l'instanciation. Le symbole de fonction généré doit être nouveau pour préserver la correction de la règle[3].

$$\frac{\exists x\ P}{P[x := f(y_1, ..., y_n)]}\ \delta$$

– f est un symbole de Skolem frais globalement pour tout le tableau ,
– $y_1, ..., y_n$ sont les variables libres apparaissant sur la branche de la formule $\exists x\ P$.

FIG. 1.6 – δ-règle (Fitting)

Exemple avec métavariables

Voyons sur un exemple comment les métavariables améliorent le traitement des quantificateurs universels et prouvons cette fois-ci la formule Ψ suivante, tirée de [45] :

$$\Psi = \exists x((\forall y \forall z P(y, f(x, y, z))) \Rightarrow (\forall y P(y, f(x, y, x) \wedge \forall y \exists z P(g(y), z))))$$

Il apparaît en regardant la figure 1.7 qu'il aurait fallu une chance assez extraordinaire pour trouver aussi directement une solution à ce tableau en utilisant les tableaux clos de la section 1.3.1. Cependant, s'il a été assez facile cette fois-ci de trouver comment instancier les métavariables introduites, quel est le processus formel qui a été utilisé ? C'est l'objet du paragraphe suivant.

[3] On peut relâcher un peu cette condition sous certaines hypothèses.

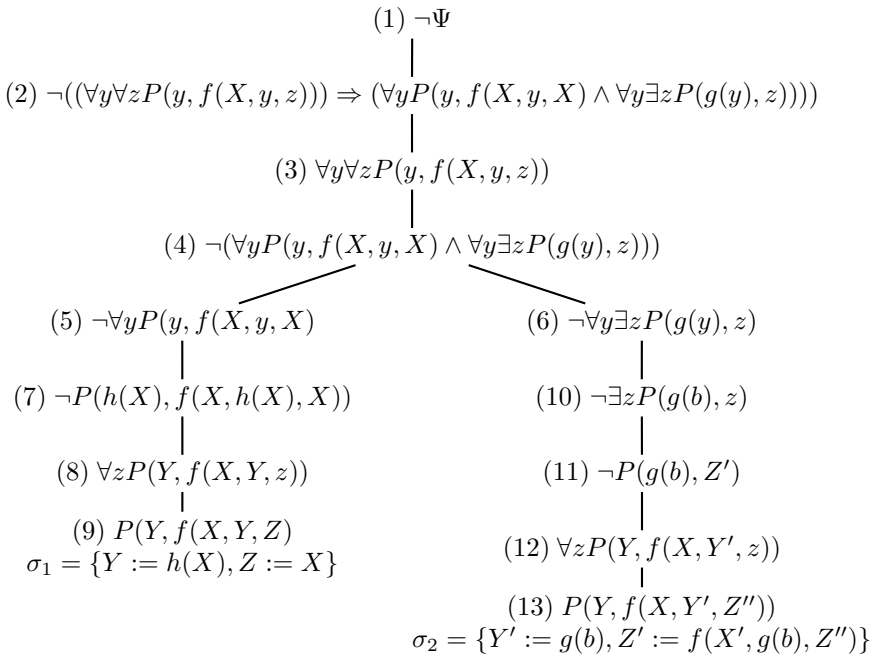


FIG. 1.7 – Tableau à métavariables fermé pour la formule $\Psi = \exists x((\forall y \forall z P(y, f(x, y, z))) \Rightarrow (\forall y P(y, f(x, y, x) \wedge \forall y \exists z P(g(y), z))))$

Substitution et unification

L'addition de métavariables dans le processus de raisonnement oblige à définir comment substituer de manière adéquate des termes du langage à ces variables dans les tableaux.

Pour fermer un tableau avec métavariables, il est nécessaire d'utiliser des techniques d'unification, dont la théorie est exposée dans [2].

En pratique, il existe plusieurs façons de faire :

- La méthode employée dans [31] calcule localement une solution permettant de fermer une branche et la propage dans le reste du tableau. Plusieurs substitutions peuvent parfois être appliquées pour fermer une branche : ainsi si on a $\neg P(X), P(a), P(b)$ on peut choisir de substituer a ou b à X pour clore la branche. Si le choix fait ne permet pas de fermer ensuite le tableau (si j'ai $P(X), \neg P(b)$ et que j'ai choisi de substituer a à X dans le premier cas), alors il faut revenir en arrière pour changer notre premier choix. On peut obtenir le même effet sans retour en arrière en rejouant la γ-expansion qui a donné $P(X)$ et en instanciant le $P(Y)$ ainsi obtenu par b.
- Une technique un peu plus avancée associe au tableau un ensemble de contraintes d'unification que l'on doit pouvoir résoudre pour fermer le tableau. On peut par exemple développer totalement un tableau avec un nombre donné de γ-expansions, déterminer s'il est possible de le fermer et recommencer avec un nombre plus grand de γ-expansions (technique d'*iterative deepening*) si la fermeture échoue. La procédure de fermeture incrémentale de [32] fait aussi partie de cette famille.

Propriétés

Theorème 5 (Correction et complétude)**.** *Soit $\Psi = \Gamma \wedge \neg\Delta$ une formule du premier ordre :*

- *si Ψ a une preuve par tableaux à métavariables, Ψ est insatisfiable.*
- *si Ψ est insatisfiable, alors Ψ a une preuve par tableaux à métavariables.*

Démonstration. La preuve (sémantique) de correction (cf [31]) est une extension de celle réalisée dans le cas propositionnel. Il est cependant nécessaire d'étendre la notion de satisfiabilité pour prendre en compte la présence de métavariables.

La preuve de complétude générique est moins difficile car les tableaux avec métavariables sont plus généraux que la version close de la section 1.3. Cependant, il est en général montré qu'une certaine restriction (une hiérarchisation donnée) de l'application des règles reste complète. Nous ne rentrerons pas dans les détails, car ces preuves sont plus liées au choix de l'auteur concernant non seulement les règles mais aussi la façon de substituer les métavariables.

Les preuves syntaxiques correspondantes restent relativement simples. Cependant, il faut considérer la substitution qui ferme le tableau pour pourvoir instancier immédiatement les variables dans les preuves en séquents. □

Le contenu sémantique du noyau propositionnel des tableaux du premier

ordre reste inchangé par rapport à ce qu'on a pu voir dans les sections 1.2.5. Il faut par contre parler du rôle des règles δ et γ. Cette dernière est la plus problématique, car comme on peut l'appliquer sans fin, il peut être impossible d'extraire un contre-modèle fini d'une réfutation par tableaux avec métavariables [4].

1.3.3 Conséquences de l'utilisation de métavariables

Hormis le raffinement des techniques d'élimination des quantificateurs existentiels, l'introduction de métavariables dans les méthodes de tableaux a permis d'exhiber d'autres caractéristiques de ces calculs.

Métavariables et rigidité

La première propriété des métavariables est liée intrinsèquement à leur rôle. Elles sont en effet qualifiées de *rigides*. Nous allons illustrer cette notion sur un exemple. Essayons de prouver de manière naïve la formule (invalide) suivante tirée de [37] :

$$\Psi = \forall x\ (P(x) \vee Q(x)) \Rightarrow (\forall(x)\ P(x) \vee \forall(x)\ Q(x))$$

Le tableau *incorrect* correspondant à cette formule est construit en figure 1.8. Si l'on ne fait pas attention, le tableau fermé obtenu permet de penser que la formule est une tautologie. Ce n'est pas le cas. L'introduction d'une métavariable fraîche dans la formule (2) s'accompagne de l'obligation (laissée implicite) d'assurer que toutes ses occurrences soient instanciées simultanément dès que l'une de ces occurrences le sera. En effet, toute occurrence de cette métavariable dans un tableau est sous la dépendance du quantificateur universel éliminé lors de l'introduction de cette métavariable.

La fraîcheur des métavariables introduites est cruciale afin de ne pas rigidifier des formules qui n'ont pas de rapport entre elles.

Nécessité de pouvoir dupliquer γ

Les tableaux à métavariables sont caractérisés par la règle γ présentée en section 1.3.2. Nous avons déjà remarqué que dans cette inférence, la formule universellement quantifiée est gardée « en cas de nécessité ». Cette manœuvre n'est pas innocente : elle permet de *conserver la complétude du calcul*.

En effet supposons que la règle ait été la suivante :

$$\frac{\Gamma_1, \forall x\ P \mid ... \mid \Gamma_n}{\Gamma_1, P[x := X] \mid ... \mid \Gamma_n}\ \zeta$$ où X est une métavariable fraîche pour le tableau

[4] Sinon on aurait une procédure de décision pour la logique du premier ordre, ce qui est impossible.

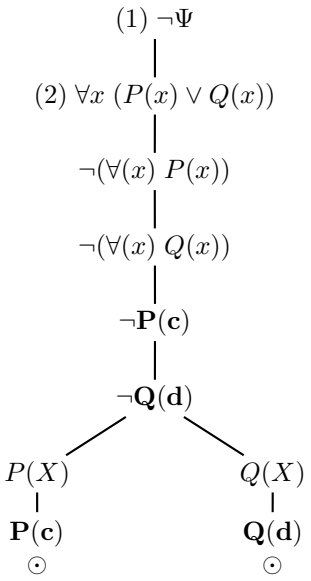


FIG. 1.8 – Tableau incorrect pour $\Psi = \forall x\ (P(x) \vee Q(x)) \Rightarrow (\forall(x)\ P(x) \vee \forall(x)\ Q(x))$

On voit que cette règle ζ ne permet d'appliquer qu'une seule fois une expansion à $\forall x\ P$: elle est clairement destructive. Adoptons la en lieu et place de *gamma* en essayons de démontrer

$$\Psi = \forall x\ P(x) \Rightarrow (P(c) \wedge P(d))$$

qui est clairement une tautologie. on arrive alors à développer l'arbre de la figure 1.9 mais on ne peut pas le fermer car aucune règle supplémentaire ne peut s'y appliquer. Or, si une tautologie n'est effectivement pas démontrable par une méthode, c'est que cette dernière est incomplète. Il est donc obligatoire d'autoriser de multiples applications de γ dans les tableaux du premier ordre. C'est d'ailleurs la seule règle qui ne peut pas être totalement destructive.

Complétude : un critère

À partir du moment où des métavariables sont introduites, la méthode de recherche de preuve devient encore moins déterministe. En particulier, la complétude n'est pas assurée pour toutes les façons d'utiliser la méthode (on a vu que c'était une conséquence par exemple de l'imposition d'une limite au nombre d'applications de la règle γ à une formule universelle). Il faut donc trouver un arrangement des règles qui permette d'épuiser de manière systématique tout l'espace de recherche.

Giese ([32]) définit une recherche de fermeture incrémentale. La procédure, définie en 1, permet de garantir la complétude d'une méthode s'inscrivant dans

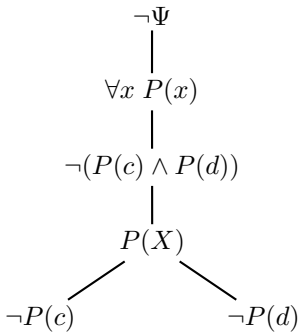


FIG. 1.9 – Incomplétude des tableaux avec γ destructive. Comment fermer si je ne peux pas réappliquer γ à $\forall x\ P(x)$?

ce cadre à condition que le choix des applications de règles soit *équitable*. Est volontairement omis ici ce qui est détaillé dans [32], c'est-à-dire la procédure (incrémentale) `fermable(T)` : on peut considérer en première approximation qu'il s'agit d'appliquer un algorithme d'unification à un ensemble de contraintes.

Algorithme 1 Fermeture incrémentale de Giese

```
T := tableau initial pour S
tant que non fermable(T) faire
  si extensible(T) alors
    choisir une règle applicable à T
    étendre T
  sinon
    retourne 'satisfiable'
  fin si
fin tant que
retourne 'insatisfiable'
```

1.3.4 Traitement avancé des quantificateurs existentiels

Les différentes δ-règles proposent des variations essentiellement sur les arguments du symbole de Skolem et sur la génération de celui-ci. On peut considérer que les améliorations successives apportées portent sur deux points : le contenu sémantique des arguments du symbole de Skolem et l'observation plus fine du contexte (le tableau courant en fin de compte) dans lequel est appliquée la règle. Cette prise en compte plus fine du contexte rapproche les règles δ dynamiques d'une skolemisation (plus statique) qui serait effectuée *avant* l'application des règles d'expansion, qui génère par nature un nombre réduit de symboles de Skolem.

Le nombre de variantes de la règle δ existant effectivement dans la littérature est assez limité mais on peut a priori en développer un grand nombre. D'ailleurs,

D.Cantone et M.Nicolosi Asmundo proposent dans [16] un cadre théorique permettant de vérifier directement la correction des règles de skolemisation. Les δ-règles déjà existantes [5] peuvent y être exprimées mais on peut imaginer en développer d'autre à l'aide de ces outils théoriques.

$$\frac{\exists x\ P(x)}{P[x := sko(P, x, \mathcal{T})]}\ \delta$$

- Γ est la branche sur laquelle se trouve la formule $\exists x\ P(x)$
- $sko(P, x, \mathcal{T})$ est une fonction de Skolem dépendant de P (la formule qui fournit un contexte local) , x (la variable à skolemiser) et $\mathcal{T}$ (le tableau que l'on développe qui fournit le contexte global).

FIG. 1.10 – Forme générale des δ-règles

La règle δ de Smullyan est la skolemisation la plus simple qui soit car les variables existentielles sont remplacées directement par des constantes. C'est quelque part le pendant de la règle γ qui susbtitue directement un terme quelconque à la variable universelle car elle génère également des instanciations en aveugle alors que le contexte contient parfois des informations plus précises. La règle de skolemisation de Fitting (figure 1.6) est déjà une évolution intéressante. Par la suite, de nouvelles idées ont été avancées afin d'améliorer la skolemisation dynamique dans les tableaux. Nous nous attarderons donc dans les sections suivantes sur les diverses améliorations apportées à la règles δ dans le but d'être plus performant dans la gestion de cette skolemisation à la volée. On verra que les efforts se sont portés à la fois sur la caractérisation chaque fois plus précise des arguments du symbole de Skolem mais aussi sur ce symbole lui-même en essayant d'analyser de manière plus fine le contexte dans lequel se déroule la preuve.

La règle δ

La première règle de skolemisation incluse dans les tableaux à variables libres est dûe à Fitting dans la première édition de [31] (cette deuxième édition emploie la règle δ^+ que nous introduisons juste après).

Cette règle correspond à un processus légèrement amélioré par rapport à la skolemisation dite externe dans [47] tout en restant plus faible que la skolemisation interne (toujours [47]), comme on peut le voir dans l'exemple 1.

Exemple 1. Prenons la formule :

$$\Psi = \forall w\ \forall x\ \forall y\ \forall x' \exists z\ (P(x, y, z) \vee R(z, w))$$

La skolemisation externe de Ψ donnera

$$\forall w\ \forall x\ \forall y\ \forall x'\ (P(x, y, f(w, x, y, x')) \vee R(g(w, x, y, x'), w))$$

[5] sauf la règle δ^{ε}

La skolemisation interne de Ψ donnera

$$\forall w\ \forall x\ \forall y\ \forall x'\ (P(x, y, f(x, y)) \vee R(g(w), w)))$$

La skolemisation de Ψ par une règle analogue à δ donnerait

$$\forall w\ \forall x\ \forall y\ \forall x'\ (P(x, y, f(x, y, w)) \vee R(g(w, x, y), w))$$

Dans la skolemisation externe (*outer skolemization*) toutes les variables universellement quantifiées sous la portée desquelles se trouve le formule à skolemiser vont être utilisées en argument du nouveau symbole de Skolem. Dans la skolemisation interne (*inner skolemization*), seules les variables libres apparaissant effectivement dans la formule à skolemiser vont faire partie des arguments de le fonction de Skolem. Dans la règle δ de la figure 1.6, on prend toutes les variables libres ayant une occurrence dans la branche dans laquelle on se situe au moment de l'application de la skolemisation.

Cette règle conduit à utiliser des fonctions de Skolem ayant plus d'arguments que nécessaire d'un point de vue sémantique. La δ-règle suivante s'emploie à minimiser le nombre d'arguments.

La règle δ^+

Cette règle, dûe à Hähnle et Schmitt ([38]) et donnée en figure 1.11, est une spécialisation de la règle précédente. Au lieu de prendre toutes les variables libres présentes sur la branche, elle requiert simplement celles qui sont effectivement présentes dans le formule considérée. Cela correspond à un processus de skolemisation interne (*inner skolemisation*). Notons que dans le cas de formules en forme prénexe, elle est strictement équivalente à la règle δ de Fitting.

$$\frac{\exists x\ P}{P[x := f(y_1, ..., y_n)]}\ \delta^+$$

– f est un symbole de Skolem frais globalement pour tout le tableau ,
– $y_1, ..., y_n$ sont les variables libres apparaissant dans la formule $\exists x\ P(x)$.

FIG. 1.11 – δ^+

La règle δ^{+^+}

δ^{+^+} (figure 1.12) est une amélioration de la précédente par Beckert, Hähnle et Schmitt ([9]). L'intérêt de cette amélioration réside dans la création limitée de nouveaux symboles de Skolem. Au lieu de générer un symbole nouveau pour tout le tableau à chaque application de la règle, un symbole de Skolem unique est associé à chaque δ-formule. La méthode des tableaux ainsi obtenue est plus efficace car elle génère moins de symboles inutiles.

Plus exactement, le même symbole de Skolem est assigné à toutes les formules faisant partie d'une même classe d'équivalence définie comme suit : soit une δ-formule δ du langage, la classe d'équivalence $[\delta]$ est l'ensemble des formules identiques à δ modulo le renommage des variables (y compris liées).

Ainsi, prenons la formule $\forall x\ \exists y P(x,y)$, et appliquons lui γ n fois d'affilée. L'application δ^+ au résultat de l'application de γ produit, à chaque fois, une formule légèrement différente qui est de la forme $P(x, f_i(X))$ où $i \in [1,n]$. La règle δ^{+^+} oblige toutes les formules $P(x, f_i(X)), i \in [1,n]$ à avoir le même symbole de Skolem ($f_1 = ... = f_n = f$) .

$$\frac{\exists x\ P(x)}{P[x := f_{[\delta]}(y_1, ..., y_n)]}\ \delta^{+^+}$$

- $f_{[\delta]}$ est un symbole de Skolem frais assigné à la classe d'équivalence de la formule $\exists x\ P(x)$
- $y_1, ..., y_n$ sont les variables libres apparaissant dans la formule $\exists x\ P(x)$.

FIG. 1.12 – δ^{+^+}

La règle δ^*

Cette nouvelle variation de la δ-règle proposée par Baaz et Fermüller ([3]) permet d'introduire un gain non élémentaire dans la taille des preuves par rapport aux versions précédentes en n'utilisant que les variables pertinentes comme argument du symbole de Skolem. Cet ensemble est par conséquent un sous-ensemble des variables libres dans la formule.

Le résultat de cette règle dynamique correspond à ce qu'on qu'on aurait obtenu en appliquant au préalable une réduction maximale de portée des quantificateurs (ou *miniscoping* [47]) à la formule en entrée.

[16] présente une version corrigée de cette règle en changeant légèrement la notion de *variable pertinente* (voir définition 1) et introduit la notion de *formule extraite pertinente*. Nous présentons cette version corrigée qui par ailleurs préserve les résultats de complexité donnés dans [3].

$$\frac{\exists x\ P(x)}{P[x := f(y_1, ..., y_n)]}\ \delta^*$$

- f est un symbole de Skolem frais globalement pour tout le tableau ,
- $y_1, ..., y_n$ sont les variables *pertinentes* apparaissant dans la formule $\exists x\ P(x)$.

FIG. 1.13 – δ^*

La définition de l'ensemble $Rel(\phi, x)$ des variables libres qui apparaissent de manière pertinente dans la formule ϕ par rapport à la variable x est la suivante :

Définition 1 (Variables pertinentes)**.** Soient ϕ une formule et S un ensemble de variables. Les variables *pertinentes* de ϕ par rapport à S, notées $\mathrm{Rel}(\phi, S)$ sont définies inductivement comme suit :

– Si $Free(\phi) \cap S = \emptyset$ alors $Rel(\phi, S) = \emptyset$

sinon :

– si ϕ est une formule atomique, alors $Rel(\phi, S) = Free(\phi) \backslash S$;
– si $\phi = \neg\Psi$, alors $Rel(\phi, S) = Rel(\Psi, S)$;
– si $\phi = \Psi_1 \vee \Psi_2$
  – si $Free(\Psi_1) \cap S = \emptyset$ alors $Rel(\phi, S) = Rel(\Psi_2, S)$;
  – de même si $Free(\Psi_2) \cap S = \emptyset$ alors $Rel(\phi, S) = Rel(\Psi_1, S)$;
  – sinon si $Free(\Psi_1) \cap S \neq \emptyset$ et $Free(\Psi_2) \cap S \neq \emptyset$ alors $Rel(\phi, S) = Free(\phi) \backslash S$

  Le même raisonnement est appliqué pour les cas $\phi = \Psi_1 \wedge \Psi_2$ et $\phi = \Psi_1 \Rightarrow \Psi_2$;
– si $\phi = (\forall | \exists) x\Psi$ alors $Rel(\phi, S) = Rel(\Psi, S \cup \{x\})$

Exemple 2. Soit $\phi_1 = P(x, y) \vee Q(z)$. Les variables pertinentes de ϕ par rapport à x sont $Rel(P(x, y) \vee Q(z), x) = Rel(P(x, y), x) = Free(P(x, y) \backslash \{x\} = \{y\}$

À partir de la notion de variable pertinente, on définit une notion voisine de *formule extraite pertinente* sur laquelle on va effectivement utiliser la règle δ^*. Lorsqu'on applique δ^* à une δ-formule, la skolemisation est en fait appliquée à la *formule extraite pertinente* de cette δ-formule. Pour préserver la satisfiabilité, on doit cependant calculer une formule B en parallèle de notre formule extraite pertinente.

Le calcul des formules extraites pertinentes est exposé dans la définition 2, où Λ est un raccourci signifiant « la formule vide ».

Définition 2 (Formule extraite pertinente)**.** Soient ϕ une formule et S un ensemble de variables. La *formule extraite pertinente* de ϕ par rapport à S, notée $RelF(\phi, S)$ et sa formule accompagnatrice $B(\phi, S)$ sont calculées comme suit :

– Si $Free(\phi) \cap S = \emptyset$ alors $RelF(\phi, S) = \Lambda = B(\phi, S)$

sinon :

– si ϕ est un littéral alors $RelF(\phi, S) = \phi$ et $B(\phi, S) = \Lambda$;
– si $\phi = \Psi_1 \wedge \Psi_2$, alors
  – si $Free(\Psi_1) \cap S = \emptyset$ alors $RelF(\phi, S) = RelF(\Psi_2, S)$ et $B(\phi, S) = B(\Psi_2, S)$;
  – si $Free(\Psi_2) \cap S = \emptyset$ alors $RelF(\phi, S) = RelF(\Psi_1, S)$ et $B(\phi, S) = B(\Psi_1, S)$;
  – $RelF(\phi, S) = \phi$ et $B(\phi, S) = \Lambda$ sinon.

  Le même calcul est appliqué pour $\phi = \Psi_1 \vee \Psi_2$.
– si $\phi = (\forall \mid \exists\ ) y\Psi$ alors $RelF(\phi, S) = (\forall \mid \exists\ ) y RelF(\Psi, S \cup \{y\})$ et $B(\phi, S) = B(\Psi, S \cup \{y\})$;

À partir de ces calculs on peut montrer la préservation de la satisfiabilité (par induction structurelle).

Lemme 1. *Soient φ une formule, S un ensemble de variables et $\Psi = RelF(\varphi, S)$. Si $\Psi \neq \Lambda$, alors*

– $\models \varphi \Rightarrow (\Psi \vee B(\varphi, S))$ *et*
– $\models (\exists\, S\varphi) \Rightarrow \forall\, S((\Psi \vee B(\varphi, S)) \Rightarrow \varphi)$

La règle δ^{*^*}

La dernière version de δ instanciable dans le cadre théorique de [16] est proposée par Cantone et Nicolosi Asmundo dans [17]. Elle est basée sur la technique dite de *skolemisation globale* et sur la notion de formule clé qui sont présentées dans [19].

$$\frac{\exists x\ P(x)}{P[x := f_\Psi(S_\Psi)\sigma]}\ \delta^{**}$$

- $\phi = Key(P(x), x)$
- $\sigma \in SubstKey(P(x), x)$
- $\Psi = RelF(\phi, \{x_0\})$
- $S_\Psi = Free(\Psi)\backslash\{x_0\}$
- Si $\Psi = \Lambda$, la formule est inchangée : la formule inférée est $P(x)$.

FIG. 1.14 – δ^{*^*}

Cette règle est capable de réduire de façon exponentielle la longueur des preuves par comparaison avec les δ-règles ci-dessus par une réduction des dépendances de variables dans les fonctions de Skolem et par une plus grande réutilisabilité du même symbole de Skolem. L'idée est d'utiliser une certaine variable (renommée x_0) dans les formules comme pivot pour calculer des formules clés équivalentes et dégager ainsi des classe d'équivalence plus grandes entre formules.

La première version ([17]) était incorrecte : elle utilisait la même définition de variables pertinentes que celle donnée dans [3], corrigée par la suite dans [16]. C'est cette dernière version que nous décrivons.

La notion de formule-clé permet de caractériser les formules du langage qui ont besoin de leur propre symbole de Skolem : à chaque formule correspond une unique formule clé.

Définition 3 (Formules canoniques)**.** Une formule φ est dite *canonique* (par rapport à la variable x_0 si :

- il existe $k \geq 0$ tel que les variables liées de φ sont $\{x_{-1}, \ldots, x_{-k}\}$, celles-ci apparaissent dans φ dans l'ordre $x_{-1}, \ldots, x_{-k}$ de la gauche vers la droite (de la formule) et chacune n'est quantifiée qu'une fois (mais peut apparaître plusieurs fois) ;
- il existe $k \geq 0$ tel que $Free(\varphi)\backslash\{x_0\} = \{x_1, \ldots, x_k\}$, ces variables apparaissent dans l'ordre $x_1, \ldots, x_k$ de gauche à droite, et chacune n'apparaît qu'une seul fois dans φ.

Chaque formule φ peut être canonisée par rapport à une certaine variable x et il existe une unique formule canonisée correspondant $\hat{\Phi}$ telle que φ et $\hat{\Phi}\sigma$ soient identiques modulo renommage des variables liées. Dans ce cas on suppose σ une substitution libre (n'introduit que des variables fraîches) pour $\hat{\Phi}$ faisant correspondre des variables avec des variables et telle que $x = x_0\sigma$. Les formules clé sont définies comme les formules canoniques les plus générales par rapport aux substitutions.

Définition 4 (Formules clé)**.** Une formule φ est une *formule clé* si
- elle est canonique par rapport à x_0 et
- pour chaque formule canonique φ, s'il existe une substitution σ qui est libre pour $\exists x_0 \Psi$, et telle que $\varphi = \Psi\sigma_{x_0}$, alors $\Psi = \varphi$.

À l'aide de ces définitions, il est prouvé dans [17] que chaque formule du langage possède une unique formule clé.

Lemme 2. *Soient Ψ une formule de notre langage, x_i une variable donnée. Il existe une* unique *formule clé φ, notée $Key(\Psi, x_i)$ et une collection non vide de substitutions libres pour φ, notée $SubstKey(\Psi, x_i)$ telle que, pour chaque $\sigma \in SubstKey(\Psi, x_i)$, on ait :*
- *Ψ et $\varphi\sigma$ sont identiques modulo renommage des variables liées*
- *$x_0\sigma = x_i$,*
- *x_i n'apparaît pas dans $x\sigma$ pour $x \neq x_0$*

Détaillons ces définitions sur un exemple afin de les éclaircir.

Exemple 3. Soit $\varphi = \exists yz(R(x, f(y), z, h(w, w)) \wedge Q(u, v))$.

Sa formule canonique par rapport à x est la suivante :

$$\varphi_1 = \exists x_{-1}x_{-2}(R(x_0, f(x_{-1}), x_{-2}, h(x_1, x_2)) \wedge Q(x_3, x_4))$$

Dans ce cas $\sigma = \{x := x_0, w := x_1, w := x_2, u := x_3, v = x_4\}$.

Sa formule clé par rapport à x est :

$$\varphi_2 = \exists x_{-1}x_{-2}(R(x_0, f(x_{-1}), x_{-2}, x_1) \wedge Q(x_2, x_3))$$

La substitution suivante satisfait les conditions du lemme 2 : $\sigma' = \{x_0 := x, x_1 := h(w, w), x_2 := u, x_3 := v\}$.

δ^ϵ

La méthode la plus récente, publiée dans [33], utilise les ϵ-termes de Hilbert. Un ϵ-terme est de la forme $\epsilon x.\Phi$, où x est une variable et Φ est une formule. La signification de cette notation est « un élément pour lequel Φ est vérifiée, si cet élément existe, et un élément arbitraire sinon ». Dans le cas où plusieurs éléments remplissent cette condition, ϵ joue le rôle d'opérateur de choix.

Bien que les ϵ-termes puissent paraître plus compliqués que la skolemisation première vue, les raffinements successifs proposés (en particulier δ^* et δ^{*^*}) peuvent être interprétés comme une évolution vers un comportement « à la ϵ-terme » dans la skolemisation. Ahrendt et Giese remarquent que la règle de la figure 1.15 peut être renforcée de manière à être strictement plus forte que la règle δ^*, ce qui assure que δ^ϵ est plus performante que toutes les δ-règles sauf δ^{*^*}. Il est notamment montré dans [33] que les preuves obtenues avec δ^ϵ peuvent être exponentiellement plus courtes que celles avec δ^{+^+}

$$\frac{\exists x\ P(x)}{P(\epsilon x.P(x))}\ \delta^{\epsilon}$$

FIG. 1.15 – δ^{ϵ}

Conclusion sur le traitement des quantificateurs existentiels

Nous avons passé en revue les améliorations successives apportées à la façon d'éliminer les quantificateurs existentiels dans le raisonnement par les méthodes de tableau.

Les propositions basées sur des skolemisations améliorées cherchent en fin de compte à :

– réduire le nombre d'arguments des fonctions de Skolem
– réduire le nombre de nouveaux symboles de Skolem

Les règles proposées permettent en général de conserver telle quelle la preuve de complétude du calcul de tableau à métavariables associé. Par contre, la correction de ces règles est parfois difficile à établir (voir dans les articles correspondants).

Les améliorations reposent sur des analyses de plus en plus subtiles et complexes des contextes de preuves locaux et globaux. Cette complexification, surtout apparente dans la famille des règles δ^*, a conduit à réutiliser les ϵ-termes de Hilbert dans [33]. De plus, c'est pour l'instant la proposition la plus efficace en terme de longueur de preuve sur le papier, hormis vis-à-vis de la règle δ^{*^*}. Pour ce cas précis, il semble raisonnable de conjecturer que la combinaison d'un miniscoping préalable aux règles d'expansion et de la règle δ^{ϵ} soit aussi performant que la règle δ^{*^*}.

La complexité des calculs mis en œuvre semblent provenir du fait que la skolemisation par δ règle est faite de manière dynamique. Une skolemisation interne préalable – et donc, en quelque sorte, statique — à toute règle d'expansion est par exemple équivalente en terme de résultats obtenus à la règle δ^{+^+} de [9] : l'idée à sa base semble pourtant plus simple.

D'après [4] [6] , les δ-règles sont particulièrement intéressantes en terme d'efficacité lorsqu'elles sont combinées avec des tableaux permettant l'introduction (restreinte) de coupures.

La discussion pourrait continuer sur l'influence de la méthode de skolemisation choisie sur le calcul mais dépasserait de loin le cadre de cette thèse. Les articles [4, 34, 47] permettront au lecteur intéressé d'approfondir le sujet.

1.4 Logique intuitionniste

Les méthodes de tableaux fournissent un cadre utilisé avec succès en dehors de la logique classique. La logique intuitionniste, de par son utilisation assez

[6] (pp. 322) (...) dynamic skolemization can yield drastically better behavior if variants of tableaux are used which allow for (restricted versions of) the analytic cut rule

répandue, est l'un des exemples les plus intéressants. Nous allons voir comment les tableaux on été adaptés pour tenir compte des contraintes de cette logique.

Un panorama détaillé de ces méthodes pour la logique intuitionniste se trouve dans [62].

1.4.1 Sémantique de la logique intuitionniste

Deux sémantiques principales sont disponibles pour la logique intuitionniste : l'algèbre de Heyting et les structures de Kripke. Cette dernière sémantique est celle que nous introduisons dans cette section car c'est celle que nous allons utiliser dans les preuves sémantiques liées aux tableaux intuitionnistes (modulo).

On peut considérer les structures de Kripke comme des suites de modèles booléens. Une structure à un seul monde est effectivement un modèle booléen.

Un *modèle de Kripke* est un quadruplet $(U, D, \leq, \models'_{\mathcal{K}})$ défini de la façon suivante :

- l'*univers* U est un ensemble non vide ordonné partiellement par $\leq$: chaque point de cet univers est appelé *monde* (noté p ou q) ;
- la *fonction de domaine* D fait correspondre un ensemble non vide de termes (du langage) à chaque monde de U de telle manière que $D(p) \subseteq D(q)$ lorsque $p \leq q$;
- $\models'_{\mathcal{K}}$ est une relation binaire entre l'univers U et l'ensemble des formules atomiques telle que, premièrement, si $p \models'_{\mathcal{K}} P$ et $p \leq q$, alors $q \models'_{\mathcal{K}} P$ et deuxièmement $\forall p \in U, p \not\models'_{\mathcal{K}} \bot$.
  Notons que l'utilisation de la notation $\models_{\mathcal{K}}$ sert uniquement ici à bien distinguer du signe utilisé dans les modèles classiques $\models$ (par la suite nous utiliserons $\models$ pour $\models_{\mathcal{K}}$).

Alors la *relation de forçage* $\models_{\mathcal{K}}$ est la plus petite relation contenant $\models'_{\mathcal{K}}$ qui est close sous les règles suivantes :

$$
\begin{array}{lll}
p \models_{\mathcal{K}} A \wedge B & \text{si} & p \models_{\mathcal{K}} A \text{ et } p \models_{\mathcal{K}} B \\
p \models_{\mathcal{K}} A \vee B & \text{si} & p \models_{\mathcal{K}} A \text{ ou } p \models_{\mathcal{K}} B \\
p \models_{\mathcal{K}} A \Rightarrow B & \text{si} & q \models_{\mathcal{K}} B \text{ lorsque } q \not\models_{\mathcal{K}} A \text{ et } q \geq p \\
p \models_{\mathcal{K}} \forall x A & \text{si} & q \models_{\mathcal{K}} A[x := t] \text{ pour chaque } q \geq z \text{ et chaque } t \in D(q) \\
p \models_{\mathcal{K}} \exists x A & \text{si} & p \models_{\mathcal{K}} A[x := t] \text{ pour un } t \in D(p)
\end{array}
$$

L'obtention des notions usuelles de validité et de contre-modèle est immédiate :

- Un séquent $\Gamma \vdash C$ est *valide* dans un modèle si, pour chaque monde p qui force chaque formule de Γ, p force C.
- Un modèle est un *contre-modèle* de $\Gamma \vdash C$ s'il contient un monde où chaque formule de Γ est forcée mais où C ne l'est pas.

1.4.2 Règles pour le premier ordre

Les règles de dérivation que nous allons présenter sont très simples, dans la lignée de celles de Smullyan pour la logique classique. Ce sont celles que l'on retrouve dans [46**?**].

La différence majeure provient de l'explicitation de la nature sémantique du raisonnement dans le cas intuitionniste. La sémantique dans le cas du premier ordre est en fait intégrée directement dans la négation $\neg$ (ou l'absence de négation) devant les formules ; par conséquent on ne l'explicite généralement pas (l'idée est cependant très clairement présente chez Smullyan [50]).

$$\frac{Tp \Vdash A \vee B}{Tp \Vdash A \mid Tp \Vdash B} \qquad \frac{Fp \Vdash A \wedge B}{Fp \Vdash A \mid Fp \Vdash B}$$

$$\frac{Tp \Vdash A \wedge B}{Tp \Vdash A, Tp \Vdash B} \qquad \frac{Fp \Vdash A \vee B}{Fp \Vdash A, Fp \Vdash B}$$

$$\frac{Tp \Vdash A \Rightarrow B}{Fp' \Vdash A \mid Tp' \Vdash B} \text{ pour tout } p' \geq p \qquad \frac{Fp \Vdash A \Rightarrow B}{Tp' \Vdash A, Fp' \Vdash B} \text{ pour un nouveau monde } p' \geq p$$

$$\frac{Tp \Vdash \neg A}{Fp' \Vdash A} \text{ pour tout } p' \geq p \qquad \frac{Fp \Vdash \neg A}{Tp' \Vdash A} \text{ pour un nouveau monde } p' \geq p$$

$$\frac{Tp \Vdash \exists x P(x)}{Tp \Vdash P(c)} \text{ pour une constante fraîche } c \qquad \frac{Fp \Vdash \exists x P(x)}{Fp \Vdash P(t)} \text{ pour tout terme } t$$

$$\frac{Tp \Vdash \forall x P(x)}{Tp' \Vdash P(t)} \text{ pour tout monde } p' \geq p \text{ et tout terme } t \qquad \frac{Fp \Vdash \forall x P(x)}{Fp' \Vdash P(c)} \text{ pour un nouveau monde } p' \geq p \text{ et une constante fraîche } c$$

FIG. 1.16 – Tableaux intuitionnistes de Nerode et Shore

Les entités sur lesquelles on raisonne dans les tableaux intuitionnistes sont les formules du premier ordre auxquelles on associe le modèle de Kripke dans lequel on les interprète. Hormis cela, les règles sont similaires aux tableaux de la section 1.3.1.

Le tableau pour démontrer $\Gamma \vdash \Delta$ prend intialement les entrées $T\emptyset \models \Gamma, F\emptyset \models \Delta$ puis applique les règles de la figure 1.16. Autrement dit on suppose un monde intial $\emptyset$ dans lequel Γ est valide (ceci est dénoté par T) et Δ invalide (ceci est dénoté par F) et on cherche à dériver un modèle. Si toutes les branches sont fermées alors aucun modèle (de Kripke) de $T\emptyset \models \Gamma, F\emptyset \models \Delta$ ne peut être trouvé.

1.4.3 Propriétés

Les tableaux de la section précédente remplissent eux aussi des critères des corrections et de complétude. La manière usuelle de démontrer ces points est

une extension de la méthode sémantique utilisée dans le cas classique, en se servant cette fois des modèles de Kripke.

Theorème 6 (Correction et complétude)**.** *Soient* Γ *un ensemble de formules et* P *une formule. Le tableau pour* $T\emptyset \Vdash \Gamma, F\emptyset \Vdash P$ *est fermé si et seulement si* $\Gamma \models P$ *pour tout modèle de Kripke.*

1.4.4 Améliorations

Les méthodes à variables libres ont été adaptées pour la logique intuitionniste. On peut en trouver l'illustration dans [60, 61].

Les techniques de skolemisation ont également fait l'objet de travaux par [49, 60] mais leur formulation est plus compliquée que dans le cas classique.

1.5 Conclusion

Nous avons développé dans ce chapitre l'adaptation de la méthode des tableaux de la logique propositionnelle à celle du premier ordre. Partant d'une vision informelle de la méthode, nous avons pu introduire les concepts principaux concernant les tableaux. Dans le cas propositionnel et dans les tableaux clos du premier ordre, le lien avec le calcul des séquents est immédiat. Celui-ci est moins visible dans les optimisations pratiques apportées pour la logique du premier ordre : métavariables et skolemisation.

Nous avons pu voir que la règle de skolemisation a fait l'objet de nombreuses améliorations, destinées à produire chaque fois des preuves plus courtes. Les résultats de gain de taille sur les preuves sont généralement calculés « dans le pire des cas » : il est difficile de déterminer le gain moyen et, encore plus ennuyeux, de savoir si le gain en longueur de preuve n'est pas compensé par une perte à cause des calculs supplémentaires (le cas de δ^{*^*} semble le plus extrême).

Enfin, nous avons abordé un exemple d'adaptation des méthodes de tableaux à une autre logique : la logique intuitionniste. Dans l'expression que nous en donnons, le côté sémantique des tableaux est mis en avant.

Chapitre 2

Égalité dans la méthode des tableaux

L'égalité est un prédicat un peu à part du fait de son ubiquité d'une part, et de ses caractéristiques, d'autre part. Ces considérations permettent de penser qu'il faut lui réserver un traitement interne particulier (optimisé si possible) dans le domaine de la preuve automatique, et donc en particulier au sein des méthodes de tableaux.

De manière générale, on peut distinguer en suivant Beckert ([8]) deux grands types d'approches dans la façon d'intégrer l'égalité dans les tableaux :

- un raisonnement qui fait appel à l'ajout de règles de tableaux pour intégrer l'égalite dans le raisonnement (ce type de méthodes est appelé *partiel* dans [8]) ;
- un raisonnement qui n'ajoute aucune règle aux tableaux mais qui va utiliser des techniques basées sur l'(E-)unification pour fermer les branches relativement à des théories équationnelles ([8] caractérise cette catégorie de raisonnement de *total*).

Nous allons voir un aperçu de l'évolution du traitement de l'égalité à travers certaines approches partielles et totales. Enfin, deux autres approches plus atypiques seront décrites pour offrir une vue plus large du domaine. [7, 8, 21] fourniront des informations en profondeur au lecteur désireux d'avoir des descriptions plus détaillées des formalismes présentés ainsi qu'un tour d'horizon plus large du domaine.

2.1 Premières approches

Les premières approches pour traiter l'égalité dans les tableaux sont faites sur les tableaux clos (en anglais *ground tableaux*). Elles apparaissent dans les années 60 à la suite des travaux de Jeffrey et d'autres. Le principe consiste à ajouter de nouvelles règles spécifiques aux tableaux pour traiter le prédicat de l'égalité. Cette démarche directe ressemble en fait à celle qui a permis de

passer de la logique propositionnelle à la logique du premier ordre en ajoutant un traitement spécial des quantificateurs. Nous n'allons en présenter que deux variations qui sont assez significatives.

Jeffrey

Jeffrey présente ([41]) une des premières méthodes partielles de traitement de l'égalité dans les tableaux sans variables libres.

$$\frac{\begin{array}{c} t \approx s \\ P(t) \end{array}}{P(s)} \approx_l$$

$$\frac{\begin{array}{c} t \approx s \\ P(s) \end{array}}{P(t)} \approx_r$$

$$\frac{\neg(t \approx t)}{\odot} \odot_{\approx}$$

FIG. 2.1 – Règles de Jeffrey pour l'égalité

Les règles de Jeffrey (figure 2.1) s'appliquent sur des tableaux sans métavariables. En dehors de cela, elles ont le désavantage d'être utilisables sans aucune restriction, ce qui génère un grand indéterminisme dans leur application. À son tour, celui-ci produit potentiellement un énorme espace de recherche dans lequel un très grand nombre de formules inutiles peuvent être obtenues. L'exemple le plus marquant est le suivant : prenons une branche qui contient les formules $f(a) \approx a$ et $P(a)$; alors celle-ci peut être étendue par toutes les formules $P(f(a)), P(f(f(a))), ..., P(f^n(a))$. On a donc dans ce cas une méthode qui non seulement génère des formules qui ne permettront jamais de clore le tableau mais qui en génère une infinité.

Reeves

Le traitement de l'égalité par Reeves est une amélioration de la méthode précédente. En effet les règles de [48] reprises dans la figure 2.2 génèrent un plus petit espace de recherche .

$$\frac{t \approx s}{s \approx t}\ \text{sym} \qquad \frac{P(t_1, ..., t_n), \neg P(s_1, ..., s_n)}{\neg(t_1 \approx s_1) \mid ... \mid \neg(t_n \approx s_n)}\ \text{pred}$$

$$\frac{\neg(t \approx t)}{\odot} \odot_{=} \qquad \frac{\neg(f(t_1, ...t_n) \approx f(s_1, ..., s_n))}{\neg(t_1 \approx s_1) \mid ... \mid \neg(t_n \approx s_n)}\ \text{fun}$$

FIG. 2.2 – Règles de Reeves pour ègalité

L'espace de recherche généré par les règles de Reeves est plus petit puisque seules les formules atomiques qui ferment potentiellement une branche sont utilisées dans ses expansions.

Cette méthode cause malheureusement un accroissement exponentiel du nombre de branches par rapport au nombre d'égalités dans la branche dans le pire des cas : les nouvelles règles d'expansion peuvent en effet aussi bien être appliquées à des égalités.

2.1.1 Un raffinement avec des variables libres

Fitting [31] propose plusieurs façons d'intégrer le traitement de l'égalité pour les tableaux du premier ordre avec métavariables. La méthode que nous allons présenter est très proche des règles de paramodulation utilisées en résolution.

Cette nouvelle approche partielle généralise les techniques de la section 2.1 en intégrant la gestion des métavariables. L'application des règles pour l'égalité peut nécessiter une instanciation de celles-ci, et donc les substitutions calculées peuvent évidemment être différentes de l'identité. Fitting ajoute par conséquent de l'unification dans les règles de tableaux, dont le résultat doit être appliqué au tableau entier : nous allons donc utiliser la représentation globale des règles de tableaux pour mieux voir leur fonctionnement.

$$\frac{\Gamma_1, t \approx s, P(u) \mid ... \mid \Gamma_n}{(\Gamma_1, t \approx s, P(u), P(s) \mid ... \mid \Gamma_n)\sigma} \approx_l,\ \sigma = mgu(t,u)$$

$$\frac{\Gamma_1, t \approx s, P(u) \mid ... \mid \Gamma_n}{(\Gamma_1, t \approx s, P(u), P(t) \mid ... \mid \Gamma_n)\sigma} \approx_r,\ \sigma = mgu(s,u)$$

$$\frac{\Gamma_1, \neg(t \approx s) \mid ... \mid \Gamma_n}{(\odot \mid ... \mid \Gamma_n)\sigma} \odot_{\approx},\ \sigma = mgu(t,s)$$

$$\frac{\Gamma_1, P.\neg Q \mid ... \mid \Gamma_n}{(\odot \mid ... \mid \Gamma_n)\sigma} \odot,\ \sigma = mgu(P,Q),\ \text{P,Q littéraux}$$

FIG. 2.3 – Règles de Fitting pour l'égalité

La complétude de la méthode de tableaux avec métavariables intégrant ce traitement de l'égalité est préservée. Par contre si on ajoute la restriction — qui reste sans influence dans le cas sans métavariable — que l'application d'égalités à d'autres égalités est interdite, la complétude est perdue. On ne peut fermer la branche composée de $a \approx b, f(h(a), h(b)) \approx g(h(a), h(b)), \neg(f(x,x) \approx g(x,x))$.

Exemple 4. Nous allons réfuter l'ensemble S de formules suivant :

$$
\begin{array}{ll}
(1) & \forall x(g(x) \approx f(x) \vee \neg(x \approx a)) \\
(2) & \forall x(g(f(x)) \approx x) \\
(3) & b \approx c \\
(4) & P(g(g(a)), b) \\
(5) & \neg P(a, c)
\end{array}
$$

Si l'on applique les règles de tableau à variables libres de la section 1.3.2, on obtient le tableau fermé suivant de la figure 2.4.

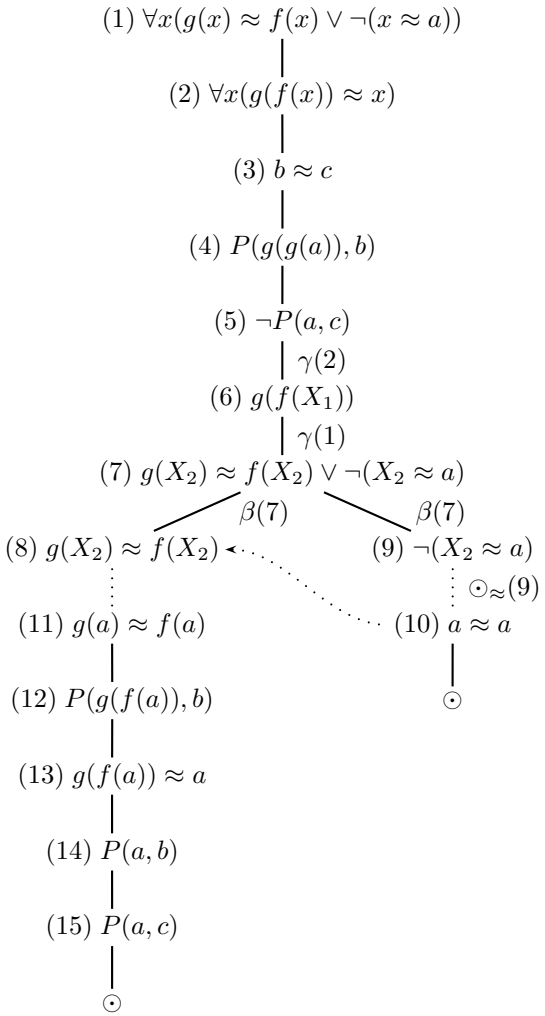


FIG. 2.4 – Tableau fermé pour l'example 4

Cet exemple souligne une difficulté rencontrée lorsque l'on utilise les règles de Fitting pour l'égalité avec variables libres : si l'égalité (8) est appliquée à (4)

de la mauvaise manière, on va générer $P(g(f(a)), b)$ au lieu de $P(f(g(a)), b)$ et le tableau ne peut être fermé aussi vite. Il faudra soit reproduire de nouvelles instances de (7),(8) et (9) à partir d'une application de γ à (1) où revenir sur ses pas et utiliser des méthodes de backtracking (dont on ne parle pas dans cette thèse).

2.2 Raisonnement total par E-unification

Le problème commun des méthodes précédentes provient du fait qu'il n'existe quasiment pas de restrictions sur l'application des règles supplémentaires. La symétrie de l'égalité amène vite une explosion dans la taille de l'espace de recherche. Des approches partielles plus élaborées semblent assez complexes à mettre en œuvre. Par conséquent, il ne reste plus qu'à essayer de raisonner sur les égalités sans ajouter des règles aux tableaux et le problème qui consiste à trouver des formules qui ferment une branche est équivalent à résoudre des problèmes d'E-unification.

Trois types d'E-unification sont importants vis-à-vis du traitement de l'égalité dans les tableaux : universelle, rigide ou mixte. Selon la version des tableaux considérée, une de ces trois méthodes sera utilisée. Dans le cas de tableaux clos, on utilisera la variante dite universelle. Dans les tableaux à métavariables, on aura la version rigide. Dans les tableaux avec formules universelles [1] (cf. [8]), non présentés dans cette thèse, on utilisera une combinaison des deux, l'E-unification mixte.

2.2.1 E-unification rigide

Commençons par les quelques définitions d'usage permettant de situer le problème.

Définition 5 (Équation rigide. E-unification rigide)**.** Une *équation rigide* est une expression de la forme $E \vdash_\forall s \approx t$, où E est une ensemble fini d'équations et s, t sont ds termes.

Le problème de l'*E-unification rigide* est de déterminer si une équation rigide donnée a une solution, c'est-à-dire de déterminer s'il existe une substitution σ telle que $E\sigma \vdash_\forall s\sigma \approx t\sigma$.

La solution à un problème d'E-unification est une substitution qui instancie les variables libres nécessaires à la preuve que deux termes sont égaux. Une variable ne peut être instanciée qu'une seule fois dans une substitution : les variables contenues dans les égalités du problème ne peuvent être utilisées qu'avec au plus une instanciation. Dans l'E-unification rigide, une variable est soit instanciée, soit traitée comme une constante. Si plusieurs instances d'une égalité

[1] L'idée est de détecter les formules dont les variables peuvent être sans risque considérées comme universelles, autrement dit non rigides. En fait seules les variables universellement quantifiées dans des formules de la forme $\forall x\ (A(x) \vee B(x))$ ne remplissent pas ce critère

sont nécessaires, il faut pouvoir fournir plusieurs copies de celle-ci avec différents variables rigides afin de pouvoir résoudre le problème.

En particulier, la différence entre l'E-unification rigide et l'E-unification tout court peut être formulée comme suit. Considérons l'ensemble d'équations

$$E = \{s_1 \approx t_1, ..., s_n \approx t_n\}$$

et l'équation $s \approx t$. Alors l'E-unification consiste à trouver une substitution σ telle que

$$\forall(s_1 \approx t_1), ..., \forall(s_n \approx t_n) \vdash s\sigma \approx t\sigma$$

et l'E-unification rigide consiste à trouver σ telle que

$$\vdash \forall(s_1\sigma \approx t_1\sigma \wedge ... \wedge s_n\sigma \approx t_n\sigma) \Rightarrow s\sigma \approx t\sigma.$$

Lors d'une procédure d'E-unification, les variables des équations $s_i \approx t_i$ sont traitées comme étant quantifiées universellement. Ce n'est pas le cas pour les variables des équations rigides : de là provient leur rigidité qui correspond à l'obligation d'appliquer une éventuelle substitution d'un terme à une variable dans *toute* l'équation.

Il existe effectivement des procédures qui permettent de calculer un ensemble fini de solutions pour une équation rigide donnée. Le problème est *décidable* au niveau d'une seule équation.

Une méthode incomplète : BSE

Degtyarev et Voronkov donnent dans [21, 23] une procédure permettant de calculer un ensemble fini (mais incomplet) de solutions pour un problème d'E-unification rigide donné. L'incomplétude de la procédure est compensée par le fait qu'elle termine systématiquement.

Le système BSE fondé sur la *superposition basique rigide* permet de résoudre des équations rigides. Les objets que l'on peut prouver à l'aide du calcul de [23] sont plus précisément des équations rigides avec contraintes : une paire composée d'une équation rigide $E \vdash_\forall s \approx t$ et d'un ensemble de contraintes équationnelles et d'ordre $\mathcal{C}$ qui lui est associé. Cette paire est dénotée

$$E \vdash_\forall s \approx t \cdot \mathcal{C}.$$

Le système BSE se compose des règles suivantes :

– les règles de superposition basique rigide (droite et gauche) :

$$\frac{E \cup \{s \approx t\} \vdash_\forall u[s'] \approx v \cdot \mathcal{C}}{E \cup \{s \approx t\} \vdash_\forall u[t] \approx v \cdot \mathcal{C} \cup \{s \succ t, u[s'] \succ v, s = s'\}} \text{ (sbrd)}$$

et

$$\frac{E \cup \{s \approx t, u[s'] \approx v\} \vdash_\forall e \cdot \mathcal{C}}{E \cup \{s \approx t, u[t] \approx v\} \vdash_\forall e \cdot \mathcal{C} \cup \{s \succ t, u[s'] \succ v, s = s'\}} \text{ (sbrg)}$$

– la règle qui permet d'établir la solution à une égalité rigide

$$\frac{E \vdash_\forall s \approx t \cdot C}{E \vdash_\forall s \approx s \cdot C \cup \{s \approx t\}} \text{ reqs}$$

Cependant les règles données ne sont applicables que sous les conditions suivantes :

1. la contrainte en conclusion de la règle est effectivement satisfiable ;
2. $s \neq t$ dans (reqs) ;
3. dans les règles de superposition basique s' n'est pas une variable ;
4. dans (sbrd), $u[t] \neq v$.

2.2.2 E-unification rigide simultanée

La situation qui consiste à résoudre une équation rigide donnée correspond à la fermeture d'une seule branche d'un tableau donné. Malheureusement, il faut résoudre la fermeture simultanée de toutes les branches d'un tableau pour décider si on peut fermer le tableau. Le passage de la résolution d'une équation rigide à la résolution d'un système d'équations rigides rend le problème *indécidable* ([22]).

BSE généralisé

L'impossibilité d'avoir en général une procédure de décision pour les problèmes d'E-unification rigide oblige à utiliser sciemment des méthodes incomplètes (mais décidables).

La méthode de Degtyarev et Voronkov présentée en section 2.2.1 peut être utilisée pour calculer branche à branche un ensemble de contraintes-solutions. La procédure extrait de chaque branche un problème d'E-unification rigide à résoudre par BSE puis détermine si l'union des contraintes d'E-unification rigide est satisfiable à l'aide des solutions individuelles calculées.

Nous reprenons l'exemple de [21] pour illustrer l'utilisation de BSE dans le cadre des tableaux à métavariables.

Exemple 5 (Tableau et BSE)**.** Considérons la formule :

$$\exists xyuv((a \approx b \Rightarrow g(x,u,v) \approx g(y,f(c),f(d))) \wedge (c \approx d \Rightarrow g(u,x,y) \approx g(v,f(a),f(b))))$$

Une dérivation d'un tableau à métavariables où γ n'est appliqué qu'une seule fois est donnée en figure 2.5. On utilise par ailleurs un ordre LPO basé sur la précédence $g \succ_F f \succ_F a \succ_F b \succ_F c \succ_F d$.

Chacune des deux branches du tableau admet une fermeture correspondant aux équations rigides :

$$a \approx b \vdash_\forall g(X,U,V) \approx g(Y,f(c),f(d))$$
$$a \approx b \vdash_\forall g(U,X,Y) \approx g(V,f(a),f(b))$$

Dans les deux cas, seule la règle (reqs) est applicable. Ces équations admettent donc respectivement $g(X,U,V) \approx g(Y,f(c),f(d))$ et $g(U,X,Y) \approx$

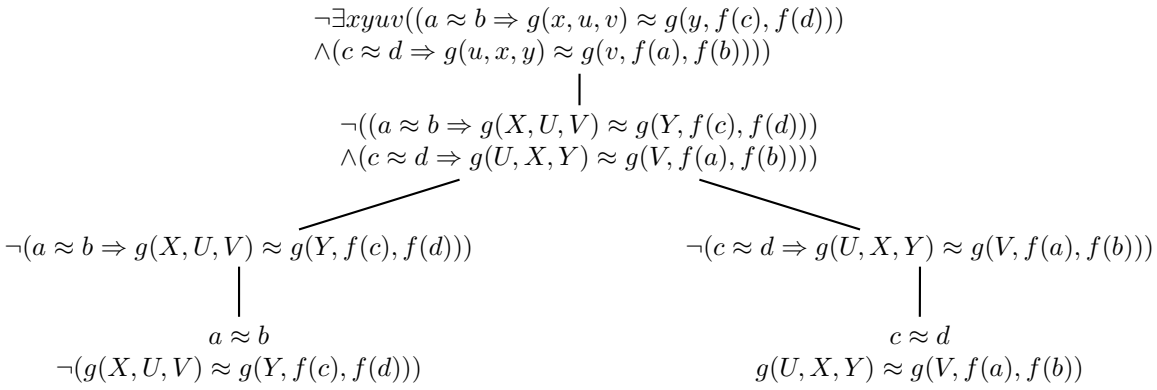


FIG. 2.5 – Dérivation pour un tableau à métavariables

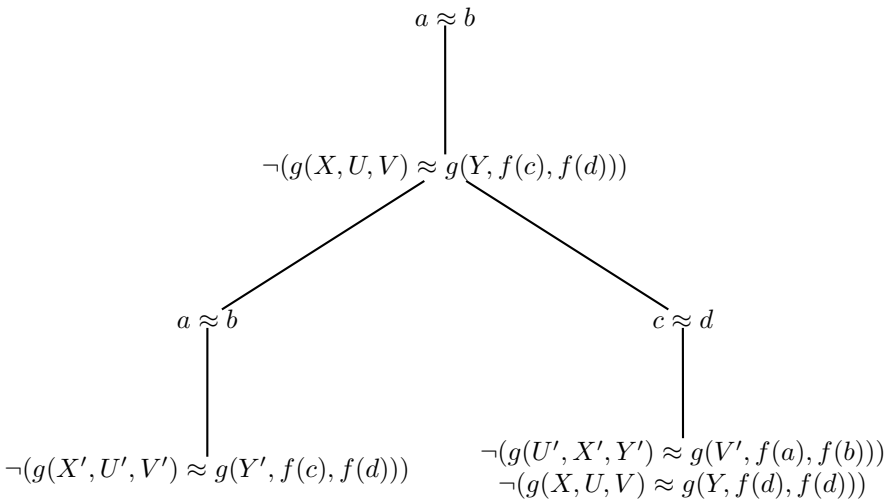


FIG. 2.6 – Partie atomique gauche du tableau 2.5 après γ-expansion supplémentaire

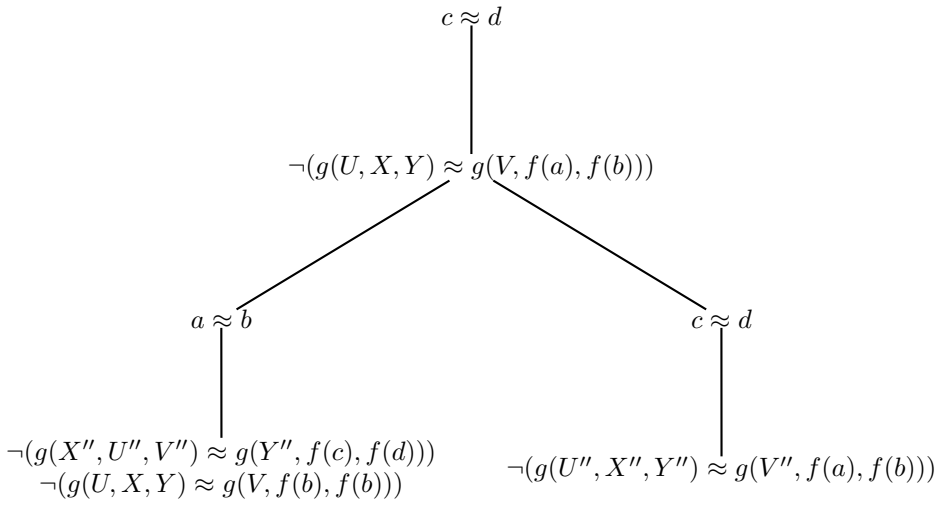


FIG. 2.7 – Partie atomique droite du tableau 2.5 après γ-expansion supplémentaire

$g(V, f(a), f(b))$ comme contrainte-réponse. L'union de ces deux contraintes n'est cependant pas satisfiable. Il faut réappliquer γ pour obtenir le tableau (partie atomique) des figure 2.6 et 2.7.

Ce tableau a quatre branches — chacune comprenant quatre littéraux — que nous nommerons de gauche à droite $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_3$. Considérons les équations suivantes pour ces branches :

$$\begin{aligned}
E(\Gamma_1) &= a \approx b \vdash_\forall g(X', U', V') \approx g(Y', f(c), f(d)) \\
E(\Gamma_2) &= a \approx b, c \approx d \vdash_\forall g(X, U, V) \approx g(Y, f(c), f(d)) \\
E(\Gamma_3) &= a \approx b, c \approx d \vdash_\forall g(U, X, Y) \approx g(V, f(a), f(b)) \\
E(\Gamma_4) &= c \approx d \vdash_\forall g(U", X", Y") \approx g(V", f(a), f(b))
\end{aligned}$$

On peut alors appliquer les règles suivantes (cf. [21])

- (reqs) à $E(\Gamma_1)$;
- (sbrd) puis (reqs) à $E(\Gamma_2)$,
- (sbrd) puis (reqs) à $E(\Gamma_3)$;
- (reqs) à $E(\Gamma_4)$

Notons que dans chaque cas, ce sont les seules règles applicables. Le résultat de ces applications est un ensemble des contraintes réponses suivantes qui correspond à l'union des contraintes produites par BSE sur chaque branche. On obtient l'ensemble de contraintes ci-dessous, où chaque ligne correspond aux

contraintes d'une des quatre branches :

$$\{g(X', U', V') \approx g(Y', f(x), f(d)),$$
$$c \succ d, g(Y, f(c), f(d)) \succ g(X, U, V), c = c, g(X, U, V) = g(Y, f(d), f(d)),$$
$$a \succ b, g(V, f(a), f(b)) \succ g(U, X, Y), g(U, X, Y) = g(V, f(b), f(b)),$$
$$g(U", X", Y") = f(V", f(a), f(b))\}$$

Cet ensemble de contraintes est satisfiable. La substitution suivante résout en effet toutes les contraintes :

$$\{X := f(b), Y := f(b), U := f(d), V := f(d),$$
$$X' := b, Y' := b, U' := f(c), V' := f(d),$$
$$X" := f(a), Y" := f(b), U" := d, V" := d\}$$

2.3 D'autres perspectives

En dehors des deux grandes familles présentées auparavant, il existe d'autres propositions pour traiter l'égalité dans le tableau. Nous allons détailler deux autres propositions : la méthode de modification et la méthode d'élimination de l'égalité. La première cherche à transformer les égalités de la méthode de preuve de façon à ce qu'elle ne soit plus que des prédicats « comme les autres ». La seconde essaie de minimiser l'impact global de l'égalité sur le reste du raisonnement en restant le plus local possible.

2.3.1 Modification de Brand

Dans la famille des méthodes qui consistent à transformer les données en entrée, Brand ([13]) est à l'origine d'une méthode dite de modification. Ce type de méthodes n'est par ailleurs pas vraiment spécifique aux tableaux. L'idée est d'éliminer le recours à des axiomes de substitution dans les fonctions et prédicats puis d'éliminer le besoin d'axiomes de symétrie et de transitivité pour l'égalité.

Nous verrons ici une version adaptée aux tableaux ([8]) car la version originelle travaille sur des formes clausales. Les formules ici sont supposées être en forme normale négative skolemisée (*SNNF*). Avec cette hypothèse, la modification à la Brand est définie de la façon suivante :

Définition 6 (E-modification)**.** Soit Ψ une formule en forme normale négative skolémisée. La *E-modification* de Ψ est le résultat de l'application du processus suivant de manière itérative, tant que possible :

1. Si un littéral de la forme $P(..., t, ...)$ (resp. $\neg P(..., t, ...)$) apparaît dans la formule et t n'est pas une variable , remplacer par :

$$\forall x\ (\neg(t \approx x) \vee P(..., x, ...))$$
$$\text{resp. } \forall x\ (\neg(t \approx x) \vee \neg P(..., x, ...))$$

où x est une nouvelle variable.

2. Si une égalité de la forme $f(...,t,...) \approx s$ ou $s = f(...,t,...)$ apparaît dans la formule, telle que t ne soit pas une variable, la remplacer par :

$$\forall x\ (\neg(t \approx x) \vee f(...,x,...) \approx t)$$

où x est une nouvelle variable.

La première phase de modification permet d'enchaîner pour arriver à la transformation souhaitée :

Définition 7 (STE-modification)**.** La *STE-modification* d'une formule Ψ est le résultat de la substitution dans la E-modification Ψ' de Ψ de toutes les égalités $s \approx t$ par

$$\forall x\ ((\neg(t \approx x) \vee s \approx x) \wedge \forall x\ (\neg(s \approx x) \vee t \approx x)$$

où x est une nouvelle variable.

Appliquons ces transformations sur un exemple tiré de [8] pour éclaircir la transformation.

Exemple 6. La E-modification de $\forall x\ \forall y\ (f(x,y) \approx g(a))$ est :

$$\forall x\ \forall y\ \forall z\ (\neg(a \approx z) \vee f(x,y) \approx g(z))$$

Par conséquent sa STE-modification est :

$$\forall x\ \forall y\ \forall z\ (\neg(a \approx z) \vee (\quad \begin{array}{l} \forall u\ (\neg(f(x,y) \approx u) \vee g(z) \approx u) \wedge \\ \forall u\ (\neg(g(z) \approx u) \vee f(x,y) \approx u))) \end{array}$$

Pour prouver l'E-instasfiabilité d'une formule STE-modifée, seul l'axiome de refexivité reste nécessaire : la symétrie, la transitivité et la monotonicité ne sont plus nécessaires.

Theorème 7. *Soit Ψ une formule close en SNNF, et soit Ψ' la STE-modification de Ψ. Alors Ψ est E-insatisfiable si et seulement si*

$$\Psi' \wedge \forall x(x \approx x)$$

est insatisfiable.

Une version améliorée de ces transformations existe sous le nom d'*élimination de contraintes d'égalité* [6].

2.3.2 Élimination de l'égalité à la Degtyarev-Voronkov

Les méthodes basées sur l'ajout de règles d'expansion ainsi que celles fondées sur des techniques d'unification ont une caractéristique commune : la substitution que l'on cherche à appliquer au tableau est *globale*. Par opposition, la méthode dite d'élimination de l'égalité fonctionne de façon *locale*.

L'idée ([21]) est d'associer aux méthodes fondées sur le calcul des séquents un solveur d'équations par superposition basique. Si la superposition basique

était incluse dans les règles de tableaux, les contraintes induites influenceraient l'ensemble du tableau. Nous allons voir par la suite que ce n'est pas tout-à-fait le cas ici.

Cette méthode est en fait seulement partiellement locale pour la méthode des tableaux. Ceci est dû à la méthode des tableaux elle-même qui est une méthode globale.

Base théorique

Afin de pouvoir présenter la méthode d'élimination de l'égalité, il faut introduire un nouvel ensemble de définitions spécifiques.

Notons au préalable qu'ici, une clause signifie simplement un (multi)-ensemble de littéraux. Par ailleurs la méthode postule que les formules sont skolémisées et en forme normale négative.

Définition 8 (Plus petite sur-formule conjonctive)**.**
- Une (occurrence d'une) sous-formule φ de ξ est dite *conjonctive* si elle apparaît dans une sous-formule $\varphi \wedge \psi$ ou $\psi \wedge \varphi$.
- Une formule φ est appelé *sur-formule* d'une formule ψ si ψ est sous-formule de φ.
- Soit φ une sous-formule de ξ. Une *sur-formule* conjonctive de φ dans ξ est une sur-formule ψ de φ qui est également une sous-formule conjonctive de ξ.
- La *plus petite sur-formule conjonctive* de φ dans ξ est la sur-formule conjonctive ψ de φ qui est telle que toute autre sur-formule conjonctive de φ est une sur-formule de ψ.

L'utilisation de cette notion de sur-formule conjonctive permet d'avoir certaines propriétés intéressantes :
- Si φ est une formule et Ψ est sa plus petite sur-formule conjonctive alors $\varphi \vdash \Psi$.
- La restriction aux sur-formules conjonctives permet d'éliminer en une fois des chaînes déterministes de α et γ règles (voir [21]).

Définition 9 (ξ-nom)**.** On énumère les sous-formules conjonctives $\xi_1, \ldots, \xi_n$ de ξ dans l'ordre de leur apparition dans ξ de telle manière à ce qu'on puisse parler sans ambigüite de la $k$ième (sous)-formule conjonctive ξ_k de ξ.

Soit maintenant $\mathcal{A}_1, \ldots, \mathcal{A}_n$ des symboles de prédicats sans occurrence dans ξ. Alors la formule atomique $\mathcal{A}_k(x_1, \ldots, x_m)$ est le *ξ-nom d'une sous-formule* φ de ξ si :
- ξ_k est la plus petite sur-formule conjonctive de φ ;
- $x_1, \ldots, x_m$ sont les variables libres de ξ_k dans leur ordre d'apparition dans ξ_k

Les ξ-noms ont pu être calculés. Nous allos maintenant décrire un calcul dit de fermetures.

On va tenter par la suite déliminer l'égalité des fermetures en les transformant à partir de fermetures initiales et d'un calcul de superposition basique.

Les fermetures initiales sont calculées de la façon suivante :

1. si $s \approx t$ (resp. $s \not\approx t$) apparaît dans ξ et C est l'ensemble de ξ-noms de cette occurrence de $s \approx t$ (resp. $s \not\approx t$) la fermeture initiale est $s \not\approx t, C \cdot \emptyset$ (resp. $s \approx t, C \cdot \emptyset$
2. si les littéraux $P(s_1, ..., t_n)$ et $\neg P(t_1, ..t_n)$ apparaissent dans ξ et C_1, C_2 sont leurs ensembles de ξ-noms. Alors la fermeture initiale est $s_1\rho \not\approx t_1, ..., s_n\rho \not\approx t_n, C_1\rho, C_2 \cdot \emptyset$, avec ρ une substitution renommant les variables telles les variables de $C_1\rho$ et C_2 soient disjointes.

À partir de ces fermetures initiales, on peut calculer des (clauses)-solutions en utilisant les règles ci-dessous.

Les règles de superposition basique sont :

$$\frac{s \approx t, C \cdot \sigma_1 \qquad u[s'] \approx v, D \cdot \sigma_2}{u[t] \approx v, C, D \cdot \sigma_1\sigma_2\Theta} \text{ (brs)} \qquad \frac{s \approx t, C \cdot \sigma_1 \qquad u[s']\neg \approx v, D \cdot \sigma_2}{u[t]\neg \approx v, C, D \cdot \sigma_1\sigma_2\Theta} \text{ (bls)}$$

1. Θ est le mgu de $s\sigma_1$ et de $s'\sigma_2$.
2. $t\sigma_1\neg \geq s\sigma_1\Theta$ et $v\sigma_2\neg \geq u[s']\sigma_2\Theta$.
3. s' n'est pas une variable.
4. $u[s'] \neq v$ est l'inéquation la plus à gauche dans la seconde prémisse de la règle (bls).

La règle de *solution* est la suivante :

$$\frac{s \not\approx t, C \cdot \sigma}{C \cdot \sigma mgu(s\sigma, t\sigma)} \text{ (eqs)}$$

Une *(clause-)solution* est un ensemble de littéraux $C\sigma$ tel que C ne contienne aucune égalité et que $C \cdot \sigma$ soit dérivable à partir de fermetures initiales dans le système composé des règles d'inférence (bls), (brs) et (eqs) qu'on a données ci-dessus.

Après avoir obtenu des clauses-solutions, on peut réduire toute démonstration en logique avec égalité vers une démonstration en logique sans égalité. Nous allons appliquer cette méthode aux tableaux.

L'élimination de l'égalité pour les tableaux

L'élimination de l'égalité peut s'appliquer aussi bien sur des variantes de résolution ou sur la méthode inverse. La méthode des tableaux avec métavariables doit être modifiée pour utiliser les ξ-noms au lieu des formules. Ceci permet d'avoir une méthode de tableau plus efficace car elle travaille sur des occurrences de sous-formules au lieu de travailler sur les sous-formule elles-mêmes.

Définition 10 (ξ-tableau, unification sous-ensemble)**.** Une ξ-branche est un multi-ensemble d'atomes de la forme $\mathcal{A}_k(t_1, \ldots, t_m)$ tel que $\mathcal{A}_k(x_1, \ldots, x_m)$ soit un ξ-nom d'une sous-formule de ξ.

Un ξ-tableau est un multi-ensemble $\{\mathcal{B}_1, \ldots, \mathcal{B}, n\}$ de ξ-branches $\mathcal{B}_1 \mid ... \mid \mathcal{B}_n$.

Un unificateur sous-ensemble est une substitution σ telle que $\mathcal{B}_1\sigma \subseteq \mathcal{B}_2\sigma$. Il est dit minimal s'il est minimal par rapport à l'ordre $\leq$ sur tous les unificateurs sous-ensemble de $\mathcal{B}_1$ contre $\mathcal{B}_2$.

Le calcul considéré sur les ξ-tableaux comporte deux règles d'inférence : l'une d'expansion (décrite ici de façon plus simple sous la forme de deux sous-cas) et l'autre de fermeture.

Soit Γ_1, Γ_2 et Γ les ensembles de ξ-noms de φ, ψ et $\varphi \wedge \psi$, cette dernière étant une sous-formule de ξ. Alors Γ est soit $\emptyset$ soit le multi-ensemble singleton $\{\mathcal{A}_i(\overline{x})\}$. Quant à Γ_1 et Γ_2, ils sont non-vides car φ et ψ sont des sous-formules conjonctives de $\varphi \wedge \psi$. Par conséquent, $\Gamma_1 = \{A_j(\overline{y})\}$ et $\Gamma_2 = \{A_k(\overline{z})\}$.

1. Si $G = \emptyset$, alors la règle d'expansion est la suivante :

$$\frac{\mathcal{B}_1 \mid \mathcal{B}_2 \mid ... \mid \mathcal{B}_m}{\mathcal{B}_1 A_j(\overline{y}) \mid \mathcal{B}_1 A_k(\overline{z}) \mid \mathcal{B}_2 \mid ... \mid \mathcal{B}_m} (te_{\xi_\emptyset})$$

2. Si $\Gamma = \mathcal{A}_i(\overline{x})$ alors la règle d'expansion devient :

$$\frac{\mathcal{B}_1 \mathcal{A}_i(\overline{t}) \mid \mathcal{B}_2 \mid ... \mid \mathcal{B}_m}{\mathcal{B}_1 A_j(\overline{y})\sigma \mid \mathcal{B}_1 A_k(\overline{z})\sigma \mid \mathcal{B}_2 \mid ... \mid \mathcal{B}_m} (te_{\xi_{\mathcal{A}_i(\overline{x})}})$$

avec $\sigma = \{\overline{x} := \overline{t}\}$. Cette règle correspond effectivement à la β-règle de Smullyan.

Soit Γ une clause (ou branche) solution. La règle de fermeture de branche est la suivante :

$$\frac{\mathcal{B}_1 \mid \mathcal{B}_2 \mid ... \mid \mathcal{B}_m}{(\mathcal{B}_2 \mid ... \mid \mathcal{B}_m)\sigma} (\odot_\xi)$$

où σ est un unificateur sous-ensemble minimal de Γ contre $\mathcal{B}_1$. Il est supposé que les variables du tableau sont disjointes des variables de Γ, ce qui peut être fait en renommant les variables de Γ.

Le calcul de tableau T_ξ est le calcul composé des règles $(\odot_\xi), (te_{\xi_\emptyset}), (te_{\xi_{\mathcal{A}_i(\overline{x})}}$.

Le théorème suivant peut-être démontré :

Theorème 8 (Correction et complétude)**.** *Les propositions suivantes sont équivalentes :*

1. *ξ est démontrable en logique du premier ordre avec égalité.*
2. *Le tableau vide $\odot$ est dérivable dans le calcul T_ξ.*

2.4 Conclusion

Nous avons vu les différentes approches préconisées pour le traitement de l'égalité dans les méthodes de tableaux que ce soit de façon partielle par l'ajout de règles, de façon globale via des contraintes d'unification ou encore par des méthodes visant à restreindre les problèmes d'explosion combinatoire de légalité

en la traitant comme un prédicat ordinaire après modification des entrées ou en l'éliminant.

Les propositions relativement simples des débuts ont comme souvent été améliorées et dépassées. Grâce à des théories restreignant de plus en plus les choix possibles, on peut réduire considérablement l'indéterminisme lié à la recherche de preuve en logique avec égalité.

D'un point de vue purement pratique, il faut cependant se poser la question suivante : jusqu'à quel point la réduction de l'espace de recherche (qui est visée par les différents améliorations) n'est-elle pas contrebalancée par les calculs supplémentaires souvent induits par les formalismes plus récents par rapport aux anciens ?

Chapitre 3

Déduction modulo

Le chapitre qui suit présente les notions de base nécessaires à la compréhension du cadre formel de la déduction modulo. Après l'exposition des motivations et des termes utilisés, on verra comment (en théorie) mêler d'un côté un outil déductif comme le calcul des séquents avec de l'autre, la réécriture sur des termes mais également sur des propositions. Certains résultats du domaine seront évoqués, avec notamment l'expression de la logique d'ordre supérieur en déduction modulo et une méthode de preuve fondée sur la résolution (ENAR). Enfin, nous discuterons les différences entre calcul et déduction avant de présenter les préoccupations actuelles dans le domaine de la preuve modulo.

3.1 Introduction

La déduction modulo permet d'établir un cadre théorique dans lequel le raisonnement est effectué modulo une congruence composée de règles de réécriture sur les termes et les propositions et d'axiomes équationnels. Ce cadre permet de délimiter de façon claire la partie calculatoire d'un raisonnement, modélisée par la congruence, de sa partie purement déductive. L'intégration de calcul dans le système déductif donne en premier lieu des preuves plus courtes, mais aussi des preuves plus proches de celles réalisées à la main. En effet, le calcul, considéré comme trivial, est « caché » dans la preuve et ce qu'on obtient en tant que preuve ne comporte en fin de compte que les étapes de déduction, considérées comme importantes.

La problématique est plus clairement illustrée encore par un exemple classique : la preuve de $2 + 2 = 4$ dans l'arithmétique de Peano. Répondre à ce problème est *trivial* par le calcul : en effet $2 + 2$ donne 4 et, bien évidemment, $4 = 4$. La même preuve dans le calcul des séquents avec la théorie de Peano est beaucoup plus longue.

En déduction modulo, les règles de déduction classique sont étendues pour gérer une congruence. Ainsi, si $P \equiv Q$ par cette congruence, toute règle utilisant P s'appliquera de la même manière sur une entrée contenant en fait Q. Donc la

preuve du séquent

$$\Gamma, 4 = 4 \vdash 2 * 2 = 4, \Delta$$

où $2 * 2 \equiv 4$ dans la congruence sera simplement

$$\frac{}{\Gamma, 4 = 4 \vdash 2 * 2 = 4, \Delta} \text{ axiome}$$

On constate que la preuve en déduction modulo est aussi simple que celle par le calcul direct : elle est triviale.

3.2 Raisonner en déduction modulo

Nous allons introduire ici les notions élémentaires de la déduction modulo. On va ensuite formaliser le raisonnement dans ce cadre. Pour ce faire, on s'appuie sur le calcul des séquents, que l'on a déjà dans le cadre des méthodes de tableaux.

Le calcul des séquents pour la déduction modulo est une extension du calcul des séquents du premier ordre. Nous allons voir qu'il est possible d'intégrer la congruence directement dans les règles d'inférence ou en ajoutant des règles de conversion au calcul des séquents classique.

3.2.1 Définitions

Définition 11. Une *règle de réécriture sur les termes* est une paire de termes $g \to d$. Les variables du membre droit d doivent apparaître dans le membre gauche g.

Un *axiome équationnel* est une paire de termes $g = d$.

Une *règle de réécriture propositionnelle* (ou sur les propositions) est une paire $P \to Q$. P doit être une proposition atomique alors que Q est une proposition arbitraire. Les variables libres apparaissant dans le membre droit Q doivent apparaître dans le membre gauche P.

Une règle de réécriture sur les termes est par exemple

$$x * 0 \to 0$$

Un exemple d'axiome équationnel est la commutativité de la multiplication :

$$x * y = y * x$$

Enfin, présentons une règle de réécriture propositionnelle :

$$x * y = 0 \to x = 0 \vee y = 0$$

Remarquons que les variables x et y sont libres dans la règle ci-dessus.

Un *système de réécriture de classe* (abrégé en système de réécriture par la suite) est ici une paire $\mathcal{RE}$ composée de :

- $\mathcal{R}$: un ensemble de règles de réécriture propositionnelles,
- $\mathcal{E}$: un ensemble de règles de réécriture sur les termes et d'axiomes équationnels.

Définition 12. Soit $\mathcal{R}$ un ensemble de règles de réécriture propositionnelles.

La proposition Ψ se $\mathcal{R}$-réécrit en φ (ou Ψ se réécrit par $\mathcal{R}$ en φ) si pour une règle $g \to d \in \mathcal{R}$, une occurrence ω dans Ψ et une substitution σ données.

- $\Psi = \Psi[\sigma(g)]_\omega$
- et $\varphi = \Phi[\sigma(d)]_\omega$

La substitution σ ne s'applique que sur les variables libres de la règle et ne doit pas entraîner de captures.

Le membre droit Q peut contenir des quantificateurs : il faut faire une α-conversion de ces variables universellement quantifiées pour éviter toute capture de variables pour la proposition filtrée par le membre gauche.

De façon usuelle, la fermeture réflexive et transitive de la relation $\to_{\mathcal{R}}$ sera notée $\to^*_{\mathcal{R}}$. Par ailleurs, les relations $=_{\mathcal{E}}$ et $=_{\mathcal{RE}}$ sont les congruences générées respectivement par $\mathcal{E}$ et $\mathcal{R} \cup \mathcal{E}$.

On peut maintenant définir la notion de $\mathcal{RE}$-réécriture.

Définition 13. Soit un système de réécriture de classe $\mathcal{RE}$. La proposition Ψ se $\mathcal{RE}$-réécrit en φ si, pour une règle de réécriture $g \to d \in R$, une substitution σ, une proposition Φ et une occurrence ω de Ψ données, $\Psi =_{\mathcal{E}} \Phi[\sigma(g)]_\omega$ et $\varphi =_{\mathcal{E}} \Phi[\sigma(d)]_\omega$.

En utilisant ces définitions, on peut définir une extension du calcul des séquents de Gentzen.

3.2.2 Extension de LK

Le calcul des séquents modulo est donné en figure 3.1. La condition ajoutée à côté des règles traditionnelles de séquents permet de travailler modulo une congruence définie par un système de réécriture de classe $\mathcal{RE}$. Ces règles sont plus libérales que les règles de Gentzen : en particulier, lorsque le système $\mathcal{RE}$ est vide, on récupère le calcul des séquents classique.

La présentation du calcul des séquents modulo par inclusion de conditions de congruence au niveau des règles d'inférence est celle qui est usuellement adoptée. On peut cependant aussi choisir d'ajouter uniquement deux règles de conversion au calcul des séquents de Gentzen qui gèrent explicitement les étapes de réécriture 3.2. Les deux options sont équivalentes (voir [39]).

3.2.3 Relier les séquents modulo aux séquents de Gentzen

Un des résultats importants de [30] consiste à montrer le lien entre les preuves obtenues dans les deux calculs de séquents. En effet, il est démontré que toute proposition prouvable dans le calcul des séquents modulo a une preuve dans le calcul des séquents usuels et réciproquement. Pour ce faire on se donne la définition suivante :

Définition 14. Un ensemble d'axiomes $\mathcal{S}$ et un système de réécriture $\mathcal{RE}$ sont dits *compatibles* si

$$\frac{}{P \vdash Q}\ \text{axiom} \qquad P =_{\mathcal{RE}} Q$$

$$\frac{\Gamma, P \vdash \Delta \qquad \Gamma \vdash Q, \Delta}{\Gamma \vdash \Delta}\ \text{cut} \qquad P =_{\mathcal{RE}} Q$$

$$\frac{\Gamma, Q_1, Q_2 \vdash \Delta}{\Gamma, P \vdash \Delta}\ \text{contr-l} \qquad P =_{\mathcal{RE}} Q_1 =_{\mathcal{RE}} Q_2$$

$$\frac{\Gamma \vdash Q_1, Q_2, \Delta}{\Gamma \vdash P, \Delta}\ \text{contr-r} \qquad P =_{\mathcal{RE}} Q_1 =_{\mathcal{RE}} Q_2$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, P \vdash \Delta}\ \text{weak-l}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash P, \Delta}\ \text{weak-r}$$

$$\frac{}{\Gamma, P \vdash \Delta}\ \bot\text{-l, si } P =_{\mathcal{RE}} \bot$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, R \vdash \Delta}\ \wedge\text{-l} \qquad R =_{\mathcal{RE}} (P \wedge Q)$$

$$\frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash R, \Delta}\ \wedge\text{-r} \qquad R =_{\mathcal{RE}} (P \wedge Q)$$

$$\frac{\Gamma, P \vdash \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, R \vdash \Delta}\ \vee\text{-l} \qquad R =_{\mathcal{RE}} (P \vee Q)$$

$$\frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash R, \Delta}\ \vee\text{-r} \qquad R =_{\mathcal{RE}} (P \vee Q)$$

$$\frac{\Gamma \vdash P, \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, R \vdash \Delta}\ \Rightarrow\text{-l} \qquad \text{si } R =_{\mathcal{RE}} (P \Rightarrow Q)$$

$$\frac{\Gamma, P \vdash Q, \Delta}{\Gamma \vdash R, \Delta}\ \Rightarrow\text{-r} \qquad \text{si } R =_{\mathcal{RE}} (P \Rightarrow Q)$$

$$\frac{\Gamma \vdash P, \Delta}{\Gamma, R \vdash \Delta}\ \neg\text{-l} \qquad R =_{\mathcal{RE}} \neg P$$

$$\frac{\Gamma, P \vdash \Delta}{\Gamma \vdash R, \Delta}\ \neg\text{-r} \qquad R =_{\mathcal{RE}} \neg P$$

$$\frac{\Gamma, \{t/x\}Q \vdash \Delta}{\Gamma, P \vdash \Delta}\ (Q, x, t)\ \forall\text{-l} \qquad P =_{\mathcal{RE}} \forall x Q \text{ et } c \text{ constante fraîche}$$

$$\frac{\Gamma \vdash \{c/x\}Q, \Delta}{\Gamma \vdash P, \Delta}\ (Q, x, c)\ \forall\text{-r} \qquad P =_{\mathcal{RE}} \forall x Q \text{ et } c \text{ constante fraîche}$$

$$\frac{\Gamma, \{c/x\}Q \vdash \Delta}{\Gamma, P \vdash \Delta}\ (Q, x, c)\ \exists\text{-l} \qquad P =_{\mathcal{RE}} \exists x Q \text{ et } c \text{ constante fraîche}$$

$$\frac{\Gamma \vdash \{t/x\}Q, \Delta}{\Gamma \vdash P, \Delta}\ (Q, x, t)\ \exists\text{-r} \qquad P =_{\mathcal{RE}} \exists x Q \text{ et } c \text{ constante fraîche}$$

FIG. 3.1 – Le calcul des séquents modulo

$$\frac{\Gamma \vdash_{\mathcal{RE}} P, \Delta}{\Gamma \vdash_{\mathcal{RE}} Q, \Delta} \text{ si } P =_{\mathcal{RE}} Q$$

$$\frac{\Gamma, P \vdash_{\mathcal{RE}} \Delta}{\Gamma, Q \vdash_{\mathcal{RE}} \Delta} \text{ si } P =_{\mathcal{RE}} Q$$

FIG. 3.2 – Règles de conversion pour LK modulo

- $P =_{\mathcal{RE}} Q$ implique $\mathcal{S} \vdash P \Leftrightarrow Q$
- pour toute proposition $P \in \mathcal{S}$, $\vdash P$

À partir de cette définition, on peut montrer que la génération d'un ensemble d'axiomes compatibles est possible pour tout système de réécriture.

Lemme 3. *Pour tout système de réécriture de classe $\mathcal{RE}$, il existe un ensemble d'axiomes $\mathcal{S}$ tel que $\mathcal{S}$ et $\mathcal{RE}$ soient compatibles.*

Démonstration. Pour toute paire de propositions $P =_{\mathcal{RE}} Q$, prendre

$$\forall x_1, ..., x_n P \Leftrightarrow Q$$

où $x_1, ..., x_n$ sont les variables libres de $P \Leftrightarrow Q$. □

Cela suffit pour démontrer la correction et la complétude des séquents modulo par rapport aux séquents de Gentzen.

Lemme 4. *Soient $\mathcal{RE}$ un système de réécriture de classe et $\mathcal{S}$ un ensemble d'axiomes compatibles avec $\mathcal{RE}$. Alors :*

$$\mathcal{S}, \Gamma \vdash \Delta \Leftrightarrow \Gamma \vdash_{\mathcal{RE}} \Delta$$

Démonstration. Voir [30]. □

3.3 L'ordre supérieur en déduction modulo

Dans les travaux à la base de la déduction modulo, la logique d'ordre supérieur occupe une place particulière. En effet, on trouve dans [29], deux expressions de cette logique sous forme de systèmes de réécriture. Nous allons présenter l'une d'entre elles : HOL-C.

HOL-C est une présentation de l'ordre supérieur en tant que théorie du premier ordre multi-sortée avec égalité. Cette présentation est fondée sur l'utilisation des combinateurs. Les sortes de cette théorie sont les types du λ-calcul simplement typé (HOL-λ) définies inductivement par :

- ι et o sont des sortes ;
- si T et U sont des sortes alors $T \to U$ est une sorte.

La notion de *rang* est définie de la manière suivante :

- Un symbole de fonction f a un rang $(T_1, ..., T_n)U$ s'il a comme arguments n termes de sortes $T_1, ..., T_n$ pour construire un terme de sorte U.

- Un symbole de prédicat P est de rang $(T_1, ..., T_n)$ s'il a comme arguments n termes de sortes $T_1, ..., T_n$.

Au côté de l'égalité le langage contient aussi :

- Les applications ou symboles de fonction $\alpha_{T,U}$ de rang $(T \to U, T)U$.
- Le symbole de prédicat (unaire) ϵ de rang (o) transformant un terme t de sorte o en proposition.
- Pour tout n-uplet de variables $x_1, ..., x_n$ de sortes $T_1, ..., T_n$ et tout terme t de sorte U formés de ces variables et des symboles d'application, on pose un combinateur (constant) :

$$x_1, ..., x_n \mapsto t \text{ de sorte } T_1 \to ... \to T_n \to U$$

- Les constantes :
  - $\dot{\Rightarrow}, \dot{\wedge}, \dot{\vee}$ de sorte $o \to o \to o$,
  - $\dot{\neg}$ de sorte $o \to o$ et $\dot{\bot}$ de sorte o,
  - $\dot{\forall}_T$ et $\dot{\exists}_T$ de sorte $(T \to o) \to o$.

En utilisant ce langage, on peut orienter les axiomes représentant ces constantes en une congruence définie par le système de réécriture suivant :

$$\begin{array}{rcl}
((x_1, ..., x_n \mapsto t)u_1...u_n) & \to & t\{x_1 := u_1, ..., x_n := u_n\} \\
\epsilon(\dot{\Rightarrow}xy) & \to & \epsilon(x) \Rightarrow \epsilon(y) \\
\epsilon(\dot{\wedge}xy) & \to & \epsilon(x) \wedge \epsilon(y) \\
\epsilon(\dot{\vee}xy) & \to & \epsilon(x) \vee \epsilon(y) \\
\epsilon(\dot{\neg}x) & \to & \neg\epsilon(x) \\
\epsilon(\dot{\bot}) & \to & \bot \\
\epsilon(\dot{\forall}x) & \to & \forall y \epsilon(xy) \\
\epsilon(\dot{\exists}x) & \to & \exists y \epsilon(xy)
\end{array}$$

$\dot{\Rightarrow}xy$ est réellement $\alpha(\alpha(\dot{\Rightarrow}, x), y)$ et $\epsilon(\dot{\Rightarrow}xy)$ est une proposition atomique.

Les axiomes d'extensionnalité

$$\forall f \forall g((\forall x((fx) = (gx)) \Rightarrow f = g$$
$$\forall x \forall y((\epsilon(x) \Leftrightarrow \epsilon(y)) \Rightarrow x = y)$$

sont nécessaires pour obtenir l'équivalence entre cette formulation et le λ-calcul simplement typé extensionnel. Cette équivalence est en fait l'expression du fait que toute proposition démontrable dans un formalisme l'est aussi dans l'autre (et réciproquement) : les preuves obtenues ne sont généralement pas similaires.

Il faut tout de même noter que l'autre présentation de la logique d'ordre supérieur de [29], appelée HOL-$\lambda\sigma$ permet de remédier à ces problèmes. HOL-$\lambda\sigma$ est en effet intentionnellement équivalente à HOL-λ et les preuves sont parallèles.

L'intérêt de ces transformations de l'ordre supérieur en logique du premier ordre modulo devient évident lorsqu'elles sont utilisées dans le domaine de la preuve automatique, où l'on peut dès lors utiliser toutes les optimisations disponibles au premier ordre pour un raisonnement à l'ordre supérieur.

3.4 ENAR : une méthode de résolution modulo

La déduction modulo semble un formalisme bien adapté à l'automatisation de preuves puisque l'idée est de calculer au maximum au lieu de déduire (qui nécessite des approximations heuristiques). Les preuves ainsi obtenues sont généralement plus courtes, plus lisibles aussi et la recherche de preuves devrait être plus rapide puisque le calcul est la capacité première des ordinateurs, .

Afin d'utiliser la déduction modulo, Dowek, Hardin et Kirchner ([30]) ont formalisé une méthode pour celle-ci fondée sur la résolution : ENAR (voir [5] pour une introduction à la résolution).

La méthode ENAR (pour Extended Narrowing And Resolution) opère en deux temps : tout d'abord la proposition à prouver est mise en forme clausale et ensuite, les clauses ainsi obtenues sont utilisées pour essayer d'obtenir la clause vide, synonyme de réussite de la preuve, au moyen des deux règles de Narrowing et de Résolution étendus.

3.4.1 Mise en forme clausale

Nous rappelons l'algorithme de mise en forme clausale en définition 15. On peut remarquer que les propositions sont munies en exposant d'un *label*, qui permet de garder explicitement en mémoire le contexte dans lequel se déroule la mise en forme clausale afin de pouvoir skolemiser les sous-formules rencontrées. Cette transformation a deux propriétés essentielles : elle est correcte d'une part, et elle termine d'autre part.

Définition 15 (Forme clausale)**.**

$$
\begin{array}{lcl}
\Gamma, (\Delta, (P \wedge Q)^l) & \rightarrow & \Gamma, (\Delta, P^l), (\Delta, Q^l) \\
\Gamma, (\Delta, \neg(P \Rightarrow Q)^l) & \rightarrow & \Gamma, (\Delta, , P^l), (\Delta, \neg Q^l) \\
\Gamma, (\Delta, \neg(P \vee Q)^l) & \rightarrow & \Gamma, (\Delta, \neg P^l), (\Delta, \neg Q^l) \\
\Gamma, (\Delta, (P \vee Q)^l)) & \rightarrow & \Gamma, (\Delta, P^l, Q^l) \\
\Gamma, (\Delta, (P \Rightarrow Q)^l) & \rightarrow & \Gamma, (\Delta, \neg P^l, Q^l) \\
\Gamma, (\Delta, \neg(P \wedge Q)^l) & \rightarrow & \Gamma, (\Delta, \neg P^l, \neg Q^l) \\
\Gamma, (\Delta, \forall x\ P^l) & \rightarrow & \Gamma, (\Delta, P^{l,x}) \\
\Gamma, (\Delta, \neg(\exists x\ P^l)) & \rightarrow & \Gamma, (\Delta, \neg P^{l,x}) \\
\Gamma, (\Delta, \exists x\ P^l) & \rightarrow & \Gamma, (\Delta, P[x := f(l)]^l) \\
\Gamma, (\Delta, \neg(\forall x\ P^l)) & \rightarrow & \Gamma, (\Delta, \neg P[x := f(l)]^l) \\
\Gamma, (\Delta, \neg\neg P^l) & \rightarrow & \Gamma, (\Delta, P^l) \\
\Gamma, (\Delta, \bot^l) & \rightarrow & \Gamma, \Delta \\
\Gamma, (\Delta, \neg\bot^l) & \rightarrow & \Gamma
\end{array}
$$

- Dans les règles précédentes, la symbole de Skolem f introduit par l'élimination du quantificateur existentiel ($\exists$ ou $\neg\forall$) est frais.
- La variable ajoutée au label dans les règles d'élimination du quantificateur universel ($\forall$ ou $\neg\exists$) ne doit pas apparaître dans le label l de la proposition.

3.4.2 Narrowing étendu et résolution étendue

Avant de présenter les règles étendues de la résolution pour la déduction modulo, il est nécessaire de définir certaines notations qui apparaissent dans ces inférences.

Définition 16. Soit $\mathcal{E}$ une théorie équationnelle, une équation modulo $\mathcal{E}$ est une paire de termes ou de propositions atomiques notée $s \stackrel{?}{=}_{\mathcal{E}} t$.

Une substitution σ est une $\mathcal{E}$-solution de $s \stackrel{?}{=}_{\mathcal{E}} t$ si $s\sigma =_{\mathcal{E}} t\sigma$. C'est une solution d'un système d'$\mathcal{E}$-équations E si elle solution de toutes les équations de ce système.

Définition 17 (Clause avec contraintes)**.** Une clause avec contraintes est une paire notée $U[C]$ où U est une clause et C un ensemble d'équations dénommées contraintes. Cela permet de caractériser l'ensemble des instances de U par les solutions de C.

ENAR comporte effectivement deux règles décrites dans la figure 3.3 qui permettent d'incorporer la congruence $\mathcal{RE}$ dans le raisonnement. Les inférences sont effectuées sur des clauses avec contraintes.

Résolution étendue

$$\frac{\{P_1, ..., P_n, Q_1, ..., Q_m\}[C_1] \qquad \{\neg R_1, ..., \neg R_p, S_1, ..., S_n\}[C_2]\}}{\{Q_1, ..., Q_m, S_1, ..., S_n\}[C_1 \cup C_2 \cup P_1 \stackrel{?}{=}_{\mathcal{E}} P_2, ..., P_1 \stackrel{?}{=}_{\mathcal{E}} P_n, P_1 \stackrel{?}{=}_{\mathcal{E}} R_1, ..., P_1 \stackrel{?}{=}_{\mathcal{E}} R_p]}$$

Narrowing étendu

$$\frac{U[C]}{U'[C \cup U_{|\omega} \stackrel{?}{=}_{\mathcal{E}} l]} \text{ si } l \to r \in \mathcal{R},\ U_{|\omega} \text{ proposition atomique et } U' \in cl(U[r]_\omega)$$

FIG. 3.3 – Narrowing étendu et résolution étendue

La règle de résolution est une extension de la règle usuelle où, au lieu de résoudre directement les équations d'unification, on les mémorise dans un ensemble de contraintes.

La règle de narrowing étendu n'est applicable que sur des propositions atomiques. Le fait que celles-ci puissent être réécrites en des propositions non-atomiques (et donc pas en forme clausale) rend obligatoire la remise en forme clausale.

Définition 18 (Dérivation ENAR)**.** Soit K un ensemble de clauses avec contraintes. On notera

$$K \underset{\mathcal{RE}}{\hookrightarrow} U[C]$$

si la clause avec contraintes $U[C]$ peut être dérivée de K en utilisant un nombre fini d'applications des règles de narrowing et de résolution étendus.

3.4.3 Théorème de correction et de complétude

Le théorème central de [30] établit la correction et la complétude de la méthode ENAR par rapport au calcul des séquents modulo de la section 3.2.

Theorème 9 (Correction et complétude d'ENAR)**.** *Soit $\mathcal{RE}$ un système de réécriture de classe tel que la relation $\xrightarrow{\mathcal{RE}}$ soit confluente. Soient $P_1, ..., P_n$ et $Q_1, ..., Q_m$ des formules closes alors on a les implications suivantes :*

- *Si*

$$cl(P_1, ..., P_n, \neg Q_1, ..., \neg Q_m)[\emptyset] \underset{\mathcal{RE}}{\hookrightarrow} \Box[C]$$

avec C un ensemble de contraintes $\mathcal{E}$-unifiable, alors le séquent

$$P_1, ..., P_n \vdash_{\mathcal{RE}} Q_1, ..., Q_m$$

est dérivable.

- *Et inversement si $P_1, ..., P_n \vdash_{\mathcal{RE}} Q_1, ..., Q_m$ est dérivable* sans coupure *alors il existe une dérivation dans ENAR*

$$cl(P_1, ..., P_n, \neg Q_1, ..., \neg Q_m)[\emptyset] \underset{\mathcal{RE}}{\hookrightarrow} \Box[C]$$

où C est un ensemble $\mathcal{E}$-unifiable de contraintes.

Il découle immédiatement de ce théorème un corollaire d'équivalence : si la règle de coupure est redondante dans le calcul des séquents modulo alors

$$\begin{gathered} cl(P_1, ..., P_n, \neg Q_1, ..., \neg Q_m)[\emptyset] \underset{\mathcal{RE}}{\hookrightarrow} \Box[C] \\ \Longleftrightarrow \\ P_1, ..., P_n \vdash_{\mathcal{RE}} Q_1, ..., Q_m \end{gathered}$$

On a aisément les mêmes propriétés sous les mêmes hypothèses avec le calcul des séquents usuel à partir du moment où on a déterminé un ensemble $\mathcal{S}$ d'axiomes compatible avec $\mathcal{RE}$.

La preuve passe par un système intermédiaire EIR dont les règles sont données en figure 3.4 dont on établit la correction et la complétude vis-à-vis du calcul des séquents modulo.

Le système EIR permet de détailler les différents mécanismes « atomiques » : Instanciation de variable, résolution simple (identique), réécriture propositionnelle et conversion sur les termes. Le principe derrière ce système est que la combinaison de ces quatre règles permet d'obtenir les règles d'ENAR.

Les résultats de correction et de complétude sur EIR sont enfin relevées vers ENAR pour obtenir la preuve du théorème 9. Nous allons utiliser un processus similaire dans le chapitre 4

3.5 Le calcul des séquents en déduction modulo

Le calcul des séquents est souvent perçu comme une procédure de déduction. Hormis la remarque (non négligeable) purement sémantique que la dénomination

$$\frac{U}{U[x \mapsto t]} \text{ Instanciation}$$

$$\frac{U}{U} \text{ Conversion si } U =_{\mathcal{E}} U'$$

$$\frac{U}{V} \text{ Réduction si } U \rightarrow_{\mathcal{R}} U' \text{ et } V \in cl(U')$$

$$\frac{U, P \qquad U', \neg P}{U \cup U'} \text{ Résolution identique}$$

FIG. 3.4 – Le système EIR

de cette mécanisation du raisonnement comporte effectivement le mot « calcul », comment peut-on dès lors expliquer que le fragment propositionnel est décidable (par un ordinateur) par la méthode des tableaux (par exemple) ? La réponse est que la méthode de Gentzen comporte effectivement une bonne part de calcul pur.

On peut par conséquent espérer représenter les séquents comme une théorie modulo. C'est une partie du travail de [25] dont nous allons discuter ici. Pour représenter le calcul des séquents, un opérateur explicite de liaison ($\bullet$) entre deux séquents est ajoutée à la (méta)syntaxe. Ainsi la règle $\wedge$-r devient :

$$\frac{\Gamma \vdash P, \Delta \bullet \Gamma \vdash Q, \Delta}{\Gamma \vdash, P \wedge Q, \Delta} \wedge\text{-r},\bullet$$

Ensuite, il s'agit de définir une congruence sur les séquents et d'identifier ce qui est orientable dans la théorie tout en préservant la confluence. En premier lieu, les axiomes d'associativité et de commutativité sur les séquents et ensembles de formules restent non orientés (figure 3.5).

$$E = \begin{cases} \Gamma, (\Delta, \Theta) \approx (\Gamma, \Delta), \Theta \\ \Gamma, \Delta \approx \Delta, \Gamma \\ \\ S_1 \bullet (S_2 \bullet S_3) \approx (S_1 \bullet S_2) \bullet S_3 \\ \mathcal{S}_1 \bullet S_2 \approx S_2 \bullet S_1 \end{cases}$$

FIG. 3.5 – Axiomes d'associativité-commutativité

Les règles de réécriture (propositionnelles) sur les séquents sont alors définies en figure 3.6.

Les règles de réécriture propositionnelles représentant le calcul des séquents sont très proches de la méthode des tableaux du chapitre 1. En fait, hormis les détails syntaxiques de représentation, l'ensemble de règles de Deplagne travaillant sur les formules du premier ordre et le fragment de méthode des tableaux correspondant sont identiques.

$$
\mathrm{ER}_{\mathrm{Seq}}
\begin{cases}
\Gamma, \nabla \leftrightarrows \Gamma \\
\Gamma, \Gamma \leftrightarrows \Gamma \\
S \bullet \diamond \leftrightarrows S \\
S \bullet S \leftrightarrows S \\
\\
\neg\neg P \leftrightarrows P \\
P \wedge P \leftrightarrows P \\
P \vee P \leftrightarrows P \\
P \Rightarrow Q \leftrightarrows \neg P \vee Q \\
\exists x\ P \leftrightarrows \neg\forall x\ \neg P \\
\\
\Gamma, \neg P \vdash Q \leftrightarrows \Gamma \vdash P, Q \\
\neg P \vdash Q \leftrightarrows \nabla \vdash P, Q \\
\Gamma \vdash \neg P, \Delta \leftrightarrows \Gamma, P \vdash \Delta \\
\Gamma \vdash \neg P \leftrightarrows \Gamma, P \vdash \nabla \\
\\
\Gamma, P \wedge Q \vdash \Delta \leftrightarrows \Gamma, P, Q \vdash \Delta \\
P \wedge Q \vdash \Delta \leftrightarrows P, Q \vdash \Delta \\
\Gamma \vdash P \vee Q, \Delta \leftrightarrows \Gamma \vdash P, Q, \Delta \\
\Gamma \vdash P \vee Q \leftrightarrows \Gamma \vdash P, Q \\
\Gamma \vdash P \wedge Q, \Delta \leftrightarrows \Gamma \vdash P, \Delta \bullet \Gamma \vdash Q, \Delta \\
\Gamma \vdash P \wedge Q \leftrightarrows \Gamma \vdash P \bullet \Gamma \vdash Q \\
\Gamma, P \vee Q \vdash \Delta \leftrightarrows \Gamma, P \vdash \Delta \bullet \Gamma, Q \vdash \Delta \\
P \vee Q \vdash \Delta \leftrightarrows P \vdash \Delta \bullet Q \vdash \Delta \\
\\
\Gamma, P \vdash P, \Delta \leftrightarrows \diamond \\
\Gamma, P \vdash P \leftrightarrows \diamond \\
P \vdash P, \Delta \leftrightarrows \diamond \\
P \vdash P \leftrightarrows \diamond \\
\\
\Gamma \vdash \Delta \bullet \Gamma, \Gamma' \vdash \Delta', \Delta \leftrightarrows \Gamma \vdash \Delta \\
\Gamma \vdash \Delta \bullet \Gamma, \Gamma' \vdash \Delta \leftrightarrows \Gamma \vdash \Delta \\
\Gamma \vdash \Delta \bullet \Gamma \vdash \Delta', \Delta \leftrightarrows \Gamma \vdash \Delta \\
\Gamma \vdash \nabla \bullet \Gamma, \Gamma' \vdash \Delta \leftrightarrows \Gamma \vdash \nabla \\
\nabla \vdash \Delta \bullet \Gamma \vdash \Delta', \Delta \leftrightarrows \nabla \vdash \Delta \\
\Gamma \vdash \nabla \bullet \Gamma \vdash \Delta \leftrightarrows \Gamma \vdash \nabla \\
\nabla \vdash \Delta \bullet \Gamma \vdash \Delta \leftrightarrows \nabla \vdash \Delta \\
\nabla \vdash \nabla \bullet \Gamma \vdash \Delta \leftrightarrows \nabla \vdash \nabla
\end{cases}
$$

FIG. 3.6 – Congruence orientée sur les séquents

La gestion des substitutions est faite par un calcul de substitutions explicites fondé sur [1, 18] décrit en figure 3.7

$$
ER_{\forall\exists\sigma} = \left\{
\begin{array}{lll}
\mathrm{app}_f & f(a_1,..,a_n)[s] & \overset{\approx}{\to} \; f(a_1[s],..,a_n[s]) \\
\mathrm{app}_P & P(a_1,..,a_n)[s] & \overset{\approx}{\to} \; P(a_1[s],..,a_n[s]) \\
\mathrm{app}_\neg & (\neg P)[s] & \overset{\approx}{\to} \; \neg(P[s]) \\
\mathrm{app}_\wedge & (P \wedge Q)[s] & \overset{\approx}{\to} \; P[s] \wedge Q[s] \\
\mathrm{app}_\vee & (P \vee Q)[s] & \overset{\approx}{\to} \; P[s] \vee Q[s] \\
\mathrm{app}_\Rightarrow & (P \Rightarrow Q)[s] & \overset{\approx}{\to} \; P[s] \Rightarrow Q[s] \\
\\
\mathrm{abs}_\forall & (\forall P)[s] & \overset{\approx}{\to} \; \forall(P[1 \cdot (s\circ \uparrow)]) \\
\mathrm{abs}_\exists & (\exists P)[s] & \overset{\approx}{\to} \; \exists(P[1 \cdot (s\circ \uparrow)]) \\
\\
\mathrm{id}_t & a[id] & \overset{\approx}{\to} \; a \\
\mathrm{id}_f & P[id] & \overset{\approx}{\to} \; P \\
\mathrm{clos}_t & (a[s])[t] & \overset{\approx}{\to} \; a[s \circ t] \\
\mathrm{clos}_f & (P[s])[t] & \overset{\approx}{\to} \; P[s \circ t] \\
\\
\mathrm{varcons} & 1[a \cdot s] & \overset{\approx}{\to} \; x \\
\mathrm{idl} & id \circ s & \overset{\approx}{\to} \; s \\
\mathrm{shiftcons} & \uparrow \circ (a \cdot s) & \overset{\approx}{\to} \; s \\
\mathrm{assenv} & (s_1 \circ s_2) \circ s_3 & \overset{\approx}{\to} \; s_1 \circ (s_2 \circ s_3) \\
\mathrm{mapenv} & (a \cdot s) \circ t & \overset{\approx}{\to} \; a[t] \cdot (s \circ t) \\
\mathrm{idr} & s \circ id & \overset{\approx}{\to} \; s \\
\mathrm{varshift} & 1 \cdot \uparrow & \overset{\approx}{\to} \; id \\
\mathrm{scons} & 1[s] \cdot (\uparrow \circ s) & \overset{\approx}{\to} \; s
\end{array}
\right.
$$

FIG. 3.7 – Calcul de substitution (pour les quantificateurs)

Il reste les règles sur les quantificateurs de la figure 3.8 qui représentent la partie déductive du processus.

La partie effectivement déductive du calcul des séquents réside donc dans les règles concernant les quantificateurs. On peut en voir une confirmation dans les méthodes de tableau du chapitre 1 : c'est également au niveau des quantificateurs que les méthodes deviennent plus complexes (métavariables, skolemisation). Elles essaient en quelque sorte de devenir « intelligentes » et simulent la déduction au moyen de calculs évolués.

$$
R = \{
\begin{array}{lllll}
 & l\forall_1 & \Gamma, \forall P \vdash \Delta & \rightarrow & \Gamma, P[t \cdot id], \forall P \vdash \Delta \\
 & l\forall_2 & \forall P \vdash \Delta & \rightarrow & P[t \cdot id], \forall P \vdash \Delta \\
 & r\forall_1 & \Gamma \vdash \forall P, \Delta & \rightarrow & \Gamma \vdash P[n \cdot id], \forall P, \Delta \\
 & r\forall_2 & \Gamma \vdash \forall P & \rightarrow & \Gamma \vdash P[n \cdot id], \forall P \\
 & \text{cut} & \Gamma \vdash \Delta & \rightarrow & \Gamma \vdash P, \Delta \bullet \Gamma, P \vdash \Delta \\
 & \Box_{\text{neutral}} & \mathcal{S} \bullet \Box & \rightarrow & \mathcal{S} \\
 & \text{congruent} & \Diamond & \rightarrow & \Box
\end{array}
$$

Dans les règles $l\forall$, t est un terme est un terme quelconque.
Dans les règles $r\forall$, n est un indexe libre frais.

FIG. 3.8 – Règles de déduction

3.6 Développements récents

De nombreux travaux ont été effectués récemment autour de la déduction modulo. La plupart de ces travaux sont centrés autour de deux grands ensembles de questions :

- Quelles théories peut-on exprimer à l'aide de la déduction modulo, et particulièrement au moyen des règles de réécriture sur les propositions ?
- Quelle est l'influence de l'ajout d'une congruence dans le système déductif ? Plus précisément, que se passe-t-il pour la règle de coupure ? Est-elle toujours admissible ? Peut-on normaliser les preuves ?

Nous allons passer en revue brièvement les nouveaux résultats obtenus dans les deux sous-thèmes. On verra que, malgré tout, l'élimination des coupures reste toujours sous-jacente dans ces travaux. Puis on présentera un travail un peu à part touchant à la déduction naturelle modulo.

3.6.1 Théories en déduction modulo

Jusqu'à présent, l'une des difficultés majeures (et donc intéressantes) de la déduction modulo réside dans les moyens de trouver comment passer d'une théorie sous forme d'axiomes à une théorie équivalente mais uniquement calculatoire sous forme de règles de réécriture. Pour l'instant, seules des solutions ad-hoc existent.

Arithmétique de Heyting

Dans [28], l'arithmétique constructive est présentée comme théorie orientée modulo. Afin de transformer les axiomes de l'arithmétique en règles de réécriture, plusieurs étapes de transformation de la théorie axiomatique sont nécessaires :

- Un symbole de prédécesseur $Pred$ est ajouté. Puis un prédicat $Null$.

– L'univers de la théorie est étendu au delà des nombres entiers. On ajoute donc un prédicat N signifiant « être un nombre entier ».
– Une deuxième sorte représentant les classes de nombres entiers est introduite pour exprimer l'égalité et le schéma d'induction.
Pour toute proposition P du langage de l'arithmétique, on ajoute un symbole de fonction $f_{x,y_1,...,y_n,P}$. On obtient en particulier :

$$\forall x \forall y_1 ... \forall y_n (x \in f_{x,y_1,...,y_n} P(y_1, ..., y_n)) \Leftrightarrow P$$

Ces modifications successives de la théorie axiomatique de Heyting permettent de la traduire vers une théorie équivalente sans axiomes ayant la propriété d'élimination des coupures. avec les règles de réécriture suivantes.

$$y = z \rightarrow \forall p (y \in p \Rightarrow z \in p)$$

$$N(n) \rightarrow \forall p (0 \in p \Rightarrow \forall y (N(y) \Rightarrow y \in p \Rightarrow S(y) \in p) \Rightarrow n \in p)$$

$$x \in f_{x,y_1,...,y_n,p}(y_1, ..., y_n) \rightarrow P$$

$$Pred(0) \rightarrow 0 \qquad Pred(S(x)) \rightarrow x$$

$$Null(0) \rightarrow \top \qquad Null(S(x)) \rightarrow \bot$$

$$0 + y \rightarrow y \qquad S(x) + y \rightarrow S(x + y)$$

$$0 \times y \rightarrow 0 \qquad S(x) \times y \rightarrow x \times y + y$$

Théorie des ensembles de Zermelo

Un résultat très récent [26] décrit comment transformer la théorie des ensembles de Zermelo en déduction modulo, c'est-à-dire où les axiomes sont remplacés par des règles de réécriture, de manière telle que le système modulo obtenu puisse être normalisable.

La méthode utilisée pour réussir cette transformation consiste à exprimer la théorie des ensembles comme une théorie sur les graphes pointés. L'égalité est interprétée dans ce contexte en tant que bisimulation et on peut exprimer les axiomes de Zermelo sous forme de primitives sur les graphes. Ces primitives peuvent être exprimées sous forme de règles de réécriture. La théorie ainsi obtenue est une extension conservatrice d'une légère extension de la théorie des ensembles de Zermelo.

3.6.2 Autour des coupures

La notion de coupure est centrale en ce qui concerne les théories du raisonnement. La propriété d'élimination des coupures permet entre de s'assurer de la cohérence du raisonnement. Par ailleurs, reconnaître le « bon endroit » où appliquer une coupure permet de raccourcir une preuve de manière drastique.

On distingue deux notion d'élimination des coupures :

– La *normalisation* donne un algorithme pour passer de toute preuve avec coupure à une preuve sans coupure.
– L'*admissibilité* de la règle de coupure montre qu'il existe une preuve sans coupure correspondant à une preuve avec coupure. Par contre, on ne sait pas toujours calculer la preuve sans coupure en fonction de celle avec coupure.

Les théories exprimées en déduction modulo de la section 3.6.1 sont également accompagnées de preuves d'élimination des coupures. Mais ici nous parlons des travaux qui portent plus spécifiquement sur la règle de coupure dans la déduction modulo.

Normalisation de preuves

L'utilisation de règles de réécriture en déduction modulo influe de manière non négligeable sur l'élimination des coupures. En effet, cette propriété peut être valable pour une formulation des axiomes sous forme de règles de réécriture mais pas pour une autre. L'étude de la notion de coupure en déduction modulo (voir [27]) est particulièrement intéressante, car celle-ci est uniforme pour toute théorie du premier ordre exprimable uniquement à l'aide de règles de réécriture.

En particulier, il peut sembler surprenant que la normalisation des preuves modulo ne soit pas toujours possible même dans le cas d'un système confluent et noetherien. C'est le cas de l'adaptation du paradoxe de Russell :

$$R \in R \rightarrow \forall y(y \simeq R \Rightarrow (y \in R \Rightarrow (A \Rightarrow A)))$$

Dans la théorie définie par cette règle de réécriture, la preuve de $\perp$ n'est pas normalisable.

Afin de caractériser sous quelles conditions la normalisation est possible [27] s'appuie sur la notion de *candidat de réductibilité* pour arriver à définir une notion suffisamment forte, celle de *pré-modèle*.

La construction d'un pré-modèle est en effet une condition suffisante pour la normalisation. Elle est utilisée pour démontrer qu'on peut normaliser les preuves modulo une congruence définie par :

– un système de réécriture sur les termes confluent et noetheriens ;
– un système de réécriture positif, i.e. sans occurrence négative de proposition atomique ;
– la théorie des types simple en déduction modulo (section 3.3).

Élimination des coupures sémantique

L'élimination des coupures peut se montrer de deux manières, comme on l'a déjà énoncé :

– soit on montre qu'il existe une procédure de normalisation des preuves dans la théorie considérée et il existe comme on vient de le voir un critère général pour la déduction modulo pour s'en assurer ;

– soit on veut montrer que la règle de coupure est admissible dans notre théorie, c'est-à-dire que la cohérence de cette théorie n'est pas changée par l'ajout de coupures.

Le deuxième point est l'objet des méthodes sémantiques développées par Hermant ([39, 40]) dans les logique classique et intuitionniste.

Ces méthodes sont fondées sur des extensions des modèles classiques et intuitionnistes usuels aux systèmes de réécriture. Elles permettent de montrer l'admissibilité de la règle de coupure pour des classes entières de systèmes de réécriture. Nous utiliserons ces outils sémantiques de manière assez détaillée dans les chapitres 5 et 6 suivants dans les preuves de complétude.

Élimination des coupures par complétion abstraite

En général, les preuves en déduction modulo sont plus directes (et donc plus simples) que les preuves correspondantes en logique du premier ordre. La contrepartie est assez lourde : l'élimination des coupures dans les preuves modulo peut ne plus être valable même dans des cas a priori aussi peu problématiques que des systèmes de réécriture confluents et noetheriens.

Une méthode de complétion « à la Knuth-Bendix » du système de réécriture permet de récupérer la normalisation des preuves.

Burel et Kirchner [15] montrent comment le calcul des séquents modulo peut être interprété comme une instance d'un système canonique abstrait. Dans ce cadre, les preuves modulo dont on ne peut éliminer les coupures sont des preuves critiques.

L'identification de ces preuves critiques permet de compléter le système de réécriture. Une fois compléter, on sait construire une preuve sans coupure de la preuve critique originelle.

Le processus de complétion abstraite permet donc de retrouver la propriété d'élimination des coupures. Il faut noter que ce n'est pas un processus de normalisation.

3.6.3 Déduction surnaturelle

Le cas de la déduction surnaturelle [63] est un peu à part des thèmes centraux de la déduction modulo. C'est une formulation différente de la déduction naturelle modulo. En déduction surnaturelle, la congruence sur les propositions de la déduction modulo est remplacée par l'introduction de nouvelles règles dans la logique. La congruence sur les termes est elle conservée. Les nouvelles règles d'inférence se comportent en quelque sorte comme des macros que l'on peut utiliser dans les preuves en déduction naturelle. Ainsi le système de réécriture représentant une partie de la théorie des ensembles

$$\mathcal{R} = \left\{ \begin{array}{lcl} X \subseteq Y & \to & \forall x(x \in X \Rightarrow x \in Y) \\ x \in \emptyset & \to & \bot \end{array} \right.$$

est transformé en l'ensemble de règles d'inférence

$$x \notin \mathrm{fv}(\Gamma)\ \frac{\Gamma, x \in X \vdash x \in Y}{\Gamma \vdash X \subseteq Y}\ (\subseteq I) \qquad \frac{\Gamma \vdash t \in \emptyset}{\Gamma \vdash \bot}\ (\emptyset E)$$

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

Lorsqu'on ajoute ces règles d'inférence à la déduction naturelle, il est intéressant de noter que la preuve de $\emptyset \subseteq A$ est plus courte que la preuve en déduction modulo correspondante.

On peut trouver dans [63] une méthode générale pour effectuer la transformation des règles de réécriture propositionnelle en règles d'inférence pour la déduction naturelle. La façon d'opérer permet de garder léquivalence entre déduction surnaturelle, déduction modulo et logique avec axiomes. De plus, sous certains critères, on est certain que la cohérence du système est assurée.

3.7 Conclusion

Nous avons pu voir les concepts nécessaires à la définition du cadre théorique de la déduction modulo. Nous avons pu montrer comment le calcul des séquents est étendu pour intégrer la congruence définie par le système de réécriture de classe modélisant le calcul pour les théories considérées.

La déduction modulo, notamment à travers les règles de réécriture propositionnelle est un outil puissante. On peut y exprimer la logique d'ordre supérieur, l'arithmétique (de Heyting) ou bien encore la théorie des ensembles de Zermelo.

Cependant cette puissance a une contrepartie : la règle de coupure est un problème encore plus central que dans les théories du raisonnement traditionnelles. En effet, même avec des systèmes respectant des critères raisonnables, on ne sait plus normaliser certaines preuves. Grâce aux méthodes sémantiques d'élimination des coupures, on peut cependant déterminer si une preuve sans coupure peut exister pour certaines classes de systèmes.

La distinction entre calcul et raisonnement est la base de la déduction modulo. Cependant, on a pu voir que la frontière peut être relativement floue entre les deux modes de recherche de preuve. Cela permet quand même de penser que l'on peut aussi granulariser de façon intéressante les preuves en déduction modulo en accordant plus ou moins de place aux deux parties.

Chapitre 4

Tableaux analytiques classiques modulo

Ce chapitre présente une formulation de la méthode des tableaux pour la déduction modulo. Plus précisément, ces tableaux s'appuient sur le calcul des séquents modulo pour la logique du premier ordre (voir section 3.2).

La méthode que l'on va présenter ici est une extension de la méthode des tableaux avec métavariables. À partir de cette base, on va ajouter les règles permettant de prendre en compte les systèmes de réécriture de classe. L'idée de cette construction n'est pas sans rapport avec la formulation du calcul des séquents modulo comme calcul des séquents avec règles de conversion supplémentaires.

Formuler une méthode de tableaux pour la déduction modulo permet de s'appuyer plus facilement sur le calcul des séquents sous-jacent que dans le cas de la méthode de résolution ENAR. Cela permet aussi de confirmer le fait que les méthodes de tableaux sont assez facilement extensibles à des nouveaux calculs, à partir du moment où la calcul des séquents correspondant est défini [1].

Nous allons d'abord introduire quelques extensions du langage du premier ordre pour expliciter le traitement des quantificateurs existentiels du langage. On présentera le fragment non modulo des tableaux du premier ordre un peu différente (implicitement plus calculatoire). Les propriétés de cette base commune permettront de passer par la suite au contenu relatif à la déduction modulo. Nous établirons alors la correction et la complétude de la méthode des tableaux modulo ainsi présentée en utilisant des arguments syntaxiques. La propriété de complétude (section 4.2.1) établie est parfois qualifiée de faible car elle énonce simplement que pour toute preuve en calcul des séquents modulo, il existe une preuve correspondante par la méthode des tableaux modulo.

[1] dans le cas précis de la déduction modulo, la résolution a quand même été étendue d'abord

4.1 La méthode TaMed

Les tableaux modulo sont formulés d'une manière un peu inhabituelle :
- ils travaillent sur des formules du premier ordre annotées par des labels ;
- ils opèrent en deux temps distincts : une phase de « normalisation » de l'entrée précède l'application des règles intégrant la déduction modulo.

Nous allons détailler ici ces particularités.

4.1.1 Les labels

La méthode que nous présentons utilise la skolemisation. Or il existe plusieurs formes skolémisées d'une même formule. La formule $\forall x\ \exists y P(0, y)$ dans laquelle la variable x n'apparaît pas peut se skolemiser en $P(0, f)$ ou $P(0, f(x))$ selon le type de skolemisation choisi. Ainsi $P(0, f)$ correspond à une skolemisation dite *interne*[2] — les arguments du symbole de Skolem sont les variables universellement quantifiées ayant effectivement une occurrence dans la formule — et $P(0, f(x))$ est le produit d'une skolemisation *externe* où les arguments de la fonction de Skolem introduite sont toutes les variables universellement quantifiées sous la portée desquelles se trouve la formule (ici $P(0, y)$).

Dans le cas de la déduction modulo, le choix se portera sur l'utilisation de la skolemisation externe, pourtant moins « performante » car elle utilise plus de variables. Ce choix est justifié par la considération suivante : si, dans la congruence, on a une équation comme $x * 0 = 0$, conduisant à avoir une $\mathcal{E}$-équivalence entre $\forall x\ \exists y\ P(0, y)$ et $\forall x\ \exists y\ P(x * 0, y)$, alors les formes skolemisées des deux formules auraient la même arité, ce qui ne serait pas le cas en utilisant la skolemisation interne. Au niveau pratique, le choix de la skolemisation externe est géré par l'utilisation de *labels* qui mémorisent les portées des quantificateurs de chaque sous-formule.

Pour clore l'introduction des labels, on définit une extension du langage du premier ordre pour pouvoir annoter les formules manipulées.

Définition 19 (Formule avec label)**.** Une *formule avec label* est une paire P^l composée d'une formule (P) et d'un ensemble fini de variables l appelé son label.

L'opération classique de substitution est également étendue aux formules avec label.

Définition 20 (Substitution dans les formules avec label)**.** Appliquer une substitution Θ à une formule P^l avec label remplace chaque variable x du label l par les variables libres de Θx.

Définition 21 (Équivalence.Réécriture)**.** Deux formules P^l et Q^j sont $\mathcal{E}$-équivalentes si $P =_{\mathcal{RE}} Q$ et $l = j$.

La formule P^l se réécrit par $\mathcal{R}$ en Q^j si P se $\mathcal{R}$-réécrit en Q et $l = j$.

[2] Les appellations *inner* et outer skolemization sont employées dans [47]

4.1.2 Expansion en forme normale de tableau

Notations

Dans ce chapitre les tableaux sont des multi-ensembles de branches. Et les branches sont des multi-ensembles de formules (du premier ordre avec métavariables) Ainsi un tableau $\mathcal{T}$ composé des branches $\mathcal{B}_1, ..., \mathcal{B}_n$ sera dénoté $\mathcal{T} = \mathcal{B}_1 \mid ... \mid \mathcal{B}_n$. Soient $\mathcal{T}$ et $\mathcal{U}$ deux tableaux, $\mathcal{B}$ et Γ deux branches, et P une formule alors on pose les notations suivantes :

$$\begin{array}{rcl} \mathcal{T} = \mathcal{U} \mid \mathcal{B} & \hat{=} & \mathcal{T} = \mathcal{U} \cup \{\mathcal{B}\} \\ \mathcal{B} = \Gamma, P & \hat{=} & \mathcal{B} = \Gamma \cup \{P\} \end{array}$$

Une branche $\mathcal{B} = \{P_1, ..., P_n\}$ est implicitement interprétée comme la conjonction de ses formules $\mathcal{B}_1 = P_1 \wedge ... \wedge P_n$. On dira qu'une branche est *complètement développée* (ou entièrement développée) lorsqu'elle n'est composée que de propositions littérales. Cette notion de développement complet est étendue aux tableaux : un tableau est complètement développé lorsque chacune de ses branches l'est.

Règles d'expansion

Nous allons maintenant présenter les règles d'expansion de TaMed. Ce sont des règles qui sont similaires à celles des tableaux du premier ordre. Elles s'appliquent cependant aux formules avec labels. Initialement toute formule à un label *vide*. Le label est calculé dynamiquement lors des étapes d'expansion par les γ-règles.

Définition 22 (Règles d'expansion et forme normale de tableau)**.** Mettre un multi-ensemble de formules closes en forme normale de tableau (*tnf*) consiste à :

- associer un label vide à chaque proposition ;
- associer à chaque γ-formule (et γ-sous-formule) un entier n_x représentant le nombre d'expansions autorisées pour cette formule ;
- appliquer les transformations (expansions) ci-dessous

$\beta -$ expansions

$$\begin{array}{lcl}
\mathcal{T} \mid (\mathcal{B}, (P \vee Q)^l)) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}, P^l \mid \mathcal{B}, Q^l \\
\mathcal{T} \mid (\mathcal{B}, \neg(P \wedge Q)^l)) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}, \neg P^l \mid \mathcal{B}, \neg Q^l \\
\mathcal{T} \mid (\mathcal{B}, (P \Rightarrow Q)^l))) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}, \neg P^l \mid \mathcal{B}, Q^l
\end{array}$$

$\alpha -$ expansions

$$\begin{array}{lcl}
\mathcal{T} \mid (\mathcal{B}, (P \wedge Q)^l) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}, P^l, Q^l \\
\mathcal{T} \mid (\mathcal{B}, \neg(P \vee Q)^l) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}, \neg P^l, \neg Q^l \\
\mathcal{T} \mid (\mathcal{B}, \neg(P \Rightarrow Q)^l) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}, P^l, \neg Q^l \\
\mathcal{T} \mid (\mathcal{B}, \neg\neg P^l) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}, P^l
\end{array}$$

$\gamma -$ expansions
La métavariable x ajoutée au label est fraîche pour le tableau

$$\begin{array}{lcl}
\mathcal{T} \mid (\mathcal{B}, (\forall x P)^l_{n_x}) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid (\mathcal{B}, (\forall x P)^l_{n_x - 1}, P^{l,x}) \quad \text{si } n_x > 1 \\
\mathcal{T} \mid (\mathcal{B}, (\forall x P)^l_1) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid (\mathcal{B}, P^{l,x}) \\
\mathcal{T} \mid (\mathcal{B}, \neg(\exists x P)^l_{n_x}) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid (\mathcal{B}, \neg(\exists x P)^l_{n_x - 1}, \neg P^{l,x}) \quad \text{si } n_x > 1 \\
\mathcal{T} \mid (\mathcal{B}, \neg(\exists x P)^l_1) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid (\mathcal{B}, \neg P^{l,x})
\end{array}$$

$\delta -$ expansions
f est un symbole de Skolem frais pour le tableau

$$\begin{array}{lcl}
\mathcal{T} \mid (\mathcal{B}, (\exists x P)^{y_1, \ldots, y_n} & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid (\mathcal{B}, (P\{x := f(y_1, \ldots, y_n)\})^{y_1, \ldots, y_n} \\
\mathcal{T} \mid (\mathcal{B}, \neg(\forall x P)^{y_1, \ldots, y_n})) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid (\mathcal{B}, (\neg P[x := f(y_1, \ldots, y_n)])^{y_1, \ldots, y_n}
\end{array}$$

Autres expansions

$$\begin{array}{lcl}
\mathcal{T} \mid (\mathcal{B}, \bot^l) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \\
\mathcal{T} \mid (\mathcal{B}, \neg \bot^l) & \overset{\mathsf{tnf}}{\hookrightarrow} & \mathcal{T} \mid \mathcal{B}
\end{array}$$

Nota Bene 1 (Abus de notation)**.** Les métavariables ajoutées dans la mise en forme normale de tableau sont toujours fraîches. La notation

$$\mathcal{T} \mid (\mathcal{B}, (\forall x P)^l_{n_x}) \overset{\mathsf{tnf}}{\hookrightarrow} \mathcal{T} \mid (\mathcal{B}, (\forall x P)^l_{n_x - 1}, P^{l,x})$$

est un abus de notation (x dans le label n'est *pas* x sous le quantificateur) pour la notation plus explicite

$$\mathcal{T} \mid (\mathcal{B}, (\forall x P)^l_{n_x}) \overset{\mathsf{tnf}}{\hookrightarrow} \mathcal{T} \mid (\mathcal{B}, (\forall x P)^l_{n_x - 1}, P[x := Z]^{l,Z})$$

où Z est une métavariable fraîche.

On utilisera souvent $\mathsf{tnf}(\mathcal{B})$ à la place de $\mathsf{tnf}(\{\mathcal{B}\})$ par abus de notation.

Nota Bene 2. Les entiers n_x associés aux γ-formules sont une explicitation d'un processus d'*iterative deepening*. Ils permettent de prouver cependant la terminaison de la tnf.

Propriétés

Nous allons démontrer rapidement que la transformation tnf termine et est correcte. Définissons l'ordre suivant pour démontrer la correction.

Définition 23 (Ordre tnf)**.** Soit un tableau $\mathcal{T} = \mathcal{B}_1 \mid ... \mid \mathcal{B}_n$. Définissons alors les multi-ensembles suivants :

1. À chaque branche $\mathcal{B}_i$ est associée le multi-ensemble $M_{\mathcal{B}_i}$ des ng des γ-formules sur cette branche. On forme alors le multi-ensemble
$$M_{\mathcal{T}} = \{M_{\mathcal{B}_1}, ..., M_{\mathcal{B}_n}\}$$
2. À chaque branche $\mathcal{B}_i$ est associée le couple (a, b) où a est le nombre de $\wedge, \vee, \Rightarrow, \forall, \exists$ au total sur $\mathcal{B}_1$ et b le nombre de négations $\neg$. On forme alors le multi-ensemble de ces couples
$$F_{\mathcal{T}} = \{(a_{\mathcal{B}_1}, b_{\mathcal{B}_1}), .., (a_{\mathcal{B}_n}, b_{\mathcal{B}_n})\}$$

L'ordre tnf est alors deéfini comme l'ordre lexicographique sur le couple $(M_{\mathcal{T}}, F_{\mathcal{T}})$ où :
- $M_{\mathcal{T}}$ est ordonné suivant l'ordre multi-ensemble ;
- $F_{\mathcal{T}}$ est ordonné suivant l'ordre multi-ensemble (sur les couples le composant est appliqué un ordre lexicographique avec l'ordre classique sur les entiers).

Cet ordre suffit pour prouver la terminaison de tnf.

Proposition 1 (Terminaison de tnf)**.** *La mise en forme normale de tableau* tnf *termine.*

Démonstration. L'ordre tnf de la définition 23 décroît pour chaque règle de tnf. En particulier, les δ-expansions n'introduisent que des substitutions sur les termes et efface le quantificateur $\exists$ (ou $\forall$). □

La propriété de correction de tnf est nécessaire pour les preuves de ce chapitre. Pour cela, on introduit une nouvelle notation

Définition 24 (Métavariables d'une branche)**.** Soit $\mathcal{B} = \{P_1, \dots P_n\}$. Les métavariables de $\mathcal{B}$ sont l'union des métavariables de $P_1, \dots, P_n$.

Définition 25 ($\bar{\forall}$)**.** Soit $\Gamma_1, ..., \Gamma_m$ des branches. Soient $x_1, \dots, x_n$ l'union de toutes les métavariables de la famille de branches Γ_i. Alors la notation $\bar{\forall}$ est définie comme :
$$\bar{\forall}(\Gamma_1 \vee \dots \vee \Gamma_m) = \forall x_1, \dots, \forall x_n(\Gamma_1 \vee \dots \vee \Gamma_m)$$

La transformation tnf est un cas restreint des tableaux du premier ordre à métavariables [31, 45]. On peut donc réutiliser la preuve de correction de [45] (pp. 165-167).

Lemme 5 (Correction de tnf)**.** *Soient* $\Gamma_1, \ldots, \Gamma_m$ *et* $\mathcal{B}_1, \ldots, \mathcal{B}_n$ *des branches. Si*

$$\{\Gamma_1 \mid ... \mid \Gamma_m\} \overset{\mathsf{tnf}}{\hookrightarrow} \{\mathcal{B}_1 \mid ... \mid \mathcal{B}_n\}$$

alors

$$\bar{\forall}(\mathcal{B}_1 \vee \ldots \vee \mathcal{B}_n) \vdash \implies \bar{\forall}(\Gamma_1 \vee \ldots \vee \Gamma_m) \vdash$$

Démonstration. tnf préserve la satisfiabilité des formules. □

4.1.3 Règles de TaMed

Il ne reste ici qu'à définir la notion de tableau avec contraintes et à donner les règles spécifiques de TaMed pour opérer dans le cadre de la déduction modulo.

Définition 26 (Tableau avec contraintes)**.** Un tableau avec contraintes est une paire $\mathcal{T} \cdot [C]$ dans laquelle T est un tableau et C un ensemble d'équations appelées *contraintes*.

Les objets sur lesquels travaille la méthode étant définis, on peut dès lors caractériser TaMed.

Définition 27. Soit RE un système de réécriture de classe et $\mathcal{T} \cdot [C]$ un tableau avec contraintes, on écrit

$$\mathcal{T} \cdot [C] \rightsquigarrow \mathcal{T}' \cdot [C']$$

si le tableau avec contraintes $\mathcal{T}' \cdot [C']$ peut être inféré de $\mathcal{T} \cdot [C]$ en utilisant un nombre fini de fois les règles de **Fermeture étendue** et de **Narrowing étendu** de la figure 4.1. Cela veut dire qu'il existe une dérivation du tableau $\mathcal{T}' \cdot [C']$ sous les hypothèses de $\mathcal{T} \cdot [C]$, autrement dit une séquence $\mathcal{T}_1 \cdot [C_1], ..., \mathcal{T}_n \cdot [C_n]$ telle que soit $n = 0$ et $\mathcal{T} \cdot [C] = \mathcal{T}' \cdot [C']$, soit $\mathcal{T}_1 \cdot [C_1] = \mathcal{T} \cdot [C]$, $\mathcal{T}_n \cdot [C_n] = \mathcal{T}' \cdot [C']$ et chacun des $\mathcal{T}_i \cdot [C_i]$ est produit par l'application d'une des deux règles de TaMed à $\mathcal{T}_{i-1} \cdot [C_{i-1}]$.

Commentons un peu les règles de la figure 4.1.

- La première règle dite de *fermeture étendue* (de branche) est une extension simple de la règle de fermeture traditionnelle pour les tableaux du premier ordre avec égalité, où, de façon similaire à [21], les contraintes d'$\mathcal{E}$-unification ne sont pas résolues mais stockées dans la partie contraintes du tableau. Les labels des formules ne jouent aucun rôle dans cette règle : ils peuvent être retirés des formules dans la partie contrainte du tableau.
- La règle de *narrowing étendu* est l'adaptation de la règle du même nom de [30] pour la résolution. Le narrowing n'est appliqué qu'aux formules atomiques et non aux termes. Puisque les formules atomiques peuvent être réécrites en formules non atomiques, il faut également remettre le tableau en forme normale tnf.

Fermeture étendue

$$\frac{\Gamma_1, P, \neg Q \mid \Gamma_2 \mid ... \mid \Gamma_n \ \cdot \ [C]}{\Gamma_2 \mid ... \mid \Gamma_n \ \cdot \ [C \cup \{P \stackrel{?}{=}_{\mathcal{E}} Q\}]}$$

Narrowing étendu

$$\frac{\Gamma_1, U \mid \Gamma_2 \mid ... \mid \Gamma_n \ \cdot \ [C]}{\mathcal{B}_1 \mid ... \mid \mathcal{B}_p \mid \Gamma_2 \mid ... \mid \Gamma_n \ \cdots \ [C \cup U_{|\omega} \stackrel{?}{=} l]}$$

si $l \to r \in \mathcal{R}$, $U_{|\omega}$ est une proposition atomique
et $\mathcal{B}_1 \mid ... \mid \mathcal{B}_p = (\ \mid ... \mid \Gamma_1, U[r]_{|\omega})$

FIG. 4.1 – TaMed rules

La présentation des tableaux modulo est assez générale. Si $\mathcal{R}$ est vide, alors la règle de *narrowing étendu* n'est jamais utilisée et TaMed devient une méthode pour traiter les tableaux avec théories équationnelles (et donc égalité). De la même façon, si $\mathcal{R}$ et $\mathcal{E}$ vides, alors TaMed est tout simplement une méthode de tableau à métavariables pour la logique du premier ordre. Cependant dans les deux cas, nous n'explicitons pas la méthode pour satisfaire les contraintes.

Dans les tableaux d'ordre supérieur de Kohlhase [42], dont nous ne donnons pas de présentation dans cette thèse, le processus d'unification est intégrée aux processus d'expansion, dont les règles structurelles sont celles du premier ordre. Les contraintes d'unification y sont gérées de manière locale. Une étude plus précise reste à faire sur le rapport entre la façon dont Kohlhase gère les contraintes d'unification et ce que permettent ici notre règle de narrowing étendu avec nos contraintes globales.

4.2 Objectif et plan détaillé

4.2.1 Théorème principal

Le théorème fondamental que nous devons montrer établit la correction et la complétude de la méthode des tableaux modulo proposée vis-à-vis du calcul des séquents modulo sous-jacent. Il s'énonce ainsi :

Theorème 10 (Correction et complétude de TaMeD.)**.** *Soit $\mathcal{RE}$ un système de réécriture de classe tel que $\to_{\mathcal{R}}\ \mathcal{E}$ soit confluent. Si $\mathcal{B}$ et Γ sont des ensembles de formules closes, et C est ensemble de contraintes $\mathcal{E}$-unifiables, alors l'implication suivante est vérifiée :*

$$\textit{Si } \mathsf{tnf}(\mathcal{B} \wedge \neg\Gamma) \ \cdot \ [\emptyset] \rightsquigarrow \odot \ \cdot \ [C] \textit{ alors } B \vdash_{\mathcal{RE}} \Gamma$$

où $\neg\Gamma = \{\neg P \mid P \in \Gamma\}$.

Si le séquent $\mathcal{B} \vdash_{\mathcal{RE}} \Gamma$ a une preuve sans coupure, alors il existe une dérivation

$$\textit{Si } B \vdash^{cf}_{\mathcal{RE}} \Gamma \textit{ alors } \mathsf{tnf}(\mathcal{B} \wedge \neg\Gamma) \ \cdot \ [\emptyset] \rightsquigarrow \odot \ \cdot \ [C]$$

On peut déduire de ce théorème le corollaire suivant :

Corollaire 1 (Correction et complétude)**.** *Si le séquent* $\mathcal{B} \vdash_{\mathcal{RE}} \Gamma$ *a une preuve sans coupure alors*

$$B \vdash^{cf}_{\mathcal{RE}} \Gamma \textit{ ssi } \mathsf{tnf}(\mathcal{B} \wedge \neg\Gamma) \cdot [\emptyset] \rightsquigarrow \odot \cdot [C]$$

L'hypothèse d'élimination des coupures est primordiale. Nous nous plaçons toujours sous cette hypothèse dans les preuves développées.

4.2.2 Plan de la preuve

Le chemin que nous allons suivre afin de prouver les propriétés de correction et de complétude passe par un système intermédiaire. Celui-ci est un système équivalent de TaMed avec des règles plus atomiques : cela permet de bien détailler les inférences possibles. Ce système pas-à-pas aide également à avoir des preuves plus lisibles.

Le plan de la preuve est représenté par le schéma suivant :

$$\begin{array}{c} \mathcal{K}, \mathcal{B}, \neg\Gamma \vdash \\ \overset{Lem.4}{\Longleftrightarrow} \\ \mathcal{B}, \neg\Gamma \vdash \\ \overset{Prop.2}{\Longleftrightarrow} \\ \overset{Prop.3}{} \\ \mathcal{B}, \neg\Gamma \mapsto \odot \\ \overset{Prop.4}{\Longleftrightarrow} \\ \overset{Prop.5}{} \\ \mathcal{B}, \neg\Gamma\ [\emptyset] \rightsquigarrow \odot\ [C] \end{array}$$

- La preuve du lemme 4 du chapitre précédent est donnée dans [30].
- La deuxième équivalence, établie par les propositions 2 et 3, énonce la correction et la complétude du système intermédiaire par rapport au calcul des séquents modulo. La preuve de complétude nécessite que les coupures puissent être éliminées du calcul des séquents modulo $\mathcal{RE}$.
- La troisième équivalence termine la preuve du théorème principal et établit la correction et la complétude de TaMed par rapport au système intermédiaire.

Ainsi, par équivalence successives, on arrive à montrer que le tableau $\mathsf{tnf}(\mathcal{B} \wedge \Gamma)$ peut être réfuté si et seulement si le séquent $\mathcal{K}, \mathcal{B} \vdash \Gamma$ est prouvable dans le calcul des séquents du premier ordre, avec $\mathcal{K}$ l'ensemble des axiomes compatibles correspondant à $\mathcal{RE}$.

4.3 Correction et complétude de ic-TaMeD

Afin de prouver le théorème principal, on commence par définir un système intermédiaire appelé *Intermediate Calculus for TaMed* ou ic-TaMeD. Les règles de ce calcul sont décrites dans la figure 4.2.

4.3.1 Règles

$$\frac{\Gamma_1 \mid ... \mid \Gamma_n}{(\Gamma_1 \mid ... \mid \Gamma_n)[x := t]} \text{ Instantiation}$$

$$\frac{\Gamma_1, P \mid \Gamma_2 \mid ... \mid \Gamma_n}{\Gamma_1, P' \mid \Gamma_2 \mid ... \mid \Gamma_n} \text{ Conversion si } P =_{\mathcal{E}} P'$$

$$\frac{\Gamma_1, P \mid \Gamma_2 \mid ... \mid \Gamma_n}{\mathcal{B}_1 \mid ... \mid \mathcal{B}_p \mid \Gamma_2 \mid ... \mid \Gamma_n} \text{ Réduction}$$
$$\text{si } P \xrightarrow{\mathcal{R}} Q \text{ et } \mathcal{B}_1 \mid ... \mid \mathcal{B}_p = \mathsf{tnf}(\Gamma_1, Q)$$

$$\frac{\Gamma_1, P, \neg P \mid \Gamma_2 \mid ... \mid \Gamma_n}{\Gamma_2 \mid ... \mid \Gamma_n} \text{ Fermeture simple}$$

FIG. 4.2 – Règles de ic-TaMeD

Avec les règles de la figure 4.2, on peut définir la notion de dérivation dans ic-TaMeD.

Définition 28 (Dérivation ic-TaMeD)**.** Soit $\mathcal{RE}$ un système de réécriture de classe et $\mathcal{T}$ un tableau, on note

$$\mathcal{T} \mapsto \mathcal{T}'$$

si le tableau $\mathcal{T}'$ peut être obtenu du tableau $\mathcal{T}$ en utilisant un nombre fini de fois les règles de ic-TaMeD de la figure 4.2. Autrement dit, il existe une séquence de tableaux $\mathcal{T}_1, ..., \mathcal{T}_n$ telle que soit $n = 0$ et $\mathcal{T} = \mathcal{T}'$ soit $n > 0, \mathcal{T} = \mathcal{T}_0, \mathcal{T}' = \mathcal{T}_n$ et chacun des $\mathcal{T}_i$ est inféré par l'application d'une règle de ic-TaMeD au tableau $\mathcal{T}_{i-1}$.

Détaillons un petit peu les règles de ic-TaMeD et discutons notamment de la gestion des labels au sein de celles-ci :

Instanciation Au niveau des labels des formules, la variables instanciée est remplacée par les variables libres du terme qui se substitue à elle.

Conversion Les labels restent inchangés par cette règle de par la définition de l'$\mathcal{E}$-équivalence des formules avec label (cf définition 21). En particulier, cette définition proscrit l'introduction de nouvelles variables libres.

Réduction Cette règle étend les labels de façon simple par la mise en forme normale du tableau.

Fermeture simple Dans ce cas, les formules qui sont utilisées pour fermer la branche n'ont pas besoin d'avoir les mêmes labels.

4.3.2 Correction d'ic-TaMeD

Passons maintenant à la première étape de notre preuve du théorème principal, c'est-à-dire la correction de notre système intermédiaire. Un unique lemme intermédiaire permet d'établir cette propriété.

Lemme 6. *Soit* $\mathcal{T} = \Gamma_1 \mid ... \mid \Gamma_n$ *un tableau dont les branches sont complètement expansées. Si*

$$\mathcal{T} = \Gamma_1 \mid ... \mid \Gamma_n \mapsto \odot$$

alors

$$\bar{\forall}\ (\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$$

Démonstration. La preuve de ce lemme est faite par induction sur la structure de la dérivation ic-TaMeD $\Gamma_1 \mid ... \mid \Gamma_n \mapsto \odot$.

Si cette dérivation est vide, aucune règle de ic-TaMeD n'a pu être appliquée à $\mathcal{T}$. Autrement dit, ce tableau a été « vidé » par sa mise en forme normale ce qui équivaut à dire que $\mathcal{T} \Leftrightarrow \bot$.

Sinon, la dérivation $\Gamma_1 \mid ... \mid \Gamma_n \mapsto \odot$ commence par la production d'un nouveau tableau $\mathcal{T}'$ tel que $\mathcal{T} \mapsto \odot$. Examinons les différents cas possibles.

Fermeture simple Il existe une branche de $\mathcal{T}$ qui contient un littéral et son opposé. Appelons cette branche Γ_1 et le littéral P, alors $\Gamma_1 = \mathcal{B} \wedge P \wedge \neg P$. Par ailleurs $\mathcal{T}' = \Gamma_2 \mid ... \mid \Gamma_n$ et $\Gamma_1 \Leftrightarrow \bot$.

On peut déduire assez simplement la série d'équivalence suivante :

$$\begin{array}{ll} & \bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}} \bar{\forall}(\Gamma_2 \vee ... \vee \Gamma_n) \\ \iff & \bar{\forall}((\mathcal{B} \wedge \neg P_1 \wedge P_1) \vee \Gamma_2 \vee \Gamma_n) \vdash_{\mathcal{RE}} \bar{\forall}(\Gamma_2 \vee ... \vee \Gamma_n) \\ \iff & \bar{\forall}(\bot \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}} \bar{\forall}(\Gamma_2 \vee ... \vee \Gamma_n) \\ \iff & \bar{\forall}(\Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}} \bar{\forall}(\Gamma_2 \vee ... \vee \Gamma_n) \end{array}$$

Par hypothèse d'induction $\mathcal{T}' \mapsto \odot$ donc $\bar{\forall}(\Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$. En appliquant la règle de coupure à ce dernier séquent et à $\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}} \bar{\forall}(\Gamma_2 \vee ... \vee \Gamma_n)$, on obtient finalement le résultat souhaité : $\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$

Instanciation Dans ce cas $\mathcal{T}' = (\Gamma_1 \mid ... \mid \Gamma_n)[x := t]$. Si on applique l'hypothèse d'induction, alors

$$\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n)[x := t] \vdash_{\mathcal{RE}}$$

Il est assez simple de construire la dérivation suivante dans le calcul des séquents :

$$\frac{\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n)[x := t] \vdash_{\mathcal{RE}}}{\forall x\ \bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}}\ \forall\text{-r}(\Gamma_1 \vee ... \vee \Gamma_n, x, t)$$

La définition de $\bar{\forall}$ assure ensuite la transition de $\forall x\ \bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$ à $\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$.

Réduction Il existe une branche de $\mathcal{T}$ contenant une formule atomique P se réduisant en Q. Nommons cette branche Γ_1 et posons $\Gamma'_1 = \Gamma_1 \backslash \{P\}, Q$ et $\mathsf{tnf}(\Gamma'_1) = \mathcal{B}_1 \mid ... \mid \mathcal{B}_p$.

La preuve voulue est construite en trois parties. Tout d'abord, on construira la preuve du séquent :

$$\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}} \bar{\forall}(\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n) \tag{4.1}$$

Puis on montrera que

$$\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n), \bar{\forall}(\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}} \tag{4.2}$$

Ces deux résultats auront pour conséquence la conclusion :

$$\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$$

en utilisant une coupure.

Commençons tout de suite par prouver le séquent 4.1 dans le calcul des séquents modulo. On a $\Gamma_1 =_{\mathcal{RE}} \Gamma'_1$ par construction et par définition de la règle axiome du calcul des séquents modulo $\Gamma_1 \vdash \Gamma'_1$ est donc prouvable. On peut donc construire le schéma de dérivation suivant :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_1 \vdash \Gamma'_1}\,\text{ax}}{\Gamma_1 \vdash \Gamma'_1 \vee \Gamma_2}\,\vee\text{-r}}{\vdots}}{\Gamma_1 \vdash \Gamma'_1 \vee ... \vee \Gamma_n}\,\vee\text{-r} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash \Gamma_2}\,\text{ax}}{\vdots}}{\Gamma_2 \vdash \Gamma'_1 \vee ... \vee \Gamma_n}\,\vee\text{-r}}{\Gamma_1 \vee \Gamma_2 \vdash \Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n}\,\vee\text{-l} \qquad \dfrac{\dfrac{}{\Gamma_n \vdash \Gamma_n}\,\text{ax}}{\vdots}}{\Gamma_1 \vee ... \vee \Gamma_n \vdash \Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n}\,\vee\text{-l}$$

À partir de ce séquent, on obtient le séquent 4.1 de la manière suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1 \vee ... \vee \Gamma_n \vdash \Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n}{\forall x_1\ (\Gamma_1 \vee ... \vee \Gamma_n) \vdash \Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n}\,\forall\text{-l}}{\vdots}}{\begin{array}{c}\forall x_1 ... \forall x_n\ (\Gamma_1 \vee ... \vee \Gamma_n) \vdash \Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n \\ (=)\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash \Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n\end{array}}\,\forall\text{-r}}{\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash \forall x_1\ (\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n)}\,\forall\text{-r}}{\vdots}}{\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash \bar{\forall}(\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n)}\,\forall\text{-r}$$

Le premier séquent 4.1 ayant été obtenu, procédons à la preuve du deuxième séquent 4.2. Et posons tout d'abord $\mathcal{K}$, un ensemble d'axiomes compatibles avec $\mathcal{RE}$ dans le sens de la définition 14.

$\mathcal{K}$ est un ensemble de formules closes : il peut donc être déplacé dans et en dehors de la portée de quantificateurs en gardant l'équivalence entre les formules. En particulier :

$$\mathcal{K} \wedge \bar{\forall}(\mathcal{B}_1 \vee ... \vee \mathcal{B}_p \vee \Gamma_2 \vee ... \vee \Gamma_n)$$
$$\Longleftrightarrow$$
$$\bar{\forall}(\mathcal{K} \wedge (\mathcal{B}_1 \vee ... \vee \mathcal{B}_p \vee \Gamma_2 \vee ... \vee \Gamma_n))$$

Dans le cas de la réduction, on a posé $\mathcal{T}' = \mathcal{B}_1 \mid ... \mid \mathcal{B}_p \mid \Gamma_2 \mid ... \mid \Gamma_n$. Par hypothèse d'induction :

$$\bar{\forall}(\mathcal{B}_1 \vee ... \vee \mathcal{B}_p \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$$

soit encore, par le lemme 4

$$\mathcal{K}, \bar{\forall}(\mathcal{B}_1 \vee ... \vee \mathcal{B}_p \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash \tag{4.3}$$

De plus, en utilisant la normalisation des tableaux :

$$\mathcal{K}, \Gamma'_1 \mid \mathcal{K}, \Gamma_2 \mid ... \mid \mathcal{K}, \Gamma_n \overset{\mathsf{tnf}}{\hookrightarrow} \mathcal{K}, \mathcal{B}_1 \mid ... \mid \mathcal{K}, \mathcal{B}_p \mid \mathcal{K}, \Gamma_2 \mid ... \mid \mathcal{K}, \Gamma_n$$

En utilisant le lemme 5 de correction de la normalisation et le séquent 4.3, alors

$$\mathcal{K}, \bar{\forall}(\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash$$

soit encore

$$\bar{\forall}(\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$$

En affaiblissant la partie gauche de ce séquent, on obtient le séquent 4.2 donné plus haut :

$$\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n), \bar{\forall}(\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$$

Or nous avions déjà montré :

$$\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}} \bar{\forall}(\Gamma'_1 \vee \Gamma_2 \vee ... \vee \Gamma_n)$$

L'application de la règle de coupure du calcul des séquents modulo à ces deux prémisses donne le résultat escompté :

$$\bar{\forall}(\Gamma_1 \vee \Gamma_2 \vee ... \vee \Gamma_n) \vdash_{\mathcal{RE}}$$

Conversion Le cas de la règle de *conversion* est montré en suivant les mêmes étapes que dans la première partie du cas précédent de la règle de *réduction* car on la même congruence $\mathcal{RE}$ entre les branches résultantes et les branches originelles.

□

Le lemme précédent permet à son tour de démontrer la propriété de correction d'ic-TaMeD par rapport au calcul des séquents modulo.

Proposition 2 (Correction d'ic-TaMeD)**.** *Soient $P_1, ..., P_n, Q_1, ..., Q_m$ des formules closes. Si*

$$\mathsf{tnf}(P_1 \wedge ... \wedge P_n \wedge \neg Q_1 \wedge ... \wedge \neg Q_m) \mapsto \odot$$

alors

$$P_1, ..., P_n \vdash Q_1, ..., Q_m$$

Démonstration. Soit $\Gamma_1 \mid ... \mid \Gamma_n = \mathsf{tnf}(P_1 \wedge \ldots \wedge P_n \wedge \neg Q_1 \wedge \ldots \wedge \neg Q_m)$.

On a $(\Gamma_1 \mid ... \mid \Gamma_n) \mapsto \odot$.

D'où par le lemme 6 :

$$\bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash$$

et par affaiblissement

$$\mathcal{K}, \bar{\forall}(\Gamma_1 \vee ... \vee \Gamma_n) \vdash$$

, où $\mathcal{K}$ est un ensemble d'axiomes compatibles avec $\mathcal{RE}$. On a

$$\{\mathcal{K}, P_1, \ldots, P_n, \neg Q_1, \ldots \neg Q_m\} \overset{\mathsf{tnf}}{\hookrightarrow} (\mathcal{K}, \Gamma_1) \mid ... \mid (\mathcal{K}, \Gamma_n)$$

La correction des règles d'expansion de tableau permet alors d'écrire :

$$\mathcal{K}, P_1, ..., P_n, \neg Q_1, ..., \neg Q_m \vdash$$

Et par le lemme 4 d'équivalence des calculs de séquents

$$P_1, ..., P_n, \neg Q_1, ..., \neg Q_m \vdash$$

autrement dit

$$P_1, ..., P_n \vdash Q_1, ..., Q_m$$

□

4.4 Complétude d'ic-TaMeD

Il nous reste maintenant à établir que pour toute preuve obtenue en calcul des séquents modulo, on arrive à obtenir une preuve par le biais des règles d'ic-TaMeD.

Tout d'abord, il est nécessaire de définir certaines opérations que l'on va utiliser au niveau des symboles de Skolem. Ces définitions sont empruntées à [30].

Définition 29 (Transformation des symboles de fonction)**.** Soient t un terme (resp. une formule), f un symbole de fonction d'arité n et u un terme dont les variables libres sont dans $x_1, ..., x_n$.

La transformation individuelle du symbole f en u est notée $(x_1, ..., x_n)u/f$. La notation $\{(x_1, ..., x_n)u/f\}t$ décrit son application à un terme (resp. une formule) t. le résultat de cette application est obtenu en remplaçant dans t les sous-termes de la forme $f(v_1, ..., v_n)$ — où $v_1, ..., v_n$ sont des termes arbitraires — par le terme $u[x_1 := v_1, ..., x_n := v_n]$.

Un ensemble fini d'indices I étant donné, le résultat de l'application d'une transformation de symboles de fonction $\rho = \{(x_1^i, ..., x_n^i)u^i/f^i\}_{i\in I}$ à un terme t (resp. une formule) est défini comme l'application simultanée de toutes les transformations individuelles de symboles à t (resp. à la formule).

On peut par ailleurs noter que ces transformations n'affectent pas les labels. Prouvons maintenant une série de lemmes sur l'interaction entre ces transformations de symboles de fonctions et les dérivations possibles dans nos calculs (mise en forme tableau-normale, puis règles d'ic-TaMeD).

Lemme 7. *Soit $\mathcal{T}$ un tableau et ρ une transformation de symboles de fonction. Les symboles de Skolem introduits par la mise en forme normale de tableau sont supposés être frais, c'est-à-dire non transformés par ρ. Alors*

$$\mathsf{tnf}(\rho\mathcal{T}) = \rho\mathsf{tnf}(\mathcal{T})$$

Démonstration. Nous allons vérifier que si deux tableaux $\mathcal{T}$ et $\mathcal{T}'$ sont tels que $\mathcal{T} \overset{\mathsf{tnf}}{\hookrightarrow} \mathcal{T}'$, alors $\rho\mathcal{T} = \rho\mathcal{T}'$. Supposons par exemple que :

$$\mathcal{T} = \mathcal{U} \mid \Gamma, \exists x\ P^{y_1,...,y_n}$$

et aussi que :

$$\mathcal{T}' = \mathcal{U} \mid \Gamma, P[x := f(y_1, ..., y_n)]^{y_1,...,y_n}$$

Alors :

$$\rho\mathcal{T} = \rho\mathcal{U} \mid \rho\Gamma, \rho(\exists x\ P^{y_1,...,y_n})$$

donne en une étape

$$\rho\mathcal{T}' = \rho\mathcal{U} \mid \rho\Gamma, \rho P[x := f(y_1, ..., y_n)]^{y_1,...,y_n} = \rho\mathcal{T}'$$

car les symboles de Skolem introduits (f en l'occurrence) sont frais et ne sont donc pas transformés par ρ par construction.

Le reste du lemme est montré par induction sur la longueur de la dérivation de la mise en forme tableau-normale $\overset{\mathsf{tnf}}{\hookrightarrow}$. □

Lemme 8. *Soit $\mathcal{T}$ un tableau et ρ une transformation de symboles de fonction n'apparaissant pas dans $\mathcal{RE}$. Si $\mathsf{tnf}(\mathcal{T}) \mapsto \odot$ alors $\mathsf{tnf}(\rho\mathcal{T}) \mapsto \odot$ et les dérivations ic-TaMeD ont la même longueur.*

Démonstration. Du lemme 7 précédent découle le fait qu'il suffit de montrer que $\rho\mathsf{tnf}(\mathcal{T}) \mapsto \odot$. La preuve est ensuite faite par induction sur la structure de la dérivation ic-TaMeD que pour tout tableau complètement expansé $\mathcal{T}$, si $\mathcal{T} \mapsto \odot$, alors $\rho\mathcal{T} \mapsto \odot$.

Instanciation Dans ce cas, on a $\rho(\mathcal{T}[x \mapsto t]) = (\rho\mathcal{T})[x \mapsto t]$.

Conversion Si $\mathcal{B} =_{\mathcal{E}} \Gamma$ alors $\rho\mathcal{B} =_{\mathcal{E}} \rho\Gamma$ car les symboles transformés par ρ n'apparaissent pas dans $\mathcal{RE}$.

Réduction Ce cas similaire au cas précédent car, si $\mathcal{B} \xrightarrow{\mathcal{RE}} \Gamma$, alors $\rho\mathcal{B} \xrightarrow{\mathcal{RE}} \rho\Gamma$ car ρ transforme des symboles n'apparaissant pas dans $\mathcal{RE}$.

Fermeture simple Ce cas est évident.

□

Lemme 9. *Soient t un terme clos, $\mathcal{B}$ une branche de tableau, x une variable.*

$$\mathsf{tnf}(\mathcal{B}[x := t]) \mapsto \odot \Rightarrow \mathsf{tnf}(\mathcal{B}) \mapsto \odot$$

Démonstration. La preuve est faite par induction sur l'ordre tnf, en suivant la structure de la transformation en forme tableau-normale.

Si $\mathcal{B}$ est une branche totalement développée, alors $\mathsf{tnf}(\mathcal{B}[x := t]) = \mathcal{B}[x := t]$ car $\mathcal{B} = \mathsf{tnf}(\mathcal{B})$. La règle d'*Instanciation* permet de dériver $\mathcal{B}[x := t]$ de $\mathcal{B}$. Par conséquent, si $\mathsf{tnf}(\mathcal{B}[x := t]) \mapsto \odot$, alors $\mathcal{B} \mapsto \mathsf{tnf}(\mathcal{B}[x := t]) \mapsto \odot$ et donc par définition d'une dérivation dans ic-TaMeD, $\mathcal{B} \mapsto \odot$

Sinon, il existe une formule $P \in \mathcal{B}$ qui n'est pas un littéral. Nous allons détailler les différents cas possibles, en posant par ailleurs $\mathcal{B} = \mathcal{B}', P$.

- Les cas pour $P = Q_1 \wedge Q_2$, $P = \neg\bot$, $P = \bot$, $P = \neg\neg Q$ sont assez directs et ne seront pas détaillés ici.
- Si $P = Q_1 \vee Q_2$ alors $P[x := t] = Q_1[x := t] \vee Q_2[x := t]$.

  $$\mathsf{tnf}(\mathcal{B}) = \mathsf{tnf}(\mathcal{B}', Q_1) \mid \mathsf{tnf}(\mathcal{B}', Q_2)$$

  donc

  $$\mathsf{tnf}(\mathcal{B}[x := t]) = \mathsf{tnf}(\mathcal{B}'[x := t], Q_1[x := t]) \mid \mathsf{tnf}(\mathcal{B}'[x := t], Q_2[x := t])$$

  Par conséquent si $\mathsf{tnf}(\mathcal{B}[x := t]) \mapsto \odot$, alors

  $$(\mathsf{tnf}(\mathcal{B}'[x := t], Q_1[x := t]) \mid \mathsf{tnf}(\mathcal{B}'[x := t], Q_2[x := t])) \mapsto \odot$$

  qui peut être réécrit de la manière suivante

  $$(\mathsf{tnf}(\mathcal{B}', Q_1) \mid \mathsf{tnf}(\mathcal{B}', Q_2))[x := t] \mapsto \odot$$

  En utilisant l'hypothèse d'induction, on obtient

  $$(\mathsf{tnf}(\mathcal{B}', Q_1) \mid \mathsf{tnf}(\mathcal{B}', Q_2])) \mapsto \odot$$

  i.e. $\mathsf{tnf}(\mathcal{B}) \mapsto \odot$.

- Si $P = \forall y\ Q$ alors $P[x := t] = \forall y\ Q[x := t]$. Il y a deux cas suivant la valeur de n_y^γ
  - Si $n_y^\gamma = 1$, $\mathsf{tnf}(\mathcal{B}) = \mathsf{tnf}(\mathcal{B}', Q)$ et

    $$\mathsf{tnf}(\mathcal{B}[x := t]) = \mathsf{tnf}(\mathcal{B}'[x := t], Q[x := t])$$

    Donc si $\mathsf{tnf}(\mathcal{B}[x := t]) \mapsto \odot$, alors

    $$\mathsf{tnf}(\mathcal{B}'[x := t], Q[x := t]) \mapsto \odot$$

    Par application de l'hypothèse d'induction, on a $\mathsf{tnf}(\mathcal{B}', Q) \mapsto \odot$, i.e. $\mathsf{tnf}(\mathcal{B})$.
  - Sinon $n_y^\gamma > 1$ et $\mathsf{tnf}(\mathcal{B}) = \mathsf{tnf}(\mathcal{B}, Q)$. Alors, de la même façon que précédemment :

    $$\mathsf{tnf}(\mathcal{B}[x := t]) = \mathsf{tnf}(\mathcal{B}[x := t], Q[x := t])$$

    Donc si $\mathsf{tnf}(\mathcal{B}[x := t]) \mapsto \odot$, alors

    $$\mathsf{tnf}(\mathcal{B}[x := t], Q[x := t]) \mapsto \odot$$

    Par application de l'hypothèse d'induction (n a effectivement été décrémenté de 1) on a $\mathsf{tnf}(\mathcal{B}, Q) \mapsto \odot$, i.e. $\mathsf{tnf}(\mathcal{B}) \mapsto \odot$.
- Si $P = \exists z\ Q$, alors $P[x := t] = (\exists z\ Q)[x := t]$ Si x n'appartient pas au label de P il n'est pas libre dans P et le cas est clos. Sinon, soit $y_1, \ldots, y_n, x$ le label de P ; alors on a :

  $$\mathsf{tnf}(\mathcal{B}[x := t]) = \mathsf{tnf}(\mathcal{B}'[x := t], Q[x := t][z := g(y_1, \ldots, y_n, x)])$$

  où g est un symbole de Skolem frais. Par hypothèse d'induction

  $$\mathsf{tnf}(\mathcal{B}'[x := t], Q[x := t][z := f(y_1, \ldots, y_n), x]) \mapsto \odot$$

  Soit $\rho = \{(y_1, \ldots, y_n) f(y_1, \ldots, y_n, t)/g\}$, alors par les lemmes 7 et 8 on a :

  $$\begin{array}{ll} & \mathsf{tnf}(\mathcal{B}'[x := t], Q[x := t][z := g(y_1, \ldots, y_n)]) \mapsto \odot \\ \Longrightarrow_\rho & \mathsf{tnf}(\mathcal{B}'[x := t], Q[x := t][z := f(y_1, \ldots, y_n, t)]) \mapsto \odot \\ \Longleftrightarrow & \mathsf{tnf}(\mathcal{B}', Q[z := f(y_1, \ldots, y_n, x)])[x := t] \mapsto \odot \end{array}$$

  On peut maintenant appliquer l'hypothèse d'induction à la ligne précédente.

  $$\mathsf{tnf}(\mathcal{B}', Q[z := f(y_1, \ldots, y_n, x)]) \mapsto \odot$$

  i.e. $\mathsf{tnf}(\mathcal{B}) \mapsto \odot$.
- Les $\neg(Q_1 \vee Q_2)$ et $\neg(Q_1 \Rightarrow Q_2)$ sont traités comme $Q_1 \wedge Q_2$. Les cas $\neg(Q_1 \wedge Q_2)$ et $Q_1 \Rightarrow Q_2$ correspondent au cas $Q_1 \vee Q_2$. Le cas $\neg(\forall z\ Q)$ est comme $\exists z\ Q$ et le cas $\neg(\exists z\ Q)$ comme $\forall z\ Q$.

□

Lemme 10. *Soient* $\mathcal{B} = \{P_1, ..., P_n\}$ *et* $\Gamma = \{Q_1, ..., Q_n\}$ *deux branches de formules avec label telles que, pour chaque* i, $P_i \rightarrow^* Q_i$.

Si $\mathsf{tnf}(\Gamma) \mapsto \odot$, *alors* $\mathsf{tnf}(\mathcal{B}) \mapsto \odot$.

Démonstration. La preuve est faite par induction sur $\mathcal{B}$ en utilisant l'ordre tnf. Si $P \rightarrow^0 Q$, le résultat découle de l'hypothèse.

Tout d'abord, si $P \rightarrow^1 Q$, alors, pour chaque branche de $\mathsf{tnf}(\mathcal{B})$, on peut trouver une branche de $\mathsf{tnf}(\Gamma)$ qui peut être obtenue en utilisant la règle de *Réduction*.

Sinon, si $P =_{\mathcal{E}} Q$, alors pour chaque branche de $\mathsf{tnf}(\mathcal{B})$, on peut trouver une branche de $\mathsf{tnf}(\Gamma)$ qui peut être obtenue en utilisant la règle de *Conversion*.

Par conséquent, si toutes les formules de $\mathcal{B}$ sont des littéraux, on peut dériver de $\mathcal{B} = \mathsf{tnf}(\mathcal{B})$ toutes les branches de $\mathsf{tnf}(\Gamma)$ en utilisant les règles de *Conversion* et *Réduction*. Dans ce cas, le résultat vient directement : comme

$$\mathsf{tnf}(\mathcal{B}) \mapsto_{\text{Conver. Réd.}} \mathsf{tnf}(\Gamma) \mapsto \odot$$

alors $\mathsf{tnf}(\mathcal{B}) \mapsto \odot$. □

Nous allons également avoir besoin d'un lemme à propos du connecteur logique $\vee$.

Lemme 11. *Soient* R, S *des formules closes,* Γ, Δ *des branches de formules closes. Si*

$$\mathsf{tnf}(R, \Gamma, \neg\Delta) \mapsto \odot \text{ et } \mathsf{tnf}(S, \Gamma, \neg\Delta) \mapsto \odot$$

alors on peut construire une dérivation

$$\mathsf{tnf}(R \vee S, \Gamma, \neg\Delta) \mapsto \odot$$

Démonstration. Un tableau est fermé ($\mapsto \odot$) lorsque toutes ses branches peuvent être fermées simultanément. On sait par ailleurs que

$$\mathsf{tnf}(R \vee S, \Gamma, \neg\Delta) = \mathsf{tnf}(R, \Gamma, \neg\Delta \mid S, \Gamma, \neg\Delta)$$

de par la définition de la forme tableau-normale et de la correction de tnf.

Nous savons aussi que :

$$\mathsf{tnf}(R, \Gamma, \neg\Delta \mid S, \Gamma, \neg\Delta) = \mathsf{tnf}(R, \Gamma, \neg\Delta) \mid \mathsf{tnf}(S, \Gamma, \neg\Delta)$$

Ainsi si

$$\mathsf{tnf}(R, \Gamma, \neg\Delta) \mapsto \odot$$

et

$$\mathsf{tnf}(S, \Gamma, \neg\Delta) \mapsto \odot$$

alors, comme toutes les formules utilisées sont closes,

$$\mathsf{tnf}(R, \Gamma, \neg\Delta \mid S, \Gamma, \neg\Delta) = \mathsf{tnf}(R, \Gamma, \neg\Delta) \mid \mathsf{tnf}(S, \Gamma, \neg\Delta) \mapsto \odot$$

et finalement

$$\mathsf{tnf}(R \vee S, \Gamma, \neg\Delta) \mapsto \odot$$

□

Nous utiliserons par la suite un lemme directement tiré de [30] permettant la restriction de l'utilisation de la congruence aux réductions. Nous référons le lecteur à cet article pour la preuve de ce lemme.

Lemme 12 (Restriction de la congruence aux réductions)**.** *Si la relation $\mathcal{RE}$ est confluente, alors :*

- *Si P et $Q \wedge R$ sont des formules closes telles que $P =_{\mathcal{RE}} Q \wedge R$, alors il existe une formule close $Q' \wedge R'$ telle que $P \rightarrow^* Q' \wedge R', Q =_{\mathcal{RE}} Q'$ et $P =_{\mathcal{RE}} P'$.*
- *Si P et $Q \vee R$ sont des formules closes telles que $P =_{\mathcal{RE}} Q \vee R$, alors il existe une formule close $Q' \vee R'$ telle que $P \rightarrow^* Q' \vee R', Q =_{\mathcal{RE}} Q'$ et $P =_{\mathcal{RE}} P'$.*
- *Si P et $Q \Rightarrow R$ sont des formules closes telles que $P =_{\mathcal{RE}} Q \Rightarrow R$, alors il existe une formule close $Q' \Rightarrow R'$ telle que $P \rightarrow^* Q' \Rightarrow R', Q =_{\mathcal{RE}} Q'$ et $P =_{\mathcal{RE}} P'$.*
- *Si P et $\neg Q$ sont des formules closes telles que $P =_{\mathcal{RE}} \neg Q$, alors il existe une formule close $\neg Q'$ telle que $P \rightarrow^* \neg Q'$ et $Q =_{\mathcal{RE}} Q'$.*
- *Si P est une formule close telle que $P =_{\mathcal{RE}} \bot$, alors $P \rightarrow^* \bot$.*
- *Si P et $\forall x Q$ sont des formules closes telles que $P =_{\mathcal{RE}} \forall x Q$, alors il existe une formule close $\forall x Q'$ telle que $P \rightarrow^* \forall x Q'$ et $Q =_{\mathcal{RE}} Q'$.*
- *Si P et $\exists x Q$ sont des formules closes telles que $P =_{\mathcal{RE}} \exists x Q$, alors il existe une formule close $\exists x Q'$ telle que $P \rightarrow^* \exists x Q'$ et $Q =_{\mathcal{RE}} Q'$.*

Démonstration. Admise de [30] □

Proposition 3 (Complétude d'ic-TaMeD)**.** *Soient $\rightarrow^*$ une relation confluente et $P_1, .., P_n, Q_1, .., Q_m$ des formules closes. Si le séquent*

$$P_1, .., P_n \vdash_{\mathcal{RE}} Q_1, ..., Q_m$$

admet une preuve sans coupure, alors

$$\mathsf{tnf}(P_1, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

Démonstration. Afin de prouver cette proposition, nous allons procéder par induction (certains cas plus simples sont omis) sur la taille de la preuve sans coupure de

$$P_1, .., P_n \vdash_{\mathcal{RE}} Q_1, ..., Q_m.$$

- Si la dernière règle est un axiome, alors $n = m = 1$ et $P_1 =_{\mathcal{RE}} Q_1$. Par confluence, il existe R et R' tels que $P_1 \rightarrow^* R$, $Q_1 \rightarrow^* R'$ et $R =_{\mathcal{E}} R'$. Par les règles de *Conversion* et de *Fermeture simple*, $\mathsf{tnf}(R, \neg R') \mapsto \odot$ et par le lemme 10 :
$$\mathsf{tnf}(P_1, \neg Q_1) \mapsto \odot$$
- Si la dernière règles est `contr-r` ou `contr-l`, alors les formes tableau normales de l'antécédent et du succédent sont les mêmes. Il suffit dès lors d'appliquer l'hypothèse d'induction.

– Si la dernière règle est `weak-l` ou `weak-r` alors la `tnf` de l'antécédent est un sous-ensemble de celle du successeur, on appliquera donc l'hypothèse d'induction.
De même, si la dernière règle est `contr-l` ou `contr-r`
– Si la dernière règle est $\vee - l$, alors il existe un P_i (mettons P_1) qui est $\mathcal{RE}$-équivalent à une disjonction $R \vee S$. Par le lemme 12, il existe R' et S' tels que $P \rightarrow^* R' \vee S'$ et $R' =_{\mathcal{E}} R$ et $S' =_{\mathcal{E}} S$. En utilisant l'hypothèse d'induction et la proposition 2, on a :

$$\mathsf{tnf}(R', P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

et

$$\mathsf{tnf}(S', P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

On peut maintenant utiliser le lemme 11 car l'instanciation de variables universellement quantifiées n'est pas retardée dans le calcul des séquents modulo : par conséquent aucune variable libre n'est introduite par le raisonnement dans les séquents. En conclusion aucune variable libre rigide n'apparît dans les branches par application de la β-expansion. D'où :

$$\mathsf{tnf}(R' \vee S', P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

. Et par le lemme 10, on obtient

$$\mathsf{tnf}(P_1, P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

.
– Si la dernière règle est $\forall$-l alors il existe un P_i (mettons P_1) qui est $\mathcal{RE}$-équivalent à une formule universelle $\forall x R$. Par le lemme 12, il existe R' tel que $P \rightarrow^* \forall x R'$ et $\mathcal{R}' =_{\mathcal{E}} R$. En utilisant l'hypothèse d'induction et la proposition 2, il vient :

$$\mathsf{tnf}(R[x := t], P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

pour au moins un terme donné t. Notons que les labels de $P_1, \forall R, R[x := t]$ ne contiennent pas de variables correspondant à x alors que celui de R' contient une variable fraîche x lui correspondant. Cette variable n'apparaît donc pas par ailleurs dans la branche, on peut alors appliquer le lemme 9 :

$$\mathsf{tnf}(R', P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

Autrement dit

$$\mathsf{tnf}(\forall x R', P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

et par le lemme 10 :

$$\mathsf{tnf}(P_1, P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

.

– Si la dernière règles est ∃-l alors il existe un P_i (mettons P_1) qui est $\mathcal{RE}$-équivalent à une formule existentielle $\exists x R$. Par le lemme 12, il existe R' tel que $P \rightarrow^* \exists x R'$ et $\mathcal{R}' =_{\mathcal{E}} R$. En utilisant l'hypothès d'induction et la proposition 2, il vient :

$$\mathsf{tnf}(R'[x := c], P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

avec c constante fraîche. Donc :

$$\mathsf{tnf}(\exists x R', P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

Ainsi par le lemme 10

$$\mathsf{tnf}(P_1, P_2, .., P_n, \neg Q_1, .., \neg Q_m) \mapsto \odot$$

– Les cas restants ∧-r et ⇒-l, ∨-r et ⇒-r, ¬-r, ∀-r, ∃-r sont respectivement traités comme ∨-l, ∧-l, ¬-l, ∃-l, ∀-l.

□

4.5 Correction et complétude de TaMed

Après avoir établi la correction et la complétude d'une méthode intermédiaire ic-TaMeD, nous allons montrer les mêmes propriétés pour TaMed en nous appuyant sur les démonstrations déjà obtenues.

4.5.1 Correction de TaMed

La preuve de correction des tableaux pour la déduction modulo s'appuie sur deux lemme intermédiaires, le premier établissant certaines propriétés concernant les substitutions, le deuxième établissant la traduction de dérivations TaMed en dérivations ic-TaMeD.

Lemme 13. *Soient* Γ *une branche de formules avec label et* Θ *une substitution close telle que les variables introduites durant la transformation en* tnf *de* Γ *n'apparaissent pas dans* Θ*.*

Alors il existe une transformation ρ *des symboles de fonction par la mise en forme tableau-normale de* Γ *telle que*

$$\mathsf{tnf}(\Theta\Gamma) = \rho\Theta\mathsf{tnf}(\Gamma)$$

Démonstration. Cette preuve est faite par induction sur Γ en utilisant l'ordre tnf.

Si toutes les formules contenues dans Γ sont des littéraux alors

$$\mathsf{tnf}(\Theta\Gamma) = \Theta\Gamma = \Theta\mathsf{tnf}(\Gamma)$$

et ρ est alors l'identité.

Sinon, il existe une formule P de Γ qui n'est pas un littéral. Posons $\Gamma = \Gamma', P$ et détaillons les cas possibles :

$P = Q_1 \wedge Q_2$ Par hypothèse d'induction, il existe ρ' tel que

$$\mathsf{tnf}(\Theta\Gamma', \Theta Q_1, \Theta Q_2) = \rho'\Theta\mathsf{tnf}(\Gamma', Q_1, Q_2)$$

Autrement dit, par définition de tnf :

$$\mathsf{tnf}(\Theta\Gamma) = \rho'\Theta\mathsf{tnf}(\Gamma)$$

Dans ce cas on prendra donc $\rho = \rho'$

$P = Q_1 \vee Q_2$ Par hypothèse d'induction, il existe ρ' et ρ" tel que

$$\mathsf{tnf}(\Theta\Gamma', \Theta Q_1) = \rho'\Theta\mathsf{tnf}(\Gamma', Q_1)$$

et

$$\mathsf{tnf}(\Theta\Gamma', \Theta Q_2) = \rho"\Theta\mathsf{tnf}(\Gamma', Q_2)$$

Comme les domaines de ρ' et ρ" sont disjoints car les symboles de Skolem introduits sont nécessairement frais, on peut écrire en utilisant les propriétés de la transformation tnf :

$$\begin{array}{rcl} \mathsf{tnf}(\Theta\Gamma) & = & \mathsf{tnf}(\Theta\Gamma', \Theta Q_1) \mid \mathsf{tnf}(\Theta\Gamma', \Theta Q_2) \\ & = & (\rho'\rho")\Theta(\mathsf{tnf}(\Gamma', Q_1) \mid \mathsf{tnf}(\Gamma', Q_2)) \\ & = & (\rho'\rho")\Theta\mathsf{tnf}(\Gamma) \end{array}$$

Dans ce cas on prendra donc $\rho = \rho'\rho$"

$P = \bot$ Le cas est trivial car $\mathsf{tnf}(\Theta\Gamma) = \emptyset = \Theta\mathsf{tnf}(\Gamma)$; il suffit donc de prendre l'identité pour ρ

$P = \neg\bot$ Dans ce cas $\mathsf{tnf}(\Gamma) = \mathsf{tnf}(\Gamma')$. Or il existe ρ' tel que $\mathsf{tnf}(\Theta\Gamma') = \rho'\Theta(\Gamma')$ par hypothèse d'induction. Il suffira donc ici encore de prendre $\rho = \rho'$.

$P = \neg\neg Q$ Par hypothèse d'induction, il existe ρ' tel que

$$\mathsf{tnf}(\Theta\Gamma', \Theta Q) = \rho'\Theta\mathsf{tnf}(\Gamma', Q)$$

autrement dit

$$\mathsf{tnf}(\Theta\Gamma) = \rho'\Theta\mathsf{tnf}(\Gamma)$$

On prendra donc $\rho = \rho'$

$P = \forall x\ Q$ Deux cas se présentent :

- Si $n_x = 1$, x n'étant pas dans le domaine de la substitution Θ, on peut écrire en utilisant l'hypothèse d'induction

$$\mathsf{tnf}(\Theta\Gamma) = \mathsf{tnf}(\Theta\Gamma', \Theta Q) = \rho'\Theta\mathsf{tnf}(\Gamma', Q) = \rho'\Theta\mathsf{tnf}(\Gamma)$$

- Le deuxième où $n_x > 1$ est à peine plus difficile. Dans ce cas $\mathsf{tnf}(\Gamma) = \mathsf{tnf}(\Gamma, Q)$. En utilisant cette égalité couplée à l'hypothèse d'induction, il vient :

$$\mathsf{tnf}(\Theta\Gamma) = \mathsf{tnf}(\Theta\Gamma, \Theta Q) = \rho'\Theta\mathsf{tnf}(\Gamma, Q) = \rho'\Theta\mathsf{tnf}(\Gamma)$$

Dans les deux cas, on prendra $\rho = \rho'$.

$P = \exists x\ Q$ C'est bien entendu dans ce cas que la preuve devient plus substantielle. Soit $y_1, ..., y_n, z_1, ..., z_p$ le label de P avec $y_1, ..., y_n$ les variables qui sont dans le domaine de Θ. Par conséquent, le label de ΘP est $z_1, .., , z_q$ puisque Θ est une substitution close. On a donc :

$$\begin{array}{rcl} \mathsf{tnf}(\Gamma) & = & \mathsf{tnf}(\Gamma', Q[x := g(y_1, ..., y_n, z_1, ..., z_q)]) \\ \mathsf{tnf}(\Theta\Gamma) & = & \mathsf{tnf}(\Theta\Gamma', (\Theta Q)[x := f(z_1, ..., z_q)]) \end{array}$$

De plus, on a

$$\Theta(Q[x := g(y_1, ..., y_n, z_1, ..., z_q)]) = (\Theta Q)[x := \Theta y_1, ..., \Theta y_n, z_1, ..., z_q]$$

Si on pose $\rho' = \{y_1, ..., y_n, z_1, ..., z_q) f(z_1, ..., z_q)/g\}$, on obtient :

$$\begin{array}{rcl} \rho'\Theta(Q[x := g(y_1, ..., y_n, z_1, ..., z_q)]) & = & (\Theta Q)[x := f(z_1, ..., z_q)] \\ & = & \Theta(Q[x := f(z_1, ..., z_q)]) \end{array}$$

On peut donc écrire

$$\rho'\Theta(\Gamma', Q[x := g(y_1, ..., y_n, z_1, ..., z_q)]) = \Theta(\Gamma', Q[x := f(z_1, ..., z_q)])$$

Par conséquent :

$$\begin{array}{rcl} \mathsf{tnf}(\Theta\Gamma) & = & \mathsf{tnf}(\Theta\Gamma', (\Theta Q)[x := f(z_1, ..., z_q)])) \\ & = & \mathsf{tnf}(\rho'\Theta(\Gamma', Q[x := g(y_1, ..., y_n, z_1, ..., z_q)])) \end{array}$$

En utilisant le lemme 7

$$\mathsf{tnf}(\Theta\Gamma) \quad = \quad \rho'\mathsf{tnf}(\Theta(\Gamma', Q[x := g(y_1, ..., y_n, z_1, ..., z_q)]))$$

Or par hypothèse d'induction :

$$\begin{array}{rcl} \mathsf{tnf}(\Theta(\Gamma', Q[x := g(y_1, ..., y_n, z_1, ..., z_q)])) & = & \rho"\Theta\mathsf{tnf}(\Gamma', Q[x := g(y_1, ..., y_n, z_1, ..., z_q)]) \\ & = & \rho"\Theta\mathsf{tnf}(\Gamma) \end{array}$$

D'où $\mathsf{tnf}(\Theta\Gamma) = \rho'\rho"\Theta\mathsf{tnf}(\Gamma)$. On prendra donc $\rho = \rho'\rho"$.

□

Lemme 14 (De TaMed vers ic-TaMeD)**.** *Soient $\mathcal{T}$ un tableau tel que :*

$$\mathcal{T} \rightsquigarrow \odot \cdot [C]$$

et Θ une substitution close, unificateur de C, associant à toutes les variables de C un terme clos. Alors :

$$\Theta\mathcal{T} \mapsto \odot$$

Démonstration. Cette preuve sera fait par induction sur la structure de la dérivation $\mathcal{T} \rightsquigarrow \odot \cdot [C]$ dont les contraintes sont satisfaites par Θ. Posons pour la suite $\mathcal{T} = \Gamma_1 \mid ... \mid \Gamma_n$.

Si la dérivation est de taille nulle, $\mathcal{T}$ a été « vidé » par la transformation tnf, ce qui permet de conclure que trivialement $\Theta\mathcal{T} \mapsto \odot$, avec Θ qui est l'identité.

Dans le cas contraire la dérivation TaMed vers le tableau fermé commence par produire un tableau $\mathcal{T}'$ qui a une plus petite dérivation vers le tableau clos. Par hypothèse d'induction on sait également qu'il existe une substitution Θ' telle que $\Theta'\mathcal{T}' \mapsto \odot$. Regardons les règles pouvant produire $\mathcal{T}'$:

Fermeture étendue Il existe une branche, disons Γ_1, de $\mathcal{T}$ qui contient deux formules P et $\neg Q$ telles que $P =_{\mathcal{E}} Q$. Comme toutes les contraintes de $\mathcal{T}$ sont des contraintes de $\mathcal{T}'$, elles sont unifiées par Θ'. Or $\Theta'\mathcal{T}'$ peut être produit depuis $\Theta'\mathcal{T}$ dans ic-TaMeD en utilisant les règles de *Conversion* et de *Fermeture simple*. Par hypothèse d'induction $\Theta'\mathcal{T}' \mapsto \odot$. Or $\Theta'\mathcal{T} \mapsto \Theta'\mathcal{T}'$ donc $\Theta'\mathcal{T} \mapsto \odot$.

Narrowing étendu Là encore, il y a une branche de $\mathcal{T}$, disons Γ_1 qui peut être réécrite en utilisant une règle de type $l \to r$ en Γ'_1 et $\mathsf{tnf}(\Gamma'_1) = \mathcal{U} = \mathcal{B}_1 \mid ... \mid \mathcal{B}_p$. Par conséquent, $\mathcal{T}' = \mathcal{B}_1 \mid ... \mid \mathcal{B}_p \mid \Gamma_2 \mid ... \mid \Gamma_n$. Nommons ω l'occurrence de Γ_1 où le *Narrowing étendu* est appliqué, tel que $\Gamma'_1 = \Gamma_1[r]_{|\omega}$. En particulier on a $\Theta\Gamma_1[l]_{|\omega} \to \Theta\Gamma'_1$.

$\mathcal{T}'$ est contraint par l'équation $\Gamma_{1_{|\omega}} \stackrel{?}{=}_{\mathcal{E}} l$. On sait donc que $\Theta\Gamma_{1_{|\omega}} =_{\mathcal{E}} \Theta l$ et $\Theta\Gamma_1 =_{\mathcal{E}} \Theta(\Gamma_1[l]_{|\omega})$. Par hypothèse, Θl est fermé on peut mettre le même label sur les formules de ces branches et dériver $\Theta(\Gamma_1[l]_{|\omega})$ depuis $\Theta\Gamma_1$ en utilisant la règle de *Conversion*.

Soit Θ_1 la restriction de Θ aux variables libres produites par les variables liées dans Γ'_1, et soit Θ_2 sa restriction aux autres variables.

D'après le lemme 13 précédent, il existe une transformation ρ des symboles de fonction introduits en mettant Γ'_1 en forme tableau-normale telle que $\mathsf{tnf}(\Theta_2\Gamma'_1) = \rho\Theta_2\mathsf{tnf}(\Gamma'_1)$. Donc

$$\mathsf{tnf}(\Theta\Gamma'_1) = \mathsf{tnf}(\Theta_2\Theta_1\Gamma'_1) = \rho\Theta_2\mathsf{tnf}(\Gamma'_1)$$

ne s'applique qu'aux variables libérées par la mise en forme normale de Γ'_1.

De plus

$$\Theta_1\mathsf{tnf}(\Theta\Gamma'_1) = \Theta_1\rho\Theta_2\mathsf{tnf}(\Gamma'_1) = \rho\Theta\mathsf{tnf}(\Gamma'_1)$$

car $\rho\Theta_1 = \Theta_1\rho$. Or le tableau $\rho\Theta\mathsf{tnf}(\Gamma'_1)$ peut être dérivé de $\Theta\Gamma[\Theta l]_{|\omega}$ en utilisant les règles ic-TaMeD de *Réduction* et d'*Instanciation*.

On a par hypothèse d'induction $\Theta\mathcal{T}' \mapsto \odot$, d'où par le lemme 8 :

$$\rho\Theta\mathcal{T}' \mapsto \odot$$

On peut exhiber la dérivation ic-TaMeD suivante à partir de $\Theta\mathcal{T}$:

$$\begin{array}{ll} & \Theta\mathcal{T} = \Theta\Gamma_1 \mid ... \mid \Theta\Gamma_n \\ \mapsto_{\text{Conversion}} & \Theta\Gamma_1[\Theta l]_{|\omega} \mid ... \mid \Theta\Gamma_n \\ \mapsto_{\text{Reduction}} & \mathsf{tnf}(\Theta\Gamma_1[\Theta \mathbf{r}]_{|\omega} \mid ... \mid \Theta\Gamma_n) \\ = & \rho\Theta(\Gamma_1[r]_{|\omega} \mid ... \mid \Gamma_n) \\ = & \rho\Theta(\mathcal{B}_1 \mid ... \mid \mathcal{B}_p \mid \Gamma_2 \mid ... \mid \Gamma_n) \\ & \mapsto \odot \end{array}$$

Donc $\Theta\mathcal{T} \mapsto \odot$

□

À présent, on peut procéder à la démonstration de la correction de TaMed.

Proposition 4 (Correction de TaMed)**.** *Soit $\mathcal{T}$ un tableau tel que*

$$\mathcal{T} \rightsquigarrow \odot \cdot [C]$$

où C est un ensemble de contraintes unifiable. Alors

$$\mathcal{T} \mapsto \odot$$

Démonstration. L'ensemble des contraintes C est satisfiable : il existe donc une substitution Θ associant aux variables de C les termes permettant la fermeture du tableau. D'après le lemme 14 précédent $\Theta\mathcal{T} \mapsto \odot$ et les branches de $\Theta\mathcal{T}$ peuvent être simplement dérivées de celles de $\mathcal{T}$ en utilisant la règle d'*Instanciation* : donc $\mathcal{T} \overset{\text{Inst}}{\mapsto}{}^* \Theta\mathcal{T}$.

Par conséquent $\mathcal{T} \mapsto \odot$ □

4.5.2 Complétude de TaMed

D'une manière similaire à ce qu'on vient de faire pour la preuve de correction, on va également relever le résultat de complétude de $ic-TaMeD$ vers $TaMed$. On va premièrement définir un lemme intermédiaire proche du lemme 14.

Lemme 15. *Soient $\mathcal{B}$ and Γ deux branches de propositions avec labels. Soit par ailleurs une substitution Θ telle qu'aucune variable liée dans $\mathcal{B}$ ne soit dans son domaine. Supposons par ailleurs $\Theta\mathcal{B} =_{\mathcal{E}} \Gamma$.*

Alors il existe une transformation ρ des symboles de fonction introduits en mettant Γ en forme normale de tableau telle que

$$\Theta\mathsf{tnf}(\mathcal{B}) = \rho\mathsf{tnf}\Gamma$$

Démonstration. La démonstration se fait par induction sur la branche $\mathcal{B}$ en utilisant l'ordre tnf.

Dans le cas de base, toutes les propositions sont des littéraux et donc :

$$\Theta\mathsf{tnf}(\mathcal{B}) = \Theta\mathcal{B} =_{\mathcal{E}} \Gamma = \mathsf{tnf}(\Gamma)$$

Il suffit alors de prendre l'identité pour ρ.

Sinon, il existe une proposition P de $\mathcal{B}$ qui n'est pas un littéral. Posons $\mathcal{B} = \{\mathcal{B}', P\}$ et $\Gamma = \{\Gamma', P'\}$ avec $\Theta\mathcal{B}' =_{\mathcal{E}} \Gamma'$ et $\Theta P =_{\mathcal{E}} P'$.

– Si $P = Q_1 \wedge Q_2$, alors $P' = Q'_1 \wedge Q'_2$ et $\Theta Q_1 = Q'_1, \Theta Q_2 = Q'_2$.
Par hypothèse d'induction

$$\Theta\mathsf{tnf}(\mathcal{B}', Q_1, Q_2) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma', Q'_1, Q'_2)$$

i.e. $\Theta\mathsf{tnf}(\mathcal{B}') =_{\mathcal{E}} \mathsf{tnf}(\Gamma')$.
– Si $P = \bot$ alors $P' = \bot$. On a alors $\Theta\mathsf{tnf}(\mathcal{B}) = \emptyset = \Theta\mathsf{tnf}(\Gamma)$. On prendra l'identité pour ρ.
– Si $P = \neg \bot$ alors $P' = \neg \bot$.
Par hypothèse d'induction

$$\Theta\mathsf{tnf}(\mathcal{B}') =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma')$$

i.e. $\Theta\mathsf{tnf}(\mathcal{B}) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma)$.
– Si $P = \neg\neg Q$ alors $P' = \neg\neg Q'$ et $\Theta Q =_{\mathcal{E}} Q'$.
Par hypothèse d'induction

$$\Theta\mathsf{tnf}(\mathcal{B}', Q) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma', Q')$$

i.e. $\mathsf{tnf}(\mathcal{B}) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma)$.
– Si $P = Q_1 \vee Q_2$ alors $P' = Q'_1 \vee Q'_2$ et $\Theta Q_1 = Q'_1, \Theta Q_2 = Q'_2$.
Par hypothèse d'induction

$$\Theta\mathsf{tnf}(\mathcal{B}', Q_1) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma', Q'_1)$$

et

$$\Theta\mathsf{tnf}(\mathcal{B}', Q_2) =_{\mathcal{E}} \rho'\mathsf{tnf}(\Gamma', Q'_2)$$

Comme les domaines de ρ et ρ' sont disjoints :

$$\begin{aligned}\Theta\mathsf{tnf}(\mathcal{B}) &= \Theta\mathsf{tnf}(\mathcal{B}', Q_1)|\Theta\mathsf{tnf}(\mathcal{B}', Q_2)\\ &=_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma', Q'_1)|\rho'\mathsf{tnf}(\Gamma', Q'_2)\\ &= (\rho \cup \rho')\mathsf{tnf}(\Gamma)\end{aligned}$$

– Si $P = \exists x Q$ alors $P' = \exists x Q'$ et $\Theta Q =_{\mathcal{E}} Q'$. Soit $y_1, \ldots, y_n$ le label de P et $z_1, \ldots, z_q$ les variables libres de $\Theta y_1, \ldots, \Theta y_n$. Le label de ΘP et P' est $z_1, \ldots, z_q$.
On a $\mathsf{tnf}(\mathcal{B}) = \mathsf{tnf}(\mathcal{B}, Q[x := f(y_1, \ldots, y_n)]$ et $\mathsf{tnf}(\Gamma) = \mathsf{tnf}(\Gamma', Q'[x := g(z_1, \ldots, z_q)])$.
Comme $\Theta Q =_{\mathcal{E}} Q'$,

$$\begin{aligned}\Theta(Q[x := f(y_1, \ldots, y_n)]) &= (\Theta Q)[x := f(\Theta y_1, \ldots, \Theta y_n)]\\ &=_{\mathcal{E}} Q'[x := f(\Theta y_1, \ldots, \Theta y_n)]\\ &= \rho Q'[x := g(z_1, \ldots, z_q)]\end{aligned}$$

avec $\rho = \{(z_1, \ldots, z_q) f(\Theta y_1, \ldots, \Theta y_n)/g\}$. Ainsi

$$\Theta(\mathcal{B}', Q[x := f(y_1, \ldots, y_n)]) =_{\mathcal{E}} \rho(\Gamma', Q'[x := g(z_1, \ldots, z_q)])$$

et par hypothèse d'induction

$$\Theta\mathsf{tnf}(\mathcal{B}', Q[x := f(y_1, \ldots, y_n)]) =_{\mathcal{E}} \rho'\mathsf{tnf}(\rho(\Gamma', Q'[x := g(z_1, \ldots, z_q)]))$$

et en appliquant le lemme 7

$$\Theta\mathsf{tnf}(\mathcal{B}', Q[x := f(y_1, \ldots, y_n)]) =_{\mathcal{E}} \rho'\rho\mathsf{tnf}(\Gamma', Q'[x := g(z_1, \ldots, z_q)])$$

i.e. $\Theta\mathsf{tnf}(\mathcal{B}) =_{\mathcal{E}} \rho'\rho\mathsf{tnf}(\Gamma)$

– Si $P = \forall x Q$ alors $P' = \forall x Q'$ et $\Theta Q =_{\mathcal{E}} Q'$. On examine les cas suivant la valeur de n_x pour ce quantificateur universel.
  – Si $n_x = 1$ alors
  $$\Theta\mathsf{tnf}(\mathcal{B}', Q) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma', Q')$$
  i.e. $\Theta\mathsf{tnf}(\mathcal{B}) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma)$.
  – Si $n_x > 1$ alors, comme $\Theta\mathcal{B}' =_{\mathcal{E}} \Gamma'$, $\Theta P =_{\mathcal{E}} P'$ et $\Theta Q =_{\mathcal{E}} Q'$
  $$\Theta\mathsf{tnf}(\mathcal{B}, Q) =_{\mathcal{E}} \rho\mathsf{tnf}(\Gamma, Q)$$

– Les cas $\neg(Q_1 \vee Q_2)$ et $\neg(Q_1 \Rightarrow Q_2)$ sont traités comme le cas $Q_1 \wedge Q_2$. Les cas $\neg(Q_1 \wedge Q_2)$ et $Q_1 \Rightarrow Q_2$ sont similaires au cas $Q_1 \vee Q_2$. Le cas $\neg(\forall z; Q)$ est semblable au cas $\exists z; Q$ et $\neg(\exists z; Q)$ à $\forall z; Q$.

□

On peut maintenant établir la complétude de TaMed vis-à-vis d'ic-TaMeD.

Proposition 5. *Soit $\mathcal{U}$ un tableau avec contraintes. Soient Θ une substitution satisfaisant les contraintes de $\mathcal{U}$ et $\mathcal{T}$ un tableau sans contraintes tels que :*

$$\Theta\mathcal{U} =_{\mathcal{E}} \mathcal{T}$$

et

$$\mathcal{T} \mapsto \odot.$$

Alors

$$\mathcal{U} \rightsquigarrow \odot \cdot [C]$$

où C est un ensemble de contraintes satisfiable.

Démonstration. Cette démonstration sera faite par induction sur la structure de la dérivation ic-TaMeD de $\mathcal{T}$. Posons pour cette preuve :

$$\begin{aligned} \mathcal{T} &= \mathcal{B}_1 \mid \ldots \mid \mathcal{B}_n \\ \mathcal{U} &= \Gamma_1 \mid \ldots \mid \Gamma_p \end{aligned}$$

Si la dérivation ic-TaMeD de $\mathcal{T}$ est vide, alors $\mathcal{T}$ est formé de branches fermées (par tnf). Par conséquent

$$\mathcal{U} \rightsquigarrow \odot \cdot \emptyset$$

et Θ est l'identité.

Sinon, la dérivation ic-TaMeD de $\mathcal{T}$ commence par produire un tableau $\mathcal{T}'$ et $\mathcal{T}'$ a une plus petite dérivation que $\mathcal{T}$ vers le tableau fermé. Détaillons les quatre règles possibles pour passer de $\mathcal{T}$ à $\mathcal{T}'$:

Instanciation : Dans ce cas

$$\mathcal{T}' = \mathcal{T}[x := t]$$

. Par hypothèse, $\Theta\mathcal{U} =_{\mathcal{E}} \mathcal{T}$. Donc

$$(\Theta\mathcal{U})[x := t] =_{\mathcal{E}} \mathcal{T}[x := t] = \mathcal{T}'$$

Soit $\Theta' = [x := t] \circ \Theta$, alors $\Theta'\mathcal{U} =_{\mathcal{E}} \mathcal{T}'$ et Θ' satisfait les contraintes de $\mathcal{U}$.

Comme $\mathcal{T}' \mapsto \odot$ par définition et $\Theta'\mathcal{U} =_{\mathcal{E}} \mathcal{T}'$, on obtient par hypothèse d'induction $\mathcal{U} \rightsquigarrow \odot$.

Conversion : Dans ce cas, la conclusion est immédiate en utilisant la définition de la règle de *Conversion* :

$$\Theta\mathcal{U} =_{\mathcal{E}} \mathcal{T} =_{\mathcal{E}} \mathcal{T}'$$

L'hypothèse d'induction peut être utilisée sur $\Theta\mathcal{U} =_{\mathcal{E}} \mathcal{T}'$ et$\mathcal{T}' \mapsto \odot$ pour obtenir que $\mathcal{U} \rightsquigarrow \odot$.

Fermeture simple : Dans ce cas, $\mathcal{T}$ a une branche, disons $\mathcal{B}_1$ qui contient un littéral P et sa négation $\neg P$. Comme $\Theta\mathcal{U} =_{\mathcal{E}} \mathcal{T}$, il existe une branche de $\mathcal{U}$, disons Γ_1, contenant deux littéraux P' et Q' tels que

$$\Theta P' =_{\mathcal{E}} P =_{\mathcal{E}} \Theta Q'$$

.

Posons $\mathcal{U}' = \mathcal{U} \backslash \{\Gamma_1\}$. Alors $\Theta\mathcal{U}' =_{\mathcal{E}} \mathcal{T}'$. En appliquant l'hypothèse d'induction à $\Theta\mathcal{U}' =_{\mathcal{E}} \mathcal{T}'$ et $\mathcal{T}' \mapsto \odot$ (par construction), on obtient

$$\mathcal{U}' \rightsquigarrow \odot \cdot [C]$$

avec Θ satisfaisant C.

L'application de la règle TaMed de *Fermeture étendue* aux littéraux P' et Q' définis précédemment produit exactement le tableau $\mathcal{U}'$. Donc :

$$\mathcal{U} \overset{\text{fer. ét.}}{\rightsquigarrow} \mathcal{U}' \rightsquigarrow \odot \cdot [C]$$

Réduction : Dans ce cas, il existe une branche de $\mathcal{T}$, disons $\mathcal{B}_1 = \{\mathcal{B}, P\}$, contenant un littéral P tel que :

$$\begin{array}{rcl} P & \rightarrow_{\mathcal{R}} & Q \\ \mathsf{tnf}(\mathcal{B}, P) & = & \mathcal{B}'_1 \mid ... \mid B'_q \end{array}$$

De plus, il existe un tableau avec contraintes $\mathcal{U} \cdot [C_1]$ contenant une branche, disons $\Gamma_1 = \{\Gamma, P'\}$, telle que $\Theta\Gamma =_{\mathcal{E}} \mathcal{B}$ et $\Theta P' =_{\mathcal{E}} P$.

Comme $\Theta P' =_{\mathcal{E}} P$, $P \rightarrow_{\mathcal{R}} Q$, $\mathcal{R}$ ne s'applique qu'aux propositions atomiques et que P' est par définition un littéral, alors on peut en déduire que

$$\Theta P' \xrightarrow{\mathcal{R}\mathcal{E}} Q$$

.

La proposition $\Theta P'$ contient donc une occurrence ω telle qu'il existe une règle $l \to r$ de $\mathcal{R}$ et une substitution σ telles que

$$\Theta P'_{|\omega} =_{\mathcal{E}} \sigma l$$

et

$$\Theta P'[\sigma r]_\omega = Q$$

On a par ailleurs que $(\Theta P')_{|\omega} = \Theta(P'_{|\omega})$.

Posons $Q' = P'[r]_{|\omega}, C_1 \cup \{P'_{|\omega} \neq^? l\}$ et $\Theta' = \Theta \circ \sigma$. le domaine de σ ne contient que des variables fraîches donc Θ' est une substitution satisfaisant C' telle que

$$\Theta'\Gamma = \Theta\Gamma =_{\mathcal{E}} \mathcal{B}$$

et

$$\Theta'Q' = \Theta'(P'[r]_{|\omega}) = (\Theta'P')[\Theta'r]_{|\omega} = (\Theta P')[\sigma r]_{|\omega} = Q$$

Or $\Theta'(\Gamma, Q') =_{\mathcal{E}} \mathcal{B}, Q$ et, puisque la substitution σ n'affecte que la branche $\{\Gamma, Q\}$:

$$\Theta'(\Gamma, Q' \mid \Gamma_2 \mid ... \mid \Gamma_p) =_{\mathcal{E}} (\mathcal{B}, Q\mathcal{B}_2 \mid ... \mid \mathcal{B}_n)$$

Soit $\mathcal{U}' = \mathsf{tnf}(\Gamma, Q' \mid \Gamma_2 \mid ... \mid \Gamma_p)$. Le lemme 15 précédent permet de dire qu'il existe une transformation des symboles de fonction ρ telle que :

$$\Theta'\mathcal{U}' =_{\mathcal{E}} \rho\mathsf{tnf}(\mathcal{B}, Q \mid \mathcal{B}_2 \mid ... \mid \mathcal{B}_n) = \rho\mathcal{T}'$$

Comme $\mathcal{T}' \mapsto \odot$, alors $\rho\mathcal{T}' \mapsto \odot$ par le lemme 8 et les deux dérivations ont la même longueur. Alors l'hypothèse d'induction peut être appliquée sur $\Theta\mathcal{U}'$ et $\rho\mathcal{T}'$ pour conclure que $\mathcal{U}' \cdot [C'] \rightsquigarrow \odot \cdot [C]$ où C est un ensemble de contraintes satisfiable.

Aappliquer la règle de *Narrowing étendu* à $\mathcal{U}$ et plus précisément à sa branche $\mathcal{B}_1$ produit $\mathcal{U}'$ en quelques étapes. Donc :

$$\mathcal{U} \overset{\text{Narrowing}}{\rightsquigarrow} \mathcal{U}' \rightsquigarrow \odot \cdot [C]$$

avec C, ensemble de contraintes satisfiable.

□

4.6 Conclusion

Nous avons formulé une méthode des tableaux pour la déduction modulo, en nous appuyant sur un raisonnement syntaxique dont la construction est semblable à celui de la méthode de résolution ENAR ([29]). Les différences obtenues proviennent essentiellement du traitement des métavariables dans les méthodes de tableaux. Cela nous oblige à définir d'une part à utiliser une procédure d'« iterative deepening » dans les règles d'expansion. D'autre part, nous sommes forcés

de gérer les contraintes globalement et d'appliquer toute substitution au tableau entier, alors que la résolution permet, elle, de traiter ces étapes localement.

Les preuves de correction et de complétude (faible) ont donc été établies en définissant un système intermédiaire dont les inférences sont plus atomiques ayant ces mêmes propriétés. Puis, celles-ci ont été relevées vers les règles propres de TaMed.

Cette formulation de TaMed reste encore relativement éloignée d'une implémentation pratique. L'utilisation de skolemisation externe dans TaMed est par exemple liée aux preuves syntaxiques, et permet d'établir les lemmes relatifs aux transformations de symbole (de Skolem). On ne la choisira pas a priori si on recherche l'efficacité.

Cependant, nous avons pu par cette formulation très générale des tableaux modulo démontrer les propriétés théoriques essentielles attendues d'une méthode de preuve.

Chapitre 5

Tableaux sémantiques classiques modulo

Depuis le chapitre 1, on a vu deux façons complémentaires de traiter les méthodes de tableaux : du point de vue analytique, on s'intéresse plutôt au contenu structurel et syntaxique du processus alors que du point de vue sémantique, on s'attache à déterminer la signification de la méthode. Au niveau des tableaux pour la déduction modulo, la formulation du chapitre 4 précédent (voir plus précisément la section 4.1) s'intéresse particulièrement au contenu syntaxique de la méthode. En effet, on y démontre de manière syntaxique que la méthode des tableaux modulo permet de trouver une preuve chaque fois qu'il en existe une dans les séquents modulo (et réciproquement).

L'approche sémantique que nous allons adopter dans le présent chapitre permet tout d'abord de développer des modèles de la déduction modulo alors que nous avons essentiellement aperçu jusque là les aspects syntaxiques du domaine. Grâce aux méthodes développées dans [39], cela permettra non seulement d'apporter une confirmation des théorèmes déjà prouvés pour les tableaux modulo mais aussi de nouveaux résultats. Par exemple, les rapports avec l'élimination des coupures seront précisés, et la construction de modèles obligera également à énoncer plus concrètement la manière dont sont appliquées les règles.

Les résultats présentés dans ce chapitre sont le fruit d'un travail commun avec Olivier Hermant [11].

5.1 Reformulation de TaMed

Les preuves sémantiques travaillent sur une reformulation de TaMed, dans laquelle nous n'utilisons pas d'axiomes équationnels. Le système de réécriture $\mathcal{R}$ que nous utilisons comprend des règles de réécriture sur les propositions et les termes. Rappelons ici que nous nous restreignons à n'avoir que des systèmes de réécriture confluents et noetheriens.

5.1.1 Règles

Les règles de la méthode des tableaux (sémantiques) modulo sont données en figure 5.1.

Dans ce calcul, les contraintes sont non seulement associées au tableau lors des étapes de fermeture mais également aux formules dans les branches lors des étapes de réécriture.

Définition 30 (Formule avec contraintes)**.** Soit P une formule et c un ensemble d'équations appelé contraintes. Alors la paire P_c est appelée formule avec contraintes.

Les contraintes associées aux formules gardent trace des problèmes d'unification qui ont dû être résolus pour permettre la réécriture. Cependant, il n'est pas immédiatement nécessaire de propager globalement ces contraintes : si les formules résultantes d'étapes de réécriture ne sont utilisées dans aucune règle de fermeture, elles sont inutiles du point de vue de la preuve : la réécriture était en fait inutile.

$$\frac{\Gamma_1, \beta(P,Q)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, \beta(P,Q)^l_c, P^l_c \mid \Gamma_1, \beta(P,Q)^l_c, Q^l_c \mid ... \mid \Gamma_n}\ \beta$$

$$\frac{\Gamma_1, \alpha(P,Q)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, \alpha(P,Q)^l_c, P^l_c, Q^l_c \mid ... \mid \Gamma_n}\ \alpha$$

$$\frac{\Gamma_1, \gamma(x,P)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, P(x := X)^{l\cup\{X\}}_c, \gamma(x,P)^l_c \mid ... \mid \Gamma_n}\ \gamma$$

$$\frac{\Gamma_1, \delta(x,P)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, P^l_c[\mathrm{x} :=\mathtt{sko}(l)], \delta(x,P)^l_c \mid ... \mid \Gamma_n}\ \delta$$

$$\frac{\Gamma_1, P^l_c \mid ... \mid \Gamma_n \cdot \mathcal{C}}{\Gamma_1, P^l_c, P^l_{\mathcal{K}}[d]_\omega \mid ... \mid \Gamma_n \cdot \mathcal{C}}\ \mathtt{rw} \text{ si } g \to d, \text{ et } \rfloor' = (c \cup \{P_{|\omega} \stackrel{?}{=} g\}$$

$$\frac{\Gamma_1, P^{l_1}_{c_1}, \neg P^{l_2}_{c_2} \mid ... \mid \Gamma_n \cdot C}{(\Gamma_2 \mid ... \mid \Gamma_n) \cdot \mathcal{C} \cup c_1 \cup c_2 \cup \{P^{l_1} \stackrel{?}{=} P^{l_2}\}}\ \mathtt{closure}\ (\odot)$$

FIG. 5.1 – Tableau modulo : règles d'expansion et de fermeture.

Les règles sont celles des tableaux à métavariables auxquelles a été ajoutée une règle gérant la réécriture sur les propositions et termes.

- Dans la γ-règle, une métavariable fraîche pour tout le tableau est substituée à la variable universellement quantifiée x. Cette métavariable est aussi ajoutée au label de la formule.
- Comme dans le chapitre précédent, la skolemisation dans la δ-règle est

traitée par l'utilisation de labels. On utilise également toujours la skolemisation externe comme méthode.

- La règle `rw` est celle qui construit les contraintes locales aux formules du tableau, dénotées par c'. La formule résultante de la réécriture porte avec elle le filtrage nécessaire entre l'occurrence ω de la prémisse à réécrire et la partie gauche l de la règle de réécriture utilisée. La propagation de cette contrainte au tableau entier ne sera faite que dans le cas où la formule créée (ou une de ses descendantes par les règles d'inférence de tableau) sera effectivement une prémisse d'une règle de fermeture.
- La règle de fermeture `closure` efface la branche à condition que les contraintes sur celles-ci soient unifiables. Les labels d'une formule et de sa négation n'ont pas besoin d'être les mêmes : ce serait le cas par exemple des formules $P(X)$ et $\neg P(a)$. On a donc besoin d'unifier certains sous-termes des deux formules-prémisses ($P^{l_1} \stackrel{?}{=} P^{l_2}$) et d'appliquer la substitution calculée à tout le tableau. C'est pourquoi l'ensemble de contraintes de fermeture $\mathcal{C}$ est global au tableau. La règle `closure` est aussi le moment de transférer les contraintes locales sur les formules au niveau global du tableau. Si cette règle ne semble pas poser de problème de correction, la complétude du calcul n'est assuré que si on l'utilise de manière raisonnée comme en section 5.1.2.

Le calcul présenté est explicitement non destructif puisque dans toutes les règles d'inférence, la formule sur laquelle s'applique la règle est présente. Cependant, on verra que la façon systématique de générer un tableau (complet) permet d'obtenir une méthode plus restrictive.

5.1.2 Génération systématique

La première étape vers la démonstration de la complétude de cette méthode est de définir un ordre d'application dans les règles et de montrer qu'en le suivant, la méthode est effectivement complète. La fermeture incrémentale de la section 1.3.3 servira de socle pour définir la façon d'utiliser nos règles. Dans cette optique, on définit une application équitable des règles et on construit graduellement un ensemble de contraintes que l'on essaie de satisfaire pour fermer le tableau

Il faut donc s'assurer premièrement que la façon d'appliquer les règles de tableaux est équitable. Puis on construira petit-à-petit une approximation d'un tableau complet dans le cas où la proposition Ψ en entrée n'est pas prouvable. Après chaque application de règles, on tente de fermer le tableau en résolvant ses contraintes d'unification et de filtrage.

L'idée est informellement d'imposer par un ordre une sorte de rotation dans les branches que l'on sélectionne en prenant toujours la plus petite branche du tableau et d'appliquer sur celle-ci les règles les plus déterministes et les moins coûteuses possibles.

- Tout d'abord on attache un booléen (`vrai`) à la branche d'entrée. La valeur de ce booléen sera utilisée pour sélectionner alternativement l'application

de γ ou de `rw` : l'application de ces règles est possiblement infinie mais il faut continuer à rester équitable. On se munit aussi d'une fonction d'accès à la valeur du booléen pour chaque branche : ainsi pour une branche $\mathcal{B}$, boolean($\mathcal{B}$) retournera la valeur de celui-ci. La fonction setbool($\mathcal{B}$, val) permettra de donner une valeur spécifique val au booléen de $\mathcal{B}$.

- On définit au préalable deux ordres pour sélectionner la formule à utiliser dans la prochaine expansion :
  - un ordre $\prec_{\mathcal{B}}$ sur les branches :
    On définit une fonction size($\mathcal{B}$) qui donne le nombre de formules sur $\mathcal{B}$: si $\mathcal{B} = \{P, Q, S \vee T\}$, alors $sz(\mathcal{B}) = 4$.
    On définit une deuxième fonction $lof(\mathcal{B}_1, \mathcal{B}_2)$ qui retourne true si $\mathcal{B}_1$ est à gauche de $\mathcal{B}_2$. Ainsi, dans le tableau $\mathcal{B}_1 \mid \mathcal{B}_2 \mid \mathcal{B}_3$, $\mathcal{B}_2$ est à gauche de $\mathcal{B}_3$ et à droite de $\mathcal{B}_1$.
    Alors

$$B_1 \prec_{\mathcal{B}} \mathcal{B}_2 \quad \hat{=} \quad \begin{array}{l} \text{size}(\mathcal{B}_1) < \text{size}(\mathcal{B}_2) \\ \text{ou size}(\mathcal{B}_1) = \text{size}(\mathcal{B}_2) \text{ et } lof(\mathcal{B}_1, \mathcal{B}_2) = \mathsf{true} \end{array}$$

  - un ordre $\prec_{\mathrm{f}}$ sur les types des formules sur les branches : étant donnée une branche $\mathcal{B}$,

$$\alpha \prec_{\mathrm{f}} \delta \prec_{\mathrm{f}} \beta \prec_{\mathrm{f}} \mathtt{rw} \prec_{\mathrm{f}} \gamma$$

    si boolean($\mathcal{B}$) =`true`, et

$$\alpha \prec_{\mathrm{f}} \delta \prec_{\mathrm{f}} \beta \prec_{\mathrm{f}} \gamma \prec_{\mathrm{f}} \mathtt{rw}$$

    sinon.
- Par ailleurs, afin de ne pas sélectionner toujours la même formule au sein d'une branche, les propositions seront annotées comme u (utilisée) on nu (non utilisée). L'enregistrement du statut des formules permet de simuler un noyau destructif dans le calcul pour les règles α, β et δ.

On va composer les ordres $\prec_{\mathcal{B}}$ et $\prec_{\mathrm{f}}$ pour sélectionner d'abord une branche puis une formule sur celle-ci.

Puis on procédera comme suit :

1. Toutes les formules de la branche en entrée sont marquées comme *nu*. Si on applique α, β ou δ, alors la formule à laquelle on a appliqué la règle devient u (cela permet par la suite d'interdire l'application de cette règle à nouveau). La branche produite (ou les branches produites dans le cas de β) hérite la valeur booléenne de la branche contenant la formule-prémisse.
   En ce qui concerne `rw`, la formule P de la branche $\mathcal{B}$ à laquelle on applique la règle $r \in \mathcal{R}$ est marquée comme $u(r)$ (utilisée pour la règle r) pour ne pas appliquer deux fois la même règle de réécriture à P. Par ailleurs le booléen de la branche devient sa négation soit setbool($\mathcal{B}$,¬boolean($\mathcal{B}$)).
   Pour γ, la seule modification à faire concerne la valeur du booléen de la branche : setbool($\mathcal{B}$,¬boolean($\mathcal{B}$)).
   Dans tous les cas, les nouvelles formules ajoutées à la branche sont marquées non utilisées *nu*.

2. Après chaque expansion on teste pour savoir si la fermeture du tableau est possible.
   (a) Sur chaque branche, on génère l'ensemble des contraintes qui peuvent la fermer (chaque paire de formules potentiellement unifiables produit une contrainte).

   La collecte de ces ensembles locaux aux branches permet de produire l'ensemble global de contraintes qui peuvent fermer simultanément le tableau. S'il existe une substitution satisfiant cet ensemble alors la formule est *insatisfiable*.

   Si une branche peut être fermée sans unification, on peut la retirer du tableau (ainsi que l'ensemble de contraintes qui lui est associé).
   (b) Sélectionner la plus petite branche que l'on peut étendre (selon $\prec_{\mathcal{B}}$). Déterminer l'ensemble S de ses plus petites formules selon $\prec_{\mathrm{f}}$. Appliquer la règle d'expansion adéquate à toutes les formules nu de S (ou $nu(r)$ si on applique `rw` avec r). Si aucune expansion n'est applicable, la formule en entrée est *satisfiable* sinon on revient à l'étape 2a.

5.1.3 Correction

Modèles booléens

Nous allons à partir de cette section aborder des preuves sémantiques fondées sur des extensions de modèles booléens pour la déduction modulo. Rappelons tout d'abord la notion de modèle boolée.

La sémantique s'appuie sur les algèbres de Boole, et plus précisément sur la forme la plus simple des algèbres de Boole : $\{0, 1\}$, où les propositions sont interprétées par deux valeurs de vérité complémentaires : Vrai (1) et Faux (0).

Définition 31. Soit $\mathcal{L}$ un langage. Une structure $\mathcal{M}$ est un ensemble formé d'un domaine M non vide, et pour chaque symbole de prédicat P d'arité n et de fonction f d'arité m du langage $\mathcal{L}$:
- une fonction $\hat{P} : M^n \mapsto \{0, 1\}$
- une fonction $\hat{f} : M^m \mapsto M$

Cette structure ne nous permet pas pour l'instant d'interpréter les propositions même atomiques, ni tous les termes de notre langage $\mathcal{L}$. Nous avons défini la seule interprétation des symboles de prédicat et des symboles de fonction (donc, les termes non composés). Il faut donc maintenant étendre cette interprétation. Pour ce faire, nous définissions l'interprétation d'un terme (non clos en règle générale), suivant un assignement σ qui à chaque variable associe un terme clos.

Définition 32. Soit $\mathcal{L}$ un langage, $\mathcal{V}$ l'ensemble de ses variables, $\mathcal{M}$ une structure, et σ un assignement. Nous définissons l'interprétation d'un terme selon σ par induction sur la structure du terme :
- $|x|_\sigma = \sigma(x)$ quand $x \in \mathcal{V}$
- $|f(t_1, ..., t_m)|_\sigma = \hat{f}(|t_1|_\sigma, ..., |t_m|_\sigma)$

Ensuite, nous définissions la valuation (ou l'interprétation) d'une proposition selon l'assignement σ par induction sur la structure de la proposition :

- $|P(t_1, ..., t_n)|_\sigma = \hat{P}(|t_1|_\sigma, ..., |t_n|_\sigma)$
- $|\neg P|_\sigma = 1$ ssi $|P|_\sigma = 0$. Sinon $|\neg P|_\sigma = 0$
- $|P \vee Q|_\sigma = 1$ ssi $|P|_\sigma = 1$ ou $|Q|_\sigma = 1$. Sinon $|P \vee Q|_\sigma = 0$
- $|P \wedge Q|_\sigma = 1$ ssi $|P|_\sigma = 1$ et $|Q|_\sigma = 1$. Sinon $|P \wedge Q|_\sigma = 0$
- $|P \vee Q|_\sigma = 1$ ssi $|P|_\sigma = 0$ ou $|Q|_\sigma = 1$. Sinon $|P \vee Q|_\sigma = 0$
- $|\exists x P|_\sigma = 1$ s'il existe $a \in M$ tel que $|P|_{\sigma:\langle x,a\rangle} = 1$. Sinon $|\exists x P|_\sigma = 0$
- $|\forall x P|_\sigma = 1$ si pour tout $a \in M$, $|P|_{\sigma:\langle x,a\rangle} = 1$. Sinon $|\forall x P|_\sigma = 0$

Nous dirons que la structure $\mathcal{M}$ est un modèle d'une théorie Γ si toutes les propositions de Γ sont interprétées par 1 dans $\mathcal{M}$, selon tous les assignements possibles σ. Nous utiliserons pour cela la notation $\mathcal{M} \models \Gamma$. Nous dirons que $\mathcal{M}$ est un modèle lorsque nous parlerons de la structure $\mathcal{M}$ équipée de l'interprétation précédente.

Enfin, nous noterons $\Gamma \models \Delta$ lorsque les modèles de Γ sont aussi des modèles de $\bigvee \Delta$. Cela correspondra (grâce au théorème de complétude) à la prouvabilité du séquent correspondant.

Un résultat intéressant est que si une proposition P est close, alors son interprétation dans $\mathcal{M}$ ne dépend pas de l'assignement choisi, et donc, nous pouvons nous permettre d'oublier l'annotation d'assignement et nous écrirons indifféremment $|P|$ ou $|P|_\sigma$.

La plupart du temps, nous interpréterons les termes clos du langage par eux-même. Plus précisément, M est l'ensemble des termes clos du langage, et la fonction $\widehat{f}$ se retrouve être la fonction suivante :

$$\begin{array}{rcl} \widehat{f} : M^m & \mapsto & M \\ t_1, ..., t_m & \rightarrow & f(t_1, ..., t_m) \end{array}$$

Une telle structure est appelée modèle syntaxique (et se généralise aussi aux modèles non booléens). Lorsque nous aurons affaire à des modèles syntaxiques, alors l'égalité suivante peut être démontrée (par induction sur la structure de P) :

$$|P|_\sigma = |\{\sigma(x_1)/x_1, ...\sigma(x_n)/x_n\}P|$$

Ainsi, nous pouvons totalement oublier la notion d'assignement dans le cas de modèles syntaxiques, et substituer directement par des termes clos.

Extension des modèles pour la preuve de correction

La méthode est correcte pour la notion suivante de modèle booléen, qui étend à la déduction modulo la définition usuelle.

Définition 33. Un modèle booléen est un modèle d'un système de réécriture $\mathcal{R}$ si et seulement si pour toute proposition $P \equiv_\mathcal{R} Q$, $|P| = |Q|$.

$|.|$ est alors notée $|.|_\mathcal{R}$.

Un modèle booléen est essentiellement une fonction totale d'interprétation des propositions $\{0,1\}$ se pliant aux conditions de la définition 34. Dans la suite de ce chapitre, la notion de *modèle* sera toujours celle de modèle booléen pour le système de réécriture $\mathcal{R}$ (qui lui dépend du contexte).

Theorème 11 (Correction)**.** *Soit $\Gamma \wedge \neg\Delta$ une formule. Si $\Gamma \wedge \neg\Delta$ a un modèle, alors le tableau fermé $\odot$ ne peut être dérivé de $\Gamma \wedge \neg\Delta$ avec les règles de la figure 5.1.*

Démonstration. La preuve est standard. On vérifie par induction sur la dérivation du tableau que dans tout tableau dont la racine est $\Gamma \wedge \neg\Delta$, au moins une branche reste vraie dans le modèle. □

5.2 Complétude sémantique : construction d'une valuation partielle

Cette section démontre que la procédure systématique de la section 5.1.2 précédente. On va construire et extraire un modèle booléen d'une branche ouverte.

Nous allons devoir définir une interprétation par modèle de toutes les propositions, y compris celles qui n'apparaissent pas dans le tableau. Nous allons aussi avoir besoin de donner certaines conditions sur le système de réécriture $\mathcal{R}$: la méthode peut en effet être incomplète même en considérant des systèmes confluents et noetheriens ([27, 40]). Deux de ces conditions sont présentées par la suite en section 5.3.

5.2.1 Définitions

Jusque là nous n'avons parlé de sémantique que de manière relativement informelle. Aussi nous allons définir les notions sur lesquelles nous allons travailler.

Commençons par définir la notion de semi-valuation, qui correspond à la notion d'ensemble d'Hintikka. L'intuition derrière les semi-valuations est qu'elles correspondent aux branches ouvertes générées dans la procédure systématique.

Définition 34 (Semi-valuation (valuation partielle))**.** Une interprétation est une fonction $V : \mathcal{P} \mapsto \{0,1\}$ dont le domaine est l'ensemble des propositions du langage $\mathcal{P}$. Elle est appelée *semi*-valuation (resp. valuation partielle) lorsque :

- si $V(\neg P) = 0$ alors (resp. ssi) $V(P) = 1$
- si $V(\neg P) = 1$ alors (resp. ssi) $V(P) = 0$
- si $V(P \vee Q) = 0$ alors (resp. ssi) $V(P) = V(Q) = 0$
- si $V(P \vee Q) = 1$ alors (resp. ssi) $V(P) = 1$ ou $V(Q) = 1$
- si $V(P \wedge Q) = 0$ alors (resp. ssi) $V(P) = 0$ ou $V(Q) = 0$
- si $V(P \wedge Q) = 1$ alors (resp. ssi) $V(P) = V(Q) = 1$
- si $V(P \Rightarrow Q) = 0$ alors (resp. ssi) $V(P) = 1$ et $V(Q) = 0$
- si $V(P \Rightarrow Q) = 1$ alors (resp. ssi) $V(P) = 0$ ou $V(Q) = 1$

- si $V(\forall x P) = 0$ alors (resp. ssi) il existe un terme clos t, $V(P[x := t]) = 0$
- si $V(\forall x P) = 1$ alors (resp. ssi) pour tout terme clos t, $V(P[x := t]) = 1$
- si $V(\exists x P) = 0$ alors (resp. ssi) pour tout terme clos t, $V(P[x := t]) = 0$
- si $V(\exists x P) = 1$ alors (resp. ssi) il existe un terme clos t, $V(P[x := t]) = 1$

À partir de cette définition, on peut introduire par extension les notions de semi-valuation et valuations partielles en déduction modulo.

Définition 35 (Semi-valuations et valuations partielles en déduction modulo)**.** Une semi-valuation (resp. valuation partielle) V est dite compatible avec un système de réécriture $\mathcal{R}$ si et seulement si : quand $P \equiv_{\mathcal{R}} Q$ et $V(P)$ est définie alors (resp. ssi) $V(Q) = V(P)$.

Une semi-valuation (resp. valuation partielle) pour $\mathcal{R}$ sera appelée $\mathcal{R}$-semi-valuation (resp. $\mathcal{R}$-valuation partielle).

Dans la preuve de complétude, l'un des objectifs est de construire effectivement de telles valuations partielles.

5.2.2 Définir une semi-valuation à partir d'une branche ouverte

La présence de métavariables rend la construction d'une semi-valuation plus compliquée. En effet, la signification de ces variables n'est pas encore fixée puisqu'elles attendent une instanciation. Cette instanciation est obtenue lorsqu'une branche est fermée par unification. Un tableau est *ultimement ouvert* si, à aucune étape de sa construction systématique, on ne peut trouver de substitution fermant simultanément toutes ses branches. Ce tableau comporte au moins une branche ouverte, dont on va extraire un modèle de la proposition en entrée.

Nous devons construire une substitution σ qui énumère tous les termes pouvant apparaître dans les γ-formules. Si une γ-formule $\forall x P(x)$ apparaît dans une branche ouverte du tableau complet, alors elle est développée à l'infini, c'est-à-dire qu'il existe un nombre infini de métavariables $X_i, i \in \mathcal{N}$ telles que $P[x := X_i]$ apparaisse.

Dans une branche ouverte les conséquences des γ-formules sont donc dénombrables, et comme les *métavariables sont fraîches lors de leur introduction*, on définit une énumération $\langle \gamma_i, X^i_j \rangle$ des paires associant les γ-formules (indexées by i) et leur ensemble de métavariables (indexées par j). Nous définissons aussi une énumération des termes du langage, incluant aussi les symboles de Skolem.

Alors la substitution σ dont on a besoin est construite par approximations successives :

- σ_0 est la substitution vide.
- $\sigma_{n+1} = \sigma_n + \langle X^{i_n}_{j_n} := t_{j_n} \rangle$

La génération systématique de tableau construit effectivement des approximation successives $\mathcal{T}_0, ..., \mathcal{T}_n, ...$ d'un tableau complet, c'est-à-dire un tableau où toutes les règles applicables ont été appliquées. Or le tableau que nous considérons est par hypothèse ouvert. Autrement dit, pour chaque n, il existe une branche $\mathcal{B} \in \mathcal{T}_n$ telle que $\mathcal{B}$ ne soit pas fermée par la substitution σ_n. Le

tableau étant ouvert, on peut choisir cette branche telle que, quelle que soit la prochaine expansion choisie lors de la procédure systématique, elle reste ouverte : elle ne sera fermée par aucune substitution σ_i.

Soient une γ-formule $\gamma(x)$ et un terme t. Alors pour un certaine étape n de la génération systématique de tableau, $\gamma(t)$ apparaît sur la branche car celle-ci est également ouverte pour σ_n, n étant tel que $t = t_{j_n}$ et $\gamma = \gamma_{i_n}$ (défini par l'énumération).

La branche ultimement ouverte que nous avons générée contient une infinité dénombrable de métavariables, chacune d'entre elles ayant été substituée sans succès lors des tentatives de fermeture suivant son introduction.

Nous définissons alors une inteprétation V des formules sur cette branche.

Définition 36 (V)**.** V est définie en associant à toute proposition de la branche ultimement ouverte (infinie) une valeur de la manière suivante :

- Si P est une proposition sans métavariables (donc sans contraintes) apparaissant sur la branche alors $V(P) = 1$.
- Si $\neg P$ est une proposition sans métavariable (donc sans contraintes) apparaissant sur la branche alors $V(P) = 0$.
- Si une proposition P_c avec métavariables dont les contraintes sont c est présente sur la branche, et si σ satisfait c, il faut poser $V(P\sigma) = 1$.
- Si la proposition $\neg P_c$ est présente sous les contraintes c, et si σ satisfait c, il faut poser $V(P\sigma) = 0$.
- Si les contraintes d'une proposition P_c (ou $\neg P_c$) ne sont pas satisfiables, la proposition n'a pas besoin d'être interprétée : elle ne rentrera pas dans la construction du modèle et donc elle peut être « oubliée ».

L'interprétation V ainsi définie vérifie la la définition. 34 :

Lemme 16. *L'interprétation V de la définition 36 est une semi-valuation.*

Démonstration. Les contraintes d'une formule ne sont pas modifiées lors de l'application α-, β-, γ-, δ-règles. Par conséquent, si par exemple une α-formule de la branche est interprétée par V, ses deux sous-formules immédiates le sont également. De même si β-formule de la branche est interprétée par V, l'une de ses deux sous-formules immédiates l'est également (car elle est sur la branche par la génération systématique). □

L'interprétation V que l'on a donnée est effectivement une semi-valuation bien définie : elle n'associe pas deux valeurs différentes à une même proposition. En effet, si une proposition a deux interprétations, on a $V(P\sigma) \neq V(P'\sigma)$ et $P\sigma = P'\sigma$. Cela veut tout simplement dire que σ permet de fermer la branche sur laquelle elles sont. Or cette branche était supposée ouverte : ce n'est donc pas possible.

5.2.3 Résultats concernant la semi-valuation V

N'oublions pas que nous devons modéliser aussi le système de réécriture $\mathcal{R}$. Donc V doit également être une semi-valuation pour celui-ci. Ce n'est pas encore

le cas car pour une règle $g \to d$, on peut très bien avoir défini $V(g)$ sans pour autant que $V(d)$ l'ait été. Afin d'étendre la semi-valuation à $\mathcal{R}$, nous allons déterminer exactement l'extension dont nous allons avoir besoin. Démontrons donc tout d'abord certaines propriétés de la semi-valuation V définie jusque-là.

La façon dont le développement systématique des tableaux est énoncé dans la section 5.1.2 permet de s'assurer du fait suivant : pour toute règle de réécriture $g \to d$ du système considéré, sous la substitution σ calculée précédemment, si $P\sigma = \Theta g$ est présent, tous ses réduits unitaires $d = Q\sigma$ apparaissent aussi et σ résoud les contraintes associées à Q. En effet nous essayons d'appliquer exhaustivement toutes les règles de réécriture à toutes les propositions atomiques rencontrées.

Lemme 17 (Semi-valuation et réduits atomiques)**.** *Soient P_0 une propositions atomiques et P_1 une proposition quelconque. Supposons que $V(P_0)$ soit définie et que*

$$P_0 \to^1 P_1$$

Alors $V(P_1)$ est définie et $V(P_1) = V(P_0)$.

Démonstration. Montrons que le réduit P_1 est effectivement interprété par la semi-valuation V. Le processus systématique de la section 5.1.2 essaie de réécrire tout littéral avec toutes les règles de réécriture possibles. Par conséquent si Q_c est une proposition apparaissant dans le tableau telle que $Q\sigma = P_0$, alors il existe dans le tableau par construction une proposition $Q'_{c'}$ telle que $P_1 = Q'\sigma$. Et les contraintes associées c' sont satisfaites par σ car σ satisfait c et $P_0 \to P_1$). $V(P_1)$ est donc effectivement définie.

De plus $V(P_1) = V(P_0)$ car P_1 et P_0 apparaissent exactement sous le même nombre de négations et donc par la définition 36 de V ont la même interprétation. 36). □

La semi-valuation que nous avons définie n'est pas incohérente dans son interprétation des propositions $\mathcal{R}$-équivalentes.

Lemme 18. *Soient P et Q deux propositions telles que $P \equiv_{\mathcal{R}} Q$. Si les semi-valuations $V(P)$ et $V(Q)$ sont définies alors*

$$V(P) = V(Q)$$

Démonstration. Comme $\mathcal{R}$ est une relation confluente, il existe une proposition ϕ telle que $P \to^n \phi \ ^m \leftarrow Q$.

Nous allons démontrer ce lemme par induction sur la paire $\langle n+m, \min(\#P, \#Q)\rangle$, où $\#P$ est le nombre de connecteurs logiques dans P.

Si $n = m = 0$, $P = \phi = Q$ et le résultat est immédiat.

Supposons que l'une des deux propositions, P ou Q soit atomique. Faisons l'hypothèse que ce soit P. Alors

$$P \to^p P' \to^1 \phi' \to^q \phi$$

où P' est la dernière proposition atomique de la chaîne de réduction. Alors le lemme 17 permet de conclure que $V(\phi')$ est définie. On réapplique dès lors l'"hypothèse d'induction sur les propositions $\mathcal{R}$-équivalentes ϕ' et Q pour conclure.

Si P et Q sont toutes deux des propositions composées (non atomiques), utilisons le fait que V est une semi-valuation et appliquons l'hypothèse d'induction.

Par exemple, si $P = \forall x\ S$, alors la confluence de $\mathcal{R}$ implique que $Q = \forall x\ S'$ et que $R = \forall x\ S$"'. Montrons par l'absurde que V ne peut en aucun cas interpréter P et Q de manière différente. Supposons que $V(P) = 1$ et que $V(Q) = 0$. Dans ce cas, il existe un terme t tel que $V(S'[x := t]) = 0$. Or d'après la définition 34 d'une semi-valuation, $V(S[x := t]) = 1$. L'application de l'hypothèse d'induction à :

$$S[x := t] \rightarrow^n S"'[x := t]^m \leftarrow S'[x := t]$$

engendre une contradiction : les interprétations de P et Q par semi-valuation V ne peuvent être différentes ! □

5.2.4 Extension de V en $\mathcal{R}$-semi-valuation

La semi-valuation V définie jusque-là n'est encore pas suffisante pour la déduction modulo. Le lemme 18 n'est pas suffisant car, pour être en accord avec la définition 35 il faut s'assurer que pour toute instance d'une règle de réécriture de $\mathcal{R}$ $P \rightarrow Q$, $V(Q)$ est définie lorsque $V(P)$ l'est. Par exemple, si $\mathcal{R}$ est la règle de réécriture $\phi \rightarrow \Psi$ et que le tableau comporte la seule proposition Ψ, on n'a interprété que Ψ alors qu'on doit interpréter aussi ϕ.

Nous allons donc étendre la semi-valuation V en une $\mathcal{R}$-semi-valuation $\mathcal{V}$. Dans ce but, on utilise une énumération P_n des propositions du langage de telle façon que tout proposition Q soit vue une infinité de fois (pour s'assurer que l'on peut effectivement l'interpréter au moment où l'on aura suffisamment d'informations sémantiques). C'est en substance une énumération $\langle Q, n\rangle$ où Q est une proposition et n un entier qui décrit $\mathcal{N}$.

Définition 37 (Construction de $\mathcal{V}$)**.** Soit $V_o = V$, la semi-valuation déjà constuite en section 5.2.2 précédente. Alors on construit une suite de semi-valuations où $V_{n+1} = V_n$ puis $V_{n+!}$ estétendue par rapport à V_n en interprétant P_n lorsque $V_n(P_n)$ n'est pas définie de la manière suivante :

1. On pose $V_{n+1}(P_n) = V_n(Q)$ s'il existe une proposition Q dont la semi-valuation $V_n(Q)$ est définie telle que $P_n \equiv_{\mathcal{R}} Q$.
2. Si P_n est une proposition composée, alors il faut regarder l'interprétation V_n de ses sous-formules immédiates. Si l'on a suffisamment d'informations, alors on pose $V_{n+1}(P_n)$ en fonction de celles-ci.
   Par exemple, si $P = Q \wedge R$ et que $V_n(Q) = V_n(R) = 1$, alors on posera $V_{n+1}(P) = 1$. Ou encore, si $P_n = \forall x\ R$, et que pour tout terme t, $V_n(R[x := t]) = 1$, alors on posera $V_{n+1}(P_n) = 1$.
   Par opposition, s'il existe au moins un terme t tel que $V_n(R[x := t]) = 0$, alors on posera $V_{n+1}(P_n) = 0$, et ce même s'il existe par ailleurs un terme t' tel que $V_{n+1}(P_n)$ ne soit pas encore définie.

On définit $\mathcal{V}$ comme la limite des semi-valuations V_n. Ainsi, pour toute proposition P, s'il existe un entier n tel que $V_n(P)$, on aura $\mathcal{V}(P) = V_n(P_n)$.

La construction de $\mathcal{V}$ est trs̀ importante dans les démonstrations qui suivent, notamment en ce qui concerne la façon d'énumérer les propositions pour construire les semi-valuations successives. La définition 37 ne donne pas d'informations particulières concernant $\mathcal{V}$: par exemple, une proposition composée peut très bien avoir été définie dans $\mathcal{V}$ par le premier cas. Nous allons donc avoir besoin du lemme suivant.

Lemme 19. *Soit n un entier donné. Notons qu'ici, nous adoptons la convention suivante que le prédécesseur de* 0 *est* 0 *($n-1 = n$ i $n = 0$). Soit P une proposition atomique.*

Alors il existe une proposition Q telle que :

1. $P \equiv_{\mathcal{R}} Q$
2. *Soit $V(Q) = V_0(Q)$ est définie, soit Q est une proposition composée et dans ce cas V_{n-1} est définie.*

La valeur de la semi-valuation obtenue est dans les deux cas la même.

Démonstration. Cette preuve se démontre par induction sur l'entier n.

Si $n = 0$, alors $V_0 = V$ par hypothèse et la conclusion est immédiate.

Si $V_{n-1}(P)$ est définie, alors on applique l'hypothèse d'induction.

Sinon, comme P est une proposition atomique, il existe P' tel que $P \equiv_{\mathcal{R}} P'$ et $V_n(P) = V_{n-1}(P)$ par construction de la suite de semi-valuation. Si P' est une proposition composée alors il suffit de prendre $Q = P'$. Sinon, en appliquant l'hypothèse d'induction sur $V_{n-1}(P')$, on trouve une proposition Q telle que $Q \equiv_{\mathcal{R}} P' \equiv_{\mathcal{R}} P$. □

Corollaire 2. *Soit P une proposition atomique et n un entier tels que $V_n(P)$ soit définie et qu'il existe une formule Q telle que $Q \equiv_{\mathcal{R}} P$.*

Alors il existe une proposition composée R telle que $P \equiv_{\mathcal{R}} Q$ et $V_{n-1}(R)$ soit définie et égale à $V_n(P)$.

Démonstration. Appliquons le lemme 19 précédent.

Si on obtient une proposition atomique S telle que $S \equiv_{\mathcal{R}} P$ et $V(s)$ soit définie, la confluence du système de réécriture $\mathcal{R}$ permet d'obtenir la chaîne de réduction suivante

$$S \to^* S_q \to^1 \to^* Q\downarrow$$

telle que R soit la premiìre proposition non-atomique ($Q\downarrow$ est aussi composée). L'application du lemme 17 à cette chaîne permet de montrer que les propositions S_q et R sont définies pour V. Par le lemme 18, elles sont même égales $V(S_q) = V(R) = V(S)$. Par le lemme 19 on a de plus que comme R est composée par hypothèse alors $V(R) = V_{n-1}(R)$. Toute valuation V_{p+1} est une extension conservatrice de V_P. On a en résumé :

$$V(P) = V(S) = V(R) = V_{n-1}(R)$$

Si S n'est pas atomique, alors elle est déjà composée et toutes les interprétations sont les mêmes. □

L'objectif d'obtenir une $\mathcal{R}$-semi-valuation est en vue. Montrons maintenant que si deux propositions sont $\mathcal{R}$-équivalentes, alors si l'une d'entre elles est interprétée par $\mathcal{V}$ alors la semi-valuation $\mathcal{V}$ est la même pour les deux.

Lemme 20. *Soient P et Q deux propositions telles $P \equiv_{\mathcal{R}} Q$. Alors si $\mathcal{V}(P)$ est définie,*

$$\mathcal{V}(P) = \mathcal{V}(Q)$$

Démonstration. Soit m le plus petit entier tel que $V_n(P)$ ou $V_n(Q)$ existe. Supposons que cela soit $V_n(Q)$.

Si $m = 0$ et $V_0(P)$ est définie, le résultat provient directement de l'application du lemme 18.

Sinon, souvenons-nous que l'énumération de la définition 37 est répétée tant que nécessaire. Par conséquent, il existe une étape n où la proposition P est effectivement prise en compte. Alors $V_n(P)$ est définie par la première règle puisque P remplit les conditions de cette règle. Donc

$$\mathcal{V}(P) = V_n(P) = V_m(Q) = \mathcal{V}(Q).$$

□

Montrons maintenant que $\mathcal{V}$ est effectivement une semi-valuation.

Lemme 21. *$\mathcal{V}$ est une semi-valuation.*

Démonstration. Démontrons par induction que si une proposition (non-atomique) P est définie à l'étape n, alors ses sous-formules immédiates sont interprétées dans $\mathcal{V}$ de manière suffisante pour que $\mathcal{V}$ soit une semi-valuation.

Si $n = 0$, alors c'est vrai par construction car $V = V_0$ est une semi-valuation.

Si P est définie à l'étape n par la seconde règle de construction, alors c'est également vrai (par construction donc), car cette règle correspond à la définition d'une semi-valuation.

Si P est définie à l'étape n par la première règle, nommons Q la proposition telle $P \equiv_{\mathcal{R}} Q$ et $V_n(P)$ est définie comme $V_{n-1}(Q)$. Le corollaire 2 permet de choisir Q non atomique. Par confluence, P et Q ont les mêmes connecteurs logiques. Par hypothèse d'induction, il existe suffisamment de sous-formules immédiates de Q interprétées par $\mathcal{V}$. Nommons $R_1, ..., R_k$ ces sous-formules et $R'_1, ..., R'_k$ leurs contreparties dans P. Pour tout i, $R_i \equiv_{\mathcal{R}} R'_i$. En appliquant le lemme 20 précédent, on obtient que $\mathcal{V}(R_i) = \mathcal{V}(R'_i)$. Par conséquent suffisamment de sous-formules de P sont interprétées par $\mathcal{V}$ également. □

Il est immédiat de prouver ensuit que $\mathcal{V}$ est une $\mathcal{R}$-semi-valuation.

Lemme 22. *$\mathcal{V}$ est une $\mathcal{R}$-semi-valuation.*

Démonstration. Le résultat est la conséquence directe des lemmes 20 et 21. □

5.2.5 Extension de la semi-valuation en valuation partielle

$\mathcal{V}$ n'est pas une valuation partielle pour $\mathcal{R}$. En effet, définissons la proposition $P(x) = Q(x) \wedge R(x)$ et supposons que la $V(\forall x\ P(x))$ ne soit pas définie et que malgré tout, pour tout terme t, $V(Q(t)) = V(R(t)) = 1$. Il serait par conséquent souhaitable que $\mathcal{V}(\forall x\ P(x)) = 1$ car, par construction, $\mathcal{V}(Q(t) \wedge R(t)) = 1$ pour tout terme t.

Le problème est que l'on ne peut pas trouver l'entier n tel que $V_n(Q(t) \wedge R(t)) = 1$ pour tout t car il existe un nombre (potentiellement) infini de termes t : on ne finira jamais de les énumérer. Il est par conséquent impossible de se retrouver avec le résultat : $V_{n+1}(\forall x\ P(x))$ dans ce cas. $\mathcal{V}$ doit être étendue comme à la section précédente 5.2.4 mais aucune étape finie de ce processus n'est suffisante.

Définissons alors $\tilde{V}$ comme le plus petit point fixe de l'opération d'extensions de la semi-valuation (définition 37) : ceci est possible car on peut définir un ordre partiel sur les semi-valuation $V \prec V'$ si V interprète moins de formules que V', et V' est conservatrice par rapport à V (les formules interprétées par V le sont aussi par V').

Alors le lemme suivant est immédiat.

Lemme 23. *$\tilde{V}$ est une $\mathcal{R}$-valuation partielle, en accord avec la semi-valuation V sous-jacente.*

Démonstration. $\tilde{V}$ est une $\mathcal{R}$-semi-valuation par construction. C'est aussi une valuation partielle : pour toute proposition Q dont nous connaissons suffisamment d'informations, alors l'opération d'extension définit également l'interprétation de Q. Comme $\tilde{V}$ est sa propre extension (c'est un point fixe), $\tilde{V}(Q)$ est définie. □

5.3 Complétude sémantique : définition de $\mathcal{R}$-modèles

Dans la section précédente, on a réussi à obtenir une valuation partielle pour $\mathcal{R}$. Dans le cas de la logique classique du premier ordre, la preuve de complétude est pratiquement terminée dès que l'on a établi une semi-valuation : elle est alors étendue en un modèle en donnant une valeur de vérité au hasard aux atomes non interprétés pas V. En déduction modulo, ce n'est plus possible car le modèle construit doit être aussi un modèle de $\mathcal{R}$ (appelé $\mathcal{R}$-modèle).

Nous cherchons à prouver le théorème suivant :

Theorème 12 (Complétude)**.** *Soit $\mathcal{R}$ un système de réécriture confluent et noetherien et Γ un ensemble de propositions. Si la construction systématique de tableau de la section 5.1.2 est appliquée à Γ, alors Γ a un modèle.*

Ce théorème n'est vrai que sous certaines conditions sur le système $\mathcal{R}$ que nous allons détailler ci-dessous.

5.3.1 Une condition d'ordre

On se place dans cette section sous l'hypothèse que nous avons un système $\mathcal{R}$ pour lequel existe une condition d'ordre.

Cette condition est introduite dans [51] et utilisée également dans [39, 40] dans les preuves de complétude (sémantiques) concernant ENAR (voir section 3.4) et le calcul des séquents sans coupure.

Définition 38 (Ordre)**.** On considère des systèmes de réécriture confluents sur lesquels on peut définir un ordre bien fondé $\prec$ comme suit :
- si $P \rightarrow_{\mathcal{R}} Q$ alors $P \prec Q$,
- si P est une sous-formule de Q, alors $P \prec Q$.

Le domaine de ce modèle est l'ensemble des termes clos apparaissant dans la valuation partielle $\tilde{V}$ construite dans la section 5.2.5. Alors nous allons construire l'interprétation par induction sur l'ordre $\prec$ que l'on vient de donner.

Définition 39 (Interprétation pour une condition d'ordre)**.** L'interprétation $|.|_{\mathcal{R}}$ des propositions est définie inductivement en suivant l'ordre $\prec$ de la définition 38 comme suit :
- si P est une proposition atomique normale (par rapport à $\mathcal{R}$), on pose $|A|_{\mathcal{R}} = \tilde{V}(A)$ si $\tilde{V}(A)$ est définie et on donnera à une valeur arbitraire à $|A|_{\mathcal{R}}$ (mettons 1).
- Si P est un atome non normal, alors on pose $|A|_{\mathcal{R}} = |A \downarrow|_{\mathcal{R}}$.
- si P est une proposition composée alors $|P|_{\mathcal{R}}$ est définie inductivement (et comme cela est fait d'habitude) en fonction des interprétations de ses sous-formules immédiates.

Cette définition est bien fondée. Il est démontré dans [39] les résultats suivants (que l'on reprend), dans l'ordre dans lequel ils sont énoncés ici :

1. $P \mapsto |P|_{\mathcal{R}}$ définit une interprétation par modèle.
2. $|P|_{\mathcal{R}} = |P \downarrow|_{\mathcal{R}}$.
3. $P \mapsto |P|_{\mathcal{R}}$ définit un modèle.
4. $P \mapsto |P|_{\mathcal{R}}$ est une extension conservatrice de la valuation partielle $\tilde{V}$.

Maintenant, démontrons le théorème 12 pour cette condition sur $\mathcal{R}$. Dans le modèle que nous venons de définir $|P|_{\mathcal{R}} = 1$ pour toute proposition $P \in \Gamma$. La méthode des tableaux modulo de ce chapitre est ainsi complète pour tout système de réécriture vérifiant la condition d'ordre énoncé en définition 38.

5.3.2 Une condition de positivité

Supposons maintenant que le système de réécriture $\mathcal{R}$, en plus de la confluence et de la terminaison, vérifie une propriété dite de *positivité*.

Définition 40 (Occurrences positives)**.** Soit P une formule, et l'occurrence d'un atome A. Nous définissons la positivité et la négativité de cette occurrence de la manière suivante :

- si P est l'atome A, alors elle est positive,
- si $P = Q \vee R$, alors les occurrences positives de A dans Q et R sont positives dans P,
- si $P = Q \wedge R$, alors les occurrences positives de A dans Q et R sont positives dans P,
- si $P = \forall x Q$, les occurrences positives de A dans Q ont positives dans P,
- si $P = \exists x Q$, les occurrences positives de A dans Q sont positives dans P,
- si $P = Q \Rightarrow R$, les occurrences positives de A dans R et négatives de A dans Q sont positives dans P,
- si $P = \neg Q$, les occurrences négatives de A dans Q sont positives dans P,
- dans tous les cas contraires, les occurrences de A dans P sont négatives.

On se place ici dans le cas où toute règle de réécriture propositionnelle $g \to d \in \mathcal{R}$ est telle que tous les atomes de la partie droite d apparaissent de manière positive, conformément à la définition 40. Dans ce cas d sera dénommée « formule positive ». Par exemple, il est clair que la règle suivante respecte la condition de positivité :

$$P(0) \to \forall x\ P(x)$$

Comme dans la condition d'ordre, le domaine est l'ensemble des termes clos. Mais l'interprétation est définie différemment.

Définition 41 (Interprétation pour la condition de positivité)**.** Soit $\mathcal{R}$ un système de réécriture positif. L'interprétation des formules est définie comme suit :

- Si P est une proposition atomique,
  - si $\tilde{V}(P)$ est définie, alors $|P|_{\mathcal{R}} = \tilde{V}(P)$,
  - sinon $|P|_{\mathcal{R}} = 1$.
- Si P est une proposition composée, $|P|_{\mathcal{R}}$ est définie inductivement (et de manière usuelle) en fonction de l'interprétation de ses sous-formules, en accord, avec la définition 34.

Pour l'instant l'interprétation des propositions atomiques n'est pas dépendante du système de réécriture $\mathcal{R}$ correspondant.

Lemme 24. *$P \mapsto |P|_{\mathcal{R}}$ définit une interprétation par modèle.*
Soit P une proposition. Si $\tilde{V}(P)$ est définie, alors $|P|_{\mathcal{R}} = \tilde{V}(P)$.

Démonstration. La première partie de ce lemme est tout simplement la conséquence de la construction de l'interprétation. La seconde partie est démontrée par induction sur la structure de P, en utilisant le fait que $\tilde{V}$ est une valuation partielle. □

Maintenant, si le tableau dont la racine est Γ ne peut être fermé, alors Γ a un modèle booléen. Mais l'interprétation choisie est-elle, comme elle doit l'être, un modèle de $\mathcal{R}$?

Si $P \equiv_{\mathcal{R}} Q$ et $\tilde{V}(P)$ est définie, alors en utilisant le fait que $\tilde{V}$ est une valuation partielle et le lemme 24, on peut établir que :

$$|P|_{\mathcal{R}} = \tilde{V}(P) = \tilde{V}(Q) = |Q|_{\mathcal{R}}).$$

Mais le problème apparaît pour des propositions qui ne sont pas définies dans la valuation partielle $\tilde{V}$. Nous n'avons, *a priori*, aucune interprétation.

Restreignons-nous tout d'abord à un sous-cas du problème que l'on considère. Prenons seulement $P \to Q$ au lieu de sa fermeture réflexive, symétrique et transitive $\equiv_{\mathcal{R}}$. Supposer P atomique n'est pas problématique car la réécriture propositionnelle s'opère uniquement sur des membres gauches atomiques.

Maintenant, rappelons que $\mathcal{R}$ est un système de réécriture positif. Donc Q est une proposition positive (dénotée Q^+). Le point crucial ici était de définir l'interprétation $|P|_{\mathcal{R}}$ des atomes non-interpétés par $\tilde{V}$ comme étant *la même pour tous les atomes* (on aurait pu choisir 0 aussi). Q^+ est positive : puisque tous ses atomes seront interprétés par 1, on voudrait que Q^+ soit aussi interprétée par 1.

Il faut cependant rester attentif : $\tilde{V}(Q^+)$ pourrait très bien avoir été définie à 0. Or, $\tilde{V}$ est une valuation partielle, ce qui veut dire que si $\tilde{V}(Q)$ est définie, alors $\tilde{V}(P)$ devrait l'être aussi. Cependant, il peut encore y avoir une sous-formule de Q interprétée par $\tilde{V}$. Généralisons le résultat afin de le démontrer :

Lemme 25. *Soit P^+ une proposition négative. Soit Q^- une proposition négative (donc $\neg Q^-$ est positive)). Supposons encore qu'aucun de ces deux propositions ne soient interprétées par $\tilde{V}$.*

Alors $|P^+|_{\mathcal{R}} = 1$ et $|Q^-|_{\mathcal{R}} = 0$.

Démonstration. La démonstration se fait par induction sur la structure des propositions considérées.

Le cas de base où P^+ est atomique est immédiat de par la définition de $P \mapsto |P|_{\mathcal{R}}$. Il n'y a pas de cas atomique pour Q^- puisque tout atome est positif.

Nous ne détaillerons ici que le cas des formules universelles. Soit $P^+ = \forall x\ R^+$ et t un terme donné. $\tilde{V}$ est une valuation partielle donc on ne peut avoir $\tilde{V}(R^+[x := t]) = 0$, sinon $\tilde{V}(P^+)$ aurait été définie (à 0). Donc $|R^+[x := t]|_{\mathcal{R}} = 1$ soit par le lemme 24 si $\tilde{V}(R^+[x := t]) = 1$ soit par hypothèse d'induction si $\tilde{V}(R^+[x := t])$ n'est pas définie. Comme cette démarche peut être répétée pour n'importe quel terme t, on en conclut que $|P|_{\mathcal{R}} = 1$.

Si $Q^- = \forall x\ R^-$. On ne peut avoir $\tilde{V}(R^-[x := t]) = 1$ pour tout terme t clos car $\tilde{V}$ est valuation partielle. Par hypothèse, on ne sait pas trouver de terme t tel que $\tilde{V}(R^-[x := t]) = 0$. Alors il existe un terme t_0 tel que la valuation partielle $\tilde{V}(R^-[x := t_0])$ ne soit pas définie. On conclut en utilisant l'hypothèse d'induction que $|R^-[x := t]|_{\mathcal{R}} = 0$. Donc $|Q^-|_{\mathcal{R}} = 0$. □

Ensuite, il est plus aisé de démontrer le lemme suivant :

Lemme 26. *L'interprétation $P \mapsto |P|_{\mathcal{R}}$ définit un modèle pour $\mathcal{R}$*

Démonstration. Soit $P \to^1 Q$. Soit $\tilde{V}(P)$ est définie et nous pouvons conclure par le lemme 24. Soit elle ne l'est pas et par le lemme 25 on a $|P|_{\mathcal{R}} = |Q|_{\mathcal{R}} = 1$. On étend ce résultat par induction structurelle aux propositions composées. Puis on l'étend Ã la relation $\equiv_{\mathcal{R}}$ par induction sur le nombre détapes de réécriture. □

La méthode des tableaux modulo est donc également complète pour les systèmes de réécriture positifs.

5.4 Exemple

Nous allons illustrer sur un exemple le fonctionnement de la génération systématique de la section 5.1.2.

Prouvons $2 = 2 \Rightarrow \exists x\ (x+x = 2)$ où $=$ est un prédicat propriété particulière. Nous omettrons les formules qui n'ont pas d'intérêt dans la découverte de la solution bien que nous listions quand même toutes les règles appliquées .

Soit $\mathcal{R}$ le fragment suivant de l'arithmétique Peano :

$$x + 0 \quad \rightarrow^* \quad 0 \tag{5.1}$$

$$x + s(y) \quad \rightarrow^* \quad s(x + y) \tag{5.2}$$

Alors la preuve est la suivante : les deux formules sous la quatrième barre d'inférence sont produites à partir de la formule au-dessus en utilisant respectivement `rw`(5.1) et `rw`(5.2).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{2 = 2, \forall x(x + x \neq 2)}{X + X \neq 2}\,\gamma}{s(X_1 + Y_1) \neq 2 \cdot \mathcal{C} := \{X \stackrel{?}{=} X_1, X \stackrel{?}{=} s(Y_1)\}}\,\texttt{rw}(5.2)}{X' + X' \neq 2}\,\gamma}{s(X_2) \neq 2 \cdot \mathcal{C}' := \mathcal{C} \cup \{X_1 \stackrel{?}{=} X_2, \mathbf{Y_1} \stackrel{?}{=} \mathbf{0}\}}\,\texttt{rw}(5.1),\texttt{rw}(5.2)}{s(s(X_3 + Y_3)) \neq 2 \cdot \mathcal{C}'' := \mathcal{C} \cup \{X_1 \stackrel{?}{=} X_3, Y_1 \stackrel{?}{=} s(Y_3)\}}\,\texttt{closure}(\mathcal{C}')$$
$$\odot \cdot \{X \stackrel{?}{=} 1\}$$

Notons qu'au niveau de la première réécriture propositionnelle, la règle 5.1 n'est pas appliquée car 0 est trivialement non unifiable avec $2 = s(s(0))$. Aussi, après cette étape de réécriture, X_1 aurait pu être remplacée par $s(Y_1)$ pour obtenir une solution plus rapide.

5.5 Conclusion

Nous avons démontré la complétude des tableaux pour la déduction modulo par une extension appropriée des modèles booléens. La nouvelle formulation de TaMed uniquement par règles d'inférence est plus proche de celle qui est habituellement employée : la phase de normalisation a disparu.

La sémantique utilisée est celle que [39] utilise pour montrer l'élimination des coupures sémantique dans le calcul des séquents modulo pour les classes de système de réécriture que l'on considère (plus un mélange de ces deux classes dont nous parlons dans le chapitre suivant. On a de cette faç un rapport direct entre tableaux et élimination des coupures dans le cadre classique modulo.

Au niveau pratique, nous aurions pu considérer une règle de skolemisation telle que δ^+ sans changer les arguments de la preuve. C'est un avantage par rapport aux preuves syntaxiques.

Les autres points positifs concernent, d'une part, l'obligation dans laquelle nous avons été de donner une méthode d'utilisation des règles de tableaux pour la preuve de complétude et d'autre part, le fait que les méthodes sémantique dégagent des classes de systèmes pour lesquelles la complétude est valable.

Chapitre 6

Tableaux intuitionnistes modulo

Dans ce chapitre, nous allons nous intéresser (à nouveau) à la logique intuitionniste (modulo). Nous allons utiliser la méthode des tableaux comme point d'entrée à l'étude notamment de l'élimination des coupures dans le cadre intuitionniste. La relation entre les preuves sans coupures et les tableaux fait partie du folklore dans le cas de la logique classique (et donc l'élimination des coupures). Cette relation ne semble en revanche pas avoir été définie dans le monde intuitionniste. C'est cette relation, pour la déduction modulo en particulier. que nous nous proposons d'étudier dans la suite de ce chapitre.

Les résultats de ce chapitre sont le fruit d'un travail en collaboration avec Olivier Hermant [12].

6.1 Introduction

La logique intuitionniste est particulièrement intéressante à étudier : elle est constructiviste, possède les propriétés de la disjonction et du témoin.

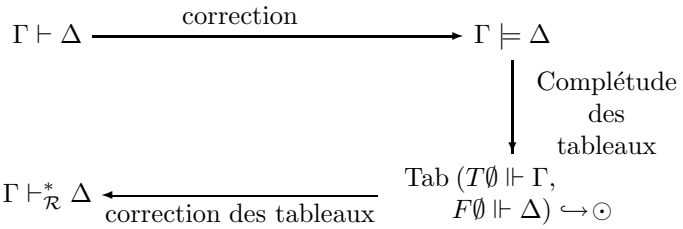


FIG. 6.1 – Élimination des coupures sémantique constructive par la méthode des tableaux

Cependant, la littérature sur les procédures de preuve automatique est moins fournie que pour la logique classique.

Nous porterons plus particulièrement attention au théorème d'élimination des coupures pour la logique intuitionniste modulo. Pour la déduction modulo, l'élimination des coupures est un problème difficile, car cette propriété n'est pas toujours valable, même dans le cas de systèmes confluents dont la terminaison peut être établie.

Nous allons montrer le lien profond qui existe également dans le cas intuitionniste entre les méthodes de tableaux et l'élimination des coupures.

Pour ce faire, nous suivrons le plan ci-dessous (résumé en figure 6.1) :

- En premier lieu, nous formulerons une méthode de recherche de preuve pour la logique intuitionniste modulo. Cette méthode sera bien sûr basée sur la méthode des tableaux.
- Puis, on prouvera la complétude de celle-ci pour certaines classes de systèmes de réécriture. La preuve est faite à partir d'arguments sémantiques à l'aide de structures de Kripke (modifiées).
- Enfin, on montrera que cette méthode de tableaux intuitionnistes modulo est correcte *par rapport au calcul des séquents intuitionnistes modulo sans coupure*. Dans le cas intuitionniste, les preuves de correction des tableaux sont systématiquement sémantiques. Le fait de se référer directement au calcul des séquents pour la preuve de correction n'est pas anodin : il s'agit de récupérer de cette manière un théorème d'élimination des coupures constructif et aussi de mettre en lumière la relation entre tableaux intuitionnistes et élimination des coupures. Ce dernier résultat, qui est bien connu dans le cas classique, semble n'avoir été jamais établi même dans les cas où on ne considère pas de règles de réécriture.

6.2 Calcul des séquents intuitionniste modulo : $\mathsf{LJ}_{\mathsf{mod}}$

Nous rappelons en figure 6.2 les règles du calcul des séquents intuitionniste modulo telles qu'on peut les trouver dans [27, 40]. Comme dans le cas classique, le calcul intuitionniste a été étendu pour manipuler des formules modulo un système de réécriture $\mathcal{R}$ sur les termes et les propositions. Notons que dans ce chapitre, *on ne considérera que des systèmes confluents et noetheriens*. On notera $\Gamma \vdash^*_{\mathcal{R}} P$ une dérivation sans coupure du séquent $\Gamma \vdash_{\mathcal{R}} P$ dans ce système.

6.2.1 Sémantique

Pour la suite, et notamment pour la démonstration de complétude de la méthode des tableaux, nous avons besoin d'établir une sémantique.

Nous utiliserons des structures de Kripke modifiées, où la négation est exprimée de manière positive. Cette modification est importante vis-à-vis de la notion de constructivité.

$$\frac{}{P \vdash_{\mathcal{R}} Q}\ \text{axiom if } P \equiv_{\mathcal{R}} Q \qquad \frac{\Gamma \vdash_{\mathcal{R}} P \qquad \Gamma, Q \vdash_{\mathcal{R}} S}{\Gamma, R \vdash_{\mathcal{R}} S}\ \Rightarrow\text{-l if } R \equiv_{\mathcal{R}} (P \Rightarrow Q)$$

$$\frac{\Gamma, P \vdash_{\mathcal{R}} Q}{\Gamma \vdash_{\mathcal{R}} R}\ \Rightarrow\text{-r if } R \equiv_{\mathcal{R}} (P \Rightarrow Q) \qquad \frac{\Gamma, P \vdash_{\mathcal{R}} S \qquad \Gamma \vdash_{\mathcal{R}} Q}{\Gamma \vdash_{\mathcal{R}} S}\ \text{cut if } P \equiv_{\mathcal{R}} Q$$

FIG. 6.2 – Règles du calcul des séquents intuitionniste modulo

Définition 42 (Structures de Kripke modifiées)**.** Une structure de Kripke modifiée $\mathcal{K}$ est un quadruplet $\langle K, \leq, D, \Vdash \rangle$ tel que :

- K est un ensemble non-vide (les mondes), partiellement ordonné par $\leq$.
- D (le domaine) est une fonction monotone sur K : si $\alpha \leq \beta$ alors $D_\alpha \subseteq D_\beta$
- $\widehat{\cdot}$ est une fonction d'interprétation (partielle) associant une fonction à un symbole de fonction f et un monde de la façon suivante : $\hat{f} : K \mapsto D_\alpha^m \mapsto D_\alpha$, telle que si $\alpha \leq \beta$, alors $\hat{f}(\alpha)(a_1, ..., a_n) = \hat{f}(\beta)(a_1, ..., a_n)$ sur D_α. (f dans β étend f dans α).
- Les symboles de prédicats sont interprétés syntaxiquement (c'est suffisant pour le contenu de ce chapitre).
- L'interprétation $|t|_\sigma^\alpha$ d'un term t dans D_α, sous une substitution σ est définie par induction sur t de manière usuelle.
- $\Vdash$ est une relation entre les mondes α et les propositions, modulo une substitution σ associant les variables à des éléments de D_α. Cette relation satisfait aux conditions suivantes :

  1. Elle est monotone sur les atomes. Pour toute proposition atomique $P(x_1, ..., x_n)$, tout monde $\beta \geq \alpha$ et termes $t_1, ..., t_n$, si
  $$\alpha \Vdash_\sigma A(t_1, ..., t_n)$$
  alors
  $$\beta \Vdash_\sigma A(t_1, ..., t_n)$$
  .
  2. $\alpha \Vdash_\sigma A \vee B$ ssi $\alpha \Vdash_\sigma A$ or $\alpha \Vdash_\sigma B$.
  3. $\alpha \Vdash_\sigma A \wedge B$ ssi $\alpha \Vdash_\sigma A$ et $\alpha \Vdash_\sigma B$.
  4. $\alpha \Vdash_\sigma A \Rightarrow B$ ssi pour tout $\beta \geq \alpha$, si $\beta \Vdash_\sigma A$ alors $\beta \Vdash_\sigma B$.
  5. $\alpha \Vdash_\sigma \neg A$ ssi pour tout $\beta \geq \alpha$, si $\beta \Vdash_\sigma A$ alors $\beta \Vdash_\sigma \bot$ (dénoté $\beta \nVdash_\sigma A$).
  6. $\alpha \Vdash_\sigma \exists x A$ ssi il existe un élément $a \in D(\alpha)$ tel que $\alpha \Vdash_{\sigma + \langle a/x \rangle} A$.
  7. $\alpha \Vdash_\sigma \forall x A$ ssi pour tout $\beta \geq \alpha$, pour tout élément $a \in D(\beta)$, $\beta \Vdash_{\sigma + \langle a/x \rangle} A$.
  8. La propriété d'explosion : si $\beta \Vdash_\sigma \bot$ alors pour tout $\alpha \in K$, toute proposition P, toute substitution θ, on a $\alpha \Vdash P$.

Une structure de Kripke modifiée $\mathcal{K}$ (que l'on appellera par la suite abusivement structure de Kripke) telle que $\alpha \Vdash \bot$ est appelée *impropre*.

Le traitement positif de la négation est essentiel pour la constructivité. Comme nous n'avons pas à considérer les structures de Kripke propres, nous éviterons l'utilisation du lemme de König pour identifier une branche infinie. Cette définition des structures de Kripke est inspirée de celles de Veldman ([59]) et Friedman ([57]). Elle est cependant peut-être plus proche de celle de Krivine ([43]), car seule une unique structure de Kripke est ajoutée à la définition usuelle (la structure impropre).

Les structures de Kripke de la définition 42 sont également contraintes de valider les règles de réécriture : ainsi pour tout monde $p \in K$, toutes formules $P \equiv_{\mathcal{R}} Q, p \Vdash P$ si et seulement si $p \Vdash Q$. C'est équivalent (voir [39]) à considérer deux formules $P \to Q$ où P est atomique et Q son réduit immédiat (une seule étape de réduction) et forcer ensuite que pour tout monde $p \in K, p \Vdash P$ ssi $p \Vdash Q$.

6.3 Tableaux intuitionnistes modulo du premier ordre

Les tableaux pour la logique intuitionniste du premier ordre modulo sont basés sur ceux de Nerode et Shore présentés en section 1.4. La différence majeure provient évidemment de la prise en compte d'un système de réécriture additionnel. La méthode présentée ici est volontairement très simple car le but principal est d'établir le théorème d'élimination des coupures. L'une des caractéristiques des tableaux est qu'ils sont l'expression d'une tentative de trouver un contre-modèle aux formules à leur racine : les structures de Kripke qui sont ici attachées aux formules du tableau nous forcent à construire des mondes dans lesquels on peut évaluer la vérité des formules.

Nous allons nous concentrer tout d'abord sur le côté sémantique des tableaux : toute branche ouverte représente un contre-modèle possible et d'une branche fermée découle un échec par contradiction dans la recherche de ce contre-modèle. La génération systématique de tableau de la définition 43 correspond à la recherche systématique d'un modèle.

Il y a peu de commentaires à faire sur les tableaux intuitionnistes modulo de la figure 6.3 car les règles d'expansion sont très proches de celles de la section 1.4.2. L'intégration de la réécriture est faite de la manière suivante :

- Toute les formules en entrée sont pré-normalisées par rapport au système de réécriture $\mathcal{R}$. On notera comme auparavant $P \downarrow$ la forme normalisée de la proposition P par rapport à $\mathcal{R}$ (confluent et noetherien, rappelons-le).
- Il faut faire attention car seules les formules normalisées peuvent apparaître dans le tableau. Seules les règles avec $Fp \Vdash \exists x P(x)$ et $Tp \Vdash \forall x P(x)$ comme prémisses sont susceptibles d'ajouter des formules non normalisées dans le processus : c'est pourquoi elles sont automatiquement normalisées dans les conséquences de ces règles d'expansion.

Comme dans le cas des tableaux modulo classiques, nous allons définir une stratégie d'application des règles qui correspond à une recherche de preuve

$$\frac{Tp \Vdash A \vee B}{Tp \Vdash A \mid Tp \Vdash B} \qquad \frac{Fp \Vdash A \wedge B}{Fp \Vdash A \mid Fp \Vdash B}$$

$$\frac{Tp \Vdash A \wedge B}{Tp \Vdash A, Tp \Vdash B} \qquad \frac{Fp \Vdash A \vee B}{Fp \Vdash A, Fp \Vdash B}$$

$$\frac{Tp \Vdash A \Rightarrow B}{Fp' \Vdash A, Tp' \Vdash B} \text{ pour tout monde } p' \geq p \qquad \frac{Fp \Vdash A \Rightarrow B}{Tp' \Vdash A, Fp' \Vdash B} \text{ pour un nouveau monde } p' \geq p$$

$$\frac{Tp \Vdash \neg A}{Fp' \Vdash A} \text{ pour tout monde } p' \geq p \qquad \frac{Fp \Vdash \neg A}{Tp' \Vdash A} \text{ pour un nouveau monde } p' \geq p$$

$$\frac{Tp \Vdash \exists x P(x)}{Tp \Vdash P(c)} \text{ pour un nouveau monde } c \qquad \frac{Fp \Vdash \exists x P(x)}{Fp \Vdash P(t)} \text{ pour tout terme } t$$

$$\frac{Tp \Vdash \forall x P(x)}{Tp' \Vdash P(t)} \text{ pour tout monde } p' \geq p \text{ et tout terme } t \qquad \frac{Fp \Vdash \forall x P(x)}{Fp' \Vdash P(c)} \text{ pour tout monde } p' \geq p \text{ et une constante fraîche } c$$

FIG. 6.3 – Tableaux intuitionnistes modulo à la Nerode et Shore

systématique (que nous allons démontrer complète par la suite).

Définition 43 (Génération systématique)**.** L'objectif de cette génération systématique est de construire un arbre représentant une variante intuitionniste des ensembles de Hintikka afin de satisfaire les conditions du lemme 27. La construction est semblable à celles qui sont définies dans [20, 46]. L'ensemble des mondes K est fait de séquences d'entiers, ordonnées par l'ordre préfixe.

Ainsi, pour tout monde, on définit par induction un ensemble de base $\mathcal{D}(p)$:

- $\mathcal{D}(\emptyset)$ est le langage construit par $\mathcal{L}_0 \cup C_\emptyset$, où $\mathcal{L}_0$ est l'ensemble des termes clos du langage du premier ordre, et $C_\emptyset$ un ensemble de constantes fraîches ;
- if q est la concaténation de la séquence p et du nombre entier k (i.e $p \bullet k$), alors $\mathcal{D}(q)$ est défini comme le langage construit à partir de $\mathcal{D}(p) \cup C(q)$. Nous supposons que nous avons un nombre suffisant d'ensembles disjoints C_p de constantes fraîches pour tout monde p.

Dans chacun de ses ensembles, on définit une énumération des termes ainsi qu'une énumération des paires (p, t) telles que p soit un monde et $t \in \mathcal{D}(p)$.

Soit aussi d une fonction mesurant la profondeur d'un nœud (longueur du chemin de la racine à ce nœud). Soit $\text{pos}(\Gamma_1, \Gamma_2)$ une fonction prenant deux branches en argument et retournant 0 si Γ_1 est à gauche de Γ_2 et 1 sinon. Alors, on définit l'ordre $\prec_n$ sur les nœuds de l'arbre représentant le tableau comme suit :

$$\begin{aligned} n_1 \prec_n n_2 \quad \hat{=} \quad & d(n_1) < d(n_2) \text{ ou} \\ & d(n_1) = d(n_2) \text{ et } \text{pos}(\Gamma_1, \Gamma_2) = 0, n_1 \in \Gamma_1, n_2 \in \Gamma_2 \end{aligned}$$

Alors le tableau est construit petit-à-petit en étendant à chaque étape le plus petit nœud non utilisé suivant $\prec_n$. Nous décrivons le processus d'expansion pour certains cas :

- Si, sur le chemin Γ, on a deux formules contradictoires, alors on peut fermer toutes les branches contenant Γ.
- Supposons que le nœud sélectionné contienne la formule $Tp \Vdash \forall x P(x)$. Soit (q,t) la plus petite paire telle que :
  - $q \geq p$,
  - $t \in \mathcal{D}(p)$,
  - le monde q apparaît sur le chemin Γ
  - et la déclaration $Tq \Vdash P(t) \downarrow$ n'y apparaisse pas.

  Alors on attache à chaque feuille dont Γ est un segment initial, les deux déclarations :
  $$\{Tp \Vdash P(t) \downarrow, Tp \Vdash \forall x P(x)\}.$$
  On recopie la déclaration $Tp \Vdash \forall x P(x)$ pour en avoir une copie inutilisée afin de pouvoir (plus tard) continuer l'énumération des termes et des mondes.
- Supposons que le nœud sélectionné contienne la formule $Fp \Vdash \forall x P(x)$. Soit k le plus petit entier tel que $p \bullet k$ n'apparaisse sur aucune branche ayant Γ comme segment initial de $Fp \Vdash \forall x P(x)$. Alors $q = p \bullet k$ n'est comparable à aucun monde sauf p. Soit aussi $c \in C_q$ une constante fraîche.

Alors on attache à chaque feuille des branches étendant Γ la déclaration $Fq \Vdash P(c)$.

Nous adoptons la convention en accord avec la définition 42 que sur une branche fermée, toutes les déclarations apparaissent (la structure de Kripke sera impropre).

La construction systématique de la définition 43 remplit alors les conditions du lemme suivant, proches des propriétés d'un ensemble de Hintikka en logique classique.

Lemme 27. *Soit Γ une branche d'un tableau complètement développé suivant la définition 43. Notons B pour « soit T soit F ». Alors :*

- *si $Bp \Vdash P$ apparaît, alors $Bp \Vdash P{\downarrow}$ également. Ainsi si $Tp \Vdash P$ apparaît, alors $Tp \Vdash P{\downarrow}$ également.*
- *si $Tp \Vdash A \wedge B$ (resp. $Fp \Vdash A \vee B$) appararaît, alors $Tp \Vdash A$ et $Tp \Vdash B$ (resp. $Fp \Vdash A$ et $Fp \Vdash B$) également.*
- *si $Fp \Vdash A \wedge B$ (resp. $Tp \Vdash A \vee B$) apparaît, alors $Fp \Vdash A$ ou $Fp \Vdash B$ (resp. $Tp \Vdash A$ ou $Tp \Vdash B$) également.*
- *si $Tp \Vdash \neg P$ (resp. $Tp \Vdash A \Rightarrow B$) et un monde $p' \geq p$ apparaissent alors $Fp' \Vdash P$ (resp. $Fp' \Vdash A$ ou $Tp' \Vdash B$) également*
- *si $Fp \Vdash \neg P$ (resp. $Fp \Vdash A \Rightarrow B$) apparaît alors il existe un monde $p' \geq p$, tel que $Tp' \Vdash P$ (resp. $Tp' \Vdash A$ and $Fp' \Vdash B$) apparaisse(nt) également.*
- *si $Fp \Vdash \exists x P(x)$ apparaît alors pour tout $t \in \mathcal{D}(p)$, $Tp \Vdash P(t){\downarrow}$ aussi.*
- *if $Tp \Vdash \exists x P(x)$ apparaît alors il existe une constante fraîche $c \in \mathcal{D}(p)$ telle que $P(c)$ apparaisse aussi.*
- *si $Fp \Vdash \forall x P(x)$ apparaît alors il existe un monde $p' \geq p$ et une constante fraîche $c \in D(p')$ tels que $Tp' \Vdash P(c)$ apparaisse.*
- *si $Tp \Vdash \forall x P(x)$ et un monde $p' \geq p$ apparaissent sur une branche, alors pour tout terme $t \in D(p')$, $Tp' \Vdash P([x := t]){\downarrow}$ aussi.*

Démonstration. Définissons $\mathcal{T} = \lim \mathcal{T}_n$ le tableau généré par la définition 43, tous les nœuds sont utilisés et nous avons effectivement énuméré tous les mondes $q \geq p$ et tous les termes de $\mathcal{D}(q)$ (pour les propositions universelles), que la branche soit fermée ou non, au vu de la convention adoptée juste avant ce lemme. □

Nota Bene 3. Ce lemme démontre qu'une branche ouverte est la contrepartie intuitionniste d'un ensemble d'Hintikka (ou une semi-valuation de Schütte). Cependant on ne peut *pas* identifier de branche ouverte dans le tableau (on ne veut pas utiliser le lemme de König) et il le faut dans une optique constructiviste.

6.4 Complétude

Comme dans le chapitre précédent la libéralisation des règles du calcul des séquent intuitionnistes fait apparaître une preuve de complétude plus difficile car la structure de Kripke que l'on va construire doit également être un modèle pour les règles de réécriture.

La complétude de la méthode n'est donc assurée que pour les mêmes classes de système de réécriture que pour la logique classique.

Theorème 13 (Complétude de la méthode de tableaux)**.** *Soit $\mathcal{R}$ un système de réécriture vérifiant une des conditions d'ordre, de positivité ou un mélange des deux (les conditions sont détaillées dans les sections suivantes).*

Soit Γ un ensemble de propositions. Soit P une proposition.

Si pour un monde quelconque α d'une structure de Kripke, $\alpha \Vdash \Gamma$ implique $\alpha \Vdash P$, alors toute branche du tableau systématique à partir de $T\emptyset \Vdash \Gamma, F\emptyset P$ est fermée.

Les preuves de complétude que l'on va faire sont légèrement différentes des démonstrations usuelles, car nous ne considérons plus les branches ouvertes. À la place, on utilisera l'approche suivante : à partir d'une branche d'un tableau complètement développé, on définira une structure de Kripke puis on démontrera qu'elle est en accord avec les déclarations présentes sur cette branche et qu'elle est un modèle de $\mathcal{R}$. Le résultat obtenu par cette méthode est $\emptyset \Vdash \Gamma$ (et par hypothèse $\emptyset \Vdash P$). On obtient aussi

$$(\emptyset \Vdash P) \Rightarrow (\emptyset \Vdash \bot)$$

autrement dit la structure de Kripke est impropre et nous rencontrerons alors sur cette branche $\Vdash \bot$: la branche est donc fermée.

6.4.1 Une condition d'ordre assurant la complétude

On considère un système confluent et un ordre bien fondé $\prec$ qui respecte les conditions suivantes :

- si $P \to Q$ alors $Q \prec P$
- si A est une sous-formule de B, $A \prec B$.

La définition d'une structure de Kripke $\mathcal{K} = \langle K, \leq, D, \Vdash \rangle$ pour une branche est faite de la manière suivante :

- $K = \{p$ séquences d'entiers $\}$. Cet ensemble est ordonné par la relation d'ordre utilisée dans la construction du tableau.
- $\mathcal{D}(p)$ est l'ensemble des termes clos qui apparaissent dans toutes les déclarations forcées mettant en cause le monde p ou un monde $q \leq p$.
- La relation de forçage $\Vdash$ est définie par induction sur la taille des formules. Pour les formules atomiques normales, $q \Vdash P$ si et seulement si $Tp \Vdash P$ est présent sur la branche pour un monde $p \leq q$ donné. La relation de forçage est étendue aux formules non atomiques en respectant la définition d'une structure de Kripke. Trois cas non standards apparaissent :
  - on aura $p \Vdash \neg P$ si, pour tout $q \leq p$, on n'a pas $q \Vdash P$;
  - si P est une proposition atomique non normale, alors $p \Vdash P$ ssi $p \Vdash P\downarrow$;
  - enfin, si $Tp \Vdash \bot$ apparaît sur une branche, on ajoute $q \Vdash P$ pour tout monde q et toute proposition P.

Cette définition est bien fondée car $\prec$ est bien fondée. On définit bien une structure de Kripke. Mais nous avons besoin d'un nouveau résultat avec les structures modifiées que nous employons.

Lemme 28. *Si la structure de Kripke est impropre, la branche est fermée.*

Démonstration. On peut avoir $p \Vdash \bot$ sur une branche pour deux raisons :
- si $Tp \Vdash \bot$ apparaît : dans ce cas, la branche est fermée.
- sinon c'est parce qu'on a un monde p et une proposition P tels que $p \Vdash P$ et $(p \Vdash P) \Rightarrow p \Vdash \bot$ apparaissent tous les deux. Le second déclaration $(p \Vdash P) \Rightarrow p \Vdash \bot$ ne peut que résulter de $q \Vdash \neg P$, pour un $q \leq p$, mais cette déclaration n'est jamais interprétée (car $p \Vdash P$) sauf si on sait déjà que $q \Vdash \bot$.

□

La structure de Kripke qu'on a définie est-elle en accord avec la branche ? C'est-à-dire si $Tp \Vdash P$ apparaît sur la branche a-t-on $p \Vdash P$? (et si $Fp \Vdash P$ a-t-on $p \nVdash P$?). La réponse est positive (et immédiate) pour les propositions atomiques normales et elle est étendue par induction sur $\prec$. L'analyse par cas est un peu différente car on interprète $p \nVdash P$ comme « si $p \Vdash P$ alors $p \Vdash \bot$ ». Détaillons certains cas :
- Si $Fp \Vdash P$ apparaît, avec P atome normal, alors $p \Vdash P$ n'est interprété dans la structure de Kripke que si celle-ci est impropre.
- Si $Fp \Vdash \neg P$ apparaît : à partir de la construction du tableau, nous avons dans la branche (en-dessous) la déclaration

$$T(p \bullet k) \Vdash P$$

pour un nouveau monde $p \bullet k$[1]. Par hypothèse d'induction $p \bullet k \Vdash P$. Si $p \Vdash \neg P$, alors par monotonicité $p \bullet k \Vdash \neg P$ et la structure de Kripke est alors impropre.

De plus, cette structure de Kripke est un modèle des règles de réécriture (par induction sur $\prec$). On a alors :

$$\emptyset \Vdash \Gamma, \emptyset \nVdash Q$$

et $\mathcal{K}$ est un modèle de Kripke de $\mathcal{R}$.

6.4.2 Une condition de positivité assurant la complétude

Comme dans le cas classique une condition de positivité peut être imposée sur les règles de réécriture propositionnelles. On suppose que si $P \to Q$, tous les prédicats atomiques ayant une occurrence dans Q apparaissent *positivement* (voir définition 40).

Si cette condition est remplie, il faut premièrement saturer la branche de telle manière que l'on puisse décider de la vérité d'un nombre de formules aussi grand que possible.

Le processus de complétion suivant la définition d'une structure de Kripke est défini comme suit : soit une énumération des paires (p, P) où $p \in K$ et la

[1] nous avons adopté la convention que, si $q \Vdash \bot$ apparaît, alors toutes les autres déclarations également

formule P est définie sur le langage $\mathcal{D}(p)$. Pour ces p et P, il faut ajouter les déclarations suivantes sur la branche :

- $Bp \Vdash P$ si $Bp \Vdash P\downarrow$ apparaît.
- $Tp \Vdash P$ (resp. $Fp \Vdash P$) si $Tq \Vdash P$ (resp. $Fq \Vdash P$) apparaît pour un monde $q \leq p$ (resp. $q \geq p$). En suivant la définition 42 d'une structure de Kripke, la vérité se propage vers le haut, et la fausseté vers le bas dans la succession des mondes de la structure.
- $Tp \Vdash P \wedge Q$ si $Tp \Vdash P$ et $Tp \Vdash Q$ apparaissent.
- $Fp \Vdash P \vee Q$ si $Fp \Vdash P$ ou $Fp \Vdash Q$ apparaît.
- $Tp \Vdash \forall x P(x)$ si pour tout monde $q \geq p$, tout terme $t \in \mathcal{D}(q)$, $q \Vdash P(t)\downarrow$ apparaît.
- $Fp \Vdash \forall x P(x)$ s'il existe un monde $q \geq p$ et un terme $t \in \mathcal{D}(q)$ tels que $Fq \Vdash P(t)\downarrow$ apparaissent

Les étapes de ce processus de complétion satisfont toujours la condition que deux déclarations de forçage opposées $Tp \Vdash P$ et $Fq \Vdash P$ pour un monde $q \geq p$ n'apparaissent sur une branche que si celle-ci est fermée (et donc le lemme 27 reste valide).

Le nombre de formules interprétées augmente à chaque itération de ce processus. Nous prenons le plus petit point fixe de cette opération comme branche dans le reste de cette section.

Lemme 29. *1. Si $Tp \Vdash P$ apparaît, alors pour tout monde $q \geq p$, $Tq \Vdash P$ aussi.*

2. Si $Fp \Vdash P$ apparaît, alors pour tout monde $q \leq p$, $Fq \Vdash P$ aussi.

3. Si $P \equiv_{\mathcal{R}} Q$ et $Bp \Vdash P$ apparaissent, alors $Bp \Vdash Q$ aussi.

4. La nouvelle branche est fermée ssi la branche originale est fermée.

5. La nouvelle branche vérifie le lemme 27

Démonstration. Le processus de complétion défini au-dessus entraîne les deux premiers points.

Les deux derniers viennent d'être démontrés ci-dessus.

Le troisième est également valide car $Bp \Vdash P$ n'apparaît que si $Bp \Vdash P\downarrow$ apparaît également, ce que l'on démontre par induction sur la taille de P. □

Ce processus est nécessaire pour définir une structure de Kripke. On doît en savoir autant que possible au sujet de chaque formule. Ceci est absolument nécessaire pour le lemme 31 ci-dessous.

Nous définissons ici l'équivalent intuitionniste de la valuation partielle de Schütte de la définition 34 du chapitre précédent. Cet équivalent satisfait plus que le lemme 27. Ainsi, $Fp \Vdash \forall x P(x)$ apparaît *si et seulement si* il existe un monde $p' \geq p$ et une constante fraîche $c \in D(p')$ tels que $Tp' \Vdash P(c)\downarrow$ apparaisse.

La différence avec les valuations partielles usuelles provient du fait que l'on peut se trouver dans un cas dégénéré. Cependant, la valuation n'est toujours pas totale car certaines formules ne sont toujours pas interprétées et nous n'avons pas encore de modèle.

Nous allons donc construire une structure de Kripke $\mathcal{K} = \langle K, \leq, D, \Vdash \rangle$ comme dans la section 6.4.1 précédente sauf que la relation de forçage est définie par induction sur la taille des formules.

- Pour toute proposition atomique P (du langage $\mathcal{D}(q)$), on pose $q \Vdash P$ si $Tq \Vdash P$ apparaît sur la branche. Si $Fp \Vdash P$ n'apparaît pas, on posera de même $p \Vdash P$.
- L'extension aux propositions composées est faite de la même manière qu'auparavant.

Ce modèle est (par construction) une structure de Kripke. Démontrons maintenant que $\mathcal{K}$ est en accord avec la branche.

Lemme 30. *Si une déclaration $Tp \Vdash P$ (resp. $Fp \Vdash P$) apparaît sur une branche, alors $p \Vdash P$ (resp. $p \nVdash P$) apparaît dans la structure de Kripke $\mathcal{K}$.*

Démonstration. Par induction sur la structure de P.

Le cas de base (P atomique) est traité par la définition.

Les autres cas sont immédiats après avoir remarqué que la branche satisfait le lemme 27. □

La structure de Kripke est en accord avec la branche donc $\emptyset \Vdash \Gamma$ et $\emptyset \nVdash P$. Montrons maintenant que cette structure de Kripke est également un modèle de $\mathcal{R}$.

Le lemme 30 nous dit que si $P \to Q$, et $Q \downarrow = P \downarrow$ apparaît dans la branche en tant que déclaration $Bp \Vdash P \downarrow$, alors les trois formules $P, Q, Q \downarrow$ partagent la même relation de forçage avec p.

Mais si $Q \downarrow$ n'apparaît pas ? C'est le moment d'utiliser l'hypothèse de positivité de $\mathcal{R}$: Q est donc positive. Démontrons alors le lemme suivant.

Lemme 31. *Soit P^+ une formule positive et Q^- une formule négative (i.e. $\neg Q$ est positive) définie à partir $\mathcal{D}(p)$.*

Si $Bp \Vdash P^+$ (resp. $Bp \Vdash Q^-$) n'apparaît pas (que $B = T$ ou $B = F$) dans la branche, alors $p \Vdash P+$ (resp. $p \nVdash Q^-$).

Démonstration. Supposons qu'aucune déclaration $Tp \Vdash \bot$ n'apparaisse dans la branche. Dans le cas contraire, $Bp \Vdash \phi$ apparaît par définition pour tout monde p et toute proposition ϕ.

Ainsi la structure de Kripke (propre) définie, $p \nVdash P$ signifie en particulier que l'on n'a pas $p \Vdash P$.

On procède alors par induction sur la structure de P et Q. Seuls quelques cas clés sont détaillés.

Si P est une proposition atomique, même non normale, elle est par hypothèse positive, et d'après la structure de Kripke construite, $p \Vdash P$.

Si $P^+ = A^+ \vee C^+$, alors comme $Tp \Vdash P^+$ n'apparaît pas, ni $Tp \Vdash A$ ni $Tp \Vdash C$ n'apparaissent. De la même façon, soit $Fp \Vdash A$, soit $Fp \Vdash C$ n'apparaît pas. Sinon $Fp \Vdash P$ aurait été interprété par le processus de complétion. Supposons que la première déclaration n'apparaisse pas, alors on applique l'hypothèse d'induction à A. On obtient alors que $p \Vdash A$. Donc $p \Vdash P$.

Si $P^- = A^- \vee C^-$, nous avons les mêmes résultats. Nous devons prouver que $p \not\Vdash A$ et $p \not\Vdash C$. Il existe alors deux cas : si $Fp \Vdash A$ on conclut par lemme 30, sinon on utilise l'hypothèse d'induction.

Si $P = \forall x A^+(x)$, soit $q \geq p$ un monde et $t \in \mathcal{D}(q)$. Si $Bp \Vdash P^+$ n'apparaît nulle part, c'est parce que $Fq \Vdash A(t)$ n'apparaît pas (sinon $Fp \Vdash P$ aurait été défini par le processus de complétion). Si $Tq \Vdash A(t)$ apparaît, $q \Vdash A(t)$ par le lemme 30. Sinon $q \Vdash A(t)$ par hypothèse d'induction. Alors par définition des structures de Kripke, $p \Vdash \forall x A^+(x)$.

Si $Q = \forall x A^-(x)$ alors il existe au moins un monde $q \geq p$ et un terme $t \in \mathcal{D}(q)$ tels que $Tq \Vdash A(t)$ n'apparaisse pas. Sinon, le processus de complétion aurait produit $Tp \Vdash \forall x A^-(x)$, contredisant notre hypothèse. Si $Fq \Vdash A(t)$ apparaît, on applique le lemme 30, sinon on applique l'hypothèse d'induction. Dans les deux cas $q \not\Vdash A(t)$. Ainsi par définition de la structure de Kripke $p \not\Vdash \forall x A^-(x)$.

Les autres connecteurs sont traités de manière similaire. □

Considérons maintenant un atome P tel que $P \to Q$, et un monde p . Si P apparaît dans une déclaration $Bp \Vdash P$ alors $Bp \Vdash Q$ par le lemme 29. De plus par le lemme 30, P et Q ont la même interprétation. Sinon, puisque Q est positive par hypothèse, $p \Vdash Q$. Et $p \Vdash P$ par définition. Ainsi, les règles de réécriture sont valides dans $\mathcal{K}$, qui est donc un modèle.

La méthode des tableaux modulo intutionniste est donc complète pour la condition de postivité.

6.4.3 Mélange des deux conditions assurant la complétude

On va maintenant voir dans quelle mesure on peut aussi être certain de la complétude de systèmes de réécriture mélangeant les conditions des sections 6.4.1 et 6.4.2.

On considère deux systèmes de réécriture $\mathcal{R}_>$ (ordonné) et $\mathcal{R}_+$ (positif). Sous les hypothèses que $\mathcal{R} = \mathcal{R}_> \cup \mathcal{R}_+$ et la condition que $\mathcal{R}_+$ est normal à droite (voir définition 44) pour $\mathcal{R}_>$, la complétude de la méthode de tableau peut être démontrée.

Définition 44 (Normalité droite)**.** Soient deux systèmes de réécriture R' et R. R' est *normal à droite* pour $\mathcal{R}$ si pour toute règle de réécriture propositionnelle $l \to r \ \in \mathcal{R}'$, toutes les instances des atomes de r par des substitutions $\mathcal{R}$-normales σ sont en forme normale pour $\mathcal{R}$.

Pour cette nouvelle condition sur les règles de réécriture, le modèle est construit de la façon suivante : saturer les branches comme en section 6.4.2 et définir le modèle par induction sur l'ordre bien fondé $>$. Les atomes non R_+-normaux sont interprétés comme en section 6.4.2 également.

La structure de Kripke $\mathcal{K}$ est en accord avec la branche, comme auparavant, et est un modèle de $\mathcal{R}_>$. Ces deux faits sont démontrables par induction sur l'ordre bien fondé $>$. Par ailleurs, $\mathcal{K}$ est aussi un modèle de $\mathcal{R}_+$.

Lemme 32. *Soit P^+ une formule positive et Q^- une formule négative définies à partir de $\mathcal{D}(p)$. Supposons que toutes les instances (par substitutions $\mathcal{R}$-normales) des propositions atomiques de P, Q soient normales pour $\mathcal{R}_>$.*

Si $Bp \Vdash P^+$ (resp. $Bp \Vdash Q^-$) n'apparaît pas (que $B = T$ ou $B = F$) dans la branche alors $p \Vdash P^+$ (resp. $p \nVdash Q^-$).

Démonstration. Comme pour le lemme 31, la démonstration est faite par induction sur la structure de la formule. Notons qu'on ne peut appliquer les règles de réécriture de $\mathcal{R}_>$. □

On peut maintenant conclure que toute règle $P \to R \in \mathcal{R}$ est valide dans la structure de Kripke $\mathcal{K}$.

6.4.4 Contenu calculatoire de la preuve de complédue

Nous exhibons ici un résultat important vis-à-vis de la relation entre l'élimination des coupures sémantique et la normalisation de preuve.

Revenons à la règle inspirée du paradoxe de Russell :

$$R \in R \to \forall y(y \simeq R \Rightarrow (y \in R \Rightarrow (A \Rightarrow A))) \tag{6.1}$$

Cette règle-ci permet de bâtir des preuves non normalisables [27, 40] dans le calcul des séquents modulo associé. Cependant la règle de coupure est admissible dans le calcul des séquents muni de la règle 6.1. Nous avons réutilisé les méthodes sémantiques qui ont permis de montrer cette admissibilité dans [39]. Nous avons repris aussi les mêmes conditions générales d'ordre et de positivité pour lesquelles l'admissibilité est plus généralement démontrée.

Les méthodes sémantiques pour montrer l'élimination des coupures pour le calcul des séquents modulo avec 6.1 permettent également de démontrer la complétude des tableaux, notamment dans le cas où la règle 6.1 est le système de réécriture considéré.

Theorème 14. *La méthode des tableaux modulo est complète pour ce système de réécriture.*

Démonstration. Étant donnée une branche, on définit la structure de Kripke $\mathcal{K}$ comme en section 6.4.2 : celle-ci est en accord avec la branche (toujours section 6.4.2).

Si la structure de Kripke est impropre, alors la branche est fermée. De plus la règle de réécriture 6.1 est valide. En effet, la formule

$$\forall y(y \simeq R \Rightarrow (y \in R \Rightarrow (A \Rightarrow A)))$$

est toujours forcée pour tout nœud de la structure de Kripke (c'est une tautologie intuitionniste). Cette preuve de complétude permet de définir un théorème d'élimination des coupures pour ce système de réécriture. □

La preuve de complétude des tableaux pour les différentes conditions d'ordre et de positivité est faite de manière *constructive*.

On sait déjà passer de $\Gamma \vdash P$ à $\Gamma \vdash_{\mathcal{R}} P$ par la preuve de correction sémantique (admise) du calcul des séquents intutionnnistes modulo (voir [39]). On vient de montrer de manière constructive comment passer de $\Gamma \vdash_{\mathcal{R}} P$ à $\mathrm{Tab}(T\emptyset \models \Gamma, F\emptyset \models P)$.

Le résultat du calcul de la preuve de complétude sémantique est une méthode des tableaux. Il ne reste plus qu'à transformer une preuve obtenue par la méthode des tableaux en preuve du calcul des séquents intuitionniste sans coupure.

6.5 Correction

Nous nous proposons maintenant de démontrer la correction de la méthode des tableaux modulo intuitionniste par rapport au calcul des séquents sans coupure modulo.

Ce fait est relativement bien établi en logique classique mais dans le cas intuitionniste, il n'est pas aussi évident. En effet, une preuve obtenue par la méthode des tableaux correspond à un séquent avec de multiples conclusions (c'est une disjonction de ces multiples conclusions), alors que le calcul des séquents n'a qu'un unique membre droit.

La correction des tableaux intuitionnistes est *toujours* démontrée par rapport aux structures de Kripke. On trouve une tentative de preuve de correction syntaxique dans [20]. Nous allons partir de ce travail pour donner quelques définitions.

Définition 45. Soient p un monde et $\mathcal{B}$ un chemin dans le tableau. Alors les ensembles $T_p(\mathcal{B})$ et $F_p(\mathcal{B})$ sont définis comme :

$$\begin{aligned} T_p(\mathcal{B}) &= \{P \mid \exists q, q \leq p \mid Tq \Vdash P \in \mathcal{B}\} \\ F_p(\mathcal{B}) &= \{P \mid Fp \Vdash P \in \mathcal{B} \end{aligned}$$

Notons $\bigvee \mathcal{S}$ la disjonction d'un nombre arbitraire d'éléments de l'ensemble $\mathcal{S}$.

Un chemin $\mathcal{B}$ est dit *cohérent* si pour tout monde p

$$T_p(\mathcal{B}) \nvdash^*_{\mathcal{R}} \bigvee F_p(\mathcal{B})$$

Nota Bene 4. L'appartenance d'une déclaration à un chemin donné indique que cette déclaration apparaît effectivement sur le chemin.

$T_p(\mathcal{B})$ contient ainsi toutes les formules vraies dans tous les mondes inférieurs à p. D'un autre côté, $F_p(\mathcal{B})$ contient les formules fausses *uniquement pour le monde p*. Ceci est bien évidemment la conséquence de la définition d'une structure de Kripke.

La différence majeure de cette définition avec [20] est celle de la cohérence d'un chemin.

Si la règle de coupure était acceptée, l'associativité du connecteur $\vee$ serait immédiate. Ici, nous avons besoin de la démontrer.

Lemme 33. *Soient P, Q, R des formules et Γ un ensemble de formules.*
Si l'on a une preuve Θ de

$$\Gamma \vdash^*_{\mathcal{R}} P \vee (Q \vee R)$$

alors on peut construire une preuve de

$$\Gamma \vdash^*_{\mathcal{R}} (P \vee Q) \vee R.$$

Démonstration. Cette démonstration est faite par induction sur la preuve Θ. On utilise une hypothèse d'induction plus forte : Θ est une preuve de $\Gamma \vdash^*_{\mathcal{R}} \mathcal{P}$ où $\mathcal{P}$ peut être $P \vee (Q \vee R)$ ou $Q \vee R$. On regarde la dernière règle appliquée dans Θ.

- Si la dernière règle de Θ est un axiome, alors la remplacer par la preuve de $\mathcal{P} \vdash^*_{\mathcal{R}} (P \vee Q) \vee R$.
- Si c'est $\vee$-r, alors nous avons une preuve de $\Gamma \vdash^*_{\mathcal{R}} P$, $\Gamma \vdash^*_{\mathcal{R}} Q$, $\Gamma \vdash^*_{\mathcal{R}} R$ ou $\Gamma \vdash^*_{\mathcal{R}} Q \vee R$.
  Pour les trois premiers cas, il suffit de compléter la preuve que l'on a effectivement en ajoutant deux règles $\vee$-r pour obtenir la preuve de $\Gamma \vdash^*_{\mathcal{R}} (P \vee Q) \vee R$.
  Dans le dernier cas, on réapplique l'hypothèse d'induction à $\Gamma \vdash^*_{\mathcal{R}} Q \vee R$.
- Dans les autres cas, il faut utiliser l'hypothèse d'induction sur les prémisse(s) de la règle (sauf si P est effacée) et appliquer la même règle à $\Gamma \vdash^*_{\mathcal{R}} (P \vee Q) \vee R$

□

La commutativité de $\vee$ est immédiate : il suffit d'inverser les prémisses.

On peut donc ne plus s'occuper des parenthèses et de l'ordre des propositions dans la notation $\bigvee \mathcal{S}$ pour la disjonction d'un sous-ensemble de $\mathcal{S}$. La conclusion peut aussi être affaiblie, en ajoutant des règles $\vee$-r afin d'obtenir la disjonction de l'ensemble $\mathcal{S}$ entier.

Nous allons maintenant établir un lemme de « renforcement » du calcul des séquents, permettant d'avoir plusieurs propositions à droite. Encore une fois, cette propriété est immédiate avec la règle de coupure.

Lemme 34. *Soient P, Q, R des formules et Γ_1, Γ_2 des ensembles de formules.*

1. *À partir des preuves*
   $$\Gamma_1 \vdash^*_{\mathcal{R}} P \vee R$$
   et
   $$\Gamma_2 \vdash^*_{\mathcal{R}} Q \vee R$$
   on peut construire une preuve de
   $$\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R$$

2. *À partir de la preuve*
$$\Gamma_1 \vdash^*_{\mathcal{R}} P(t) \vee R$$
on peut construire une preuve de
$$\Gamma_1 \vdash^*_{\mathcal{R}} (\exists P(x) \vee R)$$

3. *À partir des preuves*
$$\Gamma_1, Q \vdash^*_{\mathcal{R}} R$$
et
$$\Gamma_2 \vdash^*_{\mathcal{R}} P \vee R$$
on peut construire une preuve de
$$\Gamma_1, \Gamma_2, A \Rightarrow B \vdash^*_{\mathcal{R}} C$$

Démonstration. On va se concentrer sur le premier cas de ce lemme. Les autres cas sont démontrés en utilisant des méthodes similaires.

On va construire en partant de la conclusion des preuves π_1 et π_2 de $\Gamma_1 \vdash^*_{\mathcal{R}} P \vee R$ et $\Gamma_2 \vdash^*_{\mathcal{R}} Q \vee R$ une preuve du séquent $\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R$.

L'idée générale est la suivante : inclure une copie π_1, utilisant Γ_1 puis, aux feuilles de π_1, inclure si nécessaire une copie de π_2, utilisant Γ_2. On construit une preuve de $\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R$ par induction sur π_1. Examinons la première règle :

- Si c'est une règle dont Γ_1 est une formule active, on applique l'hypothèse d'induction aux prémisses et on applique cette même règle au résultat de l'appel inductif.
  Par exemple pour la règle $\Rightarrow$-r :
$$\dfrac{\dfrac{\pi'_1}{\Gamma_1, Q' \vdash^*_{\mathcal{R}} P \vee R} \quad \dfrac{\pi''_1}{\Gamma_1 \vdash^*_{\mathcal{R}} P'}}{\Gamma_1, P' \Rightarrow Q' \vdash^*_{\mathcal{R}} P \vee R}$$
  Alors on applique l'hypothèse d'induction à π'_1 pour obtenir une preuve π' de $\Gamma_1, Q', \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R)$ et on applique alors la règle $\Rightarrow$-r.
$$\dfrac{\dfrac{\pi'}{\Gamma_1, Q', \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R} \quad \dfrac{\pi''_1}{\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} P'}\ \text{weaks}}{\Gamma_1, P' \Rightarrow Q', \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R}$$
- un affaiblissement droit (sur $P \vee R$). On va affaiblir à la place $(P \wedge Q) \vee R$ et ajouter suffisamment d'affaiblissement à gauche pour obtenir Γ_2. On obtient alors une preuve de
$$\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R$$
- une règle d'axiome $\mathcal{A}_1$ ou une règle $\vee$-r ($\mathcal{V}_1$). Ces cas sont ceux qui requièrent le plus d'attention. On arrête l'induction sur π_1 et on commence une induction sur π_2. Les constantes fraîches de π_2 sont aussi renommées pour rester fraîches pour Γ_1. Si la première règle de π_2 est

- une règle `l-rule` manipulant la gauche du séquent. Appliquer `l-rule` aux preuves fournies par l'application de l'hypothèse d'induction aux prémisses.
- un affaiblissement droit. On le traite comme pour l'induction sur π_1.
- un axiome ($\mathcal{A}_2$) ou une règle $\vee$-r ($\mathcal{V}_2$). Il y a alors quatre cas.
  Le premier correspond à avoir $\mathcal{A}_1, \mathcal{A}_2$. Alors il existe par définition de la règle axiome en déduction modulo deux propositions $S \in \Gamma_1$ and $S' \in \Gamma_2$ telles que $S \equiv_{\mathcal{R}} A \vee C$ and $S' \equiv_{\mathcal{R}} B \vee C$. On peut alors immédiatement construire une preuve de $S, S' \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R$.
  Prenons maintenant le cas $\mathcal{V}_1$ (ou bien, de manière similaire, $\mathcal{V}_2$) . Il existe alors deux sous-cas. Si la prémisse π'_1 de $\Gamma_1 \vdash^*_{\mathcal{R}} R'$ avec $R' \equiv_{\mathcal{R}} R$, alors on construit la preuve suivante en ignorant π_2 :

$$\dfrac{\dfrac{\pi'_1}{\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} R'}\,\text{affs}}{\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R}\,\vee\text{-r}$$

  Dans l'autre sous-cas, la prémisse est une preuve π'_1 de $\Gamma_1 \vdash^*_{\mathcal{R}} P'$ où $P' \equiv_{\mathcal{R}} P$. Alors si on a $\mathcal{V}_2$ et que la prémisse est une preuve de $\Gamma_2 \vdash^*_{\mathcal{R}} R'$, on construit la même preuve que ci-dessus, en inversant les indices 1 et 2. Sinon, on a une preuve de $\Gamma_2 \vdash^*_{\mathcal{R}} Q'$ où $Q' \equiv_{\mathcal{R}} Q$, et on construit la preuve :

$$\dfrac{\dfrac{\dfrac{\pi'_2}{\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} Q'}\,\text{affs} \quad \dfrac{\pi'_1}{\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} P'}\,\text{affs}}{\Gamma_1, Q' \vdash^*_{\mathcal{R}} (P \wedge Q)}\,\wedge\text{-r}}{\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R}\,\vee\text{-r}$$

  Si l'on a $\mathcal{A}_2$, alors on construira la preuve :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_1, Q' \vdash^*_{\mathcal{R}} Q}\,\text{axiome} \quad \dfrac{\pi'_1}{\Gamma_1 \vdash^*_{\mathcal{R}} P}}{\Gamma_1, Q' \vdash^*_{\mathcal{R}} (P \wedge Q)}\,\wedge\text{-r}}{\Gamma_1, Q' \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R}\,\vee\text{-r} \quad \dfrac{\dfrac{}{\Gamma_1, R' \vdash^*_{\mathcal{R}} R}\,\text{axiome}}{\Gamma_1, R' \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R}\,\vee\text{-r}}{\Gamma_1, Q' \vee R' \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R}\,\vee\text{-l}}{\Gamma_1, \Gamma_2 \vdash^*_{\mathcal{R}} (P \wedge Q) \vee R}\,\text{affs}$$

  où $Q' \equiv_{\mathcal{R}} Q$, $R' \equiv_{\mathcal{R}} R$ et $Q' \vee R'$ la formule utilisée dans $\mathcal{A}_2$. C'est effectivement une disjonction car $\mathcal{R}$ est confluente et ses membres gauches sont atomiques, donc les connecteurs principaux de deux formules équivalentes sont les mêmes. ([39, 40]).

□

Nota Bene 5. On peut comparer ce résultat au calcul des séquents LB défini par Waaler et Wallen dans [62] : celui-ci autorise en fait les règles que l'on vient de démontrer. Cependant, la correction de LB est démontrée sémantiquement. Par conséquent notre résultat semble être nouveau.

Par la suite, on utilisera ce lemme 34 avec $\Gamma = \Gamma_1 = \Gamma_2$. Ainsi on obtiendra par exemple un preuve de $\Gamma, \Gamma \vdash^*_{\mathcal{R}} (P \vee Q) \vee R$ que l'on peut directement contracter en une preuve de $\Gamma \vdash^*_{\mathcal{R}} (P \vee Q) \vee R$.

On peut maintenant procéder à la démonstration de la correction de la construction des tableaux modulo intuitionnistes par rapport au calcul des séquents sans coupure.

Theorème 15 (Correction syntaxique des tableaux)**.** *Soient P une proposition et Γ un ensemble de propositions.*

Si $\Gamma \nvdash^*_{\mathcal{R}} P$ alors, il existe un chemin cohérent $\mathcal{B}$ dans la construction systématique de tableau ayant $T\emptyset \models \Gamma\downarrow, F\emptyset \models P\downarrow$ en entrée.

Démonstration. Nous allons montrer que si $\mathcal{B}$ est un chemin cohérent (une branche cohérente) dans un tableau partiellement développé, alors la méthode de la définition 43 l'étend obligatoirement par au moins un chemin cohérent.

La racine du tableau est cohérente : en effet avoir $\Gamma \vdash^*_{\mathcal{R}} P$ est la même chose qu'avoir $\Gamma\downarrow \vdash^*_{\mathcal{R}} P\downarrow$. Puis, nommons $Bp \Vdash P$ la plus petite déclaration apparaissant sur le chemin $\mathcal{B}$ inutilisé dans le développement du tableau (P est normal par construction car nous gardons toutes les formules en forme normale). Alors examinons la nature de $Bp \Vdash P$:

- Si $Bp \Vdash P = Tp \Vdash Q \wedge R$. $\mathcal{B}$ est alors étendu suivant les règles de la figure 6.3 par les déclarations $Tp \Vdash Q$ et $Tp \Vdash R$. Alors si le chemin $\mathcal{B}'$ ainsi créé devient incohérent, la faute en revient aux déclarations ajoutées. On a donc sur cette branche $\mathcal{B}'$ une preuve de $T_P(\mathcal{B}') \vdash^*_{\mathcal{R}} F_p(\mathcal{B}')$.
  Or par définition $T_p(\mathcal{B}') = T_p(\mathcal{B}) \cup \{Q, R\}$ et $F_p(\mathcal{B}') = F_p(\mathcal{B})$. Donc on peut construire la preuve
  $$\frac{T_P(\mathcal{B}') \vdash^*_{\mathcal{R}} F_p(\mathcal{B}')}{T_P(\mathcal{B}), P \vdash^*_{\mathcal{R}} F_p(\mathcal{B})} \wedge\text{-l},\ P = Q \wedge R$$
  qui contredit l'hypothèse $P \in T_p(\mathcal{B})$.
- Si $Bp \Vdash P = Fp \Vdash Q \wedge R$, $\mathcal{B}$ est alors étendu en deux chemins $\mathcal{B}_0$ (via Q) et $\mathcal{B}_1$ (via R). Si les deux sont incohérents, alors on obtient les preuves de :
  $$\mathcal{S} = T_P(\mathcal{B}) \vdash^*_{\mathcal{R}} Q \vee \bigvee F_p(\mathcal{B})$$
  et
  $$\mathcal{S}' = T_P(\mathcal{B}) \vdash^*_{\mathcal{R}} R \vee \bigvee F_p(\mathcal{B})$$
  avec $T_p(\mathcal{B}) = T_p(\mathcal{B}_o) = T_p(\mathcal{B}_1)$ et $F_p(\mathcal{B}_o) = F_p(\mathcal{B}) \cup \{Q\}$ et $F_p(\mathcal{B}_1) = F_p(\mathcal{B}) \cup \{R\}$.
  Par affaiblissement, si nécessaire, grâce au lemme 33, on peut considérer les occurrences de $\bigvee F_p(\mathcal{B})$ comme étant strictement identiques entre elles. Alors on applique le lemme 34 aux séquents $\mathcal{S}$ et $\mathcal{S}'$ pour obtenir une preuve de
  $$T_P(\mathcal{B}) \vdash^*_{\mathcal{R}} (Q \wedge R) \vee \bigvee F_p(\mathcal{B})$$
  autrement dit une contradiction avec l'hypothèse $Q \wedge R \in F_p(\mathcal{B})$.

- Si $Bp \Vdash P = Tp \Vdash Q \vee R$, alors, d'une manière semblable au cas précédent, si les deux chemins obtenus sont incohérents, il suffit d'appliquer la règle $\vee$-l aux séquents $T_p(\mathcal{B}), Q \vdash^*_{\mathcal{R}} \bigvee F_p(\mathcal{B})$ et $T_p(\mathcal{B}), R \vdash^*_{\mathcal{R}} \bigvee F_p(\mathcal{B})$ pour obtenir la contradiction recherchée.
- Si $Bp \Vdash P = Fp \Vdash Q \vee R$, alors si le nouveau chemin est incohérent on peut obtenir en utilisant le lemme 33, $T_p(\mathcal{B}) \vdash^*_{\mathcal{R}} (Q \vee R) \vee \bigvee F_p(\mathcal{B})$, contredisant par là le fait que $Q \vee R \in F_p(\mathcal{B})$.
- Si $Bp \Vdash P = Tp \Vdash Q \Rightarrow R$, alors si les deux chemins construits sont tout deux incohérents, on obtient des preuves de $T_q(\mathcal{B}_o) \vdash^*_{\mathcal{R}} \bigvee F_q(\mathcal{B}_0)$ et $T_q(\mathcal{B}_1) \vdash^*_{\mathcal{R}} \bigvee F_q(\mathcal{B}_1)$ car ce qui change par rapport au chemin $\mathcal{B}$ change dans le monde q. Par définition T_q et F_q, nous avons des preuves des séquents
$$T_q(\mathcal{B}) \vdash^*_{\mathcal{R}} Q \vee \bigvee F_q(\mathcal{B})$$
et
$$T_q(\mathcal{B})R \vdash^*_{\mathcal{R}} \bigvee F_q(\mathcal{B}).$$
En appliquant le lemme 34, on obtient une preuve de $T_q(\mathcal{B}), Q \Rightarrow R \vdash^*_{\mathcal{R}} \bigvee F_q(\mathcal{B})$, contredisant le fait que $Q \Rightarrow R \in F_q(\mathcal{B})$.
- Si $Bp \Vdash P = Fp \Vdash Q \Rightarrow R$, alors si le nouveau chemin $\mathcal{B}'$ est incohérent, on obtient une preuve Θ de $T_r(\mathcal{B}') \vdash^*_{\mathcal{R}} \bigvee F_r(\mathcal{B}')$. Le monde r est nouveau et comparable seulement avec un monde $q \leq p$ sur le chemin $\mathcal{B}$, on a $T_r(\mathcal{B})) = T_r(\mathcal{B}) \cup \{Q\}$ et $F_r(\mathcal{B}') = \{R\}$.
Par conséquent, on peut appliquer la règle $\Rightarrow$-r à la preuve Θ afin d'obtenir une preuve de $T_p(\mathcal{B}) \vdash^*_{\mathcal{R}} Q \Rightarrow R$, montrant l'incohérence de $\mathcal{B}$ car $Q \Rightarrow R \in F_p(\mathcal{B})$.
Dans ce cas, il est crucial de ne pas avoir d'autre choix pour $F_r(\mathcal{B}')$ que le singleton R. *C'est le moment exact où logique devient intuitionniste.* D'autres méthodes de tableaux comme celle [62] offrent également un traitement spécifique du connecteurs $\Rightarrow$ et et du quantificateur $\forall$: nous avons besoin d'un séquent à un seul membre droit.
- Si $Bp \Vdash P = Bp \Vdash \neg Q$, on traite le cas comme celui de $Bp \Vdash Q \Rightarrow \bot$.
- Si $Bp \Vdash P = Tp \Vdash \exists x Q(x)$, si le nouveau chemin est incohérent, alors on obtient une preuve du séquent $T_p(\mathcal{B}), Q(c) \vdash^*_{\mathcal{R}} \bigvee F_p(\mathcal{B})$. Il suffit d'appliquer la règle $\exists$-l à cette preuve (car c est une constante fraîche) pour montrer l'incohérence de $\mathcal{B}$.
- Si $Bp \Vdash P = Fp \Vdash \exists x Q(x)$, et le nouveau chemin est incohérent, alors nous obtenons une preuve de $T_p(\mathcal{B}) \vdash^*_{\mathcal{R}} Q(t) \downarrow \vee \bigvee F_p(\mathcal{B})$. Dans le calcul des séquents modulo, cette preuve se transforme très simplement en une preuve de $T_p(\mathcal{B}) \vdash^*_{\mathcal{R}} Q(t) \vee \bigvee F_p(\mathcal{B})$ puisque $Q(t) \equiv_{\mathcal{R}} Q(t) \downarrow$. Alors en utilisant le lemme 34, on peut construire une preuve de $T_p(\mathcal{B}) \vdash^*_{\mathcal{R}} \exists x Q(x) \vee \bigvee F_p(\mathcal{B})$, contredisant alors la cohérence de $\mathcal{B}$.
- Si $Bp \Vdash P = Tp \Vdash \exists x Q(x)$, alors si le nouveau chemin est incohérent, on obtient une preuve de $T_q(\mathcal{B}), Q(t) \downarrow \vdash^*_{\mathcal{R}} \bigvee F_q(\mathcal{B})$ que l'on convertit directement en une preuve de $T_q(\mathcal{B}), Q(t) \vdash^*_{\mathcal{R}} \bigvee F_q(\mathcal{B})$. À partir de cette preuve, il suffit d'appliquer la règle de $\forall$-l pour obtenir $T_q(\mathcal{B}), \forall x Q(x) \vdash^*_{\mathcal{R}} \bigvee F_q(\mathcal{B})$ et l'incohérence de $\mathcal{B}$.

– Si $Bp \Vdash P = Fp \Vdash \forall x Q(x)$ et que le nouveau chemin $\mathcal{B}'$ est incohérent, alors on doit avoir une preuve de $T_q(\mathcal{B}') \vdash^*_{\mathcal{R}} F_q(\mathcal{B}')$. Comme dans le cas $Fp \Vdash Q \Rightarrow R$, q est un monde comparable seulement avec p. On se retrouve en fait avec une preuve de $T_p(\mathcal{B}) \vdash^*_{\mathcal{R}} Q(c)$. Il suffit d'appliquer une règle $\forall$-r (c est une constante fraîche) pour obtenir l'incohérence de $\mathcal{B}$.

□

On vient effectivement de démontrer qu'un tableau fermé correspond à une preuve *sans coupure* dans le calcul des séquents intuitionniste modulo. Ce résultat est nouveau, même pour le calcul des séquents intuitionnistes LJ sans règles de réécriture.

De plus en combinant le théorème 15 avec le théorème de complétude 13 et le théorème de correction du calcul des séquents par rapport aux structures de Kripke (non démontré ici), on obtient un *théorème d'élimination des coupures sémantique constructive* pour les conditions de la section 6.4 (cf figure 6.4).

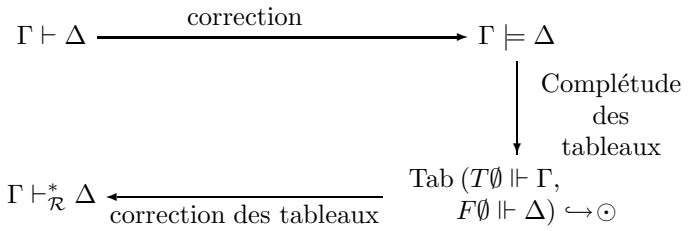


FIG. 6.4 – Élimination des coupures sémantique constructive

Theorème 16 (Élimination des coupures pour $\mathsf{LJ}_{\mathsf{mod}}$)**.** *Soient* Γ *un ensemble de propositions et* P *une proposition. Soit* $\mathcal{R}$ *un système de réécriture sur les termes et les propositions ordonné, positif ou mixte.*

Si $\Gamma \vdash_{\mathcal{R}} P$ *alors* $\Gamma \vdash^*_{\mathcal{R}} P$.

6.6 Conclusion

Nous avons dans ce chapitre produit une première version d'une méthode de preuve pour la logique intuitionniste modulo du premier ordre. La méthode est volontairement très simple. On peut d'ores et déjà penser à l'améliorer en tant que méthode de preuves en ajoutant des métavariables et de la skolemisation. Dans le même ordre d'idées, la normalisation de toute proposition n'est vraisemblablement pas efficace dans une recherche de preuve.

Nous avons prouvé de manière constructive la complétude du calcul en modifiant légèrement les structures de Kripke à la base de nos arguments sémantiques.

Puis nous avons prouvé de manière syntaxique que les tableaux intutionnistes (modulo) sont effectivement une recherche de preuve dans les calculs de séquents

intuitionnistes LJ et $\mathsf{LJ}_{\mathsf{mod}}$ correspondant. À notre connaissance, cette preuve n'existe pas non plus dans le cas intuitionniste sans règle de réécriture.

Enfin, nous nous sommes servis de ces résultats pour obtenir un théorème d'élimination des coupures constructif. Nous avons ainsi déterminé le contenu calculatoire de l'élimination des coupures.

Chapitre 7

Zenon et la déduction modulo

Comment intégrer la déduction modulo dans un prouveur automatique ? D. Doligez a développé dans le cadre du projet Focal un prouveur automatique pour la logique du premier ordre appelé Zenon. Nous allons voir dans quelle mesure on peut considérer qu'un certain degré de déduction modulo se trouve déjà dans Zenon et voir comment on peut envisager de l'étendre.

Nous présentons tout d'abord l'environnement dans lequel Zenon a vu le jour, à savoir Focal.

7.1 Focal

Le projet Focal [56] a pour but de proposer un environnement de programmation où on peut développer des programmes certifiés. Cet environnement comporte un langage de programmation empruntant à la fois aux langages fonctionnels et orientés objets et un langage de preuve.

7.1.1 Langage de programmation

Le langage de programmation de Focal permet, petit-à-petit, de passer de spécifications abstraites à leur implémentation concrète en suivant une hiérarchie d'unités de compilation similaire aux classes des langages orientés objets. Nous allons décrire ses principales caractéristiques (résumées en figure 7.1) et voir un exemple.

Le langage de Focal est fondé sur les concepts d'espèce, de collection, de type d'espèce, de type support et d'interfaces.

Les espèces sont les nœuds de la hiérarchie : elles sont formées d'un ensemble de champs appelés méthodes. Ces méthodes peuvent être seulement déclarées ou complètement définies. Trois types de méthodes existent :

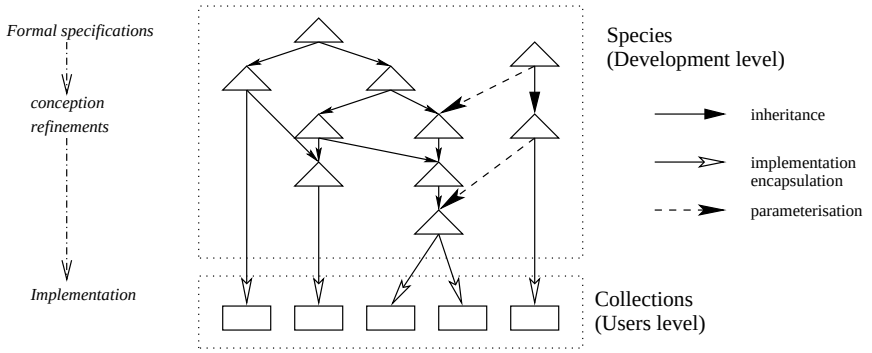


FIG. 7.1 – Hiérarchie Focal

- Le *type support* est le type des entités manipulées par les fonctions de l'espèce. Le type support n'est introduit que par un seul et unique champ d'une espèce.
- Les *fonctions* (ou *signatures* lorsqu'elles sont seulement déclarées), décrivent des opérations sur le type support.
- Les *propriétés* sont des spécifications que les implantations de l'espèce doivent vérifier. On peut également prouver des théorèmes pour les déclarations et définitions courantes des fonctions et propriétés.

Toute espèce possède un type, dit type d'espèce, constitué de la liste des déclarations/propriétés associées à chacun de ses champs. L'interface d'un espèce est obtenue par abstraction du type support dans le type de l'espèce. Héritage multiple, redéfinitions, champs qui sont des preuves font que certaines espèces ne peuvent pas recevoir d'interface. Le compilateur rejette de telles espèces. donc, toute espèce qui a pu être compilée possède une interface.

Une collection est obtenue par abstraction du type support dans la définition d'une espèce. Cette espèce doit être complète, en ce sens que toute déclaration doit être associée à une définition, toute propriété doit être prouvée. La collection a même interface que l'espèce complète qui a servi à la construire.

Une espèce $E1$ peut être paramétrée par l'interface d'une espèce déjà introduite $E2$. Cette interface peut être instanciée par n'importe quelle collection dont l'"interface contient (au sens inclusion des ensembles de champs) l'interface de E2.

Exemple

On définit une espce `s1` paramétrée par deux collections `c2` et `c3` implantant respectivement les espces `s2` et `s3`, dont ne nous donnons pas la définition ici. L'espèce `s1` hérite des espèces `s4` et `s5`.

```
species s1(c2 is s2, c3 is s3)
inherits s4, s5 =
```

On déclare alors une fonction foo qui prend un élément de self, un élément de la collection c2 implantant c2 et un élément de c3 implantant s3, et qui renvoie un élément de l'espèce s1.

```
sig foo in self -> s2 -> s3 -> self;
```

De la même façon, on définit une fonction foo2 du même type que foo, et qui est un appel direct à foo

```
let foo2(x in self, y in s2, z in s3) =
    self!foo(x, y, z);
```

La notation self!foo désigne la fonction foo de l'espèce en cours de définition.

On déclare alors une relation binaire d'égalité, et on ajoute la propriété que cette relation doit être réflexive. Nous ne mettons pas ici les propriétés de symétrie et de transitivité.

```
sig equal in self -> self -> bool;

property equal_reflexive :
  all x in self, !equal(x, x);
```

On définit alors la propriété énonçant que quels que soient les arguments passés en paramètre, le résultat renvoyé par foo est identique à celui de foo2

```
property foo_eq_foo2:
  all x in self,
  all y in s2, all z in s3,
    self!equal(self!foo(x, y, z),
               self!foo2(x, y, z));
end
```

On définit le type support de cette espèce comme étant les entiers machine, et on définit la méthode foo comme étant la fonction qui renvoie le premier argument.

```
rep = int;
let foo(x, y, z) = x;
```

L'espèce est totalement définie. On peut alors implanter l'espèce s1 par la collection coll1 (on suppose que les collections coll2 et coll3 implante respectivement les espèces s2 et s3) :

```
collection coll1
implements s1(coll2, coll3) =
```

7.1.2 Cutter : le langage de preuve

Focal permet non seulement de passer de spécifications à leur implantation mais également de prouver qu'une implantation donnée est conforme à sa spécification.

Focal posséde une traduction vers Coq [55] afin d'avoir l'environnement nécessaire pour faire les preuves en Coq. S'il est toujours possible de faire ses preuves pour Focal de cette manière, il existe maintenant un langage de preuve propre à Focal, nommé Cutter. Celui-ci est fortement inspiré des idées de Lamport [44] sur la façon d'écrire des preuves.

Cutter correspond au langage haut niveau de Zenon : c'est ici que se fait le lien entre le compilateur Focal et le prouveur automatique Zenon. À partir de Cutter, le compilateur Focal produit une entrée pour Zenon pour lui déléguer la recherche de preuves.

Syntaxe

La syntaxe Cutter correspond à la grammaire suivante :

```
<proof>  ::=  ['BY' {<ref>}*] ['DEF' {<meth>}*]
          |   [ <level> <label> <goal> <proof>]*
              <level> <label> 'QED' <proof>
          |   <empty>

<label>  ::=  num
          |   ident

<level>  ::=  '<' num '>'

<goal>   ::=  ['ASSUME' <hyp>*] 'PROVE' statement

<hyp>    ::=  ident ' :' type
          |   ident ' :' statement

<ref>    ::=  <level> <label>
          |   <level> ' :' ident

<meth>   ::=  '!' ident
          |   ident '!' ident
          |   '#' ident
          |   ident '#' ident
```

Nous n'avons pas décrit formellement les terminaux de cette syntaxe et quelques autres details.

- **num** identifie les nombres arbitraires et **ident** les identificateurs permis (toute chaîne de caractères ne commençant pas par un nombre).
- Les mots-clés de Cutter — ASSUME, PROVE, BY, DEF, QED — peuvent également être écrits en minuscules.
- Les niveaux (**level**) permettent de hiérarchiser la preuve.
- **Goal** permet d'introduire des nouveaux sous-buts à prouver.
- Les labels sont au choix des nombres ou des identificateurs.

Exemple

L'exemple suivant de preuve hiérarchique provient de la bibliothèque de Focal (fichier `sets_orders.foc`)

```
theorem order_sup_is_transitive : all x y z in self,
    !order_sup(x,y) -> !order_sup(y,z) -> !order_sup(x,z)
    (* si x=inf(x,y) et y=inf(y,z)
     alors x=inf(x,inf(y,z))=inf(inf(x,y),z)=inf(x,z)*)
    proof:
      <1>1 assume x y z in self
                  xLTy: !order_sup (x, y)
                  yLTz: !order_sup (y, z)
           prove !order_sup (x, z)
        <2>1 prove !equal (x, !sup (x, !sup (y, z)))
          by <1>:xLTy,
             <1>:yLTz,
             !sup_right_substitution_rule,
             !equal_transitive
          def !order_sup
        <2>2 prove !equal (x, !sup (!sup (x, y), z))
        by <2>1,
           !sup_is_associative,
           !equal_transitive,
           !equal_symmetric
        <2>fnord qed
        by <2>2,
           <1>:xLTy,
           !equal_symmetric,
           !equal_transitive,
           !sup_left_substitution_rule
           def !order_sup
       <1>2 qed
    ;
```

Tout d'abord, certaines hypothèses sont renommées au niveau `<1>1` où l'on énonce la propriété que l'on veut démontrer.

Ensuite, la preuve est découpée en sous-lemmes plus simples. Le premier sous-lemme `<2>1` prouve que $x = \sup(x, \sup(y, z))$ en utilisant

- les deux hypothèses (propriétés) du niveau supérieur `<1> :xLTy` (i.e. $x \leq y$) et `<1> :yLTz`,
- les propriétés de transitivité de l'égalité et de substitution par sup dans une égalité `sup_right_substitution_rule`
- et la définition de `order_sup` que l'on a dans l'environnement courant.

Dans cette partie de la bibliothèque, on a

$$\begin{aligned} \text{sup\_right\_substitution\_rule} &\equiv y = z \Rightarrow \sup(x, y) = \sup(x, z) \\ \text{order\_sup}(x, y) &\equiv x = \sup(x, y). \end{aligned}$$

`qed` permet de conclure un niveau de preuve.

7.2 Zenon

Le démonstrateur automatique de Focal est nommé d'après Zénon d'Élée, philosophe et mathématicien grec. Selon Aristote, il est le créateur de la dialectique mais il est surtout connu pour ses démonstrations paradoxales comme celles concernant la flèche et la cible (ou sa variante, Achille et la tortue).

Zenon est un prouveur basé sur la méthode des tableaux pour la logique classique du premier ordre avec égalité. Zenon ne demande pas à l'utilisateur de croire à la correction des preuves qu'il trouve : sa singularité principale est en effet de produire en sortie un terme preuve que Coq peut vérifier[1].

7.2.1 Principes généraux

Zenon intervient sur un fichier `.zv` produit par le compilateur Focal à partir d'un fichier source Focal correspondant. La preuve hiérarchique décrite dans le langage haut niveau Cutter est transformée par le compilateur Focal en une liste de lemmes à prouver (automatiquement). On associe à l'énoncé de chaque lemme les hypothèses (définitions et lemmes) nécessaires à sa démonstration. On prouve ces lemmes à l'aide des règles de MLproof (section 7.2.2). Les arbres de preuve des lemmes démontrés sont ensuit traduits en arbres de preuve LLproof(section 7.2.3). Enfin les preuves LLproof des lemmes sont transformées en preuves Coq.

7.2.2 MLproof : calcul intermédiaire

Le calcul intermédiaire MLproof (ML pour *mid-level*) est en fait le formalisme dans lequel Zenon fait ses preuves. Les règles applicables dans ce langage sont inspirées des tableaux pour la logique du premier ordre. Par ailleurs des règles spécifiques existent pour traiter non seulement l'égalité mais plus généralement des relations transitives, symétriques ou réflexives. Par conséquent le nombre de règles est plus élevé que dans le cas de base des tableaux. Ceci est accentué par le fait que la logique considérée n'est pas vraiment minimale : ainsi $\neg.\vee, \wedge, \Rightarrow, \Leftrightarrow$ sont tous des connecteurs logiques primitifs.

Il faut noter que les règles qui définissent Zenon sont sans exception *non destructives* : les formules en prémisses ne sont jamais oblitérées par l'application d'une inférence. Ces règles peuvent être classées en trois catégories :

- des règles analytiques qui s'appliquent de façon calculatoire en fonction de la structure syntaxique des propositions à prouver ;
- des règles relationnelles qui s'appliquent à des prédicats binaires représentant des relations pouvant être (de façon non exclusive) transitives, réflexives ou symétriques ;
- des règles supplémentaires permettant de fermer, de déplier des définitions ou d'instancier partiellement les formules de la preuve.

[1] La tortue va donc avoir du mal à prouver qu'elle peut gagner la course contre Achille.

Nous allons donc passer en revue ces différentes classes de règles.

Règles analytiques

Les règles purement analytiques correspondent aux catégories α, β et δ usuellement utilisées dans les méthodes de tableaux. Les deux premières types de règles décomposent les connecteurs logiques du langage et le troisième élimine le quantificateur existentiel. Elles sont formalisées en figure 7.2. Notons que l'élimination du quantificateur existentiel est faite par une ϵ-règle (cf. section 1.3.4).

$$\frac{\neg\neg P}{P}\,\neg\neg \qquad \frac{P \Leftrightarrow Q}{\neg P, \neg Q \mid P, Q}\,\Leftrightarrow \qquad \frac{\neg(P \Leftrightarrow Q)}{\neg P, Q \mid P, \neg Q}\,\neg\Leftrightarrow$$

$$\frac{P \wedge Q}{P, Q}\,\wedge \qquad \frac{\neg(P \vee Q)}{\neg P, \neg Q}\,\neg\vee \qquad \frac{\neg(P \Rightarrow Q)}{P, \neg Q}\,\Rightarrow$$

$$\frac{P \vee Q}{P \mid Q}\,\vee \qquad \frac{\neg(P \wedge Q)}{\neg P \mid \neg Q}\,\vee \qquad \frac{P \Rightarrow Q}{\neg P \mid Q}\,\vee$$

$$\frac{\exists x\ P(x)}{P(\epsilon(x).P(x))}\,\delta_{\exists} \qquad \frac{\neg\forall x\ P(x)}{\neg P(\epsilon(x).P(x))}\,\delta_{\neg\forall}$$

$$\frac{P_1 \wedge P_2 \wedge ... \wedge P_n}{P_1, P_2, ..., P_n}\,\wedge^*$$

$$\frac{P_1 \vee P_2 \vee ... \vee P_n}{P_1 \mid P_2 \mid ... \mid P_n}\,\vee^*$$

FIG. 7.2 – Règles analytiques de Zenon

Les règles de décomposition associées aux connecteurs $\wedge$ et $\vee$ sont étendues aux propositions syntaxiquement équivalentes à des conjonctions et disjonctions arbitrairement longues (ainsi $P \Rightarrow (Q \vee R)$ est traitée comme une disjonction). Cette extension permet de choisir plus judicieusement la branche dans laquelle continuer le raisonnement car on a ainsi plus d'informations sur les choix possibles.

Règles relationnelles

Ces règles comportent à la fois les inférences produisant des égalités (négatives et positives) et celles qui ont pour prémisses des relations identifiées comme transitives, réflexives ou symétriques.

Les deux premières règles sont parmi celles que S.Reeves utilise dans son traitement de l'égalité (voir section 2.1) : elles permettent de décomposer les problèmes de fermeture (non immédiate) liée aux propositions et fonctions en sous-buts plus simples à résoudre.

$$\frac{P(t_1,...,t_n) \qquad \neg P(s_1,..,s_n)}{t_1 \neq s_1 \mid ... \mid t_n \neq s_n} \qquad \frac{f(t_1,...,t_n) \neq f(s_1,...,s_n)}{t_1 \neq s_1 \mid ... \mid t_n \neq s_n} \neq_{\text{func}}$$

$$\frac{R_s(s,t) \qquad \neg R_s(u,v)}{t \neq u \mid s \neq v} \text{ sym} \qquad \frac{\neg R_r(s,t)}{s \neq t} \neg_{\text{refl}}$$

$$\frac{R_t(s,t) \qquad \neg R_t(u,v)}{u \neq s, \neg R_t(u,s) \mid t \neq v, \neg R_t(t,v)} \text{ trans}$$

$$\frac{R_{ts}(s,t) \qquad \neg R_{ts}(u,v)}{v \neq s, \neg R_{ts}(v,s) \mid t \neq u, \neg R_{ts}(t,u)} \text{ transsym}$$

$$\frac{s \equiv t \qquad R_t(u,v)}{u \neq s, \neg R(u,s) \mid \neg R(u,s), \neg R(t,v) \mid t \neq v, \neg R(t,v)} \text{ transeq}$$

$$\frac{s \equiv t \qquad R_t(u,v)}{v \neq s, \neg R(v,s) \mid \neg R(v,s), \neg R(t,u) \mid t \neq u, \neg R(t,u)} \text{ transeqsym}$$

FIG. 7.3 – Règles relationnelles de Zenon

Ensuite la règle de simplification pour les propositions est spécialisée sur les relations en fonction de leurs propriétés : réflexive, symétrique, transitive ou transitive et symétrique.

Les règles (transeq) et (transeqsym) permettent de tenir compte du fait que l'on rencontre des égalités dans les branches du tableau et de décomposer le problème.

Règles supplémentaires : fermeture, instanciation, dépliage

Les règles restantes sont regroupées ici. Elles peuvent être classées dans trois catégories différentes :

- des règles de fermeture pour fermer les branches du tableau (figure 7.4) ;
- des règles d'instanciation : cet ensemble comporte des variations sur les γ-règles des méthodes de tableau (figure 7.5).
- des règles de dépliage pour remplacer un symbole par sa définition (figure 7.6).

On dénombre cinq variantes pour fermer le tableau :

- $\odot_\bot$ et $\odot_{\neg\top}$ permettent de traiter des contradictions syntaxiques propres à la formule car $\bot$ n'est conséquence d'aucune règle de Zenon ;
- $\odot$ est la fermeture binaire classique des tableaux et s'applique à des formules de taille arbitraire (pas seulement des atomes) ;
- $\odot_r$ permet de fermer plus rapidement dans le cas de relation réflexive : cette règle pourrait être remplacée par une règle d'expansion « gratuite » (i.e. avec un ensemble vide de prémisses) permettant d'ajouter à une branche la formule $R_r(t,t)$ pour n'importe quelle relation réflexive et n'im-

$$\frac{\bot}{\odot}\,\odot_{\bot} \qquad \frac{\neg\top}{\odot}\,\odot_{\neg\top}$$

$$\frac{P \qquad \neg P}{\odot}\,\odot \qquad \frac{\neg R_r(t,t)}{\odot}\,\odot_r$$

$$\frac{R_s(a,b) \qquad \neg R_s(b,a)}{\odot}\,\odot_s$$

FIG. 7.4 – Fermeture pour Zenon

porte quel terme du langage ;

– la fermeture pour les relations symétriques $\odot_s$ est une fermeture binaire dans laquelle est codée implicitement la symétrie de la relation. C'est une règle dérivée qui peut se réécrire

$$\frac{R_s(a,b) \qquad \dfrac{\neg R_s(b,a)}{\neg R_s(a,b)}\,\text{symétrie}}{\odot}\,\odot$$

$$\frac{\forall x\ P(x)}{P(X)}\,\gamma_{\forall M} \qquad \frac{\neg\exists x\ P(x)}{\neg P(X)}\,\gamma_{\neg\exists M}$$

$$\frac{\forall x\ P(x)}{P(t)}\,\gamma_{\forall \text{inst}} \qquad \frac{\neg\exists x\ P(x)}{\neg P(t)}\,\gamma_{\neg\exists \text{inst}}$$

$$\frac{\forall x\ P(x)}{\forall x_1 ... x_n\ P(s(x_1,...,x_n))}\,\gamma_{\forall un} \qquad \frac{\neg\exists x\ P(x)}{\neg\exists x_1 ... x_n\ P(s(x_1,...,x_n))}\,\gamma_{\neg\exists un}$$

FIG. 7.5 – γ-règles

Les règles sur les formules universellement quantifiées peuvent générer ainsi à la fois des métavariables et des instanciations . Lorsque l'on est forcé de développer une γ-formule pour la première fois, on va utiliser une expansion avec métavariable en attendant d'avoir suffisamment d'informations pour l'instancier. Cependant, au lieu d'instancier directement cette formule et de propager le résultat dans la preuve de tableau courante comme le fait Fitting, on réappliquera une γ-expansion à la formule universelle originelle instanciant sa variable. Le résultat produit est exactement le même mais on évite ainsi de devoir utiliser plusieurs métavariables pour la même formule et de procéder par « iterative deepening ».

Par ailleurs, une troisième variante existe afin de résoudre les difficultés inhérentes aux problèmes d'unification de type $X \stackrel{?}{=} f(X)$. Si l'on doit instancier X par $f(X)$ dans P alors on va choisir de réappliquer d'abord $\gamma_{\forall un}$ à $\forall x\ P(x)$ où le symbole de fonction s introduit sera effectivement le f rencontré dans le problème d'unification.

Si $P(x)\hat{=}\mathrm{Def}(x)$ et $f(x)\hat{=}\mathrm{def}(x)$ alors :

$$\frac{P(x)}{\mathrm{Def}(x)}\ \text{p-unfold} \qquad \frac{\neg P(x)}{\neg\mathrm{Def}(x)}\ \text{p-unfold}_\neg$$

$$\frac{f(x)=t}{\mathrm{def}(x)=t}\ \text{f-unfold}_{l=} \qquad \frac{t=f(x)}{t=\mathrm{def}(x)}\ \text{f-unfold}_{r=}$$

$$\frac{f(x)\neq t}{\mathrm{def}(x)\neq t}\ \text{f-unfold}_{l} \qquad \frac{t\neq f(x)}{t\neq\mathrm{def}(x)}\ \text{f-unfold}_{r}$$

FIG. 7.6 – Dépliage de définition

Enfin, on peut avoir à déplier des définitions de formules ou de fonctions dans les preuves de Zenon : cela est géré par les règles de dépliage (figure 7.6).

Extensions

Les règles d'inférence de MLproof forment un noyau cohérent d'inférences pour la recherche de preuve. Zenon a un mécanisme d'extension qui permet de rajouter des règles dans le raisonnement.

Certaines des règles du noyau de Zenon pourraient ainsi être exprimées sous forme d'extensions. Ainsi, on pourrait avoir un noyau de logique minimale (avec juste $\Rightarrow$, $\bot$ et $\forall$ par exemple) et fournir les autres connecteurs sous forme d'extension.

Tri d'équivalences Cette extension répond à l'optimisation de la résolution d'un problème de TPTP [53]. Il s'agit de trier une suite d'équivalences. Ainsi la proposition

$$(A \Leftrightarrow B) \Leftrightarrow ((C \Leftrightarrow A) \Leftrightarrow C) \Leftrightarrow ((D \Leftrightarrow E) \Leftrightarrow F)$$

sera transformée en $B \Leftrightarrow (D \Leftrightarrow (E \Leftrightarrow (F \Leftrightarrow \top)))$.

Coqbool Les fonctions de Focal dont le résultat est un booléen sont traduites vers Coq en les encapsulant dans le prédicat `Is_true`. Ainsi la fonction `P(...)` sera traduite vers le prédicat `Is_true(P...)`.

Ce changement ne permet pas d'utiliser dans le recherche de preuve de Zenon le fait par exemple que P soit une fonction symétrique. Ces méthodes sont donc transformées en prédicat `Is_true_P` ayant les même propriétés.

Élagage et minimisation de la taille de l'arbre de preuve

L'élagage de l'arbre de preuve permet d'éviter de générer certaines parties du processus de déduction que l'on reconnaît comme inutiles dans la recherche de preuve. Les preuves peuvent en effet être raccourcies en observant les propositions réellement utilisées dans la preuve pour

– enlever directement des branches inutiles (élaguer) ;
– ne pas ajouter de branches inutilement ;
– ne pas dupliquer les preuves à faire.

Ces procédés, que nous allons décrire, sont employés par Zenon pour fournir des preuves de manière plus efficace.

Dans le premier cas, supposons que le tableau courant soit de la forme

$$\frac{\Gamma}{\mathcal{B}_1 \mid B_2}$$

et que l'on arrive à clore la branche $\Gamma \cup \mathcal{B}_1$. On obtient une réfutation $\mathcal{P}$ de cette branche de la forme $H \vdash \bot$, où H est l'ensemble d'hypothèses effectivement utilisées pour obtenir la preuve $\mathcal{P}$. On sait que $H \subseteq \Gamma \cup \mathcal{B}_1$. Cependant si $H \cap \mathcal{B}_1 = \emptyset$, alors $H \subseteq \Gamma$: aucune proposition spécifique à la branche $\mathcal{B}_1$ n'a été utilisée pour clore celle-ci et l'on peut donc « remonter » la preuve au niveau du nœud de séparation de l'arbre de preuve, c'est-à-dire éliminer $\mathcal{B}_2$ mais aussi $\mathcal{B}_1$ et seulement garder Γ pour obtenir la branche close

$$\frac{\Gamma}{\odot} \text{ par la preuve } \mathcal{P}$$

On a un autre cas de simplification de l'arbre de preuve lorsque l'on utilise des ensembles de propositions pour représenter les branches. En effet cela suppose de n'avoir qu'un seul représentant effectif pour une proposition. Par conséquent, supposons qu'on arrive à fermer l'arbre

$$\frac{\Gamma}{P \mid Q}$$

avec $P \in \Gamma$. Cet arbre est équivalent

$$\frac{\Gamma}{\mid Q}$$

que l'on sait fermer par hypothèse. L'expansion produisant P et Q est donc inutile.

Enfin, Zenon réutilise les preuves déjà faites dans l'arbre courant lorsqu'un ensemble d'hypothèse en « subsume » un autre. Ainsi, si une branche a été close en obtenant la réfutation $\Delta \vdash \bot$ (et la preuve $\mathcal{P}$) et que l'on est actuellement dans la branche Γ, alors si $\Delta \subseteq \Gamma$, on utilise directement la preuve $\mathcal{P}$ déjà faite pour fermer Γ.

7.2.3 LLproof : calcul bas-niveau

Lorsqu'une preuve est trouvée dans MLproof, Zenon traduit celle-ci vers une preuve de (plus) bas niveau LLproof(LL pour *low-level*). Les règles que l'on peut y appliquer sont très proches de celles du calcul des séquents classique du premier ordre. LLproof est un calcul des séquents unilatéral (*one-sided sequents*) correspondant aux tableaux non-destructifs. La règle de contraction usuelle est incorporée dans les inférences.

$$\frac{}{\bot \vdash \bot} \qquad \frac{}{\neg\top \vdash \bot} \qquad \frac{}{\Gamma, P, \neg P \vdash \bot}$$

$$\frac{}{t \neq t \vdash \bot} \qquad \frac{\Gamma, P \vdash \bot \qquad \Gamma, \neg P \vdash \bot}{\Gamma \vdash \bot} \qquad \frac{\Gamma, P, \neg\neg P \vdash \bot}{\Gamma, P \vdash \bot}$$

FIG. 7.7 – LLproof : axiomes, coupure, double négation

$$\frac{\Gamma, P \wedge Q, P, Q \vdash \bot}{\Gamma, P \wedge Q \vdash \bot}$$

$$\frac{\Gamma, P \vee Q, P \vdash \bot \qquad \Gamma, P \vee Q, Q \vdash \bot}{\Gamma, P \vee Q \vdash \bot}$$

$$\frac{\Gamma, \neg P, P \Rightarrow Q \vdash \bot \qquad \Gamma, Q, P \Rightarrow Q \vdash \bot}{\Gamma, P \Rightarrow Q \vdash \bot}$$

$$\frac{\Gamma, P \Leftrightarrow Q, \neg P, \neg Q \vdash \bot \qquad \Gamma, P \Leftrightarrow Q, P, Q \vdash \bot}{\Gamma, P \Leftrightarrow Q \vdash \bot}$$

$$\frac{\Gamma, \neg P, \neg(P \wedge Q) \vdash \bot \qquad \Gamma, \neg Q, \neg(P \wedge Q) \vdash \bot}{\Gamma, \neg(P \wedge Q) \vdash \bot}$$

$$\frac{\Gamma, \neg P, \neg Q, \neg(P \vee Q) \vdash \bot}{\Gamma, \neg(P \vee Q) \vdash \bot}$$

$$\frac{\Gamma, P, \neg Q, \neg(P \Rightarrow Q) \vdash \bot}{\Gamma, \neg(P \Rightarrow Q) \vdash \bot}$$

$$\frac{\Gamma, \neg P, Q, \neg(P \Leftrightarrow Q) \vdash \bot \qquad \Gamma, P, \neg Q, \neg(P \Leftrightarrow Q) \vdash \bot}{\Gamma, \neg(P \Leftrightarrow Q) \vdash \bot}$$

FIG. 7.8 – LLproof : noyau propositionnel

$$\frac{\Gamma, P(c), \exists x\ P(x) \vdash \bot}{\Gamma, \exists x\ P(x) \vdash \bot} \qquad \frac{\Gamma, \neg P(c), \neg\forall x\ P(x) \vdash \bot}{\Gamma, \neg\forall x\ P(x) \vdash \bot}$$

où c est une constante fraîche

$$\frac{\Gamma, P(t), \forall x\ P(x) \vdash \bot}{\Gamma, \forall x\ P(x) \vdash \bot} \qquad \frac{\Gamma, \neg P(t), \neg\exists x\ P(x) \vdash \bot}{\Gamma, \neg\exists x\ P(x) \vdash \bot}$$

où t est un terme quelconque

FIG. 7.9 – LLproof :Quantificateurs

Les règles restantes (figure 7.10) sont aussi les plus avancées. Elles permettent de traduire l'utilisation de l'égalité, des dépliages, des extensions et des lemmes.

$$\frac{\Delta, t_1 \neq u_1 \vdash \bot \qquad ... \qquad \Delta, t_n \neq u_n \vdash \bot}{\Gamma, P(t_1, ..., t_n), \neg P(u_1, ..., u_n) \vdash \bot}\ \mathrm{P}$$

en posant $\Delta = \Gamma \cup \{P(t_1, ..., t_n), \neg P(u_1, ..., u_n)\}$

$$\frac{\Delta, t_1 \neq u_1 \vdash \bot \qquad ... \qquad \Delta, t_n \neq u_n \vdash \bot}{\Gamma, f(t_1, ..., t_n) \neq f(u_1, ..., u_n) \vdash \bot}\ (\mathrm{fun})$$

en posant $\Delta = \Gamma \cup \{f(t_1, ..., t_n) \neq f(u_1, ..., u_n)\}$

$$\frac{\Gamma, C, H \vdash \bot}{\Gamma, C \vdash \bot}\ \mathrm{def}(\mathsf{nom}, C, H)$$

si on peut passer de C à H en dépliant la définition de nom.

$$\frac{\Delta, H_{11}, .., H_{1m} \vdash \bot \qquad ... \qquad \Delta, H_{n1}, ..., H_{nq} \vdash \bot}{\Gamma, C_1, .., C_p \vdash \bot}\ \mathrm{ext}(\mathsf{nom}, \mathsf{args}, [C_i], [H_{1j}, ..., H_{nk}])$$

où $\Delta = \Gamma \cup \{C_1, ..., C_p\}$
nom est le nom d'un lemme prédéfini tel que
$C_1 \wedge .. \wedge C_p \Rightarrow \bigvee_j (\bigwedge_i H_i j)$

$$\frac{}{\Gamma, C \vdash \bot}\ \mathrm{lem}(\mathsf{nom}, \mathsf{args})$$

si C est la conclusion associé à nom dans la liste de preuves.
Les arguments args correspondent aux paramètres de nom.

FIG. 7.10 – LLproof : dépliage, égalité, extension, lemme

La règle (ext) est en fait une macro correspondante aux règles définies par les extensions. La règle (lem) implante un partage de preuves dans Zenon : les preuves ne sont pas des arbres mais des DAG.

7.2.4 De MLproof vers LLproof

Nous ne détaillerons pas toute la traduction de MLproof vers LLproof. La plupart des règles de MLproof correspondent en effet directement à une inférence de LLproof. Cependant, prenons par exemple, la règle MLproof (sym) de la figure 7.3

$$\frac{R_s(s,t) \qquad \neg R_s(u,v)}{t \neq u \mid s \neq v}\ \mathrm{sym}$$

Celle-ci est traduite en preuve LLproof :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{R_s(u,v), \neg R_s(u,v) \vdash \bot}\,\text{ax} \quad \dfrac{}{R_s(v,u), \neg R_s(v,u) \vdash \bot}\,\text{ax}}{R_s(v,u), \neg R_s(u,v), R_s(v,u) \Rightarrow R_s(u,v) \vdash \bot}\,\Rightarrow \quad \dfrac{\dfrac{\pi_1}{s \neq v \vdash \bot} \quad \dfrac{\pi_2}{t \neq u \vdash \bot}}{R_s(s,t), \neg R_s(v,u) \vdash \bot}}{R_s(v,u), \neg R_s(u,v), \forall xy(R(x,y) \Rightarrow R(y,x)) \vdash \bot}\,\forall, \forall}{R_s(s,t), \neg R_s(u,v), \forall xy(R(x,y) \Rightarrow R(y,x)) \vdash \bot}\,\text{cut}$$

où seules les formules actives dans le raisonnement sont représentées (les ensembles de formules que l'on a dans le calcul des séquents LLproof ne sont donc pas indiqués).

On constate que la preuve n'est possible qu'avec l'hypothèse de symétrie $\forall xy(R(x,y) \Rightarrow R(y,x))$.

Le reste de la traduction sur les règles « absentes » de LLproof est du même ressort. Nous ne détaillerons pas les autres cas.

Lemme 35 (Correction et complétude)**.** *Toute preuve* LLproof *a une preuve* MLproof *correspondante et réciproquement.*

Démonstration. Passer de LLproof à MLproof est immédiat : toute règle de LLproof à une règle équivalente dans MLproof.

Dans le sens contraire, on utilise la traduction que l'on a ébauchée ici. Un équivalent OCaml de cette preuve est disponible dans le fichier `llproof.ml` des sources de Zenon. □

7.2.5 De LLproof vers Coq

À partir de LLproof, la traduction vers un script Coq est quasiment immédiate. Cela permet d'utiliser la vérification de la preuve par Coq et ainsi d'ajouter un niveau de confiance dans la correction pratique des preuves effectuées par Zenon.

La correction des preuves de LLproof par rapport au calcul des séquents intuitionnistes avec égalité $\mathsf{LJ}_{\mathsf{eq}}$, proche de la théorie de Coqa été démontrée par D.Doligez et D.Delahaye [24].

En pratique, chaque règle d'inférence de LLproof est définie comme un lemme de Coq. Cela permet d'avoir une traduction homogène pour les règles et les extensions en utilisant systématiquement la tactique apply.

7.3 La déduction modulo dans Zenon

Nous allons voir de quelle manière Zenon contient déjà une certaine forme de déduction modulo :

– par les règles de dépliage ;
– par les extensions ;

7.3.1 Dépliage

Le dépliage de définitions dans Zenon peut être vu comme une forme extrêmement simplifiée de déduction modulo. Prenons la règle générale

$$\frac{\Gamma, C, H \vdash \bot}{\Gamma, C \vdash \bot}\ \text{def}(\mathsf{nom}, C, H)$$

si on peut passer de C à H en dépliant la définition de nom.

C'est une méta-règle qui doit être comprise comme un ensemble de règles associées chacun à un couple (nom,conclusions) suivant les définitions de l'environnement.

Chacune de ces règles est en effet l'équivalent d'une règle de réécriture (propositionnelle) :

$$H \to C$$

que l'on aurait ajoutée à Zenon tout en se limitant à une stratégie de réécriture de tête (on ne réécrit pas les sous-formules d'une formule).

7.3.2 Extensions

Les extensions de Zenon permettent d'utiliser des règles d'inférences supplémentaires dans le raisonnement.

Ainsi les extensions décrites précédémment peuvent être vues comme des applications de la déduction modulo. Le mécanisme d'extension peut décrire n'importe quelle règle d'inférence, dont les conditions d'application peuvent être arbitrairement.compliquées. Il est plus général que les seules règles de réécriture propositionnelles. Cependant, il est vraisemblablement plus coûteux.

Actuellement ces règles ressemblent plutôt à des règles de dépliage :

$$\frac{\Gamma \vdash \mathsf{ls\_true\_g\_or\_b}(x, y)}{\Gamma \vdash x \vee y} \qquad \frac{\Gamma \vdash \mathsf{ls\_true\_g\_and\_b}(x, y)}{\Gamma \vdash xy}$$

Les règles de réécriture correspondantes peuvent s'écrire

$$\begin{aligned} \mathsf{ls\_true\_g\_or\_b}(x, y) &\to x \vee y \\ \mathsf{ls\_true\_g\_and\_b}(x, y) &\to x \wedge y \end{aligned}$$

Le cas de Coqbool est assez intéressant car cette extension est utilisée fréquemment. Il semble même dans ce cas que normaliser par le système de réécriture correspondant soit la solution la plus appropriée. Ce n'est pas possible avec la présentation sous règles d'inférence pour un calcul de séquents : les sous-formules ne peuvent être réécrites.

7.4 Conclusion

Nous avons vu dans ce chapitre la partie théorique de l'implantation du prouveur automatique Zenon du projet Focal. Fondé sur les méthodes de tableau, Zenon permet d'ores et déjà d'incorporer une version minimale de la

déduction modulo sous forme d'extensions (et de dépliages de définitions). L'extension `Coqbool` est déjà activement utilisée de manière satisfaisante. Cependant l'extension de Zenon vers un « véritable » prouveur modulo, devrait permettre d'améliorer les performances.

Conclusion et perspectives

Cette thèse traite essentiellement de la méthode des tableaux, méthode très reconnue en démonstration automatique, quelle que soit la logique sous-jacente. Dans les chapitres 1 et 2, nous avons présenté ces méthodes de tableaux, d'un point de vue syntaxique et d'un point de vue sémantique, d'abord les tableaux pour la logique propositionnelle, ensuite ceux associés à la logique du premier ordre, puis ceux liés à la logique intuitionniste, enfin ceux correspondant à la théorie de l'égalité. Nous avons détaillé les différents traitements envisagés pour la skolémisation, tentant de montrer ainsi comment l'implantation d'une méthode conduit à étudier beaucoup plus finement un mécanisme déjà bien compris sur le plan théorique. La même remarque s'applique pour le traitement de l'égalité, relativement simple tant que l'on reste au niveau étude de théories, mais qui se complique singulièrement dès qu'on envisage une implantation efficace. Nous n'avons pas dans ces deux premiers chapitres fait figurer des démonstrations des résultats énoncés, qui peuvent être trouvés dans les articles référencés. En revanche, nous avons cherché à offrir un panorama assez complet, espérant montrer ainsi la richesse mais aussi la complexité de ces méthodes de tableaux.

Le chapitre 3 présente le calcul des séquents modulo, telle qu'il a été introduit dans [30]. Cette présentation de la logique du premier ordre est fondée sur la réécriture de formules atomiques en des formules "plus précises" permettant ainsi de "progresser"plus directement dans les preuves. En effet, une formule et son réduit sont équivalentes et l'arsenal du calcul des séquents par exemple permet de déplier cette équivalence par construction de nouvelles étapes dans l'arbre de preuves. Dans la déduction modulo, l'étape de réécriture est masquée, le remplacement de la formule par son réduit étant incorporé dans une seule règle du calcul des séquents modulo. Nous mentionnons également quelques résultats récents dans le domaine de la déduction modulo, de manière très brève et incomplète. Ce domaine de recherche est actuellement en plein essor, comme le témoigne le nombre d'articles soumis ou acceptés en 2006.

À partir du chapitre 4, nous introduisons nos propres résultats, certains ayant été obtenus en collaboration avec O. Hermant[11, 12] ou résultant d'une discussion avec D. Doligez sur son outil de démonstration automatique Zenon.

Le chapitre 4 est dédié aux tableaux analytiques modulo. Dans leur article[30], les auteurs avaient introduit la résolution modulo, définie par deux règles recouvrant les différentes instances de la résoultion, selon la logique sous-jacente. Dans

ce chapitre, nous reprenons la technique générale utilisée pour l'adapter au cas des tableaux. La principale difficulté vient ici de l'utilisation de méta-variables, dont l'utilisation demande une technique d'*iterative deepening* pour dégager une notion de forme normale et de la gestion rigide de ces métavariables, qui apparaît à la fois dans la règle d'instanciation du système intermédiaire et dans la gestion globale des contraintes d'unification. Cependant, nous avons considéré les techniques de gestion de ces contraintes comme dépassant le sujet de la thèse. Nous les réservons à une étude ultérieure. Ce chapitre contient les preuves détaillées de la correction et de la complétude de la méthode de tableaux modulo, TaMed, que nous proposons. Il a été présenté dans [10].

Dans le chapitre 5, nous abordons les tableaux modulo d'un point de vue sémantique, en utilisant les techniques introduites par O. Hermant, co-auteur de la publication[11] relative à ce chapitre. Nous reformulons les règles de TaMed en nous limitant à un système de réécriture convergent. La correction repose sur la construction (facile) d'un modèle booléen de la déduction modulo. La complétude demande la construction d'une valuation modèle et l'introduction de la notion de $\mathcal{R}$-modèle, qui conduit à dégager certaines classes de systèmes de réécriture pour lesquels la complétude de la méthode est assurée.

Les tableaux intuitionistes sont étudiés dans le chapitre 6. Ce travail repose sur une collaboration avec O. Hermant[12]. Nous prouvons la correction et la complétude des tableaux intuitionnistes modulo, en utilisant une méthode originale, purement constructive. La preuve de complétude est ainsi faite de manière sémantique en modifiant les structures de Kripke usuelles de manière â rester constructif. La preuve de correction est une preuve syntaxique de correspondance les preuves des tableaux intuitionnistes modulo et celles du calcul des séquents modulo sans coupure : ce résultat est nouveau même dans le cas sans réécriture. L'addition de ces deux preuves (constructives) à la preuve sémantique de complétude du calcul des séquents modulo donne un théorème constructif d'élimination des coupures, dont on peut extraire un contenu calculatoire.

Le chapitre 7 est une présentation du système Zenon développé par D. Doligez. Nous décrivons très brièvement le langage pour lequel ce système a été intialement développé. Puis, nous donnons les règles utilisées dans Zenon en nous référant aux discussions effectuées dans les chapitres antérieurs. Nous montrons comment ces règles s'inscrivent naturellement dans le cadre des tableaux modulo. On peut donc considérer que Zenon contient déjà les prémisses d'une véritable implantation de la méthode des tableaux modulo, la méthode des tableaux apportant la stratégie de parcours d'une formule jusqu'à une occurrence d'un littéral, la réécriture permettant de remplacer ce littéral par une formule plus développée, permettant de continuer le processus de recherche de preuve. Bien sûr, des stratégies plus complexes et sans doute plus performantes peuvent être envisagées.

Nous allons examiner maintenant quelles sont les prolongations possibles de ce travail.

Zenon contient une version assez simple de la déduction modulo proche de

la déduction surnaturelle [63] sous la forme d'un mécanisme d'extension. Deux points distinguent cette implantation d'une déduction modulo « véritable » :

- Les règles d'inférence doivent être souvent appliquées dans les extensions existantes. Dans les cas traités actuellement, il serait intéressant de garder des propositions en forme normale, ce qui est relativement peu coûteux dans le cas de ces systèmes. Le traitement de Coqbool par des règles de réécriture par exemple semblerait judicieux.
- Ce n'est pas vraiment de la réécriture. En particulier on ne peut pas appliquer les règles à des sous-formules. Seule une stratégie réécrivant en tête est possible.

Les mécanismes d'extension de Zenon fournissent par ailleurs un bon point de comparaison pour tester les performances pratiques de la déduction modulo. Rappelons que les extensions sont déjà utilisées de manière satisfaisante.

L'évaluation pratique de la déduction modulo passe aussi par l'implantation de théories plus riches. Il existe par exemple une formalisation de certaines méthodes de récurrence en déduction modulo [25]. Les expressions de la logique d'ordre supérieur en tant que théorie modulo [29] semblent particulièrement intéressantes. C'est vraisemblablement l'une des plus jolies applications de la déduction modulo.

Il existe déjà des tableaux d'ordre supérieur [42] dont nous n'avons par parlé par manque de place : leurs règles structurelles sont similaires à celles du premier ordre mais la majorité des règles sont dédiées au processus d'unification qui est intégrée au processus d'expansion.

À l'aide des tableaux modulo et de HOL-$\lambda\sigma$ [29], on peut sans doute disposer d'une méthode de tableaux pour l'ordre supérieur. Nous réservons cette étude à des travaux ultérieurs.

Il restera encore à éclaircir les rapports entre les tableaux modulo HOL-$\lambda\sigma$ et le démonstrateur automatique de Kohlhase, notamment au niveau de l'unification.

Il est certain que cette méthode ne correspondra pas tout-à-fait aux tableaux de Kohlhase [42]. Ce dernier a pour objectif des tableaux extensionnels alors qu'HOL-$\lambda\sigma$ cherche à être intentionnellement équivalent au λ-calcul simplement typé.

Revenons aux implantations, et plus particulièrement à l'utilisation de la réécriture. Focal possède déjà une liaison vers CiMe pour tenter à la demande de prouver certaines propriétés. CiMe [54] est un outil puissant permettant entre autres de faire de la réécriture sur les termes avec des symboles associatifs-commutatifs, de la complétion Knuth-Bendix, de prouver la terminaison de systèmes de réécritures, de vérifier la confluence de ceux-ci...

L'idée ici serait d'utiliser CiMe comme outil complémentaire de la déduction modulo. Plusieurs utilisations semblent envisageables :

- Vérifier que le système respecte une des conditions d'ordre, de positivité (ou mélange des deux) pour garantir une certaine notion de complétude ;
- Vérifier que les systèmes de réécriture terminent ou confluent ;

– Compléter des systèmes de réécriture donnés.

La formulation théorique des tableaux modulo peut encore être améliorée pour rapprocher la théorie de l'implantation. C'est en particulier au niveau de la skolemisation qu'on peut espérer obtenir des résultats assez vite. L'introduction d'une règle de type δ^+ [9, 38] ne devrait normalement pas être problématique. En effet, les preuves sémantiques n'ont pas besoin d'être adaptées pour tenir compte de changement. Ces règles δ^+ paraissent être par ailleurs un bon compromis entre un gain effectif au niveau de la longueur des preuves et une théorie relativement simple à comprendre.

Le cas de la règle δ^ϵ est plus compliquée. Elle est intéressante car c'est celle qui est utilisée par Zenon : elle permettrait donc de rapprocher la théorie des tableaux modulo de cette implantation. Mais le symbole ϵ ne fait pas partie du langage du premier ordre. L'ajout de δ^ϵ nécessite une redéfinition des modèles utilisés.

Plusieurs manières d'étendre la déduction modulo semblent envisageables

– Au niveau des règles de réécriture, Deplagne arrive à relâcher en quelque sorte la condition d'atomicité à gauche des règles de réécriture propositionnelle. D'un point de vue pratique, il peut s'avérer intéressant de trouver des conditions dans lesquelles on est sûr de pouvoir faire cela, un peu comme les conditions d'ordre et de positivité sont utilisées dans les preuves sémantiques.
– Sur un plan purement théorique, et notamment en rapport avec les tableaux, la question se pose de savoir si la déduction modulo peut s'étendre facilement à d'autres logiques. Il faut aussi déterminer dans quelle mesure ces extensions sont intéressantes pour les logiques considérées.

Dans l'optique d'adapter la déduction modulo à d'autres logiques, un formalisme semble particulièrement intéressant : les inférences profondes.

Les inférences profondes, introduite par Guglielmi [36], se rapprochent par certains côtés de la déduction modulo. L'un des slogans du domaine, emprunté à Girard, est de « se débarasser de la bureaucratie » (des autres formalismes).

L'idée sous-jacente aux inférences profondes peut être décrite informellement de la manière suivante : on raisonne sur un multi-ensemble de formules (dont les connecteurs sont associatifs et commutatifs) et le raisonnement peut se faire aussi bien sur la tête de la formule que sur ses sous-formules.

Le cadre le plus simple pour représenter ces inférences profondes est le calcul des structures. Il est étudié dans le cas classique dans [14]. Dans ce calcul, les règles d'inférence se comportent comme des règles de réécriture. La volonté de se débarrasser de la bureaucratie a amené en quelque sorte à unifier les différents niveaux de syntaxe des séquents.

Dans le cas propositionnel par exemple, il existe seulement deux connecteurs dans la syntaxe du calcul des structures classique : $(\cdot)$ et $[\cdot]$. Le premier est une conjonction et le second une disjonction. À la différence des connecteurs traditionnels ils sont à arité variable Ainsi $[A, B] \hat{=} A \wedge B$ et $[A, B, C] \hat{=} A \wedge B \wedge C$.

(A, B) correspond quant à lui aussi bien à $(\vdash A \quad \vdash B)$ (les deux séquents prémisses de $\vee$-r dans les séquents) que $\vdash A \vee B$ (la conclusion de $\vee$-r) : c'est ici que se fondent les différents niveaux de syntaxes du calcul des séquents.

En dehors des règles correspondant directement aux règles traditionnelles de contraction, coupure, axiome ou affaiblissement, il n'y a qu'une seule règle d'inférence :

$$\frac{S\{([R, U], T)\}}{S\{[R, T], U\}}$$

où S repreśente le contexte dans lequel s'applique la règle.

En fait cette règle pourrait se traduire en déduction modulo (étendue) par une règle équivalente qui serait

$$(R \wedge U) \vee T \rightarrow (R \vee T) \wedge U$$

Notons que $(R \wedge U) \vee T \Rightarrow (R \vee T) \wedge U$ est une tautologie du calcul des séquents.

Certaines de ces caractéristiques font penser au calcul des séquents en déduction modulo de Deplagne [25] (et section 3.5), qui abolit notamment de fait le niveau méta des séquents en introduisant un opérateur de liaison entre séquents pour ne plus avoir que des règles avec une seule prémisse..

Les inférences profondes sont un formalisme qui s'adapte bien aux logiques modales par exemple. Bien comprendre le rapport entre ces inférences profondes et la déduction modulo permettrait donc aussi de mieux répondre à la question de l'intérêt de la déduction modulo pour d'autres logiques.

Pour résumer cette thèse, la déduction modulo offre un cadre théorique général adapté à l'étude syntaxique et sémantique de la démonstration automatique. Cette généralité permet de traiter de manière homogène les problèmes théoriques usuels de la théorie de la preuve, et en particulier ceux relatifs à l'expression de méthodes de preuves automatiques.

Ce formalisme permet par ailleurs une amélioration de la lisibilité des preuves produites automatiquement. En effet, le calcul y est masqué et seul les étapes de raisonnement apparaissent dans les preuves produites.

Il reste à étendre la mise en œuvre de la déduction modulo. Il faut en particulier passer par l'implantation de méthodes utilisant ce formalisme pour pouvoir valider l'approche en pratique.

Bibliographie

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit subsitutions. *Journal of Functionnal Programming*, 1(4) :375–416, 1991.

[2] F. Baader and W. Snyder. Unification theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 8. Elsevier Science Publishers B.V., 2001.

[3] M. Baaz and C. Fermüller. Non-elementary speedups between different versions of tableaux. In 4^{th} *International Workshop, Tableaux'95*, volume 918 of *LNCS*, 1995.

[4] M. Baaz, U. Egly, and A. Leitsch. Normal form transformations. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 6. Elsevier Science Publishers B.V., 2001.

[5] L. Bachmair and H. Ganzinger. *Resolution Theorem Provin*, volume 1, chapter 2. Elsevier Science Publishers B.V., 2001.

[6] L. Bachmair, H. Ganzinger, and A. Voronkov. Elimination of equality via transformation with ordering constraints. In C. Kirchner and H.Kirchner, editors, *Automated Deduction — CADE-15.* 15^{th} *International Conference on Automated Deduction*, volume 1421 of *Lecture Notes in Artificial Intelligence*, pages 175–190. Springer-Verlag, 1998.

[7] B. Beckert. Konzeption und Implementierung von Gleichheit für einen tableau-basierten Theorembeweiser. IKBS report 208, Science Center, Institute for Knowledge Based Systems, IBM Germany, 1991.

[8] B. Beckert. Equality and other theories. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1999.

[9] B. Beckert, R. Hähnle, and P. Schmitt. *The even more liberalized δ-rule in free variable Semantic Tableaux*, volume 713. June 1993.

[10] R. Bonichon. TaMeD : A tableau method for deduction modulo. In D. A. Basin and M. Rusinowitch, editors, *IJCAR*, volume 3097 of *Lecture Notes in Computer Science*, pages 445–459. Springer, 2004.

[11] R. Bonichon and O. Hermant. A semantic completeness proof of tamed. In *LPAR*, 2006. Accepté à LPAR'06.

[12] R. Bonichon and O. Hermant. On constructive cut admissibility in deduction modulo (soumis aux Proceedings of types 2006).

[13] D. Brand. Proving theorems with the modification method. *SIAM Journal of Computing*, 4 :412–420, 1975.

[14] K. Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, Sept. 2003.

[15] G. Burel and C. Kirchner. Cut elimination in deduction modulo by abstract completion.

[16] D. Cantone and M. N. Asmundo. A sound framework for delta-rule variants in free variable semantic tableaux. In R. Letz, editor, *FTP*. Universitä Koblenz-Landau, 2005. 5$^{\text{th}}$ International Workshop on First-Order Theorem Proving.

[17] D. Cantone and M. N. Asmundo. A further and effective liberalization of the delta-rule in free variable semantic tableaux. In M. P. Bonacina and U. Furbach, editors, *Second Int. Workshop on First-Order Theorem Proving, FTP'98*. Technische Universität, Wien (Austria), 1998. URL `citeseer.ist.psu.edu/290631.html`.

[18] P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43(2) : 362–397, 1996. URL `citeseer.ist.psu.edu/curien96confluence.html`.

[19] M. Davis and R. Fechter. A free variable version of the first-order predicate calculus. *J. Log. Comput.*, 1(4) :431–451, 1991.

[20] M. De Marco and J. Lipton. Completeness and cut elimination in Church's intuitionistic theory of types. *Journal of Logic and Computation*, 15 :821–854, December 2005.

[21] A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 10. Elsevier Science Publishers B.V., 2001.

[22] A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid e-unification. *Theoretical Computer Science*, 166(1-2) :291–300, 1996.

[23] A. Degtyarev and A. Voronkov. What you always wanted to know about rigid e-unification. *J. Autom. Reason.*, 20(1-2) :47–80, 1998. ISSN 0168-7433.

[24] D. Delahaye and D. Doligez. From the Zenon automated deduction system to the Coq proof assistant. Draft dans le CVS de Focal.

[25] E. Deplagne. *Système de preuve modulo récurrence*. PhD thesis, Université Henri Poincaré - Nancy 1, 2002.

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

[27] G. Dowek and B. Werner. Proof normalization modulo. *The Journal of Symbolic Logic*, 68(4) :1289–1316, 2003.

[28] G. Dowek and B. Werner. Arithmetic as a theory modulo. In J. Giesel, editor, *Term rewriting and applications*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[29] G. Dowek, T. Hardin, and C. Kirchner. HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11(1) :1–25, 2001.

[30] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31 :33–72, 2003.

[31] M. Fitting. *First Order Logic and Automated Theorem Proving*. Springer-Verlag, 2nd edition, 1996.

[32] M. Giese. Incremental Closure of Free Variable Tableaux. In *Proc. Intl. Joint Conf. on Automated Reasoning, Siena, Italy*, number 2083 in LNCS, pages 545–560. Springer-Verlag, 2001.

[33] M. Giese and W. Ahrendt. Hilbert's ϵ-terms in Automated Theorem Proving. In N. V. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, Intl. Conf. (TABLEAUX'99)*, volume 1617 of *LNAI*, pages 171–185. Springer, 1999.

[34] J. Goubault-Larrecq. A BDD-based simplification and skolemization procedure. *Journal of the IGPL*, 3(6) :827–855, 1995.

[35] J. Goubault-Larrecq and I. Mackie. *Proof Theory and Automated Deduction*. Applied Logic Series. Kluwer Academic Publishers, 1997.

[36] A. Guglieglim. Deep inference and the calculus of structures. `http://alessio.guglielmi.name/res/cos/`.

[37] R. Hähnle. Tableaux and related methods. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 3. Elsevier Science Publishers B.V., 2001.

[38] R. Hähnle and P. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13(2) :211–221, 1994.

[39] O. Hermant. *Méthodes Sémantiques en Déduction Modulo*. PhD thesis, Université Paris 7 - Denis Diderot, 2005.

[40] O. Hermant. Semantic cut elimination in the intuitionistic sequent calculus. *Typed Lambda-Calculi and Applications*, pages 221–233, 2005.

[41] R. Jeffrey. *Formal Logic : Its Scope and Limits*. McGraw Hill, 1967.

[42] M. Kohlhase. Higher-order automated theorem proving. In W. Bibel and P. Schmidt, editors, *Automated Deduction : A Basis for Applications. Volume I, Foundations : Calculi and Methods*. Kluwer Academic Publishers, Dordrecht, 1998. URL `citeseer.nj.nec.com/kohlhase98higherorder.html`.

[43] J.-L. Krivine. Une preuve formelle et intuitionniste du théorème de complétude de la logique classique. *The Bulletin of Symbolic Logic*, 2 : 405–421, 1996.

[44] L. Lamport. How to write a proof. Technical report, Digital Equipment Corporation Systems Research Center, February 1993. `<http://focal.inria.fr/misc/SRC-094.ps>`.

[45] R. Letz. First-order tableau methods. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer Academic Publishers, Dordrecht, 1999.

[46] A. Nerode and R. A. Shore. *Logic for Applications*. Springer-Verlag, 1993.

[47] A. Nonnengart and C. Weidenbach. Computing small clause normal form. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 6. Elsevier Science Publishers B.V., 2001.

[48] S. Reeves. Adding equality to semantic tableaux. *Journal of Automated Reasoning*, 3 :225–246, 1987.

[49] N. Shankar. Proof search in the intuitionistic sequent calculus. In *CADE*, pages 522–536, 1992.

[50] R. Smullyan. *First Order Logic*. Springer, 1968.

[51] J. Stuber. A model-based completeness proof of extended narrowing and resolution. *Lecture Notes in Computer Science*, 2083 :195+, 2001. URL `citeseer.nj.nec.com/376987.html`.

[52] G. Sutcliffe and C. Suttner. The cade atp system competition. `http://www.cs.miami.edu/~tptp/CASC/`.

[53] C. B. Suttner and G. Sutcliffe. The TPTP problem library. Technical Report JCU-CS-96/9, 18 1996. URL `citeseer.ist.psu.edu/article/sutcliffe97tptp.html`.

[54] The CiMe Development Team. CiMe . `http://cime.lri.fr/`.

[55] The Coq Development Team. Coq. `http://coq.inria.fr/`.

[56] The Focal Team. The focal project. `http://focal.inria.fr/`.

[57] A. S. Troelstra and D. Van Dalen. *Constructivism in Mathematics*, volume 2. North Holland, 1988.

[58] J. van Heijenoort, editor. *From Frege to Gödel : A Source Book in Mathematical Logic, 1879-1931*. Harvard University Press, 2002.

[59] W. Veldman. An intuitionistic completeness theorem for intuitionistic predicate logic. *Journal of Symbolic Logic*, 41 :159–166, 1976.

[60] A. Voronkov. Proof-search in intuitionistic logic based on constraint satisfaction. In *TABLEAUX*, pages 312–329, 1996.

[61] A. Waaler. Connections in nonclassical logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 22. Elsevier Science Publishers B.V., 2001.

[62] A. Waaler and L. Wallen. Tableaux for intuitionnistic logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1999.

[63] B. Wack. Supernatural deduction. `http://www.loria.fr/~wack/papers/supernatural.ps.gz`, 2006.